Stochastic Simulation of P2P VoIP Network Reconfiguration Using Graph Transformation

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

by

Ajab Khan

Department of Computer Science

University of Leicester

October, 2011

I would like to dedicate this thesis to my loving parents.

Acknowledgements

My first and foremost thanks go to Allah, The Almighty, for all His blessings. This thesis definitely would not have been possible without my advisor Professor Reiko Heckel, who guided and encouraged me at each and every step of my PhD and showed me the way forward. Prof. Reiko has been a catalyst in infusing the thought process in me that is an asset for a new learner. While working on this thesis, he took great interest in reviewing and proof reading the drafts and his valuable feedback has made possible this thesis in its current shape. I sincerely appreciate the time and effort that he put into this thesis.

I am very grateful to my thesis committee members Prof. Thomas Erlebach, Head of the Department of Computer Science and Dr.Fer-Jan, PhD tutor, for their valuable advices and guidance.

I owe my most sincere gratitude to Dr. Polo Torrini, who gave me the opportunity to work with him during the development phase of the Graph-based Stochastic Simulator (GraSS). He has been very kind and gave me untiring help during my difficult moments.

My gratitude is also extended to István Ráth, PhD student at Department of Measurement and Information Systems, Budapest University of Technology and Economics, who has known the answer to every question I have ever asked him regarding VIATRA2. He has been very helpful while i was converting my model and transformation rule to executable form.

I owe my thanks to the Department of Computer Science, University of Leicester for its support during my PhD studies. The Department has been extremely helpful in financing me to attend conferences which have been an important and exciting element of my research. I extend my thanks to all the personnel of the department for creating an inspiring and comfortable research atmosphere.

The financial support of the University of Malakand, Pakistan is gratefully acknowledged.

Ajab Khan

Author's Declaration

I hereby declare that this submission is my own work and that it is the result of work done during the period of registration. To the best of my knowledge, it contains no previously published materiel written by another person. None of this work has been submitted for another degree at the University of Leicester or any other University.

Parts of this thesis submission appeared in the following publications, to each of which I have made substantial contributions.

1. Ajab Khan and Reiko Heckel. Model-based super peer promotion in P2P VoIP networks using graph transformation. Proceedings of the International Conference on Data Communication (DCNET2011) in Seville Spain. Best student paper award.

2. Ajab Khan, Reiko Heckel, Paolo Torrini, and István Ráth. Model-based stochastic simulation of P2P VoIP using graph transformation system. In Proceedings of the 17th International Conference on Analytical and Stochastic Modeling Techniques and Applications (ASMTA'10), Khalid Al-Begain, Dieter Fiems, and William J. Knottenbelt (Eds.). Springer-Verlag, Berlin, Heidelberg, 204-217.

3. Ajab Khan and Muzammal. Modelling overlay routing in P2P VoIP using graph transformation system. In Proceeding of the 23rd Annual European Simulation and Modelling Conference (ESM 2009), October 26-28, 2009, Holiday Inn Leicester, Leicester, United Kingdom.

4. Ajab Khan, Paolo Torrini, Reiko Heckel. Model-based simulation of VoIP network reconfigurations using graph transformation systems. Post-Proceedings of the Doctoral Symposium of the 4th International Confidence on Graph Transformation (ICGT-DS2009), ECEASST, Volume 16.

5. Ajab Khan. Model-based analysis of network reconfigurations using graph transformation systems. In Proceedings of the the Doctoral Symposium of the 4th International Conference on Graph Transformations (ICGT '08), Hartmut Ehrig, Reiko Heckel, Grzegorz Rozenberg, and Gabriele Taentzer (Eds.). Springer-Verlag, Berlin, Heidelberg, 502-504.

Abstract

Peer-to-Peer (P2P) networks provide an alternative approach to distributed systems, relaxing the requirements for dedicated servers from the client-server model. A P2P network operates as an overlay at application layer, on top of the physical network. In the early years of P2P, most applications lacked mechanisms for enforcing a particular overlay topology. This resulted in inefficient communication schemes, such as flooding or the maintenance of large numbers of connections with other peers. However, researchers and practitioners have realized the importance of constructing and maintaining appropriate overlay topologies for efficient and robust P2P systems.

P2P-based Voice over IP (VoIP) networks, such as Skype, distinguish client peers from super peers. This results in a two-level hierarchy: Peers with powerful CPU, more free memory and greater bandwidth take on server-like responsibilities and provide services to a set of client peers. But building and maintaining a super peer-based overlay topology is not easy. In particular, the uncontrollable and unpredictable behaviour of peers results in volatile overlay topologies. This makes it challenging to design reconfigurable and stable networks that provide good Quality of Service (QoS). Various solutions have been proposed. However, peer dynamics, scale and complexity make it hard and expensive to validate them by testing. Simulation can help to validate network designs and protocols, but most existing approaches cannot cope with unbounded dynamic change of network topology.

We propose a new approach to the modelling and simulation of P2P network reconfigurations using graph transformation, a visual rule based formalism. Based on existing alternatives we classify network design variations by means of a feature tree. Focussing on P2P VoIP applications, we develop a structural model and transformation rules to compare alternative solutions to the problems of selection and connection to super peers, peer promotion, and load balancing, evaluating their QoS properties. We validate the model using statistics from the real Skype network and experimental data in the literature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet has passed through different stages of evolution. Over it's 40 years [169] of history there is a dramatic shift from static HTML pages, via client-server-based web applications, to multimedia and file sharing [58]. Most of the current sharing of voice, multimedia and files operates in Peer-to-Peer (P2P) style [54].

The popularity of P2P in the Internet community is evident through applications like Skype [149], Gnutella [48], KaZaA [96] and SETI@Home [6]. P2P networks are the subject of academic research and industrial efforts such as the P2P Working Group led by companies such as Intel, HP, Sony [107].

P2P networks can comprise hundreds of millions of peers with individual resources and capabilities. Peers are characterised by a high degree of dynamism. With nodes joining or leaving the network, their behaviour is difficult to predict. This makes it hard to design flexible and stable systems that can operate effectively with minimum cost.

In P2P, neither a central authority nor a fixed communication topology can be

employed to control various components. Instead, a dynamically changing overlay topology is maintained on top of the physical Internet. The overlay topology is maintained by the peers and is created dynamically based on the requirements of the application [111], such as connectivity, indexing and routing, availability, scalability, fault-tolerance and load balancing [33].

A vital tool of P2P systems is the overlay network architecture supplied by the embedded protocols, mechanisms and algorithms. Most P2P applications on the Internet lack mechanisms for enforcing an efficient overlay topology, resulting in the adoption of communication schemes such as flooding [111]. Several academic and industrial research projects have recognised the importance of selecting, constructing and maintaining appropriate topologies for the implementation of efficient and robust P2P systems [32; 135; 186].

However, designing an overlay network is a challenging task. A number of requirements have to be met for the network to be deployed without unduly diminishing the resources of the underlying Internet. It has been observed that most overlay designs stress specific requirements while ignoring other important ones. Such narrowly targeted approaches are inflexible in adapting to emerging requirements, which will require redesign [33]. This can be observed by analysing a well-known application like Guntella, which offers a non-hierarchical P2P approach with minimal maintenance cost. However, the employed flooding mechanism used for querying makes it unscalable [33] and causes system breakdown when the size is increased. Keeping in view the flaws in the Gnutella protocol, two approaches have been followed by P2P network designers.

The first approach is based on distributed hash table (DHTs) that index items to be mapped on the overlay network [33]. Peers are declared by Globally Unique

Identifiers. DHT-based systems aim to provide efficient routing mechanisms, but the cost of the routing grows logarithmically with the size of the overlay network for most of the proposed approaches. Although this increases the robustness of the overlay network, such a large number may not be feasible for peers with low capabilities. Further, this solution may became inefficient in scenarios where high peer dynamism exists.

The second approach introduces super peer architectures, which are the focus of this thesis. Nodes that are faster, more reliable and have more memory take on server-like responsibilities and provide services to ordinary peers. The super-peer paradigm allows decentralised networks to run efficiently by exploiting peer heterogeneity for the distribution of load [111]. The usefulness of the super-peer approach is not limited to file-sharing systems, but some popular VoIP applications such as Skype [53; 173] are also using this approach. Distributed games [78], Grid Management [65] and distributed storage are other applications of this paradigm. The approach overcomes some of the flaws of the client-server model, such as single point of failure. Building and maintaining a super peer-based overlay topology is not straightforward. Rapid architectural changes in both ordinary and super nodes require robust and efficient protocols, capable of self-reconfiguring the overlay topology in spite of both controlled and unexpected events, like the joining, leaving or crashing of nodes. In case the P2P network is used for VoIP traffic, it needs to reconfigure fast enough so that Quality of Service (QoS) is not affected [87].

A basic drawback of this approach is the requirement for super peers [33]. Super peer failures are more severe than those of normal peers as they increase the reconfiguration effort in the architecture. Super peers may also become perfor-

mance bottlenecks, if there is no sufficient number or unstable peers are promoted to the super peer-role.

P2P networks are still in their infancy, with new applications and protocols emerging frequently. However, it is difficult to evaluate them before their deployment on a large scale. Therefore they are often short-lived and disappear as fast as they emerge. One of the reasons behind this is unsatisfactory performance. As the list of failed P2P systems and protocols grows, the need for evaluation increases [161]. Testing a system before its deployment is common practice in software engineering. There are two main possible streams: Experimentation with the actual system or with a model of the system [161]. Testing a running P2P network or protocol in a realistic environment is often not feasible. However, it is possible to simulate a model of the network to evaluate it in a controlled environment

1.1 Problem Statement

Availability and reliability of P2P networks is gaining importance as their size and demand increase. Most current research in P2P networks is based on the cooperative model. It is considered that, although there can be rogue peers in a system, most of the peers are trustworthy and follow the same protocol. But this assumption does not always hold true, especially when the network is large, dynamic and distributed, such as in a super-peer based P2P VoIP network.

Overlay networks are subject to functional requirements like connectivity and routing and non functional requirements like performance and reliability. The problem in an open network is that peers and super peers belong to end users,

and without any centralized authority both can join or leave the network as they wish. A super peer may behave selfishly and drop messages which it was supposed to forward to conserve bandwidth or other resources. The selection of stable peers for promotion to super peer and super peer selection by clients are other notable events that affect the health of the super peer topology. Similarly, when a super peer switches its role to peer-only, as a result of local recovery or load balancing, the dependent peers need to find another host quickly enough so that QoS is not unduly affected. Said problems require for their solution architectural changes and, if the network fails to reconfigure in time, may result in loss of communication.

The dynamic nature of peers and super peers poses several questions for the design of network protocols:

- Which peer should be promoted to the role of super peer?
- When should a super peer be demoted to the role of peer?
- Which super peer should a new client peer connect to?
- Can we predict that a super peer will be capable of providing VoIP services to all connected peers?
- What shall we do when a super peer selfishly leaves the network?
- How can the load on super peers be balanced?

The performance of a protocol can be measured by answering the following questions:

- How many clients are provided with a good network connection through super peer?
- How many clients are currently linked to super peers?
- How many are waiting to connect?
- What is the number of peers and super peers, and their ratio?

Several approaches have been proposed to address issues of promotion, selection of suitable super peers and effects of peer dynamics. However, the complexity of P2P networks makes it difficult to validate these solutions. Geographic distribution of peers, network dynamics, and lack of central control make testing impossible. Simulation of network reconfigurations is not easy, as existing simulators provide limited support for networks with dynamic topology [61; 87; 113].

We propose to model complex network reconfigurations in P2P VoIP networks by graph transformation systems, a visual rule-based formalism, and use a new approach to the stochastic simulation of such systems to evaluate the performance of network protocols. We consider the network architecture as a graph in which nodes are represented by graph vertices and graph edges represent network connections. Reconfiguration can naturally be modelled by graph transformation [61; 85; 87]. Stochastic simulation techniques for validation have been developed in [86; 87]. The aim is to model different protocols in order to evaluate and improve their QoS properties with focus on VoIP applications.

1.2 Modelling and Simulation Requirements

P2P VoIP networks such as Skype [97] operate on unstructured topologies distinguishing client peers from super nodes. The Skype network consists of millions of clients resulting in a high degree of peer dynamics, with a continuous flow of clients going online, offline and crashing. Another crucial consideration is the intrinsic heterogeneity of physical capabilities, which aggravates the problem of workload distribution. In the following, we discuss some of the requirements that are essential for modelling, simulation and evaluation of super-peer based P2P VoIP networks [31; 113; 114; 138; 159].

1. **Promotion, Demotion and Reconnection**

   Super peers are selected from the existing pool of clients which meet the requirement for this role. Promotion is not a user choice but maintained by the protocol architecture. Similarly, if a super peer no longer meets the requirements of this role, it should be demoted and client only status restored. When super-peer status is attained, new peer connection requests will start to arrive until no new connection can be accepted. Similarly, when the role is switched to client-only, existing dependent client peers will reconfigure to connect to a new super peer. This results in a dynamic topology where clients switch their role and connections in the network.

2. **Model of Churn**

   The dynamics of peer participation, or churn, are an inherent property of P2P systems and critical for their design and evaluation. Accurately characterizing churn requires precise and unbiased information about the

arrival and departure of peers, which is challenging to acquire. Churn in a Skype-like VoIP network follows two possible mechanisms. Peers and super peers can leave and join the system either cooperatively using the proper exit procedure, or in a selfish manner, such as by crashing. The cooperative exit will enable the dependent client and super peers to retract their overlay connection or dependency link. In the event of selfish exit, dependent and super peers need to detect and reconfigure fast enough for new connections to be established.

3. **Alternative Solutions**

   There are a number of recommendations and algorithms for the selection of peers for promotion. Some solutions concentrate on requirements, such as promotion based on bandwidth, memory etc., whereas others suggest promoting as many super peers as possible. Apart from promotion, the selection of a suitable super peer by a client is another important concern. A number of approaches have been proposed, including random selection, latency-based selection, local super peer selection, and stable super peer selection. We require the approach to be flexible enough to evaluate and compare these solutions.

4. **User Behaviour**

   The approach should be able to model the application layer and user behaviour. Modelling user behaviour is essential, as distribution of peers and local user control over the services they contribute result in a highly dynamic network. Selfish user behaviour makes the P2P recovery process more vulnerable, especially when a user leaves the network by crashing his

system. In this case, neither probe messages are generated nor time is given to dependent hosts to recover.

5. **Simulation Model**

   In order to model realistic P2P VoIP, the simulator should support stochastic modelling. This will enable us to generate randomly parameters like bandwidth, memory, type of Internet connection and background traffic. These random events will create a controlled environment close to the real networks.

6. **Scalability**

   The approach should be able to simulate peers and super peers in hundreds or more, in order to provide convincing results. In order to observe meaningful intervals the tool must be able to run long simulations.

7. **Statistics**

   The study of the literature shows that most of the simulation tools lack flexibility in terms of gathering and displaying results. The usability of an approach depends on the ability to provide results with minimal effort, to include new parameters and support statistical tests, such as confidence intervals.

1.3 Contributions

As discussed in the previous sections, this thesis addresses the challenges in modelling and simulation of overlay networks for large-scale, dynamic and heterogeneous P2P systems. Our contributions are summarized below.

1. **P2P VoIP Protocol Variations as Feature Diagram**

   The first step in modelling a system is to analyze its requirements. By investigating relevant case study applications such as Skype, a rich set of features have been identified and modelled using feature diagrams (Chapter 4). These features provide input for the development of a valid model. Further, we identify a plethora of existing solutions proposed for super-peer selection, promotion and load balancing. Feature diagrams are used to model these alternative solutions.

2. **Modelling and Stochastic Analysis Approach**

   In this thesis, we have proposed a new approach for the stochastic modelling and analysis of super-peer based P2P VoIP networks. The approach is based on graph transformations, a visual-rule based formalism that has been supported by facilities for stochastic simulation.

3. **Metamodel and Rules for Super Peer-based P2P VoIP**

   This thesis also proposes a new visual model for churn and reconfiguration in P2P VoIP protocols, following the requirements stated in Section 1.2. The newly developed approach consists of a metamodel for super peer-based P2P VoIP and transformation rules describing different protocols (Chapter 6).

4. **Mapping to Executable Simulation Model**

   We have converted the metamodel into a VIATRA2 model space and the rules into VIATRA2 transformation rules (Chapter 7 and Appendix B). The aim is to convert the model and transformation rules into an executable form for stochastic simulation.

5. **Analysis of Protocol Variations by Simulation**

   We have evaluated the alternative solutions to super peer selection, promotion, load balancing, etc., using stochastic simulation. The results are compared to enable the designer to look into various alternatives in a single model and rank their performance (Chapter 8).

6. **Proposal for Improved Protocol**

   Based on the evaluation, a custom-built protocol has been proposed, which performs better than the parent protocols (Chapter 8).

7. **Model Validation and Threats to Validity**

   The model is validated using animation, event validation and a comparison to the real system (Chapter 9). Threats to the validity of the simulation results have been identified and analysed (Chapter 9).

1.4 Outline of the Thesis

The contents of the dissertation are divided into eleven chapters. Following this introductory chapter.

Chapter 2
Introduces the concepts of graph transformation used for modelling P2P VoIP networks. We also explain the algorithm underlying the simulation tool.

Chapter 3
Discusses P2P networks and their evolution from structured to unstructured and super-peer architectures. It also provides an overview of popular P2P applications running over the internet.

Chapter 4

Discusses P2P VoIP networks and identifies the core features of their operation. Alternative solutions to the problems of super-peer promotion, selection, and load balancing, etc., are modelled using feature diagrams.

Chapter 5

Describes in more detail the case study of Skype including Skype's network architecture, operation, and statistics. It provides a basis for the model and their validation.

Chapter 6

Presents a structural model of a Skype-like P2P VoIP protocol. The model is based on the discussion of the case study and the design variations studied before. It will be given in the form of a metamodel supporting the design alternatives. In this chapter, we also introduce a set of transformation rules in graphical form based on the case study in Chapter 5. The transformation rules present a complete view of the VoIP protocol and associated variations as discussed in Chapter 4.

Chapter 7

Discusses the operation and use of the Graph Based Stochastic Simulator (GraSS). We provide an overview on how this simulator could be used. This chapter also provides background on the conversion of rules to VIATRA2 and details of confidence intervals. In the last part of this chapter, we discuss the contribution of this thesis towards the development of GraSS. The metamodel and the transformation rules presented in Chapter 6 are converted into their representation inside the simulation tool.

Chapter 8

Describes the simulation of the models and associated variations. This chapter will use the feature model presented in Chapter 4 along with the model presented in Chapter 6 and in its executable form in Chapter 7.

Chapter 9

Will validate the model, evaluate the approach and comment on the simulation results. In this chapter, we also discuss threats to the validity of the results.

Chapter 10

Discuss related work including various alternatives proposed to cope with modelling churn, peer promotion, demotion and reconnection. In the last part of this chapter, we discus various simulation tools and their shortcomings. We conclude this chapter by comparing existing approaches against the requirements of this thesis.

Chapter 11

Concludes this thesis and provides directions for future research in this area.

Chapter 2

Stochastic Graph Transformation

This chapter introduces the concepts of graph transformation used for modelling P2P VoIP networks. We start by defining type and instance graphs, node type inheritance, and attributed graphs. Our graph transformation systems consist of rules transforming typed attributed graphs following the algebraic single pushout approach. We explain stochastic graph transformation in this thesis based on different probability distributions and illustrate the simulation algorithm used.

2.1 Graphs

Graph transformation systems provide a language for manipulating graphs and graph-based structures as they come about in representations of programming language semantics, databases, object-oriented systems, and distributed systems. Because of their formal, operational semantics these descriptions can be analyzed as well as executed using suitable graph transformation tools.

In the literature, graph transformation approaches are broadly categorised in two mathematical techniques: the set-theoretic approaches and the algebraic approaches [56]. The difference is the definition of application of transformation rules to the host graph. The algebraic approaches uses pushouts to model the gluing of graphs. The double pushout approach proposed by Ehrig, Pfender and Schneider in the early seventies [41] uses two gluing constructions to model a graph transformation step. The singl-pushout approach (SPO) uses a single pushout for modelling transformations. In this thesis, we will use the SPO.

Graphs are directed and can have parallel edges as well as loops. The following definition is reproduced from [56].

Definition 2.1 (Graph).
A graph $G = (V, E, src, tar)$ consists of a set V of nodes (also known as vertices), a set E of edges, and two functions $src, tar : E \to V$, the source and target functions:

$$E \underset{tar}{\overset{src}{\rightrightarrows}} V$$

Definition 2.2 (Graph Morphism).

A graph morphism $f : G \to H$ between graphs G and H is a pair of total functions $f = (f_V : G_V \to H_V, f_E : G_E \to H_E)$ preserving source and target, i. e., satisfying $src_H \circ f_E = f_V \circ src_G$ and $tar_H \circ f_E = f_V \circ tar_G$.

2.2 Type Graphs and Typing Morphisms

The simplest way to restrict the shape of a object is to specify a *type* for that object. In graph transformation this is achieved by means of a type *graph*.

Definition 2.3 (Type Graph).

A type graph is a graph $TG = (V_{TG}, E_{TG}, src, tar)$. An instance graph is a tuple $(G, type)$ of a graph G and morphism $type : G \to TG$

Type graphs can be presented by UML class diagram, as shown in Fig. 6.1. Similarly, instance graphs can be represented by corresponding *UML object diagrams* as shown in Fig. 2.2. As per UML syntax, each node is labelled by an identifier followed by a reference to its type in the type graph. Edge labels do not contain any identifier but refer to the respective edge type only.

2.3 Node Type Inheritance

In this thesis, we are using inheritance to allow *abstract node types* analogous to *abstract classes*. The definitions below are based on [12; 13].

Definition 2.4 (Type Graph with Inheritance).

A type graph with inheritance is a triple $\widehat{TG} = (TG, I, A)$ consisting of a type graph $TG = (TG_N, TG_E, src, tar)$, a set $I \subseteq TG_E$ of inheritance edges, and a

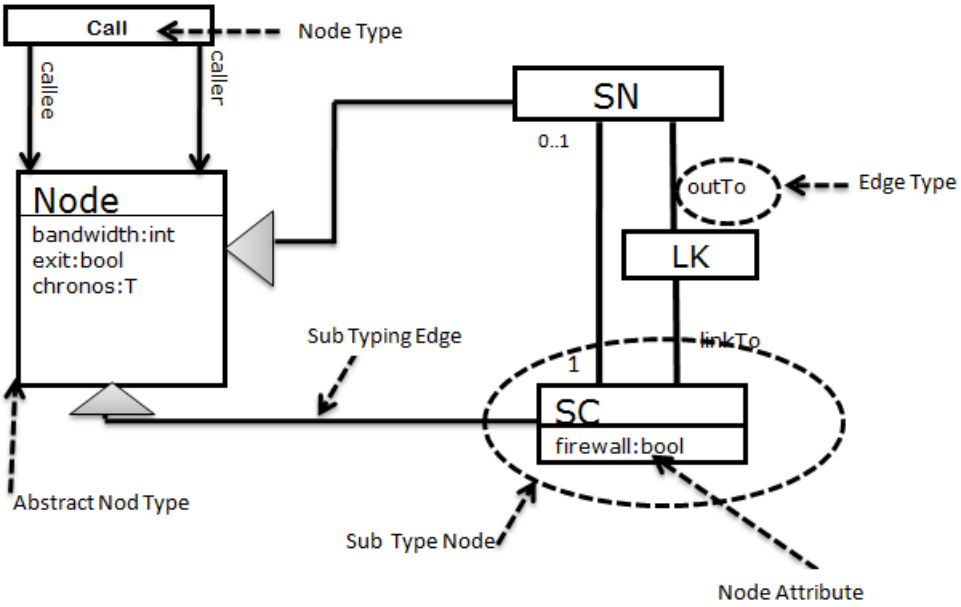

Figure 2.1: Type graph as UML class diagram

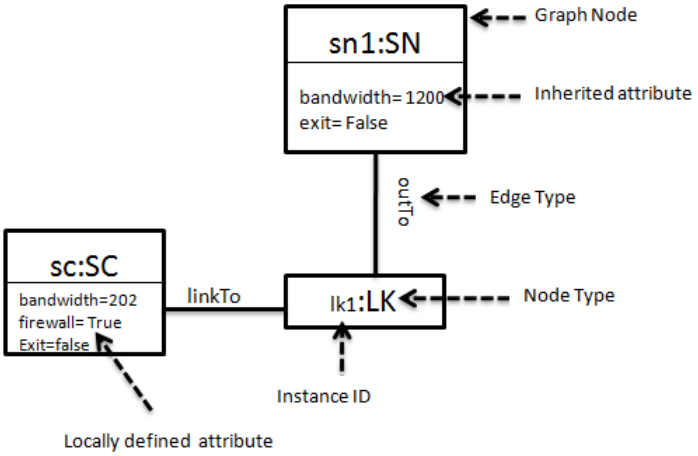

Figure 2.2: Instance graph as UML object diagram

set $A \subseteq TG_N$ *of abstract nodes. The subgraph* $TG|_I = (TG_N, I, src|_I, tar|_I)$ *is called* inheritance graph. $TG|_I$ *has to be acyclic. For each node type* $n \in TG_N$*, the set of its subtypes, also called* inheritance clan*, is defined by* $clan(n) = \{n' \in TG_N \mid \exists \text{ path } n' \xrightarrow{*} n \text{ in } TG|_I\}$ *where paths of length 0 are included, i. e.,* $n \in clan(n)$.

Fig. 6.1 represents the type graph as a UML class diagram. The type graph has one abstract node type Node and two subtypes SC and SN. The node type Call is used to model calls. As this type is connected to the abstract node, both SN and SC can use it. In the type graph we see that Node has three attributes *bandwidth*, *exit* and *chronos*. Each subtype can adopt these attributes as well as add a new local attribute if required. In Fig. 6.1 we see that SC has a locally defined attribute firewall. The type graph also shows relation between SC and SN, established through another node type LK. The type graph further shows cardinality constraints between SC and SN to ensure that an SC can have only one extended role of SN.

When defining instance graphs for type graphs with inheritance, we have to allow instances of a subtype to have incoming or outgoing edges that are specified for one of its supertypes. This type of inheritance-enabled typing is expressed using a mapping, called clan morphism [12; 13].

Definition 2.5 (Clan Morphism). *Given a type graph with inheritance* $\widehat{TG} = (TG, I, A)$ *with* $TG = (TG_N, TG_E, src_{TG}, tar_{TG})$ *and a graph* $G = (G_N, G_E, src_G, tar_G)$*; a* clan morphism $type_G : G \to TG$ *is a pair of functions* $type_G = (type_N : G_N \to TG_N, type_E : G_E \to TG_E \setminus I)$ *with:*

- $\forall e \in G_E : \ type_N \circ src_G(e) \in clan(src_{TG} \circ type_E(e))$ *and*

- $\forall e \in G_E : \ type_N \circ tar_G(e) \in clan(tar_{TG} \circ type_E(e))$.

If $f : H \to G$ is a graph morphism, then $type_G \circ f : H \to TG$ is a clan morphism, too [12]. With the help of clan morphisms, we can now provide a more general definition of instance graphs respecting inheritance:

Definition 2.6 (Instance Graph).
Given a type graph with inheritance $\widehat{TG}$, a $\widehat{TG}$-typed instance graph $\langle G, type_G \rangle$ is a graph G equipped with a clan morphism $type_G : G \to TG$. Simply we will use G instead of $\langle G, type_G \rangle$ if the context reveals that we are considering instance graphs.

Fig. 2.2 shows the instance graph. We see that two instances of abstract node type Node have been linked with each other through another instance of node type LK.

2.4 Attributes and Typed Attributed Graphs

Nodes can be enriched with attributes used to store additional information. As in object-oriented languages, an attribute has a unique name and data value. However, in the context of a type graph, we have to declare the attributes that belong to a certain node type by a name followed by their data type. After declaration, each node in the instance graph can store different values in the attribute, but of similar data type. Further, the attributes also play an important role in the application of GT rules. A rule will only be applied if a certain node meets the attributes constraints of the rule. In the case study of this thesis, an SC

will only be promoted to the extended role of SN if an SCs bandwidth attribute has a value of more than 1.5 Mbps.

Typed attributed graphs and attributed graph transformations have been formally defined in [42; 62]. In this thesis, we use the approach of [158]. The definition of attributed graphs is reproduced from [158].

Definition 2.7 (Attributed Graph).
A graph $G = (G_N, G_E, src_G, tar_G)$ *is attributed over a data node set* A*, if* $A \subseteq G_N$ *and* $\forall e \in G_E : src_G(e) \notin A$.

In this thesis, the set of possible values that can be assigned to attributes is referred to as *domain*. The values belonging to the domain are partitioned into disjoint sorts representing different basic data types.

Definition 2.8 (Domain). *A domain* $Dom = (SN, V, X, sort)$ *is a tuple with a set* SN *of sort names, a set* V *of attribute values including a subset* $X \subseteq V$ *of variable names, and a function* $sort : V \to SN$ *associating every value and variable with a sort.*

Fig. 6.1 shows the type graph where the abstract Node type has three attributes bandwidth, exit and chronos. The bandwidth is an integer attribute, exit is a boolean attribute and chronos is of type T (Time).

As we want to declare node attributes in a type graph using sort names as attribute values, type graphs are attributed over the set of sort names SN. To formally define the attributed type graph, we use the definition by [158].

Definition 2.9 (Attributed Type Graph).

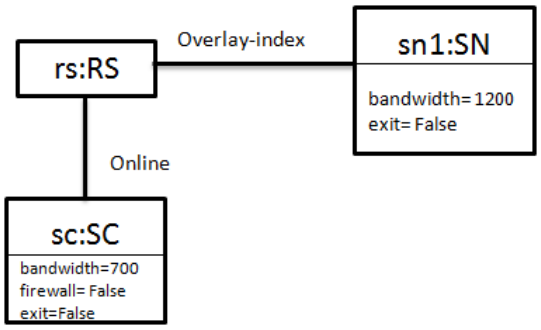

Figure 2.3: Instance attributed graph

For a given domain $Dom = (SN, V, X, sort)$*, a Dom-attributed type graph is a type graph with node inheritance* $\widehat{TG} = (TG, I, A)$ *where* TG *is attributed over* SN*.*

The instance graphs of attributed type graphs are attributed over concrete values and variables and respect the inherited attributes. In order to define attributed instance graphs we use the definition from [158].

Definition 2.10 (Attributed Instance Graph).
Given a domain $Dom = (SN, V, X, sort)$ *and an Dom-attributed type graph* $\widehat{TG}$*, an attributed instance graph is a* $\widehat{TG}$*-typed instance graph* $\langle G, type_G \rangle$ *with* G *attributed over* V *and* $type_N(v) = sort(v)$ *for all* $v \in V$*.*

Fig. 2.3 shows an instance graph with three nodes types RS, SN and SC. The type SC has inherited attributes bandwidth and exit, and a locally defined attribute firewall. Further, this shows how different values are assigned to there attributes.

2.5 Graph Transformation Rules

Graph transformation rules consist of a left-hand side (LHS) and a right-hand side (RHS) each. The left-hand side and right-hand side are instance graphs L and R. The left-hand side represents the pre-conditions of the rule, whereas the right-hand side represents its effects as post-conditions. It is important to note that L and R may be typed by abstract types and their attributes may have variables as values.

Whenever there is an occurrence of the left-hand side L in an instance graph G that conform the typing and attribute values in L, this occurrence is also know as match of L in G. If the rule is applied, the match of L is replaced by the graph in R. In this event all those elements which are present in L, but are not present in R, will be removed from the host graph.

Definition 2.11 (Graph Transformation Rule).
For a fixed domain $Dom = (SN, V, X, sort)$, a graph transformation rule over a Dom-attributed type graph $\widehat{TG}$ is given by $p = (L \xrightarrow{r} R, NAC)$, where $p = (L \xrightarrow{r} R, NAC)$ consists of a rule name p and graphs L, R attributed over V such that $L \cap R$ is well defined.

L represents the pattern of elements required for the application of the rule, R represents the pattern used to replace the pattern selected as L. NAC is a set $nac = (N, n)$ with N being a graph and $n : L \to N$ a graph morphism.

The diagram in Fig. 2.4 illustrates the definition. The arrows describe graph morphisms and dashed arrows describe clan morphisms. N is the graph of a negative application condition where $N \setminus L$ represents the forbidden structure.

Fig. 2.5 depicts an example of a graph transformation rule. According to

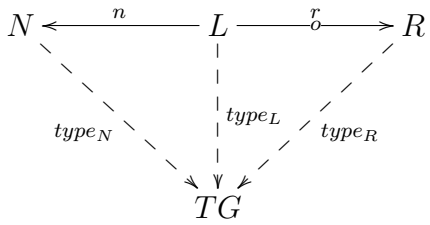

Figure 2.4: Algebraic constituents of a graph transformation rule

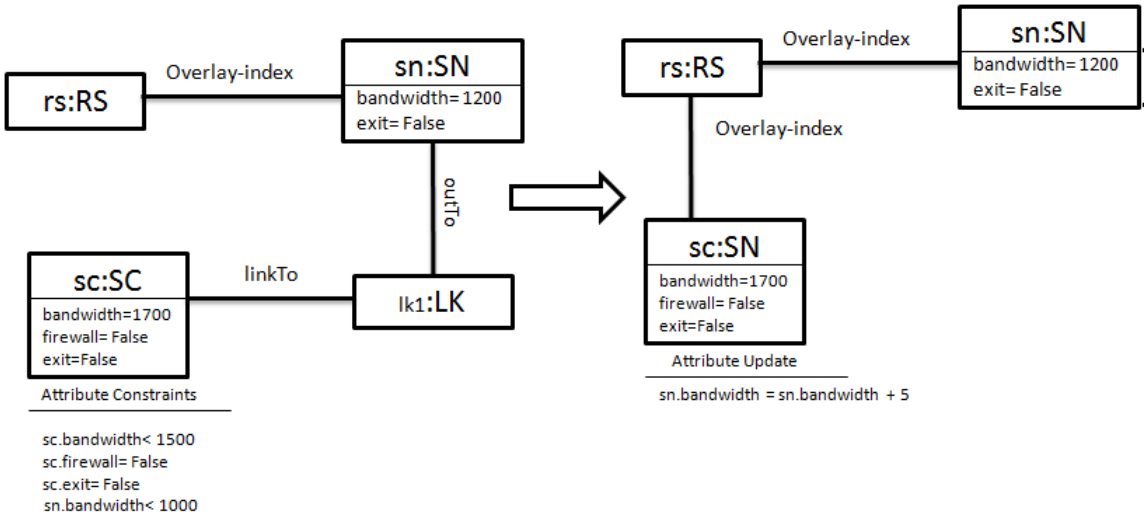

Figure 2.5: Graph Transformation Rule

its left-hand side L, it requires as pre-condition the existence of a node of type SC and three other nodes of types RS, SN and LK. The rule further states the attribute constraint that SC must have a current bandwidth value of more than 1.5 Mbps and the exit attribute is set to false. According to the right-hand side R, applying this rule deletes the first LK-node, promotes SC to type SN and links it with RS.

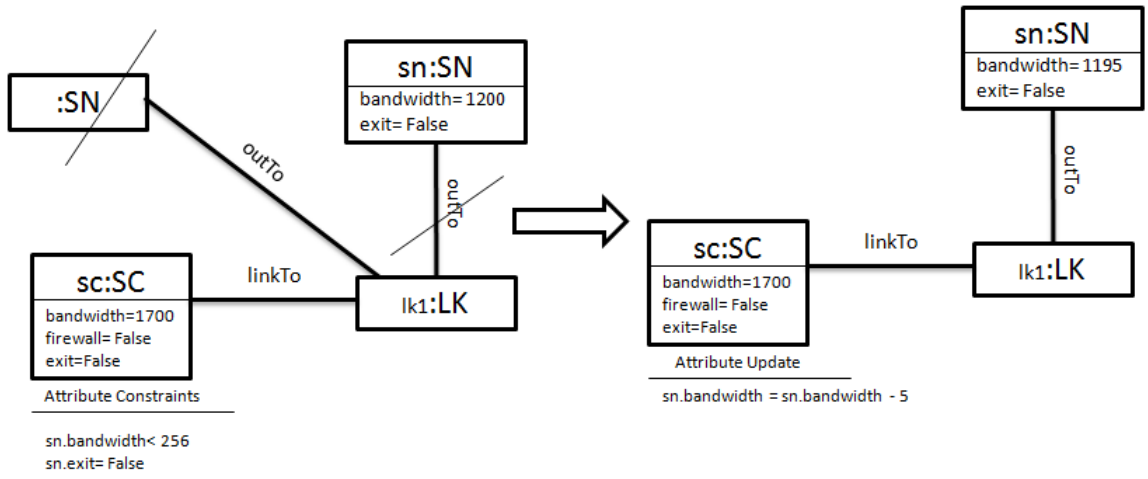

Figure 2.6: Negative application condition

2.6 Negative Application Conditions

Until now we have discussed that the left-hand side L of a transformation rule specifies which pattern has to be present in the instance graph in order to allow their application. But sometimes we require that the rule should only be applied if certain edges or nodes are not present in the instance graphs. In order to cope with this situation, we add *negative application conditions* (NACs) to the transformation rule [15; 57].

Fig. 2.5 depicts an example of a graph transformation rule with two negative conditions. The edges or nodes crossed with a single line model the negative conditions. The first condition makes sure that the SC is not linked with an SN through a node LK. The second NAC make sure that the newly selected SN is not linked with an SN. If these condition are evaluated as true and other conditions, such as attribute constraints, are satisfied, then this rule will re-link a disconnected SC with a new SN.

2.7 Graph Transformations System

A graph transformation system (GTS) consists of a type graph, transformation rules and a start graph. In this thesis, we follow the algebraic *single pushout approach* (SPO) to graph transformation [56].

Definition 2.12 (Graph Transformation System (GTS)).
A graph transformation system $G = \langle TG, P, \pi, G_0 \rangle$ *consists [91] of a type graph* TG*, a set* P *of rule names, a function mapping each rule name* p *to a* TG*-typed rule* $p = (L \xrightarrow{r} R, NAC)$*, and an initial* TG*-typed graph* G_0*.*

2.8 Stochastic Graph Transformation Systems

In order to reason stochastically about a GTS, we want to associate with each rule a distribution function governing the timing of the application of the corresponding step. Thus, a stochastic graph transformation system (SGTS) is a graph transformation system that associates distributions to rules. In this thesis we support normal and exponential distributions.

We say that $S = \langle G, F \rangle$ is a *stochastic graph transformation system* whenever G is a GTS and $F : P \to (R \to [0, 1])$ is a function which associates with every rule in G a continuous distribution function.

Our interest in stochastic graph transformation systems is closely associated with their simulation, where the stochastic aspect is useful in order to resolve the non-deterministic character of ordinary GTSs. We rely on standard notions of stochastic process and discrete event system [23] for an intuitive presentation of our approach. The behaviour of a stochastic GTS can be described as a stochastic

process over continuous time, where reachable graphs form a discrete state space, the application of transformation rules defines state transitions as instantaneous events, and interevent times, determined by the application of transformation rules, are dominated by continuous probability distributions. More precisely, we associate each rule, as it becomes enabled by a match, with an independent random variable (*timer*) which represents the time expected to elapse (*scheduled delay*) before the rule is applied to the match. This timer is set randomly, based on a continuous probability distribution function. The null delay condition means that the probability of a null delay at enabling time is always zero.

Thus, for a GTS to become a SGTS, the GT rules defined in the respective GTS require to be associated with continuous probability distribution functions (CDF) such as normal or exponential. In order to complete this task, a clear difference has to be identified between the rules carrying normal and exponential distribution. The selection of an appropriate distribution in this study is based on the real statistics and network analysis performed. Further, distribution fitting is the procedure of selecting statistical distribution that best fits the existing data set [156]. However, it is not always possible to have data for selection of a probability distribution. Instead, such a selection can also be made based on distribution properties and theories such as the memory-less property and the central limit theorem. For example, the exponential distribution is the only memoryless continuous probability distribution. Hence, memory lessness is a unique categorization for the exponential distribution.

If we know how many times in a day, a user tries to go online in the Skype network, or how often an on-line person will activate a VoIP call, then these events can be considered as following an exponential distribution. On the other hand if

we know the average length of time it takes to complete an action, such as the average time of the VoIP call, then it can be categorized as a normal distribution or a log-normal distribution, which defines a positive normal distribution.

2.9 Probability Distributions

In a SGTS a probability distribution is associated with a GT rule. In this thesis, we have divided our GT rules in two groups based on their probability distribution. We use normal or log-normal distribution and exponential distribution.

The exponential distribution is always the choice to represent the time between events that happen at a constant average rate such as inter-arrival time and rate of decay. Because of this behavior, the exponential distribution is usually used to model the mean time between occurrences, such as arrivals or failures. Exponential distributions tie together mean and variance, respectively $1/\lambda$ and $1/\lambda^2$, where λ is the exponential rate — so it is possible to decrease standard deviation only by decreasing the expected mean.

The normal distribution is always the choice to represent the average time for an event, such as the average time a barber takes to cut a persons hair. Moreover, normal distributions have a quite stable character, intuitively visualised by the bell curve functions associated with them — meaning that essentially, experimental values tend to hurdle within short range from the mean in terms of standard deviation. The normal distribution requires mean μ and variance σ^2.

The log-normal distribution can be defined as a variable resulting from the multiplicative product of many independent random variables each of which is positive. According to probability theory, a log-normal distribution is a proba-

bility distribution which defines a positive normal distribution. The log-normal distribution is also represented by mean μ and variance σ^2.

2.10 Stochastic Simulation

Simulation is one of the prime tool for study, analysis and comparison of P2P approaches and algorithms. The cost of implementation of simulation models is far lower then large-scale experiment. Further, if a simulation model is carefully constructed, then it can be more realistic [136].

For example the bandwidth or latency, in the deterministic approach has to be specified from the beginning of the simulation such as what is peer bandwidth and what is the latency of a particular channel. In a stochastic model, the peer will be assigned bandwidth randomly as it joins the network, and similarly the latency will randomized.

We simulate our model using GraSS (for Graph-based Stochastic Simulation), a new tool introduced in [162]. The tool has been developed in Java-Eclipse, as plugin of a graph transformation engine called VIATRA2. VIATRA2 [16] relies on a RETE-style implementation of incremental pattern-matching, in which precomputed matching information is stored and updated as transformation proceeds.

Essentially, the stochastic engine receives the set of enabled rule matches, (i.e. the active events) from the transformation engine, turns them into timed events, by assigning to each of them an expected time value, randomly determined on the basis of the probability distribution which is associated with the event type, and sends the events that has been scheduled first back to the transformation engine

for execution. In GraSS a GTS is represented as a VIATRA2 model, consisting of the model space with the current graph and the transformation rules.

The simulation algorithm for stochastic simulation consists of the following steps [63; 87]. The algorithm relies on calls to Random Number Generator (RNG) in order to provide delay values for the timers associated with newly activated events. Active timed events are managed as a list ordered by the time for the times events are scheduled for, so that the first element of the list turns out to be the event scheduled to take place first. Simulation time needs to be recorded explicitly.

Assume a Generalised semi-Markov Process (GSMP) $P = \langle Z,\ E,\ active,\ new,\ \Delta, s_0 \rangle$ [87; 91] where Z is a set of system states; E is a set of implicitly timed events; $active$ is the activation function, so that $active(s)$ is the finite set of active events associated with s; new is the transition function depending on states and events; Δ is the distribution assignment, so that $\Delta(e)$ is the probability distribution function associated with the delay of event e; and s_0 is the initial state. Given a GSMP $P = \langle Z,\ E,\ active,\ new,\ \Delta, s_0 \rangle$

1. The simulation time is initialised to t_0.

2. The set of the activated events $A = active(s_0)$ is computed.

3. For every event $e \in A$, a scheduling time t_e is computed by adding t_0 to a random delay value d_e provided by the RNG on the basis of the probability distribution function $\Delta(e)$;

4. The active events with their scheduling times are collected in the scheduled event list $l_{s_0} = \{(e, t_e) | e \in A\}$, ordered by the scheduling times.

Then, for each simulation step, given the current state $s \in S$ and the scheduled event list associated with it $l_s = \{(e,t)|e \in active(s)\}$

1. the first element $k = (e,t)$ is removed from l_s;
2. the simulation time t_S is updated by increasing it t;
3. the new state s' is computed as $s' = new(s,k)$;
4. the list $m_{s'}$ of the surviving events is computed, by removing from l_s all the element that become inactive, i.e., all the elements (z,x) of l_s such that $z \notin active(s')$;
5. a list $n_{s'}$ of the newly activated events is built, containing a single element (z,t) for each event z such that $z \in active(s') \backslash active(s)$ and some scheduling time $t = t_S + d_z$, where d_z is a random delay value provided by the RNG on the basis of the distribution function $\Delta(z)$;
6. the new scheduled event list $l_{s'}$ is obtained by reordering with respect to scheduling times the concatenation of $m_{s'}$ and $n_{s'}$.

The simulation algorithm can be explained with the help of an example consisting of a simple GTS with five transformation rules using only log-normal distributions with mean μ and variance σ^2. Table 2.1 shows the GT rules with mean and variance. An example run of the simulation is consist of the following steps:

Step 1. The simulation time t is set to 0.

Step 2. The set of enabled GT rule's active matches is obtained from the VIATRA2 engine.

- active(S_0)= 1 $match for GT\_Rule\_A$,
  3 $matches for GT\_Rule\_C$,
  2 $matches of GT\_Rule\_E$

Step 3. A scheduling time t_e is computed by adding t_0 to a random delay value d_e provided by the RNG on the basis of the probability distribution function $\Delta(e)$;

- (GT_Rule_A, m_1) has delay 1.35
- (GT_Rule_C, m_1) has delay 2.35
- (GT_Rule_C, m_2) has delay 3.00
- (GT_Rule_C, m_3) has delay 4.50
- (GT_Rule_E, m_1) has delay 1.75

Step 4. The timed active events are placed in a list ordered by scheduling times, we present the time line in the Fig. 2.7.

- (GT_Rule_A, m_1) is scheduled for 1.35
- (GT_Rule_E, m_1) is scheduled for 1.75
- (GT_Rule_C, m_1) is scheduled for 2.35
- (GT_Rule_C, m_2) is scheduled for 3.00
- (GT_Rule_C, m_3) is scheduled for 4.50

Step 5. The first element of the ordered event list is removed from the list. The event is despatched to the VIATRA2 GT engine for execution. The simulation time is increased to the time of the event that has been despatched.

- (GT_Rule, m_1) is despatched
- t = 1.35

Step 6. As the recently received GT rule is executed by the VIATRA2 GT engine, new GT rules will be enabled as a result, while existing enables rules will become disabled. As a result of the execution of GT_Rule_A, say GT_Rule_E is *disabled* and two new matches of GT_Rule_B are *enabled*. The single match of GT_Rule_E is removed from the active event list l_s, making it $l_{s'}$.

- $m_{s'} = 3 matches of$ GT_Rule_C
- $n_{s'} = 3 matches of$ GT_Rule_B

Step 7. The new scheduling time is assigned to active events $n_{s'}$ based on the RNG and the probability distributions associated with the GT rules.

- (GT_Rule_B, m_1) has delay 3.35
- (GT_Rule_C, m_2) has delay 1.95

Step 8. The new scheduled event list $l_{s'}$ is obtained by reordering with respect to scheduling times the concatenation of $m_{s'}$ and $n_{s'}$. The timeline is shown in Fig. 2.8.

- (GT_Rule_B, m_2) is scheduled for 1.95
- (GT_Rule_C, m_1) is scheduled for 2.35
- (GT_Rule_B, m_1) is scheduled for 3.35
- (GT_Rule_C, m_2) is scheduled for 3.00

No	Event	Mean μ	Variance σ^2
1	GT_Rule_A	1	2.5
2	GT_Rule_B	2	2.0
3	GT_Rule_C	3	4.5
4	GT_Rule_D	2	1.0
5	GT_Rule_E	4	2.0

Table 2.1: GT rule log-normal distribution

- (GT_Rule_C, m_3) is scheduled for 4.50

Step 9. The first active event is removed from the list and dispatched to the VITRA2 GT engine.

- (GT_Rule_B, m_1) is despatched

Step 10. The simulation time t is advanced by the time of the last triggered event,

- t= 1.35 + 195= 3.20

Step 11. The simulation continues this process of executing the rule match with the lowest delay and a new updated list is obtained. This process repeats itself until terminated either by the maximum number of steps or completion of the simulation time.

2.11 Summary

In this chapter, we have introduced the concepts of graphs and graph transformation with attributes, constraints, and application conditions. We have introduced

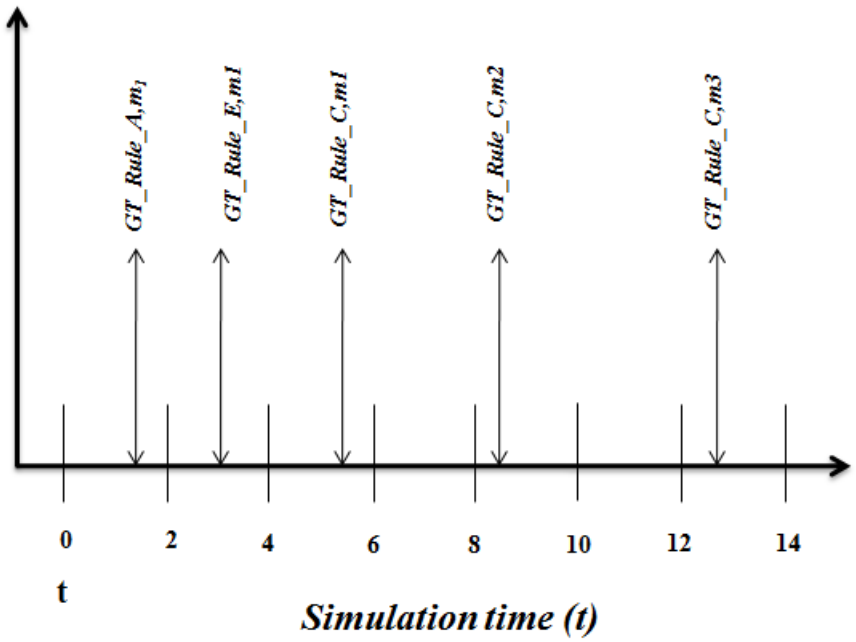

Figure 2.7: Scheduling timeline at $t = 0$

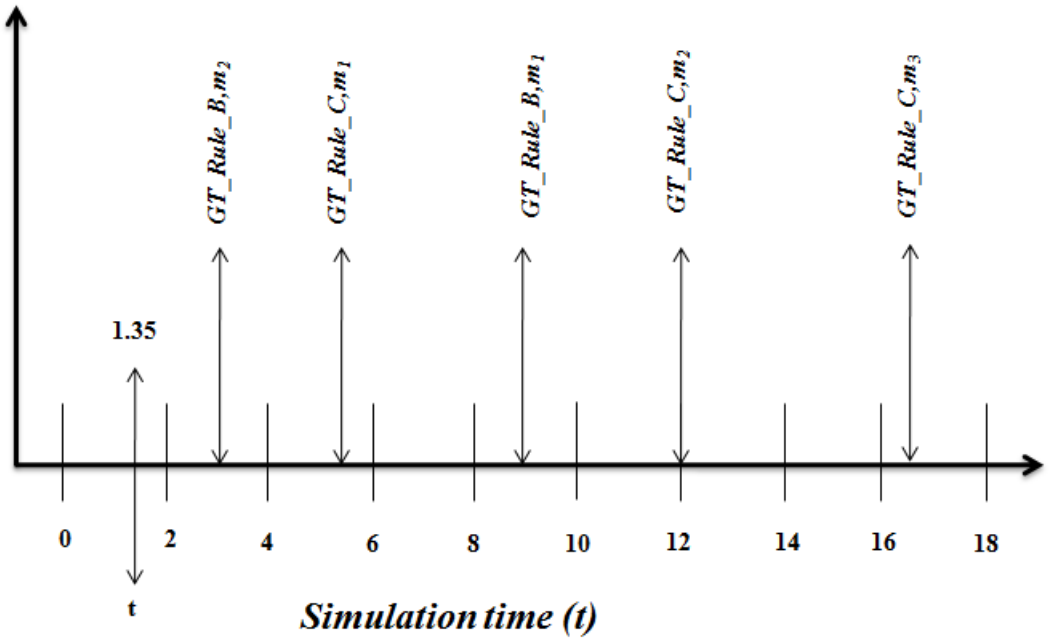

Figure 2.8: Scheduling timeline at $t = 1.35$

the concept of stochastic GTS and the procedure for selecting a probability distribution. We also introduced normal, log-normal and exponential distributions. We illustrate the simulation algorithm used through an example.

Chapter 3

P2P Networks

In this chapter, we focus on P2P systems and architectures. We first study definitions proposed by different authors. We classify these definitions based on the characteristics they consider to define P2P systems. We also agree on a single definition of P2P in this thesis. This chapter further aims to explain types of P2P architectures. In the last part we briefly discuss different P2P applications.

3.1 Peer-to-Peer Computing

Statistics show that almost a quarter of the world population has already got access to the Internet. This boosts the technology industry which directly increases the availability of new kinds of devices and technology.

It is important to note that today each user on the Internet possess CPU power more than 100 times that of an early model of super computer [107], but at the same time 95% of this power is wasted. To efficiently utilise this massive computing power the concept of P2P communication was developed. According to the Merriam Webster dictionary the word peer means, *one that is of equal standing with another* [105], that is P2P is communication with equal rights [94]. This is much like a telephone network, where caller and callee can be seen as equal partners. In a distributed setting systems setting peers are running on different machines with their own specific capabilities and resources. Different design paradigms have been developed to make use of these resources, leading to a differentiation in the basic concept of P2P networks.

P2P refers to a class of networks that make use of distributed resources to perform functions in a decentralized manner. The resources could be computing power, memory, data or network bandwidth. P2P applications are distributed systems in which various components, connected by a network, communicate and coordinate their actions by passing messages [43]. Despite several complications at the architecture level, such as peer churn [153] and selfish peer behaviour, the paradigm is becoming popular in areas such as distributed and collaborative computing, not only on the Internet but also in ad-hoc networks [107]. Distributed systems can be distinguished into two types: client-server model and P2P model.

In client-server models each client submits its request to the respective server and the server is responsible to answer the query and provide services. The core idea behind a P2P network is that a peer can act as a server as well as client in one application [14]. A peer, while acting as a server, can share resources like processing power, memory and bandwidth with other participating peers on the virtual network, called overlay. Some of the applications have central login servers, but most are lacking a central point of contact.

There exists no universally accepted and agreed definition for P2P. The term has been used to cover systems with vastly different architectures, ranging from file-sharing, distributed computing, instant messaging, voice and multimedia and content distribution. In the literature [7; 107] different taxonomy trees have been used to classify P2P systems stressing the decentralised nature, overlay network architecture and type of data being shared among peers. The existing definitions of P2P fall into two classes depending on whether they focus on the role of leaf peers. Peers which are connected to the Internet, but whether this connectivity is variable, e.g, due to shared bandwidth or temporary IPs assigned, are often called *computers at the edge of the Internet* [119]. Based on this classification we are going to discuss the existing definitions.

3.1.1 Use of Leaf Peers

1. One of the early books on P2P published in the year 2001 defines P2P computing by identifying two requirements P2P [119; 148]:

   - *Does it allow for variable connectivity and temporary network addresses?*
   - *Does it give peers at the edge of the network significant autonomy?*

2. A white paper initially published in the year 2003, which was updated in year 2006 because of many technological changes and improvements in P2P systems, states the following definition: *P2P networking is the utilization of the relatively powerful computers (personal computers) that exist at the edge of the Internet and is used for more than just client-based computing.* [[106] Paragraph 1].

3. In 2004, Andy Oram [120] defined P2P systems as: *Any networking technology where the crucial responsibility lies at the end point.*

4. In the year 2000, Clay Shirky et. al. used the following definition [145]: *P2P is a class of applications that takes advantages of resources– storage, cycles, content, human presence– available at the edge of the Internet.*

The definitions of this group concentrate on the use of computing resources at the edge of the network/Internet. These nodes have temporary IPs and are normally behind Network Address Translation (NAT) or local firewall. Therefore, these machines are inaccessible from other machines on the Internet. According to the

above group of definitions, the potential of P2P applications lies in their ability to utilize these nodes. However, it is worth mentioning that these definitions do not cover applications which perform functions such as collaboration. P2P applications also perform other tasks such as providing overlay routes, coordinating topology formation and searching. Therefore, classification of P2P based on the ability to use computers at the edge of the internet appears too limited for our purpose.

3.1.2 Server-Less Topology

In this section we discuss definitions used by other researchers in order to classify an application as P2P, in chronological order.

1. In the year 2003, Detlef Schoder and Kai Fischbach proposed the following definition for a P2P application [142; 148]: *P2P refers to a technology that enables two or more peers to collaborate spontaneously in a network of equals by using appropriate information and communication systems without the necessity for central coordination.*

   This definition identifies three important aspects of P2P systems: no need for centralized coordination, participation in a network composed of similar types of peers, and spontaneous collaboration. It is almost the same concept as that used for server-less designs. In P2P systems, each peer can communicate with other respective peers without permission of servers or other central authority. Peers are considered equal, although they may possess different computational and communication capabilities. Spontaneous collaboration is one of the key advantages and is considered as direct result

of the server-less design approach. This approach was partially adopted for hybrid systems and in this connection an interesting effort was made in the year 1999 by introducing Napster to the P2P world [115]. Hybrid systems do have a central server, which performs tasks such as authentication, resource discovery, global index monitoring, but all other tasks after successful login continue without involvement of the central server. Therefore, these systems can also be classified as P2P.

The central servers in hybrid systems do have more capabilities, functionalities and responsibilities than peers in the network, but all the other peers are equivalent in their functionalities. If we consider Napster as example of hybrid P2P systems, a Napster server is used to search a particular file and its respective address, while the actual process of file transfer is conducted in P2P fashion. Thus, P2P systems do not need to be completely decentralised. This is also evident from the modern implementation of Gnutella known as KaZaa [83], which makes use of the centralised/decentralised design approach not only to increase the efficiency of searching but to scale the network. This approach is implemented using the concept of super peers, which store file locations. Super peer selection is based on constraints such as bandwidth, CPU power, memory and time in the network. These criteria reduce the number of super peers available.

2. The Intel working group proposes the following definition for P2P systems [14]: *A type of network model, where peers (computers) share resources and services by direct exchange between networked peers.*

3. In the year 2001 at the IEEE P2P conference, Rüdiger Schollmeier presented

the following definition [143]: *A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (service and content) providers as well as resource (service and content) requesters.*

4. A technical report published by Intel in the year 2002 defines P2P as follows [34]: *P2P can be defined as any application or network solution that supports the direct exchange of resources between computers without relying entirely on a central server.*

In the above definitions, 4 and 6 seem to be using the server-less design topology as a key characteristic for P2P applications. All other definitions emphasise direct communication between peers for sharing of services and resources. The study of the literature shows that P2P got its popularity because of exchange of files among peers, however today's P2P is not limited to the exchange of resources. It does perform several other important tasks such as transmission of real time voice and video traffic [149], content distribution [27] and knowledge management [34]. Therefore, exchange of resources by direct communication should not be considered as the core classifying criterion.

To summarise the definitions of both groups, Table 3.1 shows the definition number and classification criteria used. Based on the above discussion and Table

Table 3.1: Defining classification key based on definitions

Definition	Classification Key				
	Same Peers Level	Server Less Topology	Use Edge Node	Resource Sharing	Collaboration
1			✓		
2			✓		
3			✓		
4			✓		
5	✓	✓	✓		✓
6		✓		✓	
7		✓		✓	
8	✓	✓		✓	

3.1, P2P systems are defined as follows for rest of this thesis:

P2P networks are distributed systems without central control for topology formation and communication, where most peers are equivalent in functionality while peers with extraordinary capabilities can perform additional tasks, serving a group of ordinary peers.

3.2 Overlay Networks

An overlay network is build on top of the physical network and provides additional features. Overlay networks have emerged to provide promising platforms

for customizable and reliable services to support multiple applications, such as multicast, resilient connectivity, content delivery, VoIP, distributed hash table services [133; 185] and many others (e.g Akamai [3], RON [5], Skype [149]).

Overlay networks consist of peers which may be situated in multiple domains and geographically diverse locations. These peers are connected to each other through virtual links maintained at the application layer. The main idea behind overlay networks is to give applications more independence in routing decisions. Those would normally be taken by the IP layer/physical layer [133]. With the availability of various active measuring techniques, it has become feasible for overlay networks to monitor and maintain multiple paths among peers and even select the path based on requirements such as latency, bandwidth availability, congestion and errors.

In overlay networks, in the worst case a peer can be connected to all other peers in the network, but this may affect scalability. Therefore, in most designs a peer keeps a connection with a subset of peers. One of the crucial points in P2P is the topology of peers in the overlay network. The topology can be considered as a graph in which peers are vertices and the overlay connections amongst the peers are the edges (see Figure 3.1). Most P2P overlay topologies support reconfiguration and such systems are capable of handling events like peers joining, departing and crashing, etc.

Overlay networks are used by many different applications ranging from file sharing to VoIP, each based on its own set of requirements. Overlay networks become most effective when peers are behind NAT and firewall. Some of the core tasks of the overlay include searching for peers, resources and services and transmission of keep alive messages.

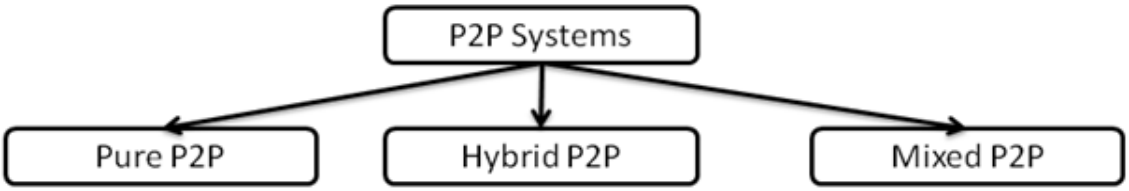

Figure 3.1: A P2P overlay Network

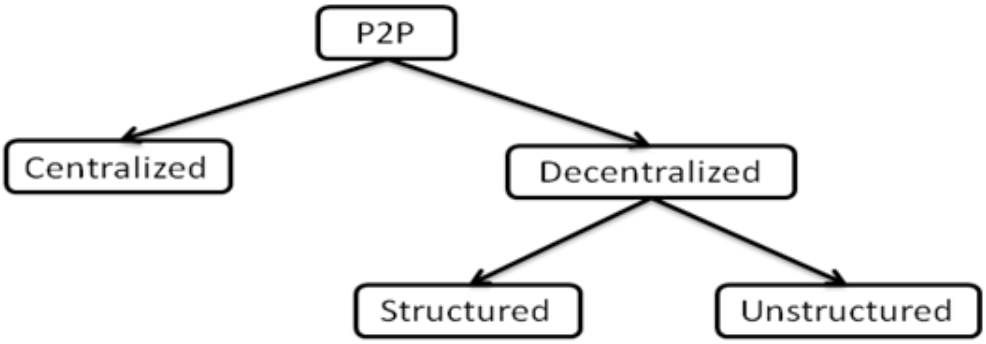

Figure 3.2: Taxonomy of P2P architectures

3.3 Types of P2P Networks

The core challenge in P2P is the organisation of peers into meaningful structure. On the basis of the organisation of peers in the overlay network, P2P networks can be divided into three broad categories (see Figure 3.2): centralized, decentralized structured, and decentralized unstructured [84; 107].

3.3.1 Centralized P2P

Early P2P systems such as Napster [115] and SETI@home [6] successfully introduced the concept of resource and data sharing through peers belonging to end users. Both Napster and SETI@home placed a central server in the network to coordinate peers. Today P2P systems that follow the approach of a central server are also know as hybrid system, because this approach combines the classic client

server model with the dynamics of P2P networks. While the central server may have more capabilities and responsibilities than other participating peers, the rest of the peers are equivalent in their functionalities [107].

In hybrid systems, when a peer joins the network it registers itself with the central server. Because of this process of registration, the server has accurate information regarding the peers and resource available. As in the case of Napster, every peer upon registration provides a list of files it is sharing. If the server receives a search query for a file, it is in a position to tell which peer is sharing that particular file.

The main drawback of the hybrid P2P approach is that the central server may become a central point of disaster. If the server fails the whole system will be down. Another potential problem is that when the number of peers increases, this has direct impact on the server [92].

Despite these drawbacks, a number of P2P applications use this approach (e.g BitTorrent [19], eDonkey [39] and distributed.net [38]). The hybrid systems seem the best choice for systems that could not afford inconsistencies but do require a moderate amount of resources for the coordination task. If the central server is capable to manage the load and chances of failure, this approach can perform efficient coordination of peers without requiring them to communicate themselves. The result of this will be low bandwidth requirement for network organisation and peer coordination.

3.3.2 Decentralized P2P

Decentralised architectures are also known as pure P2P, referring to P2P architectures where all peers participate at equal level without the umbrella of a central server. The decentralized architectures are further subdivided into unstructured and structured P2P [92; 107].

3.3.2.1 Unstructured P2P

In unstructured P2P networks, whenever a new peer joins the network it has to first find a peer on the network and after successful handshake it broadcasts a join message to the neighbours of that existing peer. In this way peers are connected randomly to each other. In this kind of topology no peer has a special location, while keeping small set of neighbours for the purpose of searching and relaying messages.

With a random network and absence of central server, in this class of P2P it is almost impossible to perform tasks that require an exact global view of the network. In unstructured P2P, if a peer wants to send a query, the query will be first forwarded to its local neighbours and the neighbours will forward further to their neighbours. To overcome flooding in the network, the message is enriched with a time-to-live (TTL) attribute which is decremented as the message proceeds.

The main drawback of this approach is that it does not scale when the network grows large. The query will only reach a small set of peers because of the TTL, possibly missing some of the peers which could have the response. This will have worst results if the search request is initiated for a unique resource. The resource

could be available further away in the network where the message could not reach because of the TTL. Thus, we may receive an answer of non availability although the resource may still be available.

3.3.2.2 Super Peer Hierarchy in Unstructured P2P

To solve the problem of searching in unstructured P2P, two options have been considered. The first option suggests increasing the size of the broadcast horizon and second suggests reducing searchable network size.

The first suggestion was implemented in Gnutella clients. The main problem with this approach is if the TTL is increased the number of message recipients is increased exponentially [92]. The implementation results showed that search horizon increase has created a lot of problems in Gnutella network and resulted in arguments, whether this could be considered the downfall of the network. The ultimate result was that this approach does not scale well [94].

The second approach proved to be much more successful and was implemented by KaZaA [83], Grokster [51] and many more. With the concept of super peer, super node or ultra-peer the network proved to scale very well. The core idea is to divide the network into two layers. The first layer consists of a number of ordinary peers or clients and the second layer consists of a small number of super peers. A peer which has sufficiently large resources, especially bandwidth, is promoted to the role of super peer [92]. Promotion can be based on a number of constraints as required differently by different applications. Some applications may promote based on need whereas others may promote as many peers as possible. The leaf node or ordinary peer will connect to the super peer. Ordinary peers will not form any kind of network among themselves. The super peers act as servers and

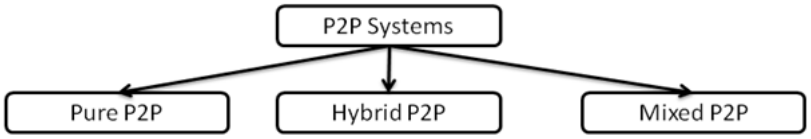

Figure 3.3: Types of P2P systems

keep records of the files that their clients are sharing and forward their queries to other super peers to find a particular resource. The super peers also reduce the network overhead by routing messages to dependent peers only which are identified to be the resource owner. However, when a peer is identified to have a particular resource the rest of the communication is direct among the dependent peers.

The super peer approach is still considered the dominating technology in the P2P world because it improves the search and also removes the single point of failure from the centralised P2P model. The dependent peer can reconfigure and connect to a new super peer if the current super peer has crashed or a user has decided to leave the network. Some of the applications have even combined the super peer unstructured P2P approach with the centralised model and the result is known as mixed P2P [6] (see Figure 3.3 for types of P2P systems). In this approach the central server is used for authentication and the rest of the network is maintained in a decentralised manner. Upon successful authentication, the peer has to select one of the super peers as their local host. Several super peer selection strategies will be discussed in the coming sections. In this thesis we focus on mixed P2P networks.

3.4 Popular P2P Applications

New P2P applications are emerging regularly, often to disappear again after a short period of time. Some P2P applications are discussed below.

1. **Gnutella** [48] is a file sharing protocol. The early version of Gnutella was based on the decentralized approach and queries were transmitted using viral propagation [127].

2. **Napster** [115] is a file sharing protocol based on the centralised approach. The search is through a central server but file transfer is direct between participating peers.

3. **BONIC** [21] is a CPU service sharing system inspired by the SETI@home project. It operates in centralised manner. Each client logs into the server downloads the data to be processed and submits the results back to the server.

4. **Freenet** [25] is a distributed content storage and retrieval application based on the structured P2P approach. The prime goal is to ensure the anonymity of the publisher.

5. **KaZaa** [83] is a file sharing application that uses the concept of super peers to organise peers into two layers. The super peer layer is used for the purpose of searching while file transfer is direct between peers.

6. **Jxta** [29] is a framework for developing P2P applications. It specifies a number of protocols that enable the creation as well as maintenance of P2P networks.

7. **IM** [103] is a messaging system where a user joins the network using a registered identity and can search as well as communicate with other identities. These systems normally have a central server which is the authentication point and also maps IP addresses with identities registered.

8. **Skype** [149] is a P2P Voice over IP (VoIP) application that controls the login process in a centralised fashion and uses the super peer approach for searching as well as to redirect conversation for peers behind firewall or NAT. These systems are the core focus of this thesis and will be explained in more detail the coming chapters.

This list of application software shows the importance of P2P applications and their topology of operation. Further, from the list above we see that each application has its own set of requirements that is essential for ensuring service quality. For example in KaZaA the focus is searching for a file and then to download it. Skype is concentrating on searching as well as transferring real time conversation packets. This set of requirements also affects the topology of the application.

3.5 Summary

This chapter started with considering P2P networks from many different angles based on the definitions proposed in existing literature. At the end of the section we agreed on a single definition of P2P for this thesis, which states that P2P networks are distributed systems without central control for topology formation and communication.

P2P networks use overlay networks for their topology formation. Overlay networks are virtual networks built on top of the physical networks. They concentrate on the application layer and enable peers to communicate directly with each other without using publically advertised routes at the physical layer. Skype is considered as one of the popular applications used for VoIP.

There are three basic types of P2P architectures which are known as centralized, unstructured and structured. P2P systems are classified into pure P2P, hybrid P2P and mixed P2P. In mixed P2P we have a central server which is used as authentication point, but the rest of the network is organised in an unstructured way using the super peer approach. This kind of P2P is the focus of this thesis.

Chapter 4

Reconfiguration of P2P VoIP Networks

In this chapter, we focus on P2P protocols and their application in the VoIP domain. Based on existing design alternatives, first we sketch the general structure of a P2P VoIP protocol and then follow the design variations by means of a feature tree. The feature tree represents the core mandatory and optional features of P2P VoIP protocols, and their associated variations selected from the literature.

This chapter also aims at introducing various reconfiguration and communication mechanisms used in P2P VoIP systems. Further, we explain how peer dynamism and selfishness affect the quality of the VoIP applications

4.1 Introduction to P2P VoIP

VoIP can be defined as set of technologies that enable voice calls to be transported over the internet or any other network designed for data transportation, rather than the classical Public Switched Telephone Network (PSTN).

The term VoIP was used by the VoIP Internet forum in the year 1966. The main driving force behind the introduction was to cut the existing cost of telephone calls. The classical telephone system works on circuit switching approach, where a dedicated circuit or channel is setup between the *caller* and *callee* before the conversation is started. The dedicated circuit approach seems the same, as was used in manual switching based telephone exchanges, where an operator has to connect the two callers. The biggest advantage of a dedicated circuit is the excellent quality of service, but at the same time it is very expensive.

The introduction of VoIP posed many challenges for the telecommunication industry. VoIP applications use existing IP protocols originally designed for data transportation. VoIP breaks voice calls into packets. These packets travel towards the callee through public routes on the internet, where they are assembled and decoded. Voice packets are much more sensitive to delay, packet loss and jitter as compared to than data packets. Researchers realised that instead of using publically advertised routes, overlay networks could minimize the delay. This coined the idea of P2P VoIP, where caller and callee communicate in directly without using intermeddles.

Today almost 30% of the traffic over the internet is VoIP and this figure is increasing day by day. The reason behind this is low cost. PSTN calls are charged based on the time used, while in VoIP the user pays only for the Internet service.

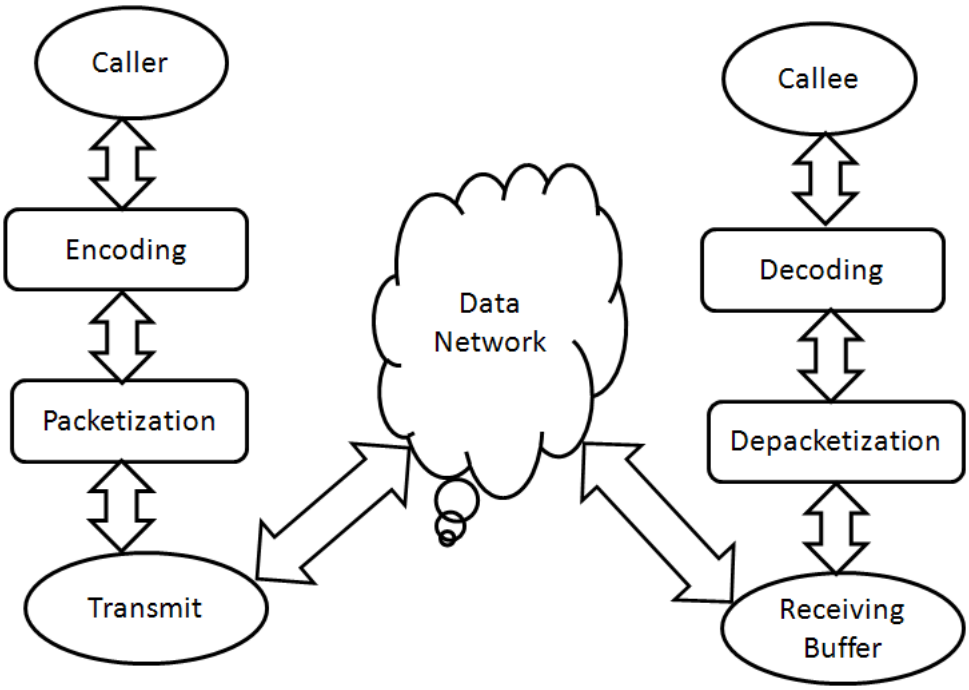

Figure 4.1: VoIP Architecture

The facility of conferencing and group meetings on VoIP are also among the core attractions.

Figure 4.1 illustrates a basic VoIP architecture, although different applications may use different topologies for organising clients in the network. A typical call scenario will involve the following steps at the callers end.

1. Encoding of the caller's voice into digital format.

2. The digitally encoded signals are compressed and divided into packets of equal size. These packets are time-stamped and provided with information regarding destination and origin. Time stamping will enable the callee to assemble these packets in correct order.

3. These packets are transmitted over the internet travelling to their destina-

tion. Different packets may follow different routes in the network. This transmission of packets over different routes makes VoIP different from circuit switch call operation where a dedicated circuit for each call is established and maintained for duration of the call.

At the callee end the following steps are followed:

1. The callee side receives these time-stamped packets and uses a buffer to organise them in order of transmission time. These packets are depacketized for conversion to analog form.

2. The depacketized streams are converted to analog form by a decoder. After this conversion the analog signal is transmitted to human ear.

It is clear from the above discussion that VoIP uses digital packets for transmission of voice over the data networks, while telephone network uses the analog signals. The data network can be an intranet maintained within the organisation, a leased network, PSTN or the Internet [50]. The process of compression and encoding of analog signals into digital packets is achieved through the use of codecs. There are many codecs available for digitizing speech. Table 4.1 gives some of the characteristics of a few commonly used International Telecommunication Union-Telecommunication (ITU-T) standard codecs [137]. The first column shows the name of the codec, the second shows the data rate used by the codec for packet transmission, the third column shows the time taken by codec to convert voice into digital and back to voice, and the last column shows the bandwidth cost of the calls. The bandwidth cost here is for a full duplex channel. In this thesis we will use the information in Table 4.1 to compute the cost of bandwidth for VoIP call. Based on the information we have fixed the cost of VoIP calls at 80 Kbps.

Table 4.1: ITU-T codecs and defaults values

Codec	Data Rate *(kbps)*	Datagram Size*(ms)*	A/D Conversion Delay *(ms)*	Bandwidth Bidirectional *(kbps)*
G.711u	64.0	20	1.0	180.80
G.711a	64.0	20	1.0	180.80
G.729(	8.0	20	25.0	68.80
G.723.1(MPMLQ)	6.3	30	67.5	47.80
G.723.1(ACELP)	5.3	30	67.5	45.80

4.2 Quality Issues in VoIP Implementation

The classical PSTN based telephony service is more than hundred years old and during these years it has passed through various technological developments. The ultimate result of these developments is the 99.999% guaranteed quality of service [24]. VoIP is relatively new technology implemented over a less reliable data network [152]. Therefore, there are justifiable concerns such as voice quality, reliability of service, access to emergency numbers like 999, directories and addressing. An important point in VoIP traffic is the real time nature of communication. When voice packets are transmitted, any problem such as congestion, link down or relay node departure will immediately affect the quality and reliability. All these factors will result in the callee experiencing poor voice quality by missing part of the conversation, experiencing echo on the line. According to [24; 126] the main concerns of voice service over IP networks are jitter, packet loss and echo. If these stated parameters are controlled in an efficient way, VoIP can provide a quality almost equivalent to PSTN.

4.2.1 Latency

VoIP is the transportation of real-time conversation over a network and therefore sensitive to communication delays. Latency has a direct effect on voice quality. Latency is defined as *the time taken by packet to travel from caller to callee*. It is composed of two kinds of a delays: propagation delay or actual transportation delay (time taken by packets to travel in the network from caller to callee) and preparation delay (such as encoding and packetization) [24]. This is also known as mouth-to-ear (M2E) delay [75]. The lower the latency, the more natural is the conversation [128].

Increased latency results from a number of factors such as congestion in the network, low bandwidth, large packet size, the number of hops that packets have to pass through [165], or loss of connection. If the codec G.714 is used, the total maximum one way delay accepted is 150 ms [69] and in [71] a delay of 200 ms was also considered acceptable. If the delay experienced is higher, the callee may experience breaks in the conversation. In this thesis we will use this information to link client peer with super peer. This will help to reduce latency and improve call quality.

4.2.2 Jitter

Jitter results from the variable arrival time of packets. In order to clearly understand the conversation, the packets must arrive at the callee end with equal intervals. Jitter results from congestion on the route a particular packet follows. Normally, if variation between consecutive packets is not too large, it is compensated by the jitter buffer at the callee end. The literature [183] shows that less

than 40 ms is ideal but 75 ms variation can be tolerated.

4.2.3 Packet Loss

Packets in the network can be lost either during transmission, due to departure of intermediate peers, or failure of connection. In VoIP if a peer behaves selfishly and stops routing traffic for others, leaves the network selfishly or crashes, the result will be packet loss. Packet loss is a very frequent phenomenon in data networks, but many applications are designed to provide a reliable service using network protocols such as TCP [15; 28] to request the retransmission of the lost packets. But most VoIP application use UDP [163] for the transmission of voice packets, which discards the missing packets. This results in gaps in the conversation [49]. Most VoIP applications are designed to handle packet loss up to 3% [24; 124].

4.3 VoIP Classification

In order to see how VoIP is used in various communication scenarios, we classify VoIP on the basis of network architecture and ownership [60].

4.3.1 VoIP as Backbone

This class of VoIP is dedicated to connecting to switching stations. Mostly this class makes use of circuit switching techniques to connect two terminal stations while the core network may be operated by the packet switching technique. The biggest example of this approach is the interconnection of local exchanges or call centre operation. This model is considered to be vertically integrated, i. e., all services are controlled by a single telecom company.

4.3.2 Dedicated VoIP Solutions

In this class of VoIP, a user who has internet access also purchases a dedicated VoIP service, either from the same Internet Service Provider (ISP) or a different VoIP service provider. The ISP will be responsible for carrying the voice signalling using TCP/IP [28] and audio over UDP/IP [163] UDP. This model is not vertically integrated as the user can have different companies for internet service and VoIP operation.

Dedicated VoIP solutions like Vonage, provide monthly subscription and their services could be used as VoIP phone. This enables the user not only to call PC to PC but can also make calls like PC to PSTN. These companies offer soft phone or SIP based hardware which are directly connected to the Internet. These types of applications are stable as they are using paid server and intermediate gateways.

4.3.3 P2P VoIP

In this class of VoIP, an Internet user simply downloads the P2P enabled VoIP application. After successful registration they get access to the service. Different P2P VoIP application use different topology formation approaches that could be either centralised, hybrid, or mixed.

In these applications centralised servers are used for authentication purposes while the rest of the operations are performed in decentralised fashion. After completion of the registration process the P2P VoIP clients are connected to each other based on the topology implemented by P2P VoIP application. For example in Skype [149], after registration the client is connected to the super peer. If the connection is lost due to departure or crashing, the client has to

reconfigure itself and connect to a new super peer.

In P2P VoIP applications, if a client wishes to make a call to a callee, first the on-line status is searched through a distributed directory service maintained by the application either through a server or super peers. Then the call request is initiated and the application software performs the task of managing signalling and audio coding. The virtual circuit/path is established between the clients and the ISP carries the signalling over TCP/IP and audio over UDP.

This class is not vertically integrated as the P2P VoIP provides only the application and directory service. As the network is maintained through cooperation of peers, P2P VoIP services are usually free. But due to the volatile nature of P2P and in the presence of a high degree of churn, this has never been the primary service of choice. However, research and development is continuing to improve the quality of P2P by using different topology formation approaches in the presence of churn. P2P VoIP is also unique in that the other two classes impose restrictions on the clients to be connected using permanent IPs whereas P2P VoIP applications do support temporary IPs obtained by NAT or firewall. The focus of this thesis is this class of VoIP.

4.4 P2P VoIP Network Design

4.4.1 Architecture of P2P VoIP Networks

The most widely-used client-based P2P technologies include Napster and Skype [53]. As discussed earlier, P2P systems operates in a decentralised environment, which has very little reliance on a central server. Almost all tasks are performed

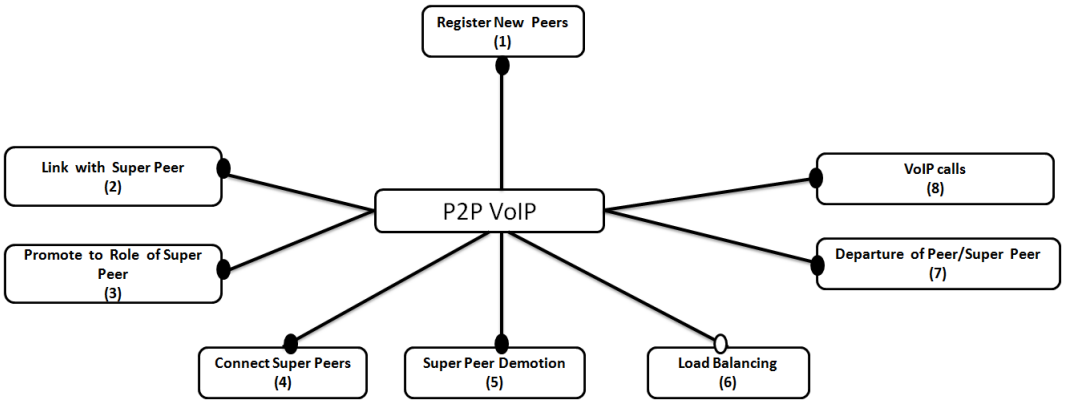

Figure 4.2: P2P VoIP features tree

in a decentralised fashion. Clients act as nodes, sharing as well as consuming resources. Fig. 4.2 represents P2P VoIP architectures by a feature tree [81]. A feature connected by a black dot is mandatory while the unfilled circle show that a feature is optional.

The first feature in Fig. 4.2 models the peer registering process with a central server as shown in the simple scenario in Fig. 4.3. After successful registration it has to select one of the super peers as local host (2nd feature in Fig. 4.2). Selection of super peers depends a on number of factors, such as bandwidth, latency or locality. If a peer has more computing capabilities, it may be promoted to the extended role of super peer while retaining the primary function of being a client (3rd feature in in Fig. 4.2).

In order to minimize the search horizon, the super peers form an overlay network among themselves (4th feature in in Fig. 4.2). The function of being host is maintained at the cost of the super peer's own bandwidth. If the super peer's bandwidth falls to critical conditions due to the existing load, the super peer is demoted to client only (5th feature in Fig. 4.2). In addition, a super peer opts to restore itself to normal condition by transferring some of its load to other

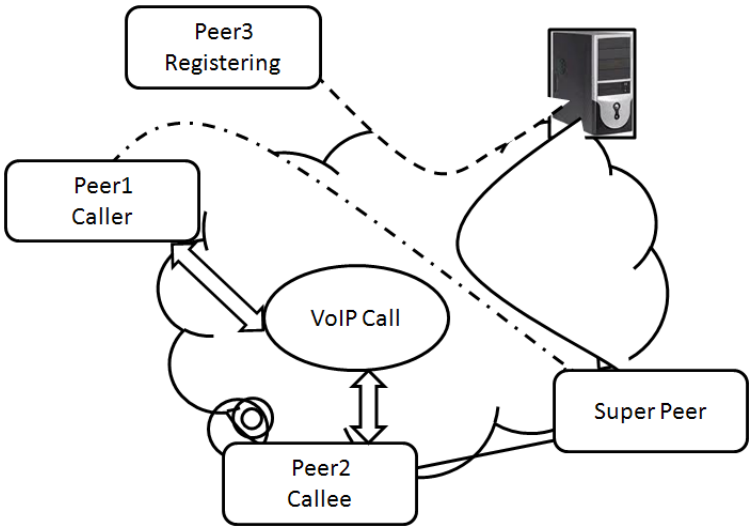

Figure 4.3: Simple P2P VoIP scenario

hosts, either by selfish disconnection or cooperative transfer (6th feature in in Fig. 4.2).

P2P network are known to have a high degree of peer dynamism, i.e, peers as well as super peers can leave the network without any constraints (7th feature in in Fig. 4.2). This is the case because peers and super peers are controlled by the user, who may decide to disconnect from the network, leaving it to the other peers to recover from the loss of connectivity.

Peers search their online peers through the super peer overlay. If callee peer and caller peer are connected through static IPs [36], after successful search the rest of the VoIP call transmission is direct between the caller and callee as in Fig. 4.3. However, it is not the case that only peers with static IPs can make calls. Peers who are behind a firewall and NAT can make calls to other peers through their host super peer [15]. At the same time super peers are also clients, so they can also make calls (6th feature in in Fig. 4.2).

There are a number proposals for each feature shown in Fig. 4.2. These

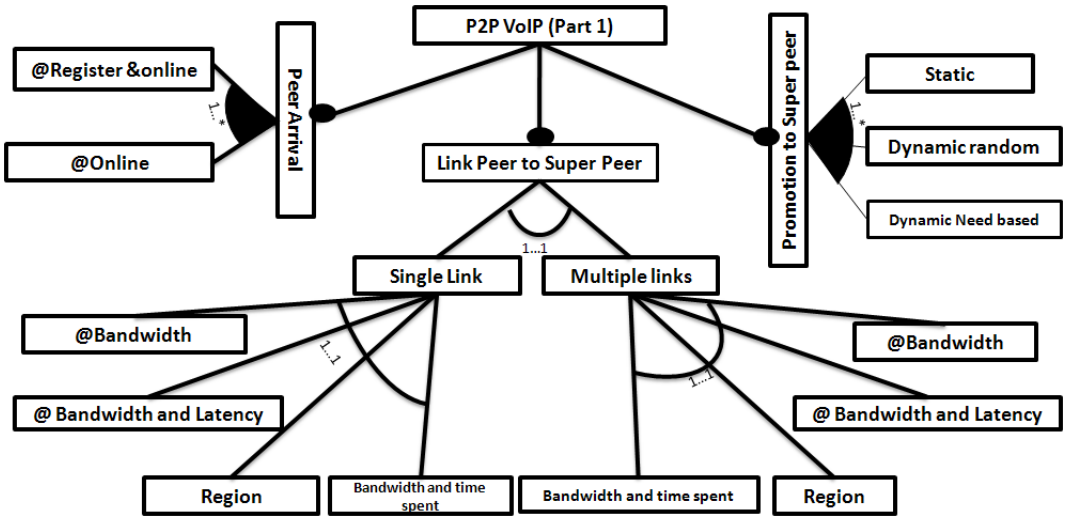

Figure 4.4: P2P VoIP protocol as feature tree*(first part)*

variations are modelled using three feature diagrams. The design alternatives presented in the feature diagram in Fig. 4.4, 4.5, 4.6 are based on the literature [15; 30; 34; 53; 99; 111; 118; 121; 153; 161; 166; 168; 176; 187]. In this section we are going to discuss these variations.

Peer Arrival In Fig. 4.4, we have two variations for peers going online in the network. The first variation is based on the assumption that a user first gets registered. After successful registration, the user can go online in the network. The second variation states that an already registered peer can go online in the network. This option does not let new peers get register and restricts the number of peers in the model. As in the feature diagram these two options are connected by the OR operator, the model permits to use either a single feature in one time or both together.

Number of Links between Peer and Super Peer In Fig. 4.4, the second feature is the process of linking peers to the super peers. The first decision is

the number of connections that a peer can make. [53; 111; 121; 176] state that a client should keep a single connection to a super peer; [176] suggests that multiple connection with different super peers will help in avoiding a single point of failure. If a super peer leaves the network, all the dependent peers will be disconnected until they relink to another super peer, so if a peer is connected to multiple super peers, departure of a single super peer will not completely disconnect a peer. [118] proposes that a peer must keep multiple connections with super peers. The peer must periodically check the status of these connections, so as to have updates if the super peer leaves the network. This will help the peer to establish a new connection to other super peers, so that multiplicity is maintained.

Links Peer and Super Peer In Fig. 4.4 the next decision is the choice of super peers among the population, either to maintain single or multiple connections [15]. Super peer selection is demanding because of the P2P architecture, a large number of super peers and highly dynamic network in which neither peer characteristics nor topology are known in advance. There are many solutions to cope with this situation. [99; 121] suggests that the bandwidth of the super peer shall be used as choice for making peer to super peer connection; [168] suggest that bandwidth, round trip time (RTT) and similarity of interest shall be used. [180] propose delay-aware P2P systems (DAPP), based on the concept that the shortest path in the overlay cannot ensure efficiency in the actual physical media. DAAP suggests sending ping-type time-stamped messages to get an idea of the actual latency. According to [9; 99], peers from the same physical network always give low latency as compared to remote or overseas peers with different geographic location as well as different physical network. The latency is observed to increase

the further we go in the network interms of network hops. Their approach is based on the notion that a local peer will be the minimum number of hops away.

In the feature tree at Fig. 4.4, based on the above discussion, we have shown four variations connecting peer to a super peer. All four variations are modelled using the alternative operator. The alternative operator permits the use of a single option at time. According to this, a protocol may choose to link a peer to super peer based on *the latency and available bandwidth*, or only use the *bandwidth* of the super peer, or only attempt to connect to *local super peer*, or select a super peer based on its *bandwidth and time spent* in the network.

Promotion to Super Peer The third mandatory feature in Fig. 4.4 is the promotion of peers to the role of super peer. The important decisions at this point are: When a peer be promoted to super peer? Which peer shall be selected among the available peers? How many super peers a network have?

[35; 111; 175] suggest that any peer that has fast reliable internet, more memory and CPU power shall be immediately promoted to the role of super peer. Being super peer is not the choice of the peer owner, so a network should have as many super peers as possible. [99] suggests that a fixed ratio must be maintained between super peers and peers. The concept is based on the idea that super peer should be arranged only when required. [99] suggests an algorithm which states that promotion to super peer shall be made on the basis of bandwidth and time spent in the network. [168] suggests promotion based on a parameter referred to as capacity. They are modelled using alternative (XOR) in the feature diagram at Fig. 4.4 [85]. We call the first choice static protocol. In this protocol, a peer promotion decision is made on the basis of bandwidth as soon as the peer goes

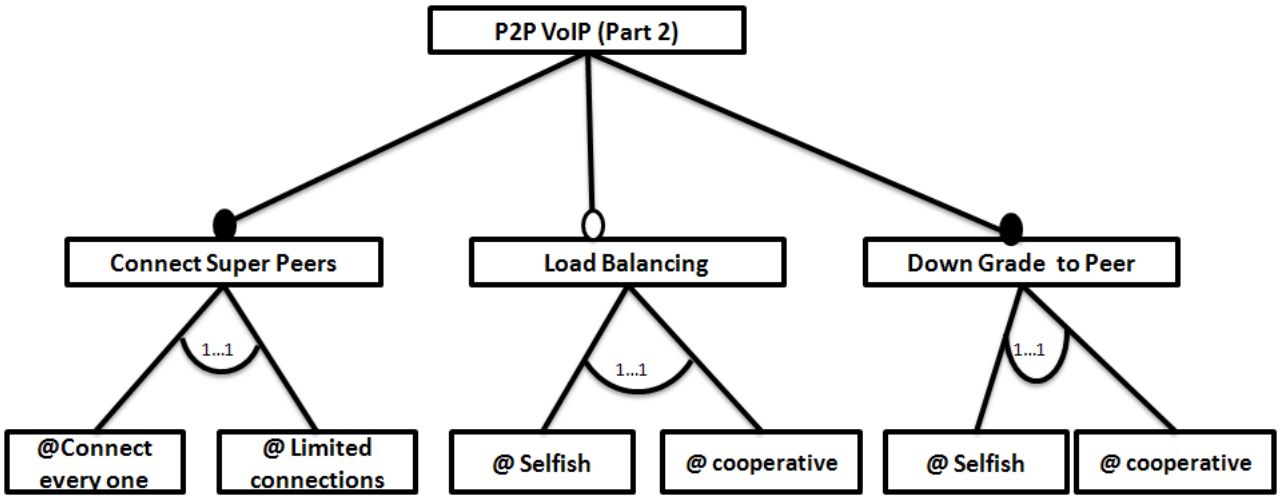

Figure 4.5: P2P VoIP protocol as feature tree*(second part)*

online in the network. This option excludes all those peers from promotion who at the time of joining fail to meet the requirements. We call the second variation the dynamic random any-time protocol. This protocol states that peers can be promoted either as they peer become online or later, as soon as it meets the requirements of the role. We call the third variation the dynamic need-based protocol. This protocol is based on the concept that being a super peer is costly, so promotion should be strictly based on the requirements. Another justification for this protocol is that too many super peers will broaden the search horizon and search will results in too much traffic among super peers. The cost of bandwidth is a key issue in super peer-based architecture. Due to this many, organisation have banned super peer-based P2P applications from their network.

Super Peer Overlay Network In Fig 4.5 we have shown the other three mandatory features and their design variations. The core concept behind the approach of the super peer-based protocols is to improve search and reduce un-

necessary traffic. In order to connect super peers among themselves to form overlay network, [61] suggests that they should have as many connections as possible, as this reduces the chance of being disconnected in the presence of churn. [15; 53] suggest that Skype super peers are connected to a fixed number of other super peers, because these connections are maintained at the cost of bandwidth [97]. Based on these ideas, we have two alternative features, first to connect every super peer with all other super peers, and second to connect super peers with a limited number of super peers only.

Load Balancing The optional feature in Fig. 4.5 is load balancing. Some of the super peer-based approaches [187] restrict the load per super peer by putting a strict limit on the number of peers they can support. This helps the network to share the load evenly. [30] proposes to use an approach based on moving peers from overloaded to free super peers. [187] explains how selfish or self-interested peers act in the P2P overlay and how load can be balanced in the presence of selfish nodes. Similarly, a self-interested super peer will simply disconnect the dependent peers, so as to make itself stable without considering that other peers will lose connection to the network. We have two alternatives here. First if a super peer is under stress due to dependent peers, it may selfishly disconnect randomly a dependent clients. The second solution is based on the approach to disconnect them through a cooperative procedure by informing them through message to find alternative. When they find an other super peer they get disconnected.

Demotion to Peer The last core feature in the feature diagram in Fig. 4.5 is the dynamic demotion to the role of client only. If a super peer's current band-

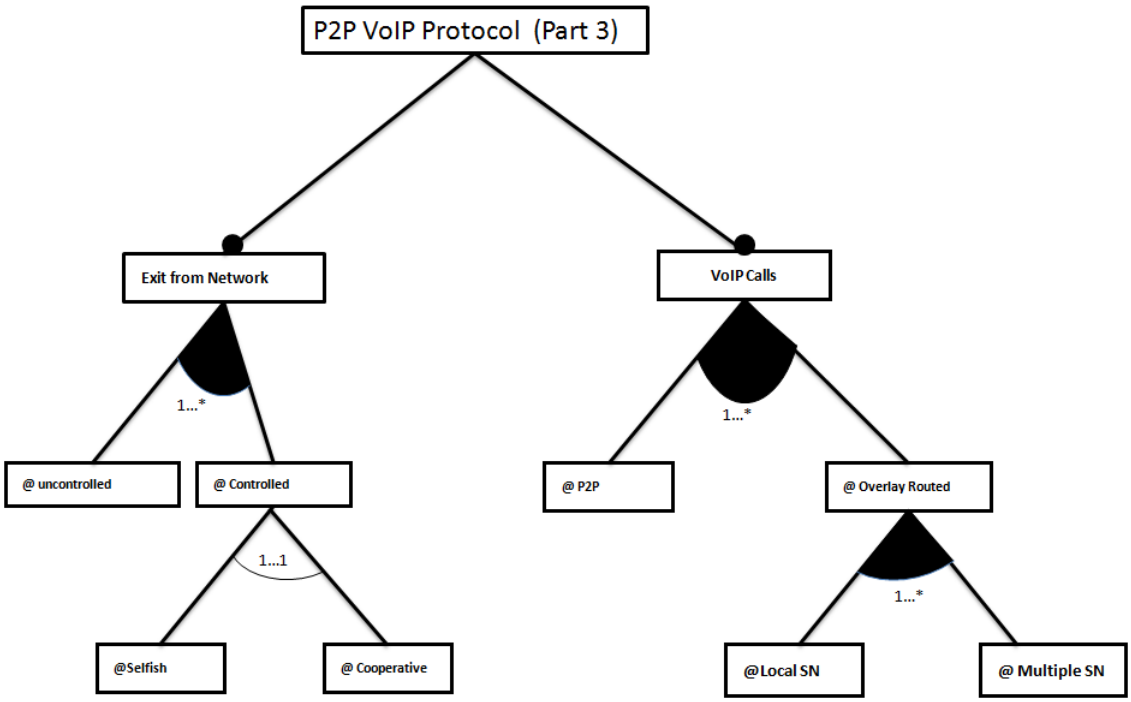

Figure 4.6: P2P VoIP protocol as feature tree*(third part)*

width reaches the critical condition, the super peer will be demoted to the role of client only. To handle this situation, we have two alternative approaches. A protocol may selfishly [170] disconnect all dependent peers and connect itself to a super peer. The second solution is performing this demotion by a cooperative procedure [99], which includes informing all dependent peers and when all dependants have safely transferred to other super peers, get demoted and find a host for itself.

Departure/Exit from Network In Fig. 4.6 we have shown the last two mandatory features of the P2P VoIP protocol. The most prominent is departure from the network. That could be either due to selfish behaviour or by using the proper exit option provided in the application. [111] states that sudden crash is

very hard to handle, as this behaviour makes peers and super peers disappear from the network without informing any other peers or super peers. Following [9] peers connected through static IPs are more stable, but those at the edge of the Internet are exposed to frequent in and out events. P2P VoIP must support both departure mechanisms. In uncontrolled exit, the network does not have any alternative, but to recover from drastic failure. In the case of cooperative exit, there are two design alternatives. The first is based on the concept to disconnect and exit from the network. The second solution is to cooperatively inform all the peers and super peers, and then exit from network.

VoIP Calls In Fig. 4.6 the last mandatory feature is the operation of VoIP calls. A P2P VoIP [15; 53; 134] supports direct P2P calls and super peer-routed calls by traversing the NAT and firewall. If calls are routed through a super peer, they can be operated by a single super peer in one domain or by involving multiple super peers in other domains.

4.4.2 Reconfiguration in P2P VoIP Networks

The dynamics of peers as well as super peers are called churn [153]. This is considered as one of the prime challenge in the design evaluation of P2P systems. In P2P VoIP, when a user starts the application, a peer joins the network after passing through the process of authentication. Based on the computing resources in its possession, it may be immediately promoted to the role of super peer and will continue to offer its resources to other peers or it may simply become dependent on other super peers. Peers may make calls or may support other peers' calls and then eventually leave the system. This process of join-participate-leave is referred

to as a cycle or session. As P2P networks are composed of thousands or millions of peers, each following this cycle, this creates significant churn.

One of the essentials of modern dynamic P2P systems is the reconfiguration of peers resulted from the churn. Since each peer is independent and autonomous [74], peers can take actions that do not require any global view of the topology. Based on this characteristic, peers in different applications handle reconfiguration at different level of priority depending on the type of service they are offering. In VoIP applications which are time sensitive, in the event of crashing or departure, the peer or super peer needs to reconfigure itself fast enough so that QoS is not affected. Peers as well as super peers are exposed to the following reconfiguration capabilities and restrictions:

- A peer can join the network at any time and can leave the network by crashing, cooperative, or selfish exits.
- Peers can be promoted to super peer while retaining their primary role, if they meet the higher resource requirements of the extended role. This dynamic promotion is not a peer choice and peers cannot stop themselves from acting as super peer.
- A super peer that no longer fulfils the minimum requirements of the extended role will be demoted and peer only status will be assigned.
- If a dependent peer loses connection to the host super peer (due to network loss, congestion or selfish exit, etc), the peer must reconfigure itself to connect to a new super peer as soon as possible.
- If a super peer loses its current connection with other super peers (due to

super peer departure or demotion), the super peer must connect to another super peer to recover its overlay formation.

- When a super peer is switched to peer-only, the peer must reconfigure fast enough to find a host for itself.

4.4.3 Communication in P2P VoIP

Communication between peers and super peers is as important as dynamic reconfiguration to restore the network connection. In P2P VoIP, if a peer wants to get an idea of the physical latency resulting from a possible connection before it is established, it has to communicate with the proposed super peer. In the P2P VoIP model which we are going to propose the model shall capture the following assumptions of communication.

- Full duplex or two-way communication is supported by a single connection, either between peer and super peer or super peer to super peer.
- Peers as well as super peers can send voice packets and control packets for finding latencies or connection requests. Both peers at the edge of the network and peers with static IP can make calls.
- Peers as well as super peers can receive voice packets and control packets.
- Super peers can answer a connection request either by saying yes or no.
- Super peers can forward voice packets to peers behind NAT or firewall.

4.5 Summary

The idea of P2P VoIP was coined when overlay routes were considered as resulting in low latency as compared to publically advertised routes. Today almost 30% of traffic over the Internet is VoIP traffic. The reason behind this massive growth is the low cost of calls as compared to the PSTN.

As the Internet is initially designed for transportation of digital data packets, the sound waves are converted to digital packets and they are reassembled and converted back to analogue sound waves for hearing purposes. The transportation of these individual packets via overlay routes result in QoS issues in P2P VoIP, such as increased latency, jitter, packet loss and echo.

In this chapter, we also studied different architectural designs for P2P VoIP protocols. We use feature trees to show different designs alternatives. The aim is to enable the reader to get an idea how many different approaches exist to perform a particular function. In the last part we studied different architectural reconfiguration and communication mechanism.

Chapter 5

Case Study

In this chapter, we are going to present a case study based on the popular VoIP application Skype. Skype is P2P VoIP application developed in 2003 by the owners of KaZaA. Skype claims that it has more than 56 million users. Real statistics shows that on average 20 million users are online. It currently contributes more than 4.4% to the total VoIP traffic over the Internet. Skype keeps its directory structure decentralized. This decentralized architecture enables it to scale its network to millions of clients without requiring complex dedicated infrastructure.

In this chapter, we also present real network statistics obtained through tracing of the Skype network. In addition, we present data collected by researchers by monitoring Skype traffic. These will later be used to validate the model.

5.1 Skype Background and Definitions

The first beta version of the Skype P2P VoIP application was distributed in 2003 [101]. By its first anniversary, in August 2004, the cooperation reported a remarkable growth in the number of downloads and registered users. During that year, the VoIP software was downloaded more than *20 million* times and the number of registered users were above 9 million [131]. The founders of Skype had previously contributed KaZaA [83], a P2P application for downloading and sharing music files [100]. The idea of using P2P topology for VoIP enables the company to maintain the network with low costs remarkable scalability, and provide an easy registration process.

Today Skype claims that it has more than 56 million users. Real statistics shows on average 20 million users are remain online [101]. It currently contributes more than 4.4% to the total VoIP traffic over the Internet. Skype permits registered users to make free PC to PC voice and video calls, send real-time text messages and to make calls to the Public Switched Telephone Network numbers by paying small charges. Skype operates a decentralized P2P network, in contrast to controlling all communication through a central server. Skype is unique, as it has the ability to traverse NAT (Network Address Translation) and firewalls, while encrypting user's conversations and storing user information in a decentralized way [131]. This enables it to scale to a large number of clients without requiring complex dedicated infrastructure [96].

The number of registered users indicate the popularity of the Skype, but due to its proprietary network and encryption mechanisms used, very little is known about its architecture and operation. [46] states that Skype is related to KaZaA,

network traffic analysis confirms that KaZaA and Skype use the same procedure for network connection. The Skype topology formation is achieved through a two layer P2P network using super peers and ordinary peers.

Most of the network administrators consider Skype as a security and bandwidth consumption threat [40; 90]. Other network experts are of the opinion that it should never be allowed on a well organised network. The reason is that Skype uses strong encryption, which makes it a black box whose architecture is unknown. Skype software is available on many platforms ranging from Ipads to mobile phones, and operating systems such as Windows, Mac OS and Linux.

Before we discuss the operation and architecture of the Skype network protocol we need to understand its basic entities and concepts, like Skype Client (SC), Super Peer (SN), Ports, Codecs, Encryption, NAT, Firewall and Skype Registration server (RS).

1. **Skype Client (SC)**

   The Skype Client executes the Skype VoIP application [59; 131]. Further, a client has consumable functionality. It means that it is using the network but does not share the load with other peers or clients. In Skype, all those users are considered as client which do not meet the requirements of the extended role of super peer. An SC keeps a permanent connection with a super peer for searching as well as VoIP calls. The SC also uses it for transferring call packets if the SC is behind NAT or firewall.

2. **Super Peer (SN)**

   A super peer is a Skype client that has a public IP address and sufficient CPU, memory and network bandwidth [15; 46; 59; 131]. This approach clearly favours the overall availability of the system. The author in [53] states that the population of the super peers selected by Skype tends to be relatively stable. Therefore, this represents a compelling point in the P2P design space where heterogeneity is leveraged to manage churn, not just to cope with it. The SN form an overlay network which reduces the number of searching clients, improving overall searching. [59] states that if SN is relaying voice traffic than 4 Kbps bandwidth are consumed.

3. **Skype Registration Server (RS)**

   This is the only central element in the Skype network. It keeps records of the user names and passwords. It also ensures that Skype user names are unique across the Skype network [15; 53]. An SC must first authenticate itself with the RS for a successful login. The RS also keeps a record of the super peers. This list of super peer is provided to every new client at first login.

4. **Ports**

   An SC opens a TCP and UDP listening port at particular port numbers, which are normally assigned randomly at the time of installation [15; 18]. In addition to the random ports the SC also opens a TCP port at the port number 80 the http port and port number 443 the https port. These are used for transmition and reception of control signals for the purpose of maintaining links.

5. **Codecs**

   If we want to transmit analogue sound waves on the digital network of the internet, we need conversion at both ends of the call. This process of conversion from analogue to digital and the reverse is achieved by mean of a (CODEC). There are many standard algorithms [137] but as this is a complex process so every application use their own preferred. The author [15] states that Skype uses iLBC [26], iSAC [68] or their self created codecs which is unknown, although, GlobalIPSound [67], who originally implemented the iLBC and iSAC, do claim that Skype is their partner in their codecs. The real experiments in [15] confirm the use of wideband codec.

6. **Encryption**

   [15; 18] states that Skype is utilizing the AES (Advanced Encryption Standard), also known as Rijndel. The same is used by the U.S Govt. Organisation, to protect sensitive information. The encryption is composed of 256 bits and offers a total of 1.1 x 1077 possible keys [15]. This helps Skype to encrypt data and voice traffic.

5.2 Three Layer Architecture

Skype is a proprietary protocol [2] so very little is published by the company regarding its operation and network architecture. However, due to its popularity, it has always remained on the agenda of researchers [2]. The author in [121] used a reverse engineering approach to find the architecture of the Skype network while others [15; 22; 46; 53; 134; 154] used real network traffic tracing.

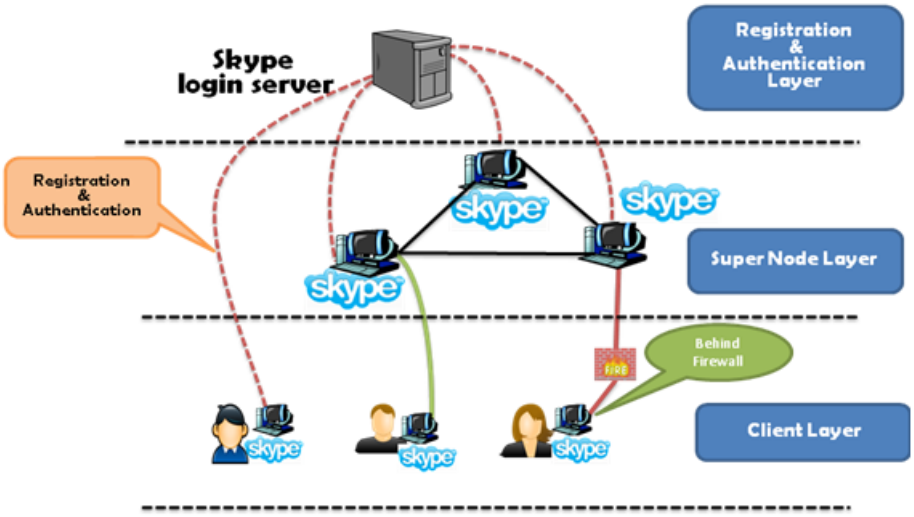

Figure 5.1: Skype architecture showing three layers

The reverse engineering and deep packet analysis of Skype network show that its protocol consist of three layers. Fig. 5.1 the first layer is known as the registration and authentication layer. This layer consists of a single Skype server. The server has the primary role of authentication and registration. In addition, this server also keeps a record of active super peers in the network

The second layer is the super peer layer. The super peers are randomly selected from the pool of users based on constraints on bandwidth, memory and CPU power [15; 53; 178]. These constraints and solutions proposed in the literature have been discussed in Chapter 4.

The last layer is the client layer. Every user is a client when authenticated and registered. After authentication and registration, it can either stay in the client only role or get promoted to the role of the super peer if it meets the requirements. If client status is retained it has to select one of the super peers as a host.

In this thesis, we are going to model Skype like protocol, so the above infor-

mation will help to model the Skype network as closes as possible.

Skype topology formation or network building process can be further sub divided into sub events like login, promotion to SN, SC connection to SN, SN connection to SN, user search and call placement.

5.3 Skype Network Operation

Skype topology formation or network building process can be further sub divided into sub events like login, promotion to SN, SC connection to SN, SN connection to SN, user search and call placement.

Login is one of the most influential and critical function required for the Skype network topology formation and operation [15; 53; 109; 178; 182]. At this stage, not only SC authenticates its user name and password with central login server but also inform other participants and friends about its presence. This activity is shown by edge marked as 1 in the Fig. 5.2. Other crucial task at this stage is the determination of NAT and firewall if used by the connecting SC. After successful login, it discovers SNs. According to the analysis performed by [15] there are two type SCs that go online in the network, freshly installed Skype new user and existing registered user. If a fresh user tries to register with the central server, first the user name selected is checked against the user database, if not issued previously, the same is assigned to the user. In the second step, the software version is checked against the latest update issued [15]. In the third step, the host cache (HC) is provided with a list of IP address and ports numbers of SN that SC refreshes at regular intervals in order to have an updated list of the SNs. If already registered SC tries to connect to the Skype Network, it first

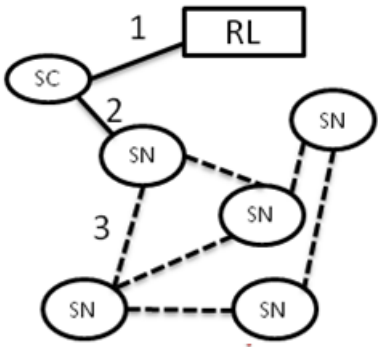

Figure 5.2: Skype login, registration and overlay formation

sends UDP messages to bootstrap SNs in HC. This is a handshake request if reply received with in 5 *seconds* [15] then connection with SN is established. If reply fails, SC try TCP with bootstrap nodes and if replied connection is made. If all the bootstrap nodes failed, SC may contact registration server for an updated list.

At the login stage, an SC identifies itself either with public IP or behind NAT or firewall. This task is achieved through the use of Session Traversal Using NAT (STUN) [70; 157] protocol.

If SC is connected with the internet through a public IP and have sufficient spare bandwidth, memory and CPU cycles, it may be selected for the extended role of the super peer. If SC meets the constraints, it can advertise itself with the RS as SN and it can start serving other SCs. [178] States that an SN can offer support from 55 to 80 SCs.

Once the SN is promoted it starts building its neighbourhood by connecting to other SNs. The SNs maintain an overlay network among themselves, whereas in this overlay each SN is connected to a limited number of SNs rather than every SN is connected to other SN [15; 53; 109; 178]. The experiments in [178] claim

that every SN keeps almost 0.05% connections as overlay with other SN. This formation of overlay network among SNs is shown in Fig. 5.2 as marked number 3. An SC in Skype network has to select one (or a small number of) SNs as their local host [15; 109; 178]. The selection of SN can be random or either based on bandwidth or latency. This activity is marked as edge 2 in the Fig. 5.2.

After successful login and subsequent connection with SN. SC provides detail of its availability to the global index maintained by SNs overlay. Skype claims that it is using Global Index (GI) [66] technology to search for a user. Skype claims that search is distributed and it guarantees to find a user if it exists and has logged in during the last 72 hours [166].

In a Skype network if either SC or SN wants to place a call on the network, there are three possible types. It is necessary to mention that call signalling information or request is always initiated through the TCP. If callee is not in the friend list of the caller. This call may take little longer as first the user will be searched and then call requite will be initiated.

In the first type of VoIP call, both the caller and callee are connected to the internet through public IPs. In such a situation, If both are in the friends list of each other and both are currently online then once the call button is pressed, call signalling information are exchanged. If the callee accepts the call, the flow of conversation packets will start. However, the timing to complete the call may change if callee is not in the friend list. Further, this call is direct, and packet flow from caller to callee.

In the second setup, if the caller is behind port restricted firewall or NAT and callee is on public IP. In this case, the caller cannot send signalling as well as actual voice packets directly to the callee. The caller will first send the call request to its

respective host SN, and then the host SN will forward the request to the callee. If callee is online and the request is acknowledged, than actual voice packets will be first forwarded by the caller to host SN and then subsequently forwarded to the callee. The callee response will also follow the same route [15; 166].

In third setup is when both the caller and the callee are behind port restricted firewall or NAT. In this case, Skype first uses STUN-like NAT traversal to establish the direct connection. In the event that the direct connection fails, Skype used alternate mechanism where the VoIP session is relayed by a publicly reachable supernode. This latter approach is invoked when NAT traversal fails, or a firewall blocks some Skype packets[15; 166]. Similarly, if the callee is behind port restricted firewall or NAT and the caller is connected to the network through public IP. The use of STUN enables the node behind the NAT to initiate/respond the TCP/UDP media session regardless of which end requested the VoIP call.

5.4 Statistics

Aaytch organization is dedicated to the collection of real Skype network statistics. They claim to be collecting Skype user information every 15 seconds. Table 5.1 developed based on the data from [150], shows that the total number of registered users is more than 65 million whle about 25 million were online on 17th January 2011 at 14.00 PM UK time. The average number of users who were online during the 2nd week of January 2011 was about 20 million. The Table 5.1 also gives details of the peak number of users attained during week.

Table 5.1: Skype Real Time Online Users Statistics

Description	Users	Date and Time
Skype Users Online Now:	25,150,422	1/17/11 14:00
Skype Real Users“:	65,814,593	1/17/11 14:00
All-Time peak:	27,474,821	01/11/2011 19:00
Average online this week:	20,112,727	1/17/11 14:00

In order to get a closer picture, we are interested in the numbers of users over a period of 24 hours. This will later help us to adjust the rate associated with transformation rules. In Fig. 5.3 we have plotted a graph of seven days showing the minimum, maximum and average numbers of users online in the second week of January 2011. If we can produce the same numbers of users on average, this would confirm the validity of the model and simulation approach.

In order to get an answer to questions like how much time do people spend on calling, or what is the duration of the calls made through Skype. [53] states that during their experiments the minimum call duration observed was 2m 50s, while the average was 12m 53s. The longest call during their experiments lasted about 3h 26m.

Over a period of 24 hours, the average number of clients that joined the network varies by over 40% of the active registered accounts, whereas the average number of super peers in this period is below 24% of the active population [53]. Similarly, the author in [171] state that in December 2010 there were 1.4 million super peers and 25 million users were online, when network crashed due to non-availability of super peers. This figure confirms that almost 6% of the on-line user were super peers, and still Skype network crashed. This further confirm that Skype super peer population must be over 6% of the online users in order to make the network stable.

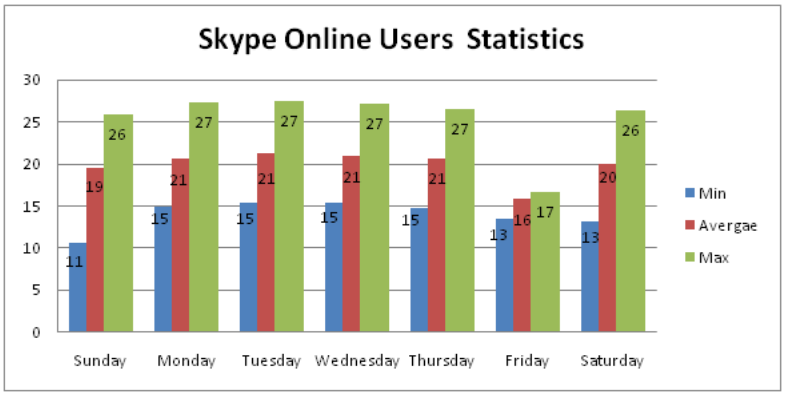

Figure 5.3: Skype online users statistics over a period of seven days

5.5 Summary

In this chapter, a case study based on the popular VoIP application Skype was presented. In the case study, we discussed that Skype protocol has a three layer architecture. We have presented the real statistics collected from the Skype network. These information will be used in the stochastic simulation to select suitable probability distributions for graph transformation rules and to adjust their rats of application, so that similar average are produced.

Chapter 6

Modelling P2P VoIP Networks

In the first part of this chapter we are going to present a structural model of a P2P VoIP Skype like protocol. The model is based on the discussion of the case study and the design variations studied in Chapter 4. This chapter will the present a model in terms a of type graph. The type graph will support all the design alternatives with attribute and cordinality constraints. In the second part of this chapter, we introduce a set of transformation rules according to the feature diagram in Chapter 4.

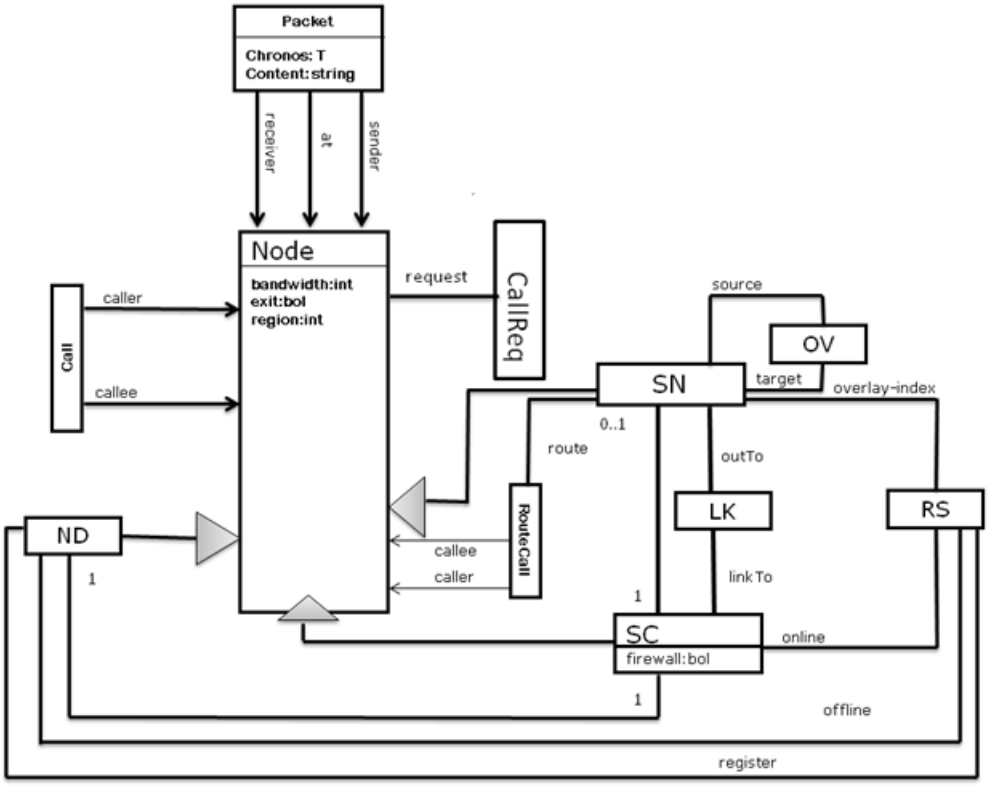

Figure 6.1: Type graph

6.1 Graph Model for the Skype Protocol

The type graph TG in Fig. 6.1 represents a structural model for Skype architecture as described earlier in the case study. It defines types for registration server (RS), super peer (SN), Skype client (SC), client (ND) and their common super type (Node). The node type LK is used to model links between SN and SC, while OV represents the overlay connection between existing SNs. The edge types *registration*, *online*, *offline* and *overlay-index* are used to show connection between RS, SC, ND and SN. The node type *Packet* is used to send control or probe messages between SC and SN. The edges types *sender*, *receiver* and *at* are used to model the sender, receiver and current location of the packets. The node

types *Call* and *RouteCall* are used to model different types of calls supported by the model. The edge types *caller* and *callee* are used to model the caller and recipient of VoIP call, whereas the edge type route is used to model calls routed through the SN. The bandwidth attribute is used to model the bandwidths of SC and SN, whereas the *exit* attribute is used to model the controlled departure of SCs and SNs. The *firewall* attribute is used to model the *NAT and firewall* status of the SC.

The objective of the model is to evaluate different strategies of connecting SCs to SNs such as random selection, selection based on bandwidth and time spent in the network, selection based on bandwidth and latency, and region-based connectivity. The model will also be used to evaluate single and multiple connections between SCs and SNs. The model will evaluate different strategies for promotion such as promotion of SC to SN at the start, dynamic any-time and dynamic need-based.

The performance result of these strategies are compared on the basis of the number of SCs happy with their current SN, SCs connected, and SNs promoted. The model also provides information of the overall number of SCs and SNs in the network and calls generated either directly or through an SN.

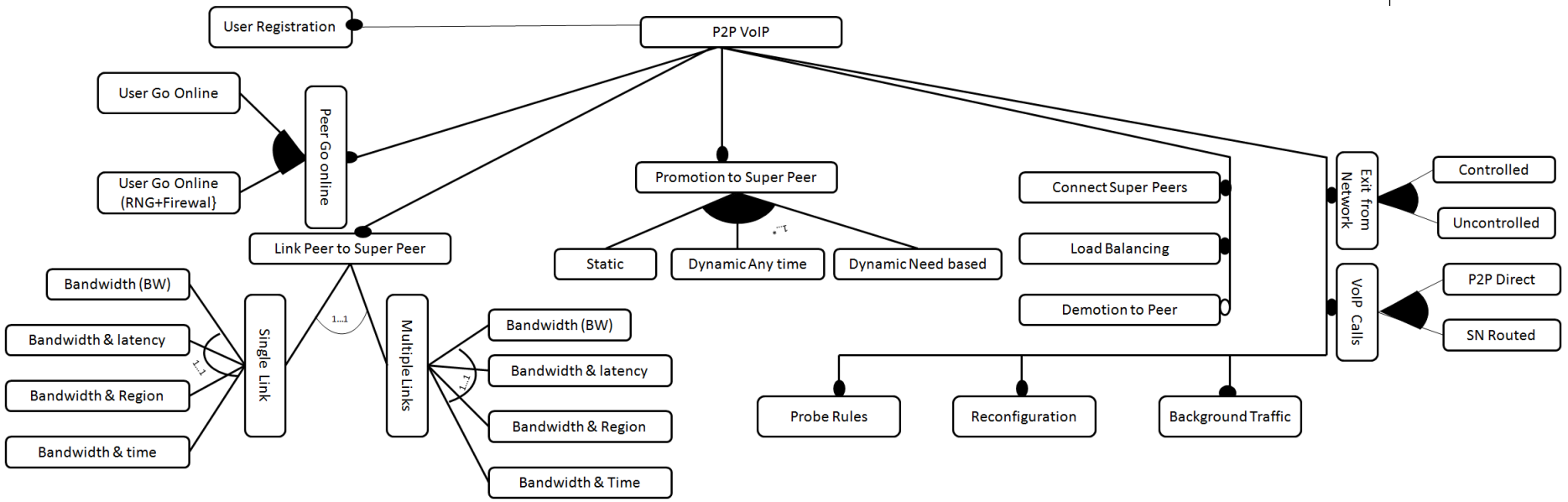

Figure 6.2: Design variations of P2P VoIP protocols

6.2 Graph Transformation Rules Modelling Protocol Behaviour

We are now going to present the transformation rules in the order of the features in Fig 6.2, i.e., registration, going online, linking SC to SN, promotion to SN, SN and SC departure, load balancing, call placement, background traffic, reconfiguration and probe rules. The feature diagram in Fig 6.2 is based on the design variations discussed in Chapter 4, the case study in Chapter 5, and the model in the previous section.

6.2.1 Registration of New User

In the model, whenever a new user wishes to join the network, it has to first register with the RS server. Once registered the user will still be offline. If a registered user wants to use the network, it can go online at any time. The *online* edge in Fig. 6.1, show when an *offline* user ND goes online, it is first assigned the status of client SC.

6.2.1.1 New User Registration

The rule in Fig. 6.3 is used to register a new user with the server. The LHS of the rule models a single server RS, and the RHS shows that whenever a new user arrives, it has to register with the central server RS.

6.2.1.2 Successful Registration

The rule in Fig. 6.4 is used to show the successful registration of a new user. When a new user is successfully registered, its existing *register* edge with RS is

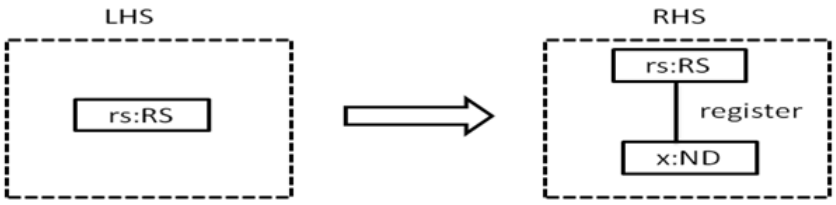

Figure 6.3: New user registration

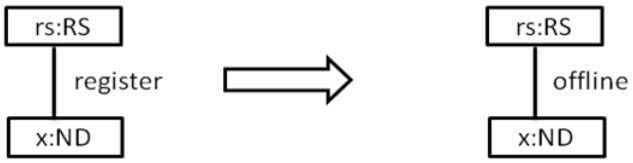

Figure 6.4: Successful registration

replaced by an edge of type offline. The user's subsequent role of client (SC) and super peer SN will be decided later when the ND tries to go online.

6.2.2 Client Goes Online

A registered user can go online at any time. Once online the type is changed to SC modelling a Skype client. The model supports here two variations. The first randomly assigns bandwidth to SCs, and decides which SC is behind a firewall and which is connected using a static IP. The second choice is to separate firewalls and static IPs. This client's resources separately to be able to increase or decrease the number of clients behind firewall or NAT. This will help to comment on the performance of the model in extreme conditions.

6.2.2.1 User Goes Online with Random Firewall

This rule in Fig. 6.5 is used to model the transformation of a registered user (ND) to an online client of the VoIP network. This rule consider a registered user

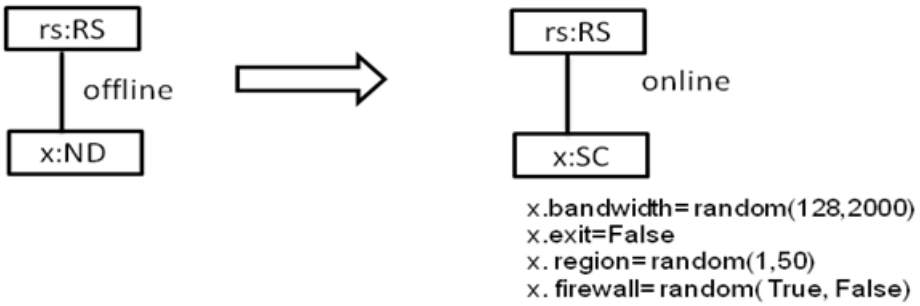

Figure 6.5: User goes online with random firewall

which is currently offline and makes them online. IT assigns each user random bandwidth between 128 Kbps to 2 Mbps and sets the exit attribute to false, modelling that the client does not intend to leave the network. The firewall attribute is set to disabled or enabled randomly. This rule further assigns the client a region number between 1 and 50, which models the locality of the client [96]. The purpose of the region attribute is to identify clients that belong to the same region [1].

6.2.2.2 Separating Firewall Decision

Go Online with Firewall Disabled

The rule in Fig. 6.6 is used to model the transformation of a registered user(ND) into an online client (SC). The SC is assigned random bandwidth between 256 Kbps to 2 Mbps. The exit attribute is set to false and the firewall is statically set to false. This rule further assigns a random value to the region attribute modelling the locality of the SC in the network.

Go Online with Firewall Enabled

The rule in Fig. 6.7 is used make client go online with firewall enabled. The LHS of the rule is same as that of the rule in the Fig. 6.6. The only difference,

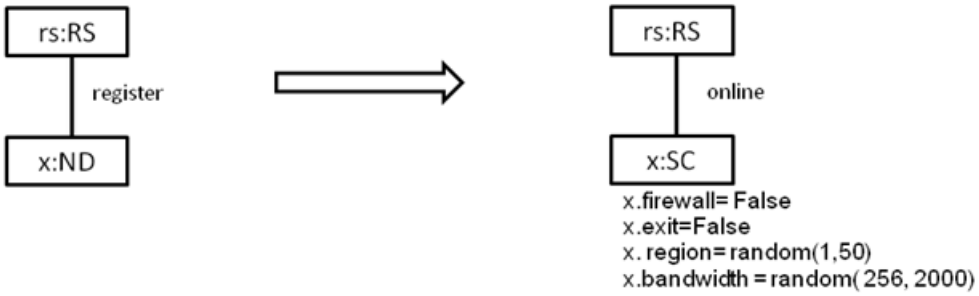

Figure 6.6: Users go online with firewall disabled

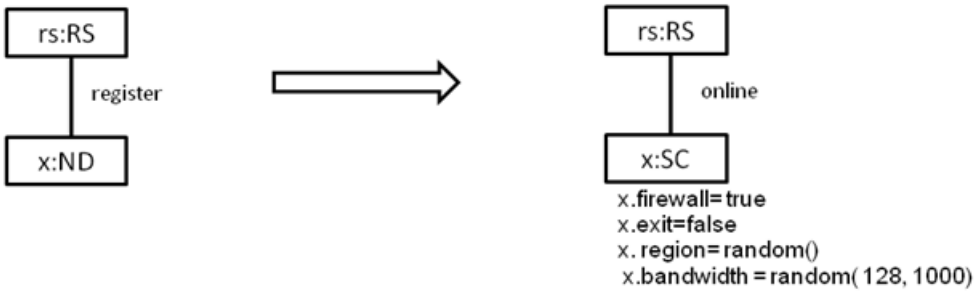

Figure 6.7: User go online with firewall enabled

is in the attributes on the RHS of the rule. As the client is behind a firewall, we reduce the bandwidth range for the client. The bandwidth available for this client is between 128 Kbps to 1 Mbps. This rule sets the value for the firewall to true. The exit and region attributes hold the same values false and random. This rule is helpful, when we want to increase the population of SCs with firewall enabled.

6.2.3 Selection of SN by SC

Once an SC goes online with the RS, next it has to select one or more SNs as host. The connection between SC and SN is modelled through node LK, where the edge type *linkTo* is placed between the SC and LK. The edge type *outTo* is placed between LK and SN. The bandwidth attributes of SC and SN are reduced

by 5 Kbps, modelling the cost of this connection. In the model, an SC can either choose to have a single connection or multiple connections to SNs. The model allows SN to accept and reject new connections based on the local awareness of current bandwidth and exit attributes. SN will reject new connections if its current bandwidth is less than 256 Kbps or its exit attribute is set to true. In the same way, the SC can select an SN as the host if latency between the connecting SC and SN is less than 300 ms (round trip time). To find the latency a packet carrying a time stamp is sent as shown in Fig. 6.1. If the round trip time taken by a packet is more than 300 ms, the SN is rejected and connection with new SN is tried, sending another packet. Similarly, an SC can select an SN on the basis of bandwidth and time spent by the SN in the network, serving other SCs. If an SC wants to connect to an SN which is close to the SC in terms of network locality, the SC can select SNs based on the region attribute value. All these variations can be seen in the feature diagram at Fig. 6.2.

6.2.3.1 Selection of a Single SN as Local Host

1. **Random Selection of SN on Bandwidth**

   The rule in Fig. 6.8 links an SC with an SN. The LHS of this rule uses a NAC to prevent an already linked SC from relinking. This NAC condition is represented by node LK crossed. This rule also selects an SN from the population of available SNs, if an SN has a current *bandwidth* more than 256 Kbps, and is not leaving the network, SN is a candidate for selection. In the RHS of this rule, a linking SC terminates its current edge with the RS and create a new connection with SN. Both host SN and connecting SC pay a bandwidth cost of 5 Kbps for keeping this connection.

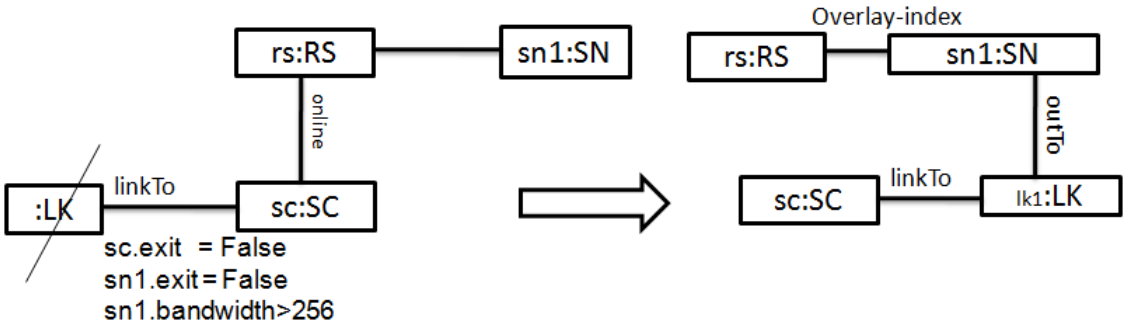

Figure 6.8: Random selection of SN on Bandwidth

2. **Selection of SN based on Bandwidth and Latency**

   The transformation rules in this section are used to model a scenario in which an SC makes an attempt to link a random SN based on bandwidth as well as latency [1; 9; 15; 53; 173]. In this approach, the latency between SC and SN is computed using a time stamped packet.

   (a) **Create and Send Time Stamped Packet**

   The rule in Fig. 6.9 creates a packet p1 and sets the chronos attribute of the packet p1 to the current simulation time. However, p1.chronos is just a static time stamp. This rule also sets the chronos attribute for SC to the current time. This will enable the SC to keep timing information such as when the packet was transmitted. This rule sets the contents of the packet to *ping*. The ping packet will be considered by an SN as a connection request. This rule make sure the packet is transmitted to an SN, not intending to leave the network in the near future. The edge *at* is used to show the current location of the packet.

   (b) **Reply Packet with AcK**

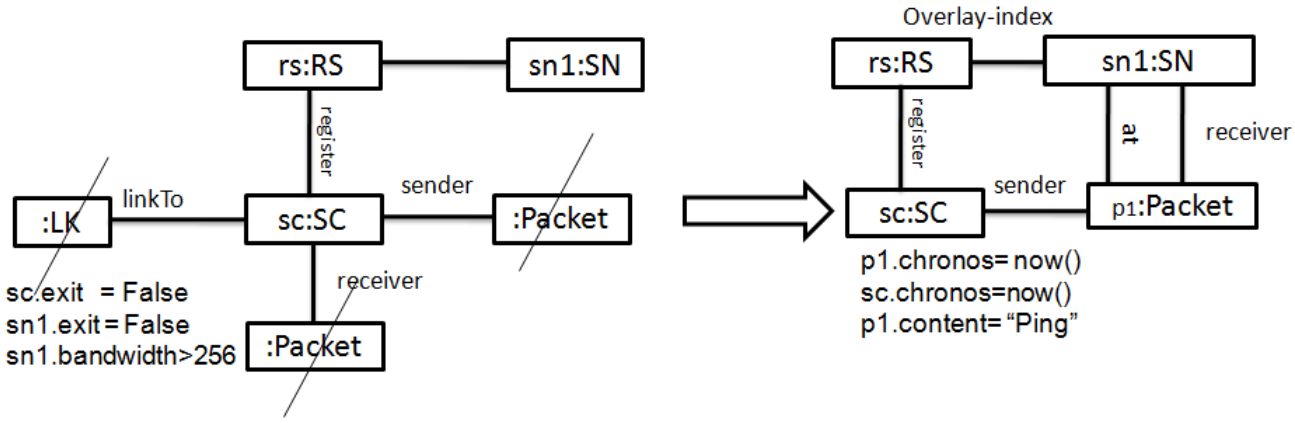

Figure 6.9: Create and send time stamped ping packet

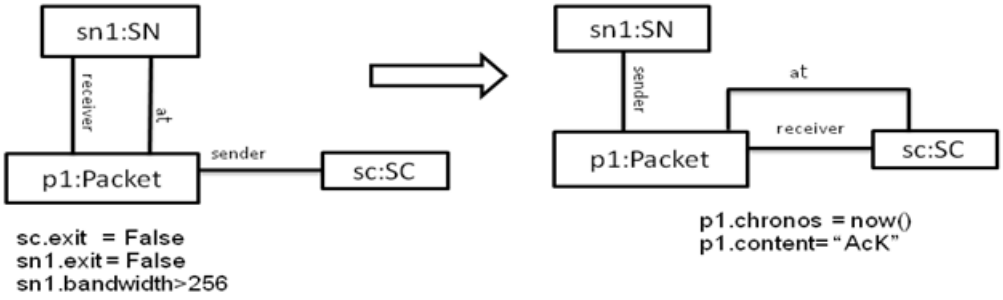

Figure 6.10: Reply packet with AcK

If the current available bandwidth at an SN is more than 256 Kbps and the exit attribute is set to false, the rule in Fig. 6.10 will return the packet p1 with message content *AcK*. Once the packet reaches SC, it updates the chronos value of p1 with current time. This will help the SC to find the time taken by the packet to complete a round trip (RTT).

(c) **Reply Packet with DnY**

The SN through the rule in Fig. 6.11, returns the packet p1 with message content *DnY* (Denial of Service), if the available bandwidth

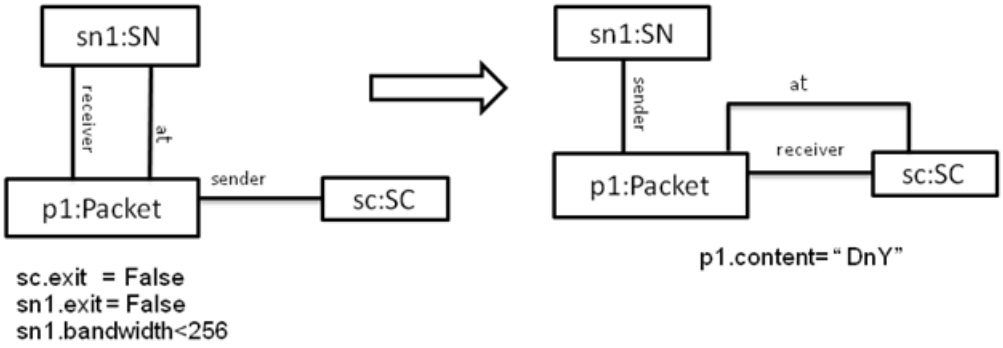

Figure 6.11: Reply packet with DnY

at SN is less than 256 Kbps. This models the situation when the SN is experiencing either too many connections or congestion.

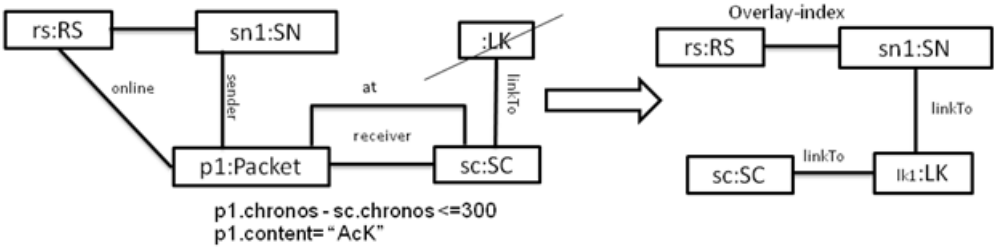

Figure 6.12: Link SC with the SN based on latency

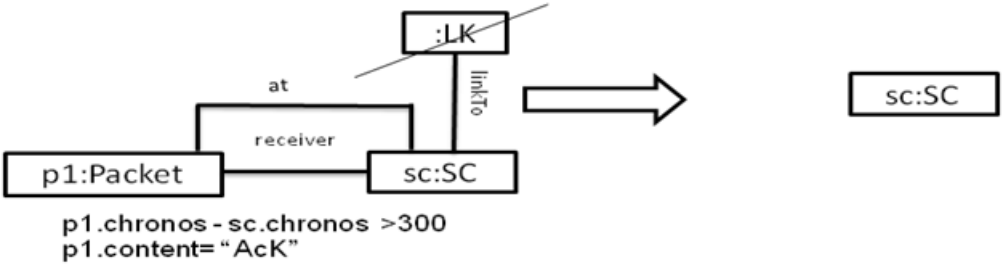

Figure 6.13: Reject SN based on latency

(d) **Links SC with SN Based on Latency**

If an SN returns the packet p1 with a message content *AcK*, and the RTT taken by packet p1 is not more than 300 ms, the requesting SC will select the current SN as their host for the rest of stay in the network. Both SC and SN pay a 5 Kbps of bandwidth as cost for keeping this connection. The rule in Fig. 6.12 will also disconnect the SC from RS by removing edge online.

(e) **Reject SN Based on Latency**

If SN returns a packet p1 with *AcK* content, but the RTT is computed to be more than 300 ms during the round trip time. The SC will reject the current SN by applying the rule in the Fig. 6.13.

(f) **Reject SN Based on Return of DnY Packet**

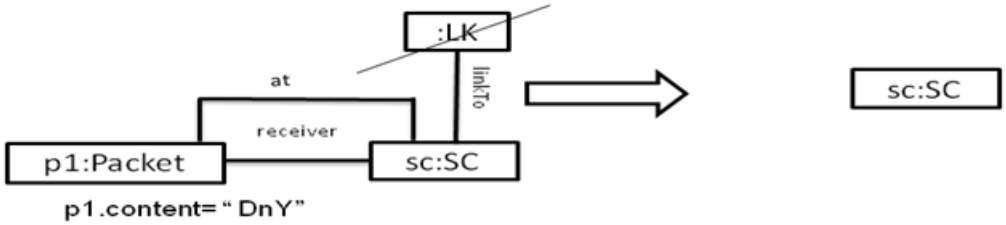

Figure 6.14: Reject SN Based on Return of DnY packet

The rule in Fig. 6.14 is used to reject the current selection of SN. As the SN has refused a new connection by sending packet DnY. Once this rule is applied, the existing Packet p1 is deleted and a new selection of SN will be made by applying the rule in Fig. 6.9.

3. **Selection of SN as based on Region**

This version of linking to an SN is based on concept that local SN results in less RTT than a non local one [1]. This approach gives equal importance to spare bandwidth and locality of the SN. The strict version of this protocol only connects an SC with an SN, if both of them share common region number. Since this may increase the waiting time for an SC to connect to an SN, the enhanced version makes an attempt to connect to a local SN and if this is not available, connects to a non local one. The enhanced version uses both strict linking and dynamic linking, so we present both rules for better understanding.

(a) **Links SC to SN on Bandwidth and Region**

The rule in Fig. 6.15 is used to select a local SN for linking an SC. The selected SN has the same region number as the connecting SC. Further, the selected SN has a current spare bandwidth of more than

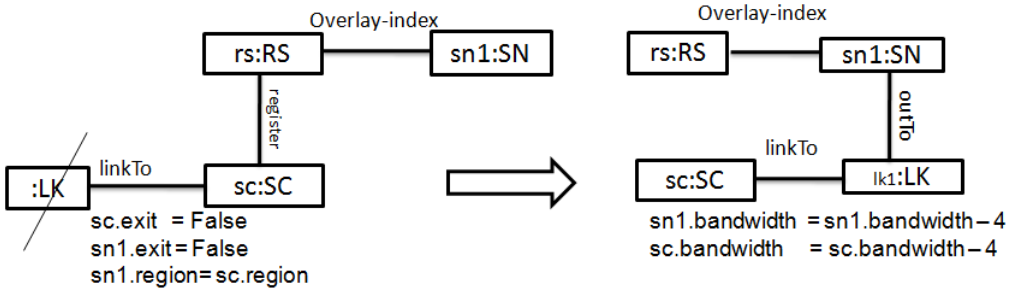

Figure 6.15: Links SC to SN on bandwidth and region

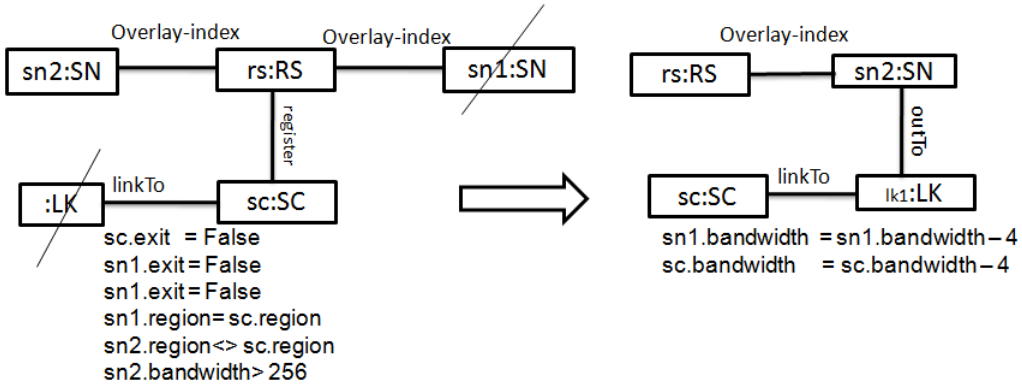

Figure 6.16: Linking to SN on bandwidth

256 Kbps. The SC is linked with the SN and both pay 5 Kbps as compensation.

(b) **linking to SN on Bandwidth**

The rule in Fig. 6.16 makes an attempt to link an SC with a random SN if there is no local SN in the current graph. This rule uses a NAC to confirm that there is no SN in the current graph which has the same region number as SC. If the NAC is true, a random SN is selected without taking care of the region. If this SN accepts new connections, the SC is linked with this non local SN.

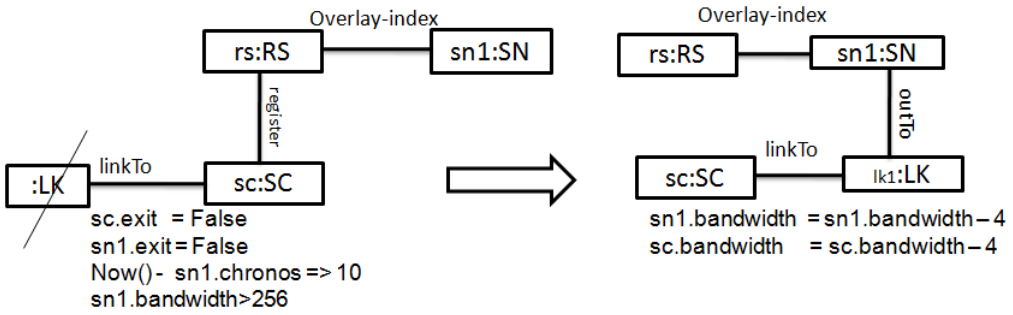

Figure 6.17: Link SC to an SN based on bandwidth and stability

4. **Selection based on Bandwidth and Time spent in Network**

   In this rule, an SC makes an attempt to select an SN which is more stable [99]. The rule in Fig. 6.17 not only considers bandwidth, but also the time SN has been serving. As LHS of this rule, an SN with current spare bandwidth of more than 256 Kbps is selected, and the time of serving in this role is computed by looking to the chronos attribute. If an SN is serving for more than 10 minutes, SC is linked.

6.2.3.2 SC Selection of Multiple SNs as Host

The feature diagram in Fig. 6.2 shows that an SC has the choice to connect to multiple SNs [118; 176]. This approach has some drawbacks such as the higher cost in bandwidth. These factors will increase the burden on the available SNs and may reduce the number of happy clients in the network. We discuss in this section the rules available to connect an SC to multiple SNs.

1. **Link to Multiple SN based on Bandwidth**

   The rule in Fig. 6.18 is used along with the rule in Fig. 6.8 to connect an already linked SC with another SN. This rule uses two NACs. The first

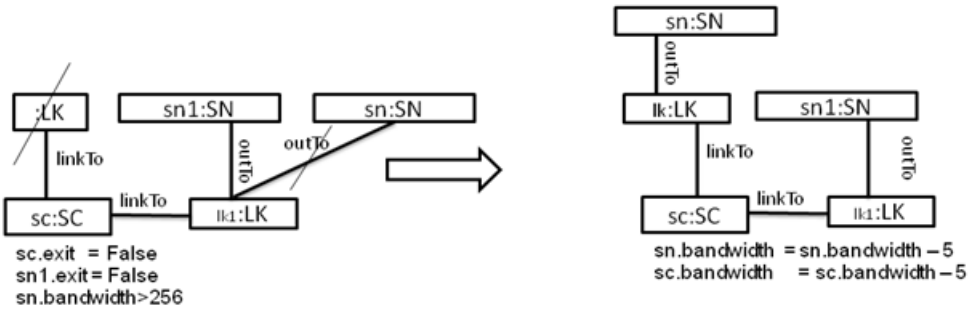

Figure 6.18: Link to multiple SN based on bandwidth

condition makes sure that the current SC is not linked to two SNs in the current graph. The second NAC makes sure that one new SN selected for the second connection is not the same already linked. The attribute constraints on *exit* and *bandwidth* are checked as precondition of the rule. If the precondition is satisfied, a link is established with a second SN.

2. **Link Multiple SNs Based on Bandwidth and Latency**

   The transformation rules in this section are used to connect SCs to multiple SN. This approach is based on computing latency before establishing the link with an SN. The procedure and approach is the same as for the single link. The only difference here is that an SC is linked multiple SNs instead a single SN.

   (a) **Create and send Packet to the Second SN**

   The rule in Fig. 6.19 sends a time stamped packet to a newly selected SN for establishing a second link.

   (b) **Reply Packet With AcK Content**

   The rule in Fig. 6.20, return the time stamped packet with *AcK*.

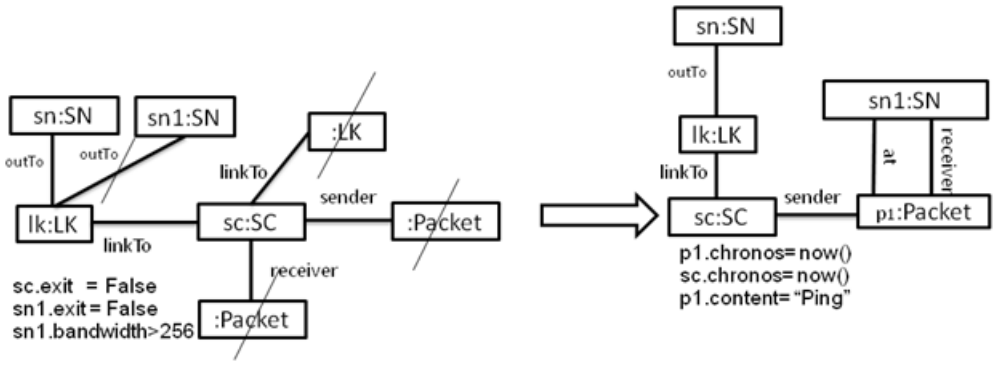

Figure 6.19: Create and send packet to the second SN

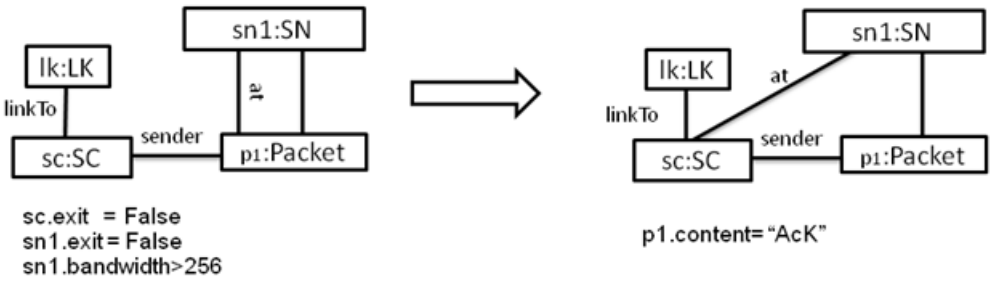

Figure 6.20: Reply packet with AcK

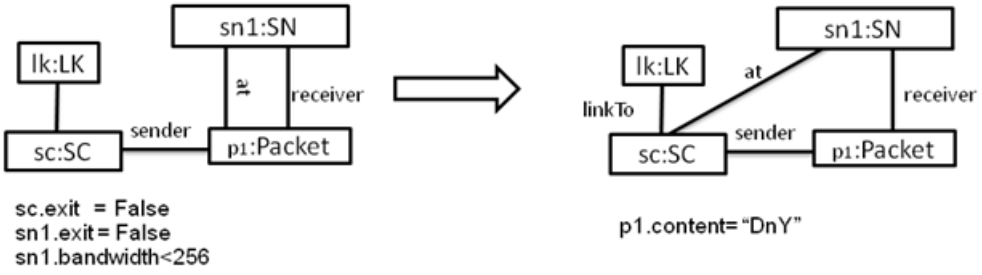

Figure 6.21: Reply Packet With DnY

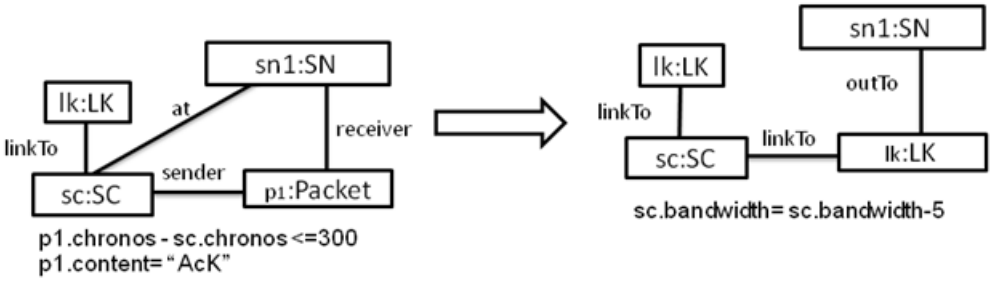

Figure 6.22: Link an SC with second SN

(c) **Reply Packet With DnY**

The rule in Fig. 6.21, is used return the time stamped packet with *DnY*.

(d) **Link SC with SN based on Latency**

The rule in Fig. 6.22, is used to link an SC with second SN.

(e) **Reject SN Based Reply of DnY Packet**

The rule in Fig. 6.23 is used to reject SN.

(f) **Reject SN Based on Latency**

The rule in Fig. 6.24, is used to delete a packet p1 and reject sn1.

3. **Link to Multiple SNs Based on Bandwidth and Locality**

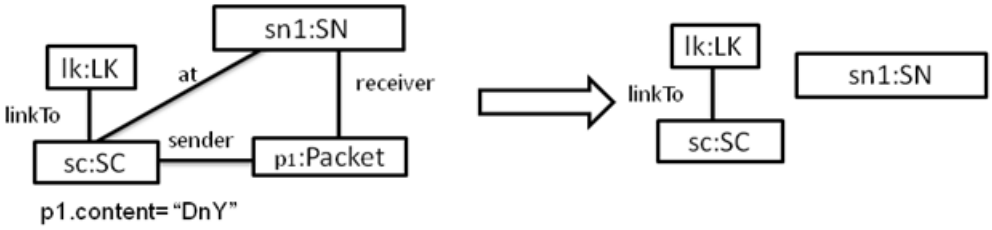

Figure 6.23: Reject SN based on reply of DnY

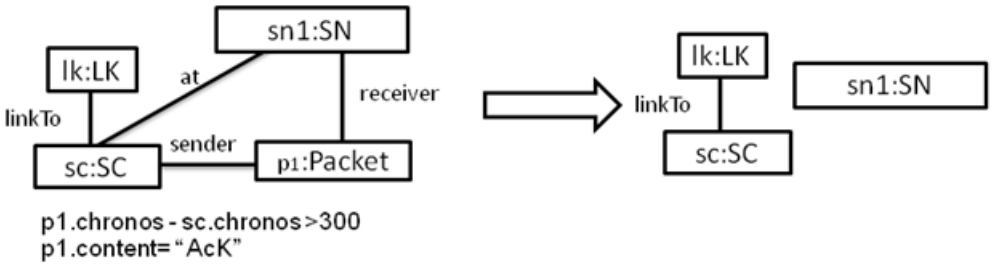

Figure 6.24: Reject SN based on latency

The rules in this sections are used to link SC to multiple SNs, based on current bandwidth at SN and region attribute.

(a) **Link to Multiple SN based on Bandwidth and Region**

The rule in Fig. 6.25 is used to link an SC to multiple SNs. This rule simply selects a new random SN based on the region number attribute. If the region number of SC and SN is the same and the current bandwidth of the SN is more than 256 Kbps, the SC will establish a second link with this SN. The NAC will confirm that this SN is not the same as SC is already connected, and the SC is not currently linked with two SNs.

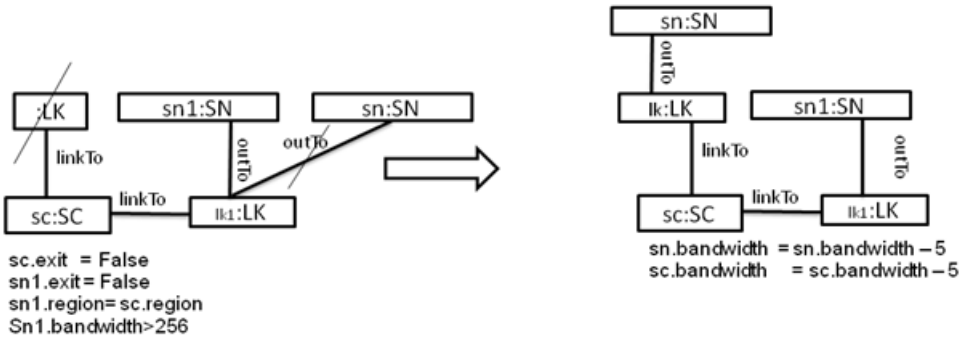

Figure 6.25: Link to multiple SN based on regions

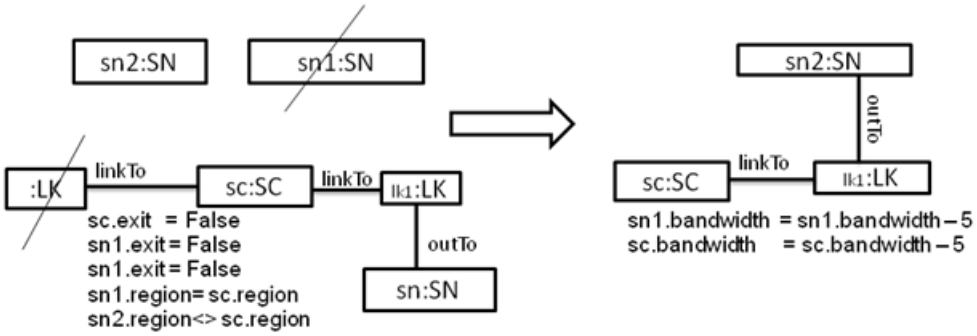

Figure 6.26: Link to multiple SN based on bandwidth

(b) **Link to Multiple SN Based on Bandwidth**

The rule in Fig. 6.26 is used to link SC to multiple SNs. This rule uses the same NACs as the single link.

4. **Link Multiple SN Based on Bandwidth and Time Spent**

This approach gives importance to the stability of SN and the rule in Fig. 6.27 use the same concepts as in single link rule.

6.2.4 Promotion to SN

If a user goes online and its firewall attribute is not set to true, this SC is candidate for SN. The model supports the promotion of the eligible SCs at three stages in

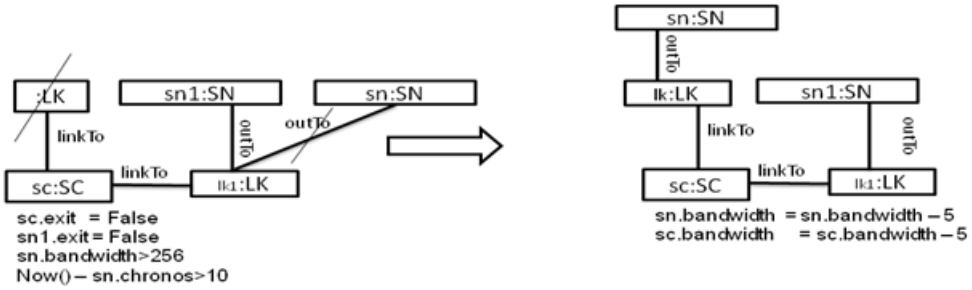

Figure 6.27: Link multiple SN based on bandwidth and time spent

the network. First, when a user goes online, second any time, and third whenever there is a need for an SN. These variations are presented in the feature diagram in Fig. 6.46. We have named these three choices as static, dynamic and need based [85]. The static protocol promotes an SC to SN as soon as it gets online and the bandwidth is more than 1.5 Mbps. The dynamic protocol can promote an SC at any time during the session in the network, but still the bandwidth will be considered. The need-based protocol promotes an SC to an SN only whenever there is a need for a new SN. The following transformation rules implement these three protocols. Once the SC is promoted to the extended role of SN, it retain, the primary role of the SC while serve other SCs in the network. After promotion, the SN also updates the RS regarding the new role.

6.2.4.1 Static Promotion of SC to SN

The rule in Fig. 6.28 is used for static promotion of SC to SN. In this rule, a recently on-line Skype client will be promoted to its extended role as SN. This rule uses an attribute constraint on bandwidth which makes sure that the client current bandwidth is more than 1.5 Mbps. In addition to bandwidth, the rule also confirms that the clients firewall attribute is set to false. Once these conditions

Figure 6.28: Static promotion of SC to SN

are satisfied, the client will be promoted to SN. Once successfully promoted, client SN informs the RS about the recent changes to its role in the network. In this protocol, the decision for client promotion is taken when joining.

6.2.4.2 Dynamic Promotion of SC to SN

The rule in Fig. 6.29, along with rule in Fig. 6.28, is used to implement dynamic promotion to SN at the start as well as later on. The dynamic protocol is based on the concept that an SC may not have sufficient bandwidth at the start, but later the SC may change, making this SC a candidate for SN. The rule in Fig. 6.29 is used to promote the SC to SN. The LHS makes sure that SC is linked with SN. The rule further confirms that SC has a current bandwidth of more than 1.5 Mbps and that firewall and exit attributes are set to false. If the selected SC satisfies these constraints, it is promoted to SN. Once the SC is promoted it terminates its current link with host SN. The new SN informs RS about change to its role in the network.

6.2.4.3 Need Based Promotion of SC to SN

The rule in Fig. 6.30 is used to implement the need-based protocol for promotion to SN. This protocol is based on the approach that bandwidth is expensive, so

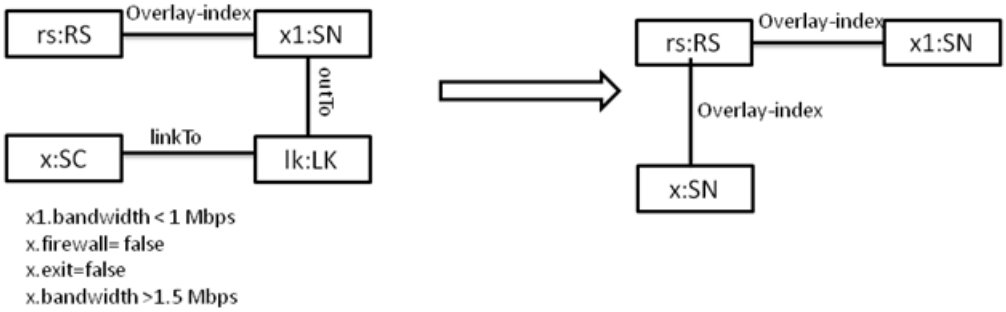

Figure 6.29: SC is promoted to SN while SC is linked with SN

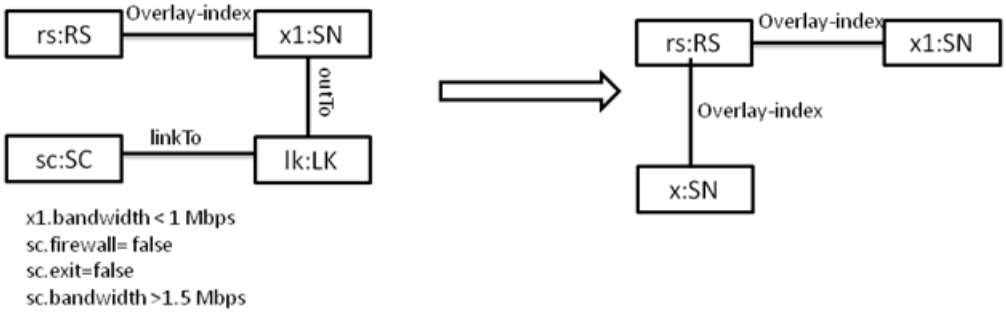

Figure 6.30: SC is promoted to SN based on the need of a new SN

promotion shall be strictly based on the need of the network. In this approach an SN, may select one of its clients for promotion. This rule selects an SN which has a current bandwidth of less than 1 Mbps. Further, the rule selects an SC from the SN dependent. The SC is promoted to the role of SN. However, the rule confirms that client meet the bandwidth requirements of the promotion.

6.2.5 Overlay Network among SNs

The rule in Fig. 6.31 is used to link an SN with four other SNs to form an overlay network. The recently online SC provides detail of its availability to the global index (GI) maintained by SNs overlay. The user availability is searched in the GI maintained by SNs. This rule uses two NACs. The first ensures that the SN

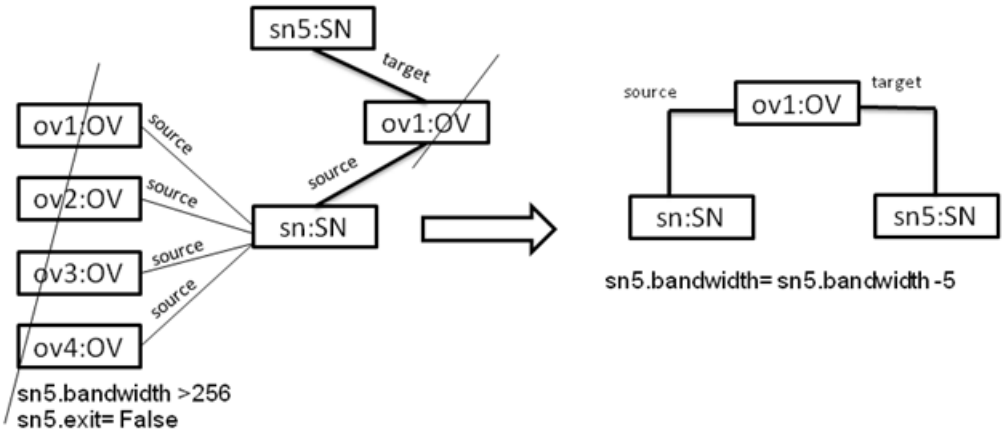

Figure 6.31: Overlay network among SNs

should only connect to four other SNs. The second makes sure that current SN can only link an SN, it is linked with. As RHS, SN establishes a new overlay connection. Both SNs pay a bandwidth of 5 Kbps as the cost for keeping the GI and connection status.

6.2.6 Load Balancing

The rule in Fig. 6.32 is used to transfer an SC from an overloaded SN to underloaded SN. This rule make sure that the current host SN has a bandwidth of less than 800 Kbps and there exists a new SN that has spare bandwidth more than 1.5 Mbps. The newly selected SN is not intending to leave the network. Once these conditions are satisfied, the SC is disconnected from the current SN and relinked with a new SN.

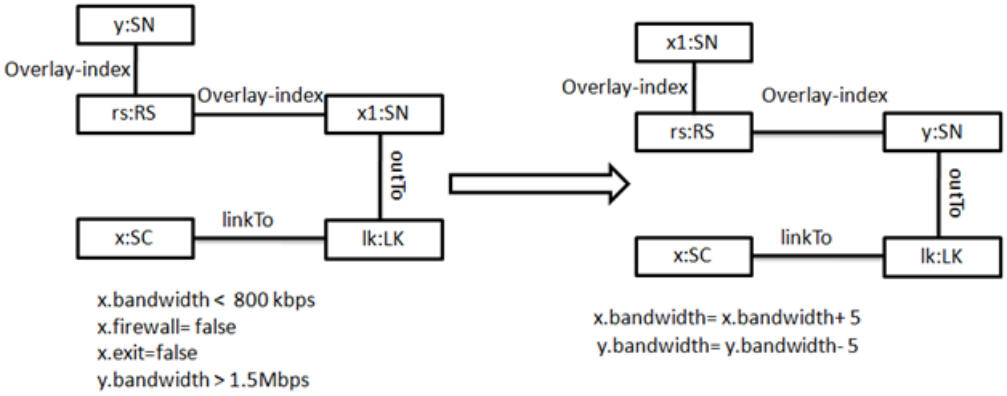

Figure 6.32: Reconfigure and transfer from overloaded

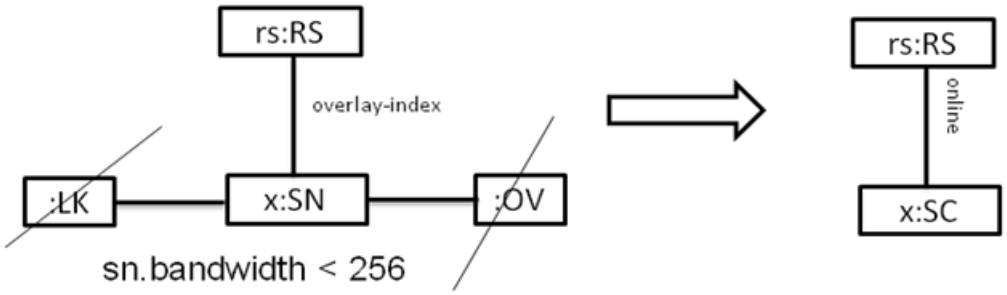

Figure 6.33: SC extended role of SN is cancelled

6.2.7 Demotion of Super Peer

The rule in Fig. 6.33 is used to cancel the extended role of a client as super peer (SN) and a client only status is restored. This rule uses two NACs. The first ensures that extended role should only be cancelled when current SN is not serving any client. The second makes sure that extended role should only be cancelled, if it does not have any current OV links in the overlay network. This uses attribute constraint which make sure that the current available bandwidth of SN is less than 256 Kbps. Once all these conditions are evaluated as true extended role as SN is demoted, and client only status is maintained. The demoted SN informs the RS by replacing the edge overlay-index by the edge online.

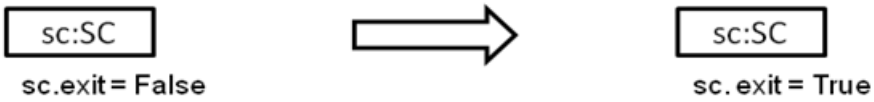

Figure 6.34: Control departure selection of SC

6.2.8 Departures of SC and SN

In the model an SC as well as SN can join, participate and exit the network by crashing and using the exit procedure prescribed in the application. We first explain the control departure from a network followed by the uncontrolled departure.

6.2.8.1 Control Departure of SC

In control departure, the user of the VoIP application uses the exit procedure provide by the application GUI. The use of this procedure enables the SC to inform an SN regarding the intended departure from a network.

1. **Control Departure Selection of SC**

   The rule in Fig. 6.34 is used to set the control exit attribute to true. This rule select a random SC and enables its exit attribute to true.

2. **Control Departure Preparation of SC**

   The rule in Fig. 6.35 is used as a result the rule in Fig. 6.34. Once an SC opts for leaving the network, and informs the host as well as involved clients by setting the exit attribute to true. In the next step it cancels its dependency link with SN. As link LK is removed both SC and SN recover their cost of compensation, i. e., 5 Kbps.

3. **Control Final Departure of SC**

Figure 6.35: Control departure preparation of SC

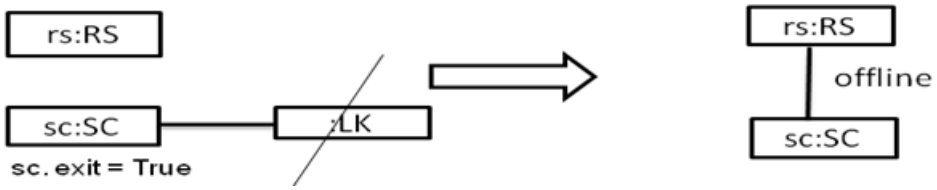

Figure 6.36: Control final departure of SC

The rule in Fig. 6.36 is used to make an SC offline.

6.2.8.2 Uncontrol Departure of SC

The second departure that is available to SC in the model is to leave a network selfishly without informing host SN or clients involved in call with current SC. The rule in Fig. 6.37 is used to model selfish departure of the S. In the LHS of this rule, we see that there are only two nodes RS and client SC. There are no attribute constraints or NACs. This rule simply make client SC offline.

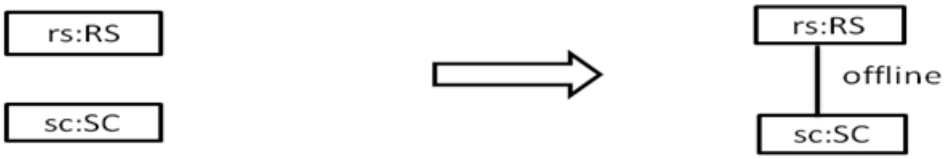

Figure 6.37: Uncontrol departure of SC

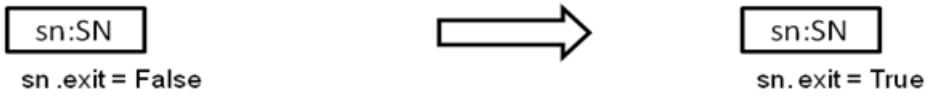

Figure 6.38: Control departure preparation of SN

6.2.8.3 Control Departure of SN

An SN can leave the network through a control procedure available in the GUI of the VoIP application. However, SN exit or departure is different from SC because SN not only has dependent SCs, but also have searchable overlay with other SNs. In the event of control exit procedure for SN, it has to inform both dependents SNs and SCs.

1. **Control Departure Selection of SN**

   The rule in Fig. 6.38 is used to set the control exit attribute of the SN to true. This rule selects a random SN, and set its exit attribute to true.

2. **SN Links Between SC and SN is Removed**

   The rule in Fig. 6.39 is used as post implementation of the rule in Fig. 6.38. Once an SN opts for control exit, existing linked SCs break their current link with SN and report back to the RS. The departing SN and SC recover their bandwidth compensation cost for the link.

3. **SN Overlay Links Removed**

   The rule in Fig. 6.40, is used as post implementation for rule in Fig. 6.38. In this rule the SNs in overlay disconnects their link and recover their cost of compensation for the link.

4. **Control Final Departure SN**

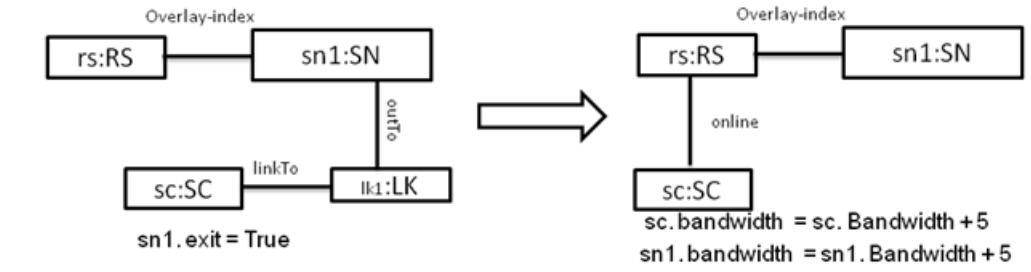

Figure 6.39: SN links between SC and SN is removed

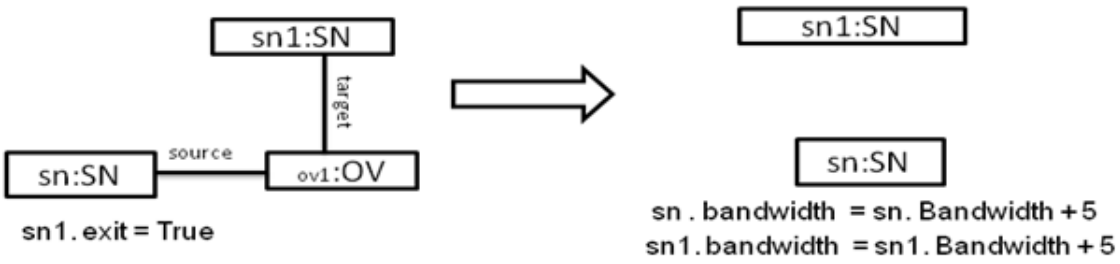

Figure 6.40: SN overlay links removed

The rule in Fig. 6.41 is used to make SN offline. This rule uses two NAC conditions. First make sure SN is not serving as host for SCs. The second makes sure that SN is not sharing searchable overlay network with other SNs. Once confirmed that no active connection exists, SN go offline.

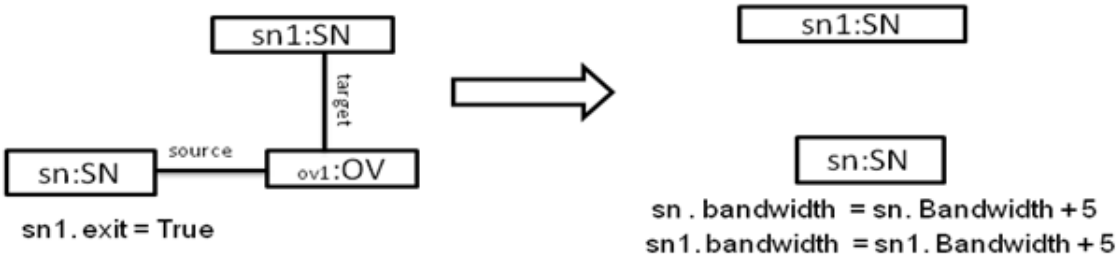

Figure 6.41: SN finally go offline

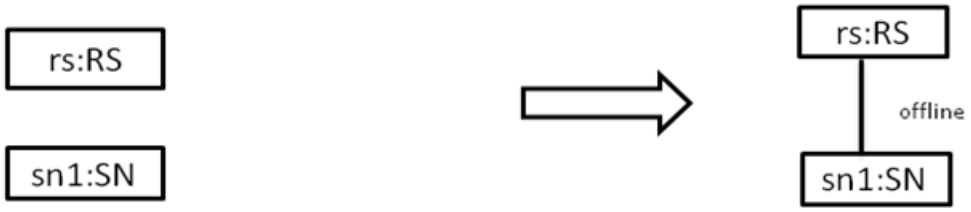

Figure 6.42: Uncontrol departure of SN

6.2.8.4 Uncontrolled Departure of SN

In uncontrolled approach the SN leaves the network without informing SNs or SCs. The rest is upto the dependent SC and SN to find a new host. The rule in Fig. 6.42 is used to make SN offline.

6.2.9 VoIP Calls

In the model shown in Fig. 6.1 two types of nodes are used to model VoIP calls. If caller and callee both have their firewall attribute set to false, both can make VoIP calls directly to each other without needing the assistance of the host SNs. It is worth mentioning that in this type of call, both the caller and callee can be either SCs or SNs. This type of call is represented by node call in the model, and the edges types *caller* and *callee* point to the caller and recipient of the call. The second type of call is modeled through a node called RouteCall. All those SCs with their firewall attribute set to true will use their host SN for the establishment of VoIP calls. The edge type *route* points to those SNs, which are relaying traffic of calls. As there are many codecs available, and each requires different bandwidth, for the purpose of simplicity in the model we assume that 80 Kbps is the cost of VoIP call. If the call is direct between the caller and callee, then only the caller's bandwidths are decremented, but if the call is routed though

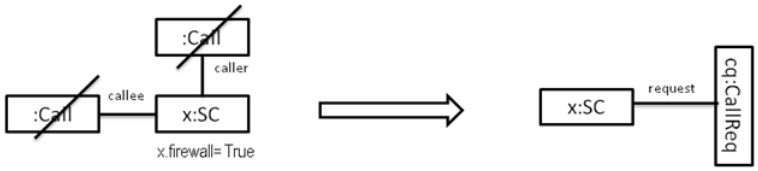

Figure 6.43: Call request by SC behind firewall/NAT

SNs they also pay the cost of the call. In the model, VoIP calls consist of three stages: request call, complete call and terminate call.

6.2.9.1 Call Request

Each client or super peer will first submit request for a VoIP call. If the user is online, the call between caller and callee will be established.

1. **Call Request by SC behind Firewall/NAT**

   rule in Fig. 6.43 is used to enable an SC behind a port-restricted firewall to request a VoIP call with the other client. The on-line participant can be either SN or SC. The callee of the call can be either connected through public IP or behind a firewall. In all cases, the VoIP traffic from this SC will be relayed by the respective host SN of the requesting SC. The rule in Fig. 6.43 use two NACs. These make sure that the requesting SC is not involved in existing VoIP calls before requesting a new call. CallReq is used to model and register requests of the caller for a new VoIP call.

2. **Call Request by SC having Static IP**

   The rule in Fig. 6.44 is used to enable an SC with static IP to request a VoIP call. The callee can be either SN or SC. The callee of the call can be either directly connected through live IP or may have firewall-enabled

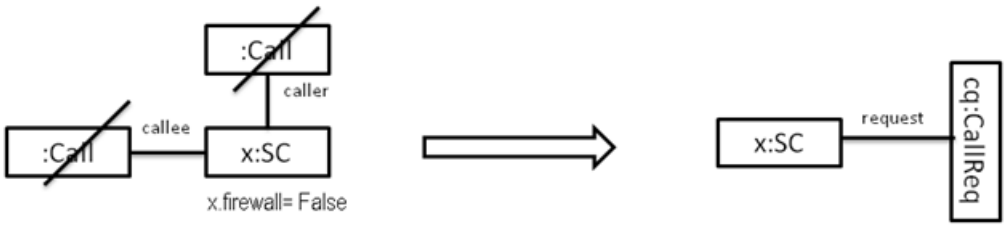

Figure 6.44: Call request by SC having static IP

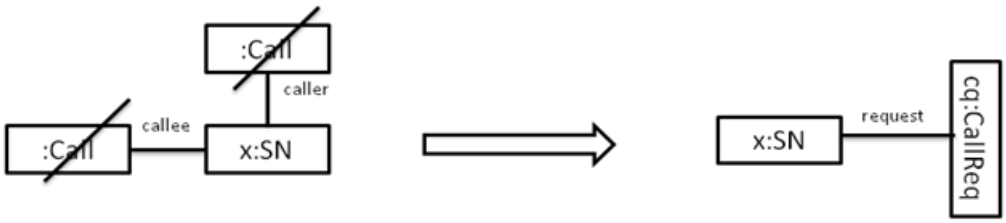

Figure 6.45: Call Request by SN

access. This rule uses two NAC conditions. They prevent a new call if an existing call is in progress. This rule uses CallReq to model the call initiation request.

3. **Call Request by SN**

   An SN can serve other clients, but retains the client functionality. The rule in Fig. 6.51 is used to register an SN's request for VoIP call.The use of NAC prevents the SN to initiate a call if an existing call is in progress.

6.2.9.2 Call Placement

The most crucial part of the VoIP P2P network is the voice conversation. Call placement is the second step in VoIP call. In this section, we discuss the type of calls that the model supports.

1. **Multiple SNs Routed VoIP Call**

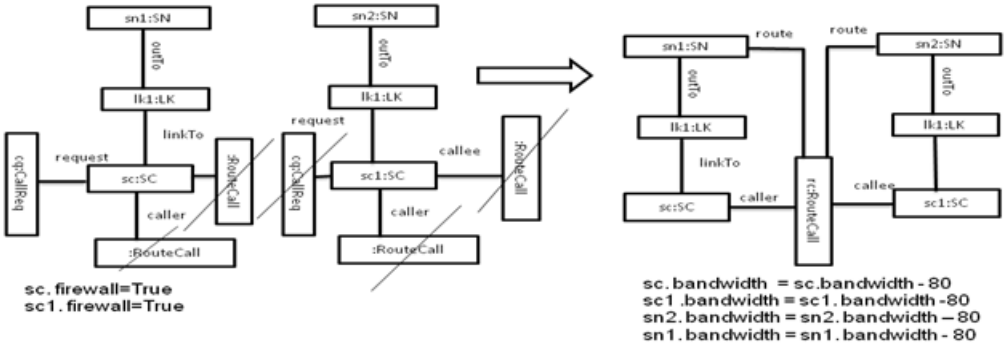

Figure 6.46: Multiple SNs routed VoIP call

The rule in Fig. 6.46 is used to complete call requests of a caller SC behind a firewall. The rule considers that callee is also behind a firewall. The NACs make sure that neither caller nor callee are already in a call. In this rule, caller and callee are both linked with different SNs. As both caller and callee are behind a firewall, the choice available for this call completion is the SN-routed call. VoIP call data will be transmitted and received through their respective SNs. We can see in the RHS of the rule that RouteCall is established between the caller and callee is routed through SNs. The route edge shows that calls are routed through SNs. In this VoIP Call, caller, callee and SNs use up bandwidth of 80 Kbps for this call.

2. **Single SN Routed VoIP call**

   The rule in Fig. 6.47 completes the call request of an SC behind firewall. The caller and callee are connected to the same SN. The call type will be RouteCall and data will be routed by a single SN. In this call, we still use the constraint of not being in a call. Once the call is established caller, callee and SN use up bandwidth of 80 Kbps.

3. **P2P Call Between SC behind Firewall and SN**

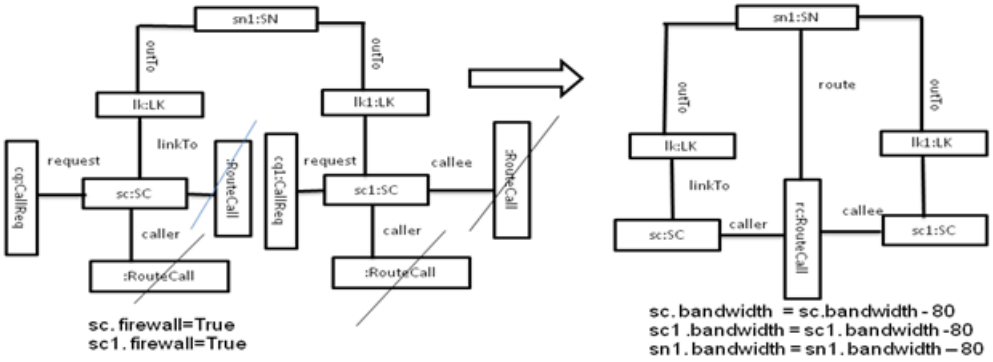

Figure 6.47: Singe SN routed VoIP call

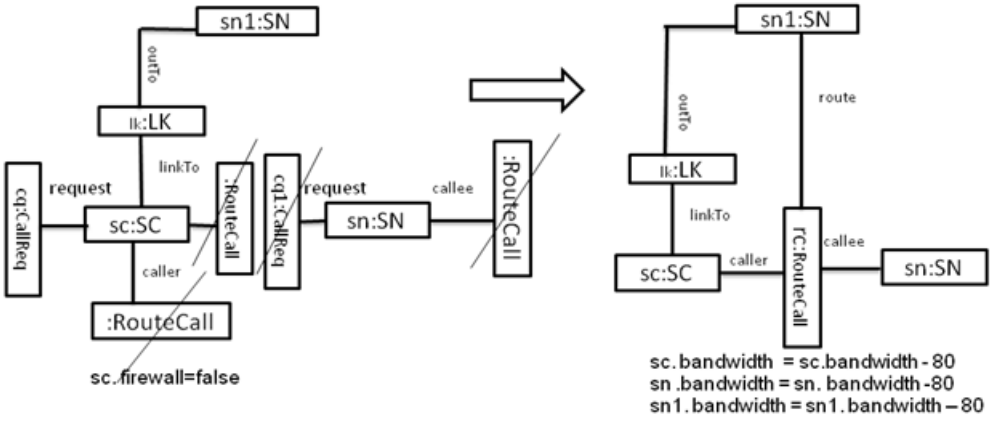

Figure 6.48: P2P Call between SC behind firewall and SN

The rule in Fig. 6.48 is used to model a call between an SC behind a firewall and an SN. In this call, the caller is SC whereas the callee is SN. The use of NACs makes sure neither caller nor callee is in a call at present. The call will be of type RouteCall as the caller cannot communicate directly with the SN. Instead, all call transmission will be through the respective host of the caller. The call is established in the RHS of the rule, and the three nodes pay the cost of 80 Kbps each.

4. **P2P Direct Call between SCs**

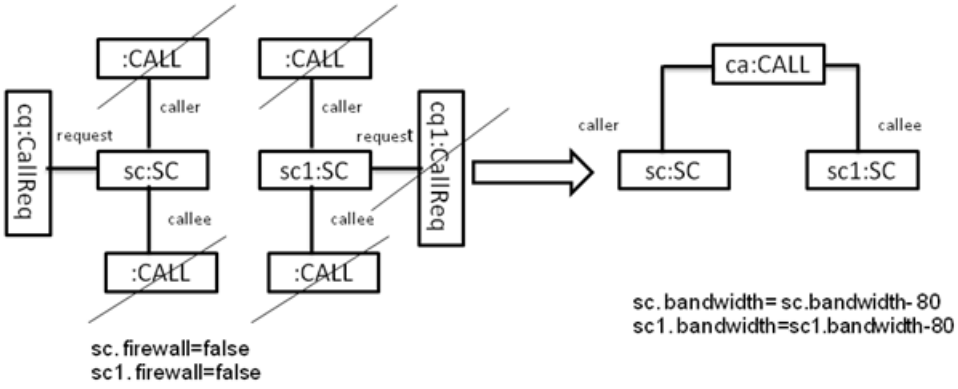

Figure 6.49: P2P direct call between SCs

The rule in Fig. 6.49 is used after the rule in Fig. 6.44 to complete a call between caller and callee. Both caller and callee are connected to the Internet through live IPs. The NACs make sure that neither callee nor caller is in a call at present. As RHS of the rule, a direct P2P call is established between Callers. Both caller and callee use up a bandwidth of 80 Kbps as the cost for maintaining this call.

5. **P2P Direct Call between SC and SN**

   The rule in Fig. 6.50 is used to show a direct P2P VoIP call between caller SC and callee SN. As caller and callee are connected to the Internet through live IPs, the direct call will be established as shown in the RHS of the rule through the node Call. Both caller and callee use up a bandwidth of 80 Kbps each.

6. **P2P Direct Call between SNs** The rule in Fig. 6.51 is used to model SN to SN VoIP Call.

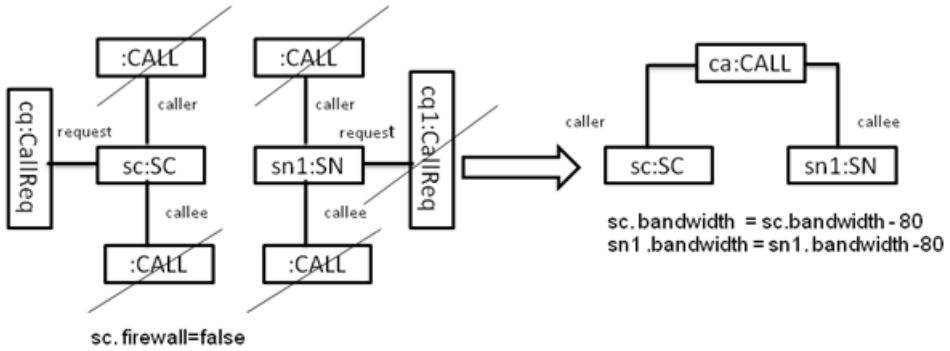

Figure 6.50: P2P direct Call between SC and SN

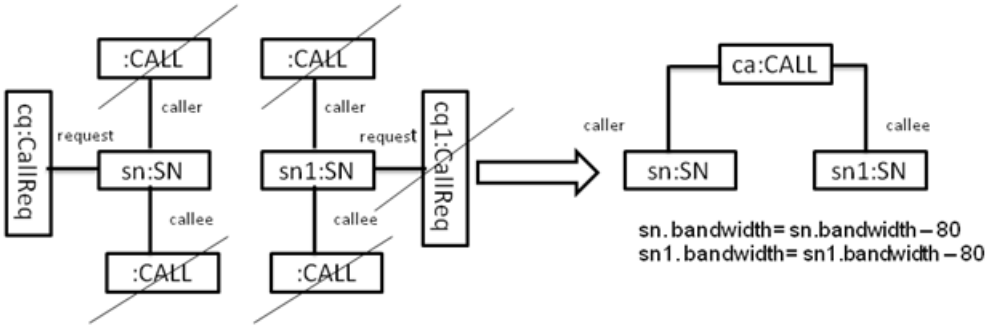

Figure 6.51: P2P direct Call between SNs

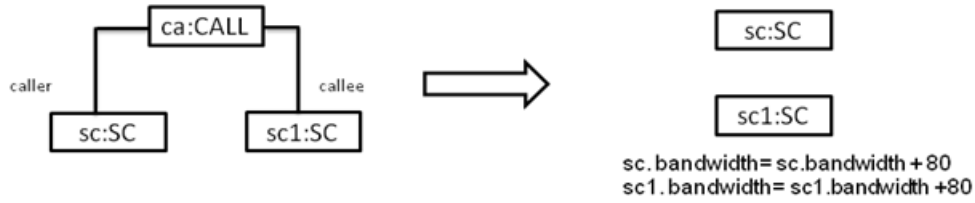

Figure 6.52: P2P direct VoIP call termination between SCs

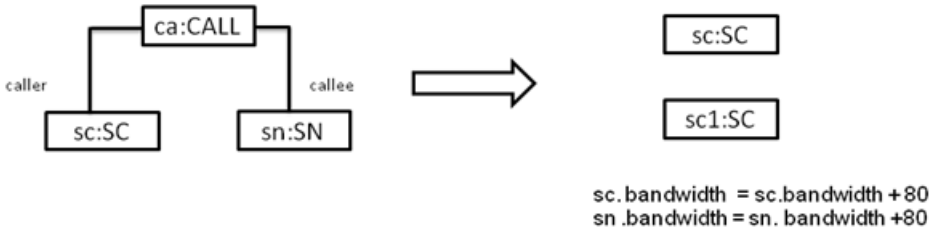

Figure 6.53: Terminate SC to SN P2P direct VoIP call

6.2.9.3 VoIP Call Terminations

Once calls are completed, they must be terminated. The following are the calls termination rules.

1. **P2P Direct VoIP call Termination Between SCs**

   The rule in Fig. 6.52 is used to terminate P2P VoIP call between the caller SC and callee SC by removing the call node. After termination, caller and callee update their bandwidth by 80 Kbps.

2. **P2P Direct VoIP call Termination Between SC and SN**

   The rule in Fig. 6.53 is used to terminate P2P VoIP direct calls between caller SC and callee SN. After termination, bandwidth cost is recovered.

3. **P2P Direct VoIP call Termination Between SN and SC**

   The rule in Fig. 6.54 is used to terminate a P2P VoIP direct call between caller SN and callee SC. After termination, bandwidth cost is recovered.

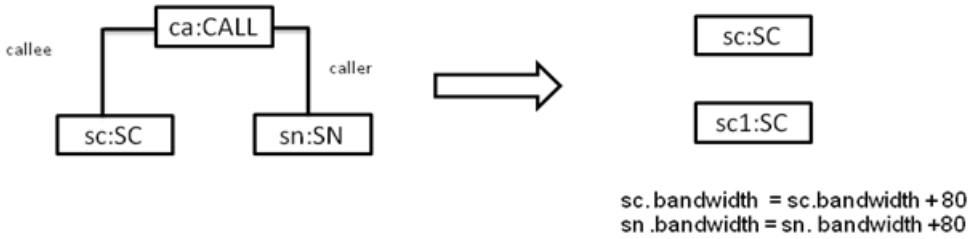

Figure 6.54: Terminate SN to SC P2P direct VoIP call

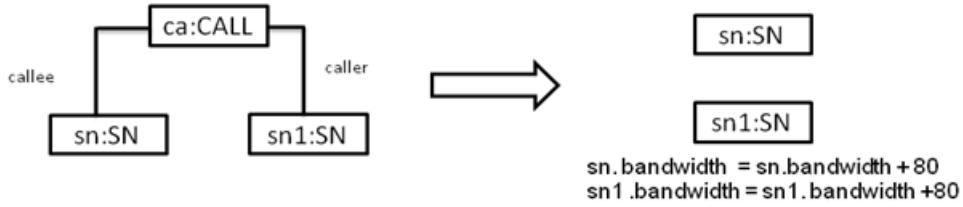

Figure 6.55: Terminate SN to SN P2P direct VoIP call

4. **P2P Direct VoIP call Termination between SNs**

   The rule in Fig. 6.55 is used to terminate a direct P2P VoIP call between SNs. Both SNs, after call termination, recover their bandwidth of 80 Kbps each.

5. **P2P Single SN Routed VoIP Call Termination**

   The rule in Fig. 6.56 is used to terminate an SN routed call established between caller and callee. As both caller and callee share the same common SN, their call is routed by a single SN. We can see in the RHS of the rule that RouteCall node is removed and bandwidth is recovered by the caller, callee and intermediate SN.

6. **Terminate Multiple SNs routed VoIP call**

   The rule in Fig. 6.57 is used to terminate a multiple SN routed call. In the LHS of this rule, a RouteCall is connected by route edges. These two

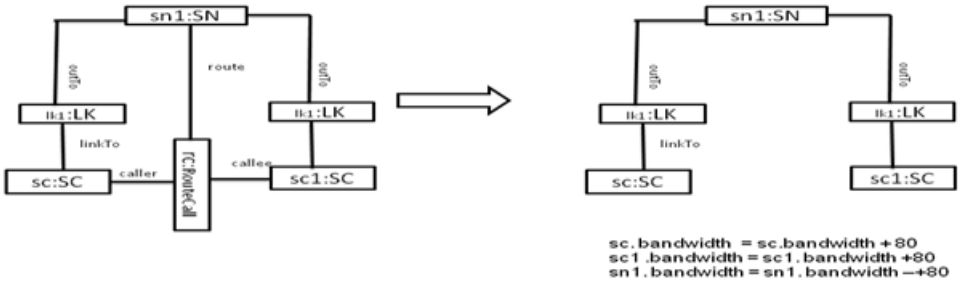

Figure 6.56: P2P single SN routed VoIP call termination

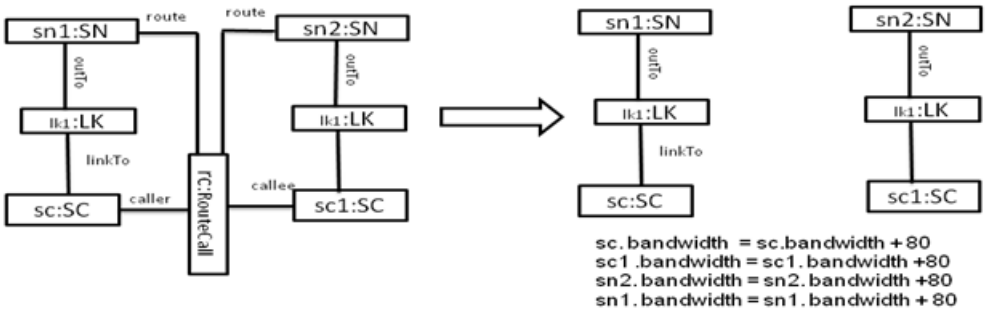

Figure 6.57: Terminate multiple SNs routed VoIP call

SNs are used to route VoIP call packets from caller to callee and back. As shown in the RHS of the rule, once the call is terminated the RouteCall is removed all participants recover their bandwidth compensation for this call.

7. **P2P Routed VoIP call Termination between SC and SN**

The rule in Fig. 6.58 is used to terminate a call between a caller SC behind a firewall and an SN connected through live IP. In this type of call, the call packets travel through the host of the caller SC. Once this call is terminated, the bandwidth cost is recovered by caller, callee and host of the caller.

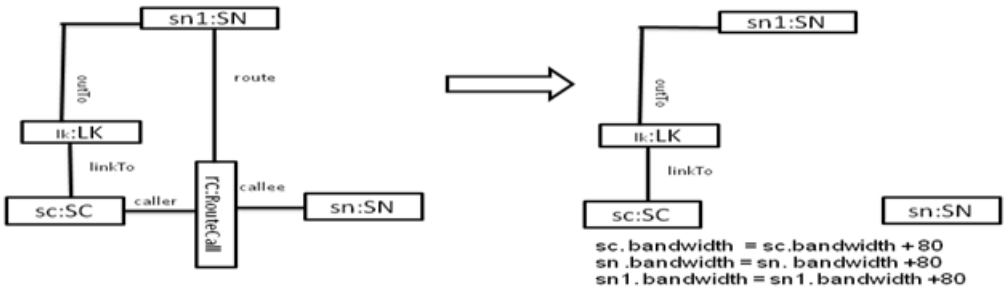

Figure 6.58: Terminate SC to SN local host routed VoIP call

6.2.10 Reconfiguration

If an SN departs from the network either cooperatively or by crashing, the SCs can reconfigure to connect to a new SN based on one of the connecting approaches discussed above. If the SN intends to leave the network by cooperating, it informs all other overlay SNs and dependent SCs by setting the *exit* attribute to *true*. This enables the involved SNs and SCs to reconfigure quickly to new SNs. Once all load is transferred, the SN can leave the network.

6.2.10.1 Link to a New SN when Existing One leaves

In Section 6.2.3 of this chapter, we discussed that SC has four choices to find a suitable SN. In case of reconfiguration the same choices will be used. In this section we explain the reconfigurations for relinking to a new SN.

1. **Reconfigure and link to a New Random SN**

   The rule in Fig. 6.59 is used to re-link an SC to a new SN. The new SN is selected randomly based on the constraint of bandwidth being greater than 256 Kbps. The LHS of the rule use two NAC. The first makes sure that the LK node is not linked with any other SN in the current graph. The second NAC makes sure thatthe LK node is not having an existing outTo

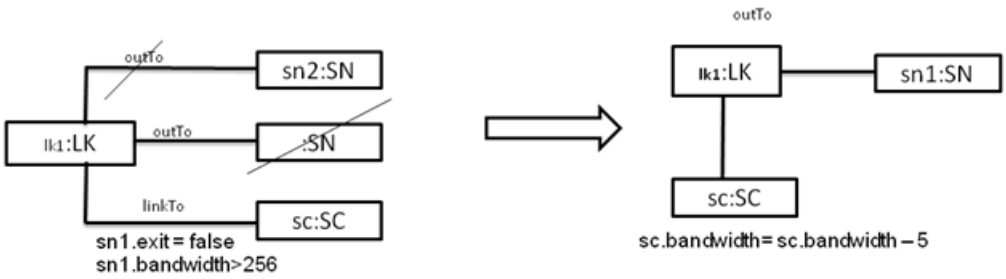

Figure 6.59: Reconfigure and link SC to a new SN

edge with the newly selected SN. Once these conditions evaluated as true, the SC can re-link with a new SN. The new SN compensates a bandwidth of 5 kbps for this newly linked SC.

2. **Reconfigure and link to a New SN(based on Bandwidth and Latency)**

   The LHS of this rule uses four NACs and two attribute constraints. The NACs make sure that SC is currently disconnected from The SN, it has no existing edge with the newly selected SN, and The SC has neither sent nor received any ping packet yet. The attribute constraints make sure that the newly selected SN has bandwidth greater than 256 Kbps and SN is not intending to leave the network . Once all these NACs and constraints are evaluated, the rule sends a time stamped packet to the newly selected SN. After this step the reply rule and link rules are used for re-linking this SC.

3. **Reconfigure and Link to a New SN based on Bandwidth and Region**

   The rule in Fig. 6.61 uses two NACs. The first makes sure that the SC is currently disconnected and node LK is not having an outTo edge with

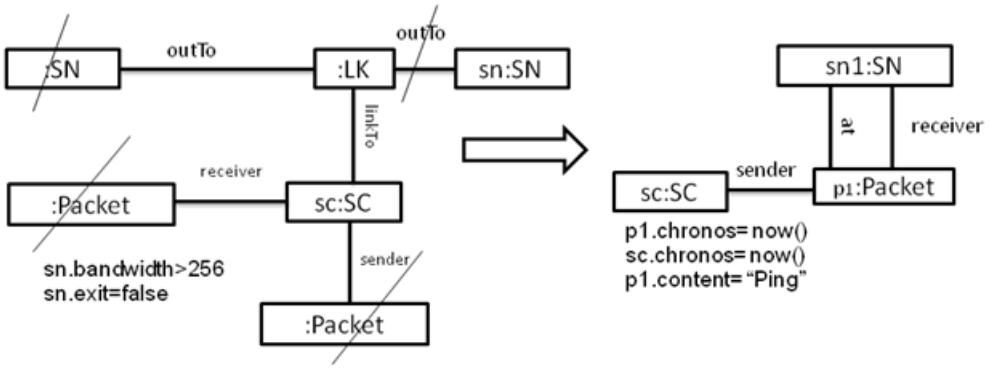

Figure 6.60: Reconfigure and send a time stamped packet to a new SN

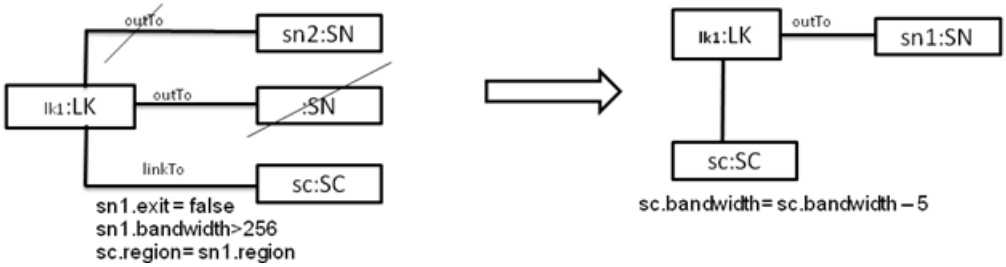

Figure 6.61: Reconfigure to link to a new SN based on bandwidth and region

any of the SN, in the current graph. The second NAC makes sure that the SC is not having an existing edge with the newly selected SN. Apart from NACs, this rule uses three attribute constraints. The first attribute constraint makes sure that the new random SN has a bandwidth greater than 256 Kbps. The second attribute makes sure that it is not intending to leave the network. The last attribute makes sure that the disconnected SC and the new SN have same region number. If all these conditions are true this SC is linked with the SN.

4. **Reconfigure and link to a New SN based on bandwidth and time spent**

   The rule in Fig. 6.62 is used to re-link a disconnected SC to a new SN

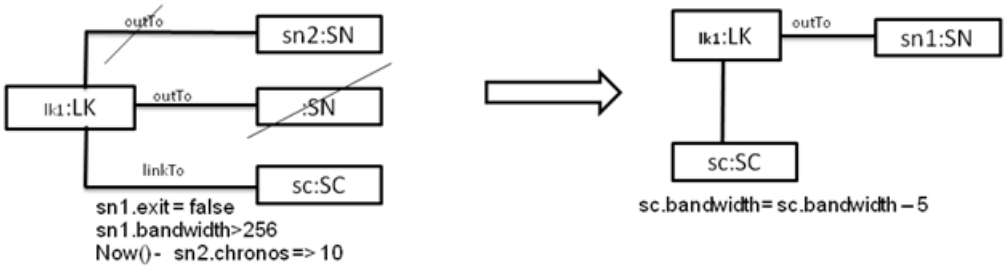

Figure 6.62: Reconfigure link to a new SN based on bandwidth and time spent

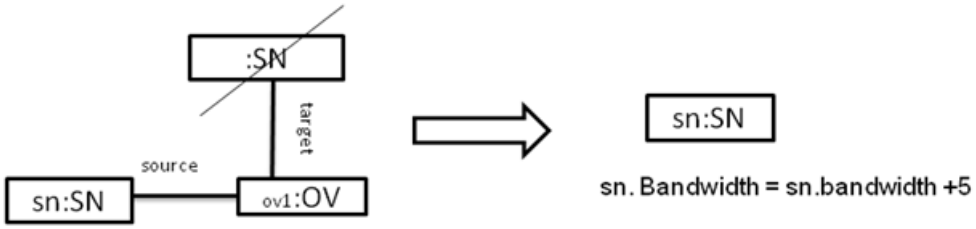

Figure 6.63: Reconfigure and remove overlay target edge

based on the time spent by the SN in the network. This rule uses the same bandwidth and exit attribute constraint they other reconfiguration rules. Only one additional constraint is added. This makes sure that the newly selected SN must have served the role for more than 10 minutes.

6.2.10.2 Recover Bandwidth due to Crashing SN (Target)

The rule in Fig. 6.63 is used to enable an SN to recover its overlay connections. The SN in this rule detects the crashing of the SN in the overlay and retracts the overlay link and bandwidth.

6.2.10.3 Recover Bandwidth due to Crashing SN (Source)

The rule in Fig. 6.64 is used to enable an SN to recover and complete its overlay network. This rule uses a single NAC that checks that OV is not connected with any SN in the network. This rule deletes the link OV and recovers its bandwidth

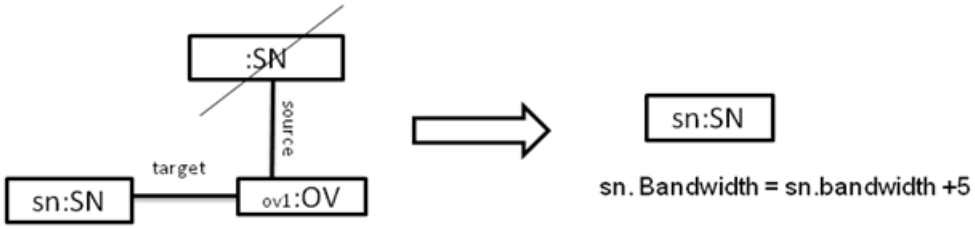

Figure 6.64: Reconfigure and remove overlay source edge

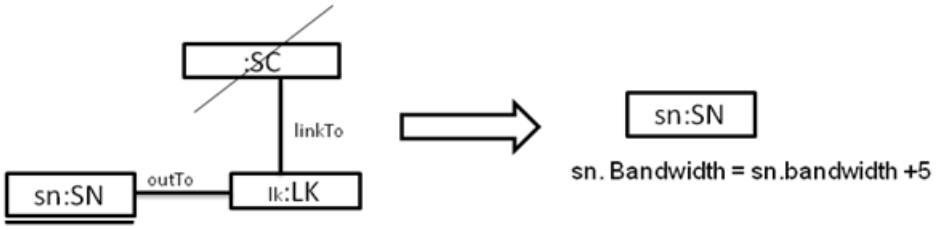

Figure 6.65: Reconfigure and remove linking edge with SC, and update bandwidth

of 5 Kbps.

6.2.10.4 Recover Bandwidth due to Crashing of SC

Similarly, an SC may also leave a network by crashing, and as result its SN will need to reconfigure. This rule enables the SN to retract the dependency link and update its bandwidth. The rule in Fig. 6.65 uses a single NAC, which makes sure that the SN is not linked with the SC through LK. The rule removes node LK and update its bandwidth.

6.2.11 Background Traffic

As the VoIP client is sharing The Internet bandwidth with other applications it will not always have guaranteed bandwidth. In order to model increase and decrease in the availability of bandwidth, we use a random function to randomise the bandwidth during the simulation. In a second approach, we subtract and add

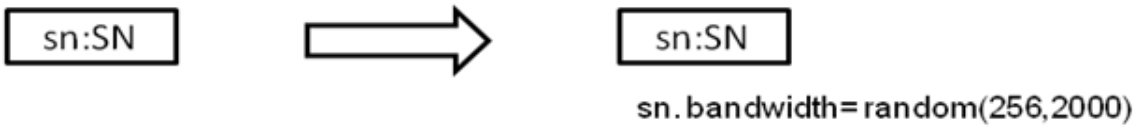

Figure 6.66: Random other traffic at SN

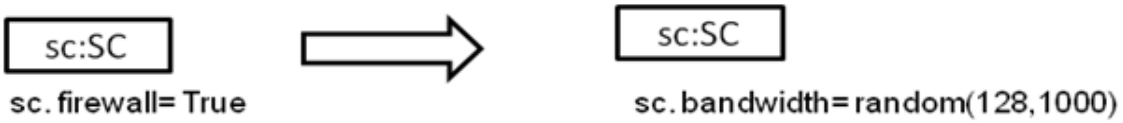

Figure 6.67: Random other traffic at SC behind firewall

a uniform value to create the effect.

6.2.11.1 Random Traffic Modelling

1. **Random other Traffic at SN**

   The rule in Fig. 6.66, is used to model the background traffic generated by other applications while the VoIP client is participating in the network. This rule selects an SN in the current graph and assigns it a random bandwidth between 256 Kbps and 2 Mbps.

2. **Random Other Traffic at SC (behind firewall)**

   The rule in Fig. 6.67 is used to model a background traffic generated by a Skype client, while behind a firewall. This rule selects an SC which has firewall and randomizes its bandwidth between 128 Kbps and 1 Mbps.

3. **Random Other Traffic at SC (behind firewall**

   The rule in Fig. 6.68 is used to model background traffic generated by a Skype client that has live Internet connection. This rule selects an SC and

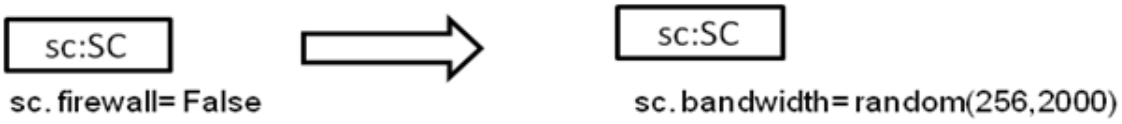

Figure 6.68: Random other bandwidth at SC

Figure 6.69: Increase bandwidth at SN

randomizes its bandwidth between 128 Kbps and 2 Mbps.

6.2.11.2 Uniform Traffic Modelling

1. **SN Transmitting and Receiving Uniform other Traffic**

   The rule in Fig. 6.69 is used to model the background traffic generated by other applications while super peer SN is participating in the network. This rule selects an SN and then subtracts a uniform value of 80 Kbps. It also uses the rule in Fig. 6.70 to increase the bandwidth value by adding 80 Kbps.

2. **SC Transmitting and Receiving Uniform other Traffic**

   The rule in Fig. 6.71 is used to model the background traffic generated by other applications while client SC is participating in the network. This rule selects an SC and then subtract a uniform value of 80 Kbps. It also use the

Figure 6.70: Reduce bandwidth at SN

Figure 6.71: Increase bandwidth at SC

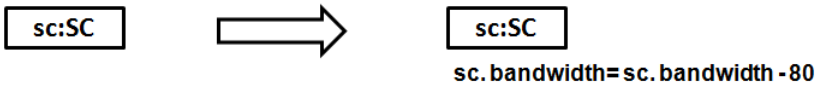

Figure 6.72: Reduce bandwidth at SC

rule in Fig. 6.72 to increase the bandwidth value by 80 Kbps.

6.2.12 Probe Rules for Collection of Statistics

Probe rules are used to count the number occurrence if certain pattern during a simulation run. These pattern are encode into the precondition of the rules, while the postconditions are empty. In the model , an SC can be connected to a single SN, but as an alternative approach it can be connected to multiple SNs. This number of connections also changes the structure of the probe rules. In this section, we present two sets of probe rules. The first set will be used with the single-link approach whereas the other will be used with the multiple-links approach.

6.2.12.1 Probe Rule for Single Link

In order to collect the statistics of the simulation, rules with empty postconditions are used as probes. Each probe rule returns the number matches in the current graph for each state of the transformation system.

Figure 6.73: Number of SCs

Figure 6.74: Number of SNs

1. **Number of SCs**

   This rule in Fig. 6.73 is used to count the number of SCs in the current graph.

2. **Number of SNs**

   This rule in Fig. 6.74 is used to count the number of SNs in the current graph.

3. **Number of SCs linked with SN**

   The rule in Fig. 6.75 is used to count the number of SCs linked with SNs in the network.

4. **Number of SCs Happy with current SN**

   The rule in Fig. 6.76 is used to count the number of SCs that are currently linked with SNs, they have a spare bandwidth more than 1 Mbps.

5. **Number of SCs re-linked with new SN**

Figure 6.75: Number of SCs linked with SN

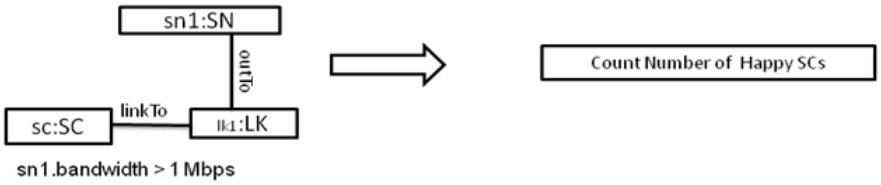

Figure 6.76: Number of SCs happy SCs with current SN

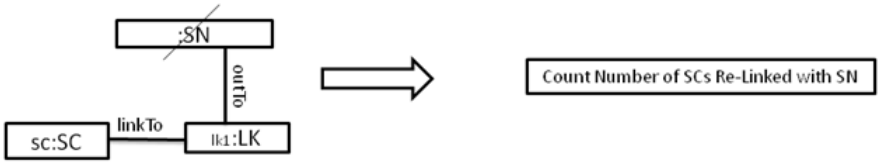

Figure 6.77: Number of SCs reconfigured with new SN

The rule in Fig. 6.77 is used to count the number of SCs that have lost connection with their current host in the graph and are looking to re-link to a new SN.

6. **Number of P2P Direct Calls**

   The rule in Fig. 6.78 is used to count the number of P2P direct calls.

7. **Number of SN-Routed Calls**

   The rule in Fig. 6.79 is used to count the number of SN routed Calls.

8. **Number of Calls Disconnected Due to Crashing of Callee**

   The probe rule in Fig. 6.80 is used to count the number of P2P direct calls that are disconnected due to crashing of callee.

9. **Number of Calls Disconnected Due to Crashing of Caller**

Figure 6.78: Number of P2P direct calls

Figure 6.79: Number SN-routed calls

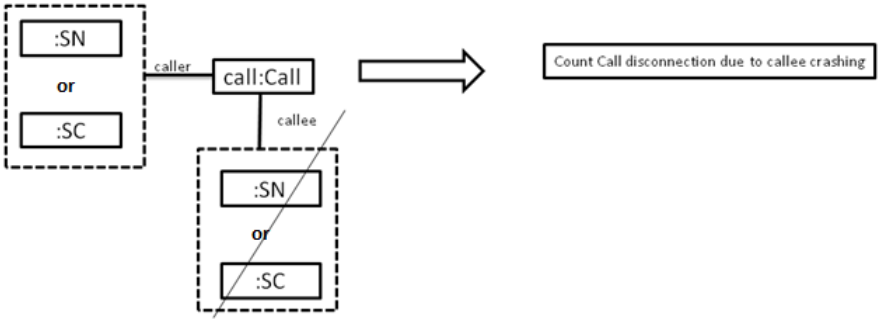

Figure 6.80: Number P2P direct calls disconnection due to callee

The probe rule in Fig. 6.81 is used to count the number of P2P direct calls that are disconnected due to crashing of caller.

10. **Number of Routed Call Disconnection due to SN**

    The probe rule in Fig. 6.82 is used to count the number of routed calls that are disconnected due to crashing of SN.

6.2.12.2 Probe Rule For Multiple Links

1. **Number of SCs linked with SNs**

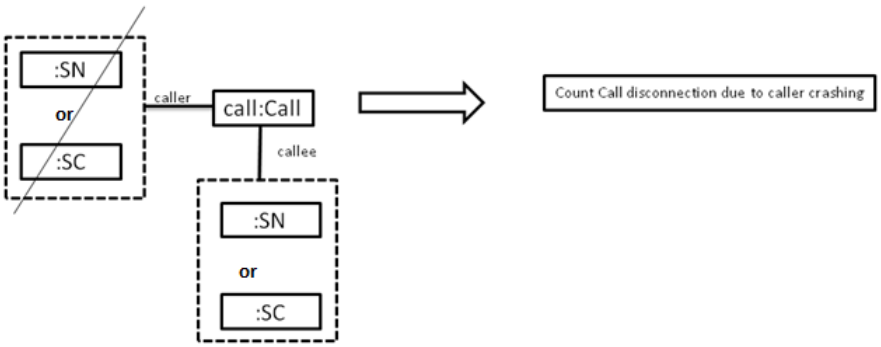

Figure 6.81: Number P2P direct calls disconnection due to caller

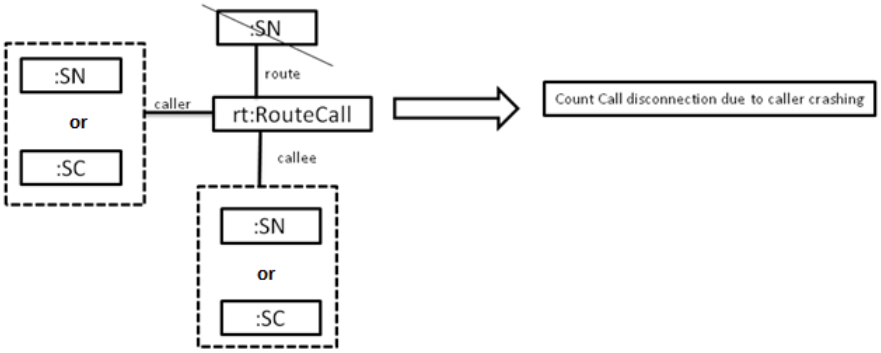

Figure 6.82: Number of routed call disconnection due to SN

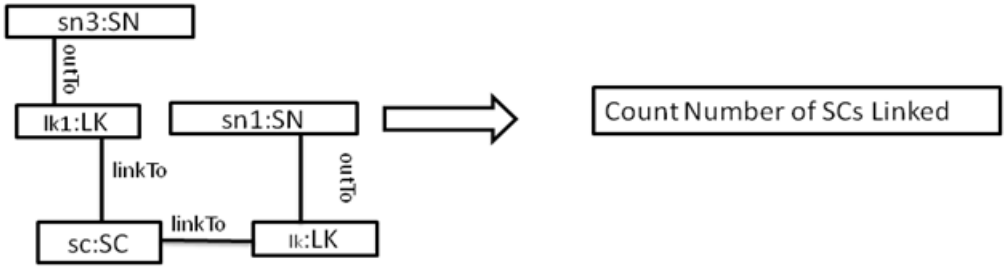

Figure 6.83: Number of SCs linked SNs

The probe rule in Fig. 6.83 is used to count the number of SCs that are linked with multiple SCs.

2. **Count Number of SCs Happy with SNs**

   The rule in Fig. 6.84 is used to count the number of SCs that are happy with their current SNs. This is the case the SN has a current bandwidth of 1 Mbps.

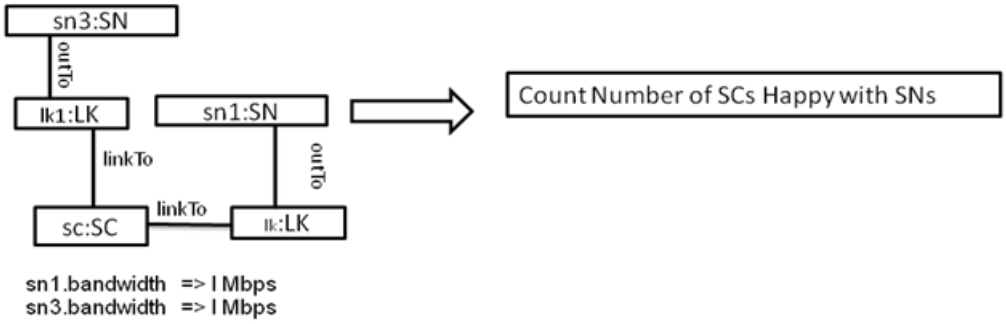

Figure 6.84: Probe rule counts number of happy SCs

6.3 Summary

In this chapter, we defined a type graph as a model for P2P VoIP like protocol. The type graph is based on the discussion of the case study. Further, we also presented the transformation rules for modelling the behaviour of the network, including all it's variants. Typed graph and transformation rules will be converted into VIATRA2 the coming chapter for use in simulation.

Chapter 7

Simulation Tool Support

In order to implement the approach proposed in this thesis, we require tool support for the simulation of graph transformation. In this chapter we will discuss the Graph-based Stochastic Simulator (GraSS) which has been developed alongside the research reported in this thesis. In particular, we will explain how the tool is used to perform the simulations required to evaluate our P2P protocols, including the simulation parameters used and the statistics produces as a result. We will also give a brief overview of the tool's architecture. GraSS requires a type graph as well as GT rules in the syntax of VIATRA2, a model transformation language and tool. This chapter will give an overview of the syntax and semantics and explain the conversion of our graphical rules into this textual model transformation language.

7.1 Graph-based Stochastic Simulation (GraSS)

GraSS is a stochastic simulator [162] based on the algorithm [87] for generalised stochastic graph transformation described in Chapter 2. GraSS provides various forms of statistical computations such as confidence intervals, averages and mean, etc. The underlying stochastic model is that of a semi-Markov process allowing for exponential and log-normal distributions. The tool has been developed in Java-Eclipse, as a plugin of a graph transformation engine called VIATRA2 [16].

GraSS takes as input an XML file with the rule names and definitions of the distributions associated with transformation rules, as well as the list of rules with empty postconditions that are to be used as probes. Additional parameters needed for a simulation run are provided to GraSS as part of the VIATRA2 model, which will be explained in the coming sections.

In GraSS a GTS is represented as a VIATRA2 model, consisting of the model space containing the current graph and the transformation rules. The model represents a discrete event system, given by the set of all reachable graphs as states and transformation steps (pairs of rules and matches) as events. In GraSS each event is associated to a random variable that represents the scheduled delay as specified by a continuous cumulative distribution function (CDF).

GraSS collects statistics by means of transformation rules used as probes (rules with empty postconditions). The statistics are provided by a Java library for stochastic simulation (SSJ) TallyStore class reports [93].

7.1.1 Simulation Parameters

The experiment can be structured as a sequence of variations, each defined by a batch of simulation runs. However, all runs of a batch share the same stochastic parameters. Runs from different batches can vary, either in stochastic assignment or in maximum length. In GraSS [162] the statistics are produced for each run, batch and experiment. In order to run simulation experiments, the following are the inputs that have to be provided to the tool, either as a part of the model space or in a separate XML file. The `parameter.xml` file used in this thesis is shown in Fig. 7.2.

1. **Graph Transformation System**

   The metamodel representing the type graph of the GTS, as shown in Fig. 7.1. The initial model (start graph) as shown in Fig. 7.1, consisting two nodes RS and SN. The transformation rules, converted from the graphical rules of our model.

2. **Probability Distribution and Rate**

   The parameter `inputXML`(string) is used to specify the name of the XML file where rule names, probability distributions and rates are stored. The Fig. 7.3 shows a section of the XML file used in this thesis.

3. **Rule Set**

   The parameter `ruleSet`(string) is used if we want to simulate a single model with different sets of transformation rules. Assigning the rule set name will enable the simulator to use only that set of rules.

4. **Length of Simulation Run**

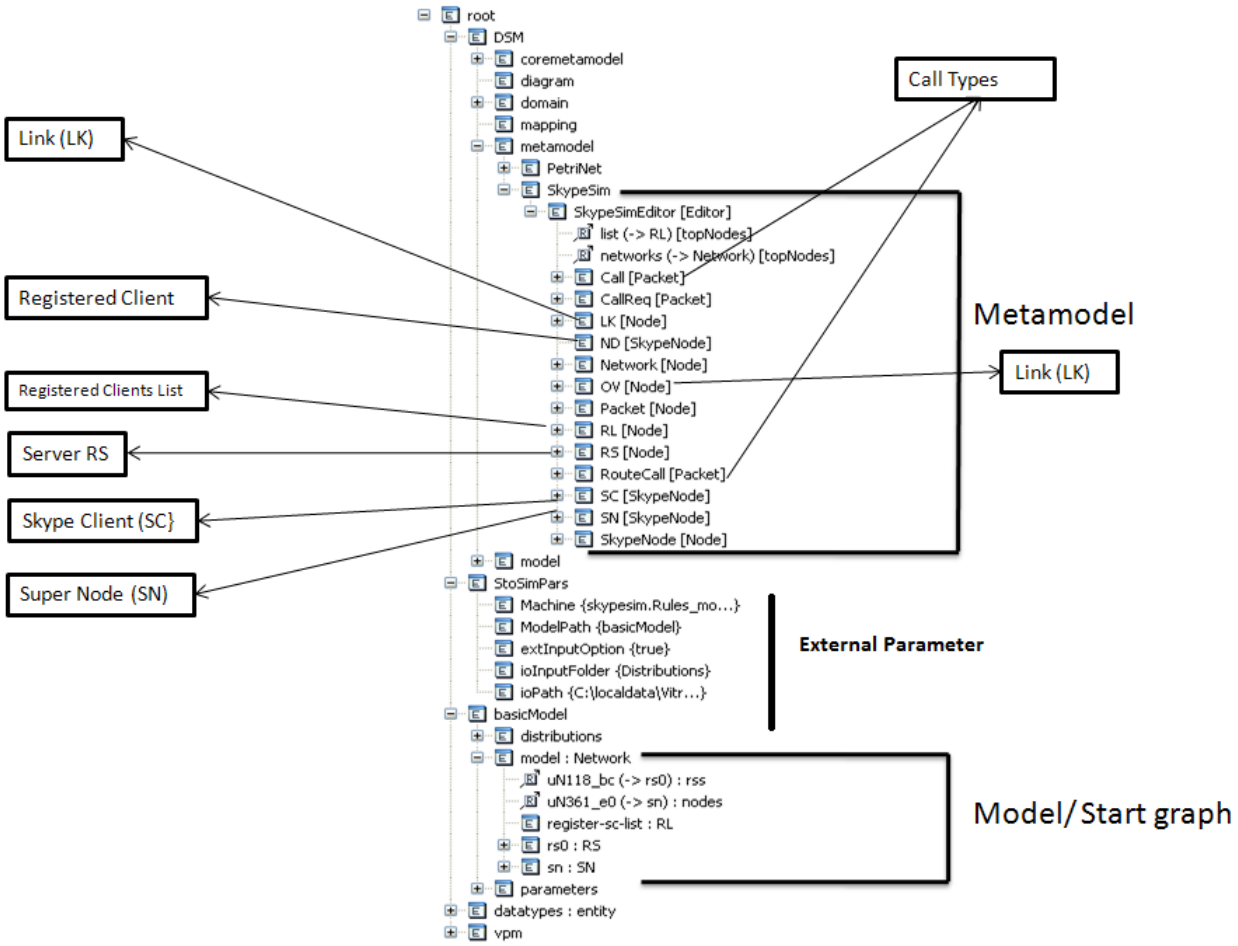

Figure 7.1: Skype VPM Model Space

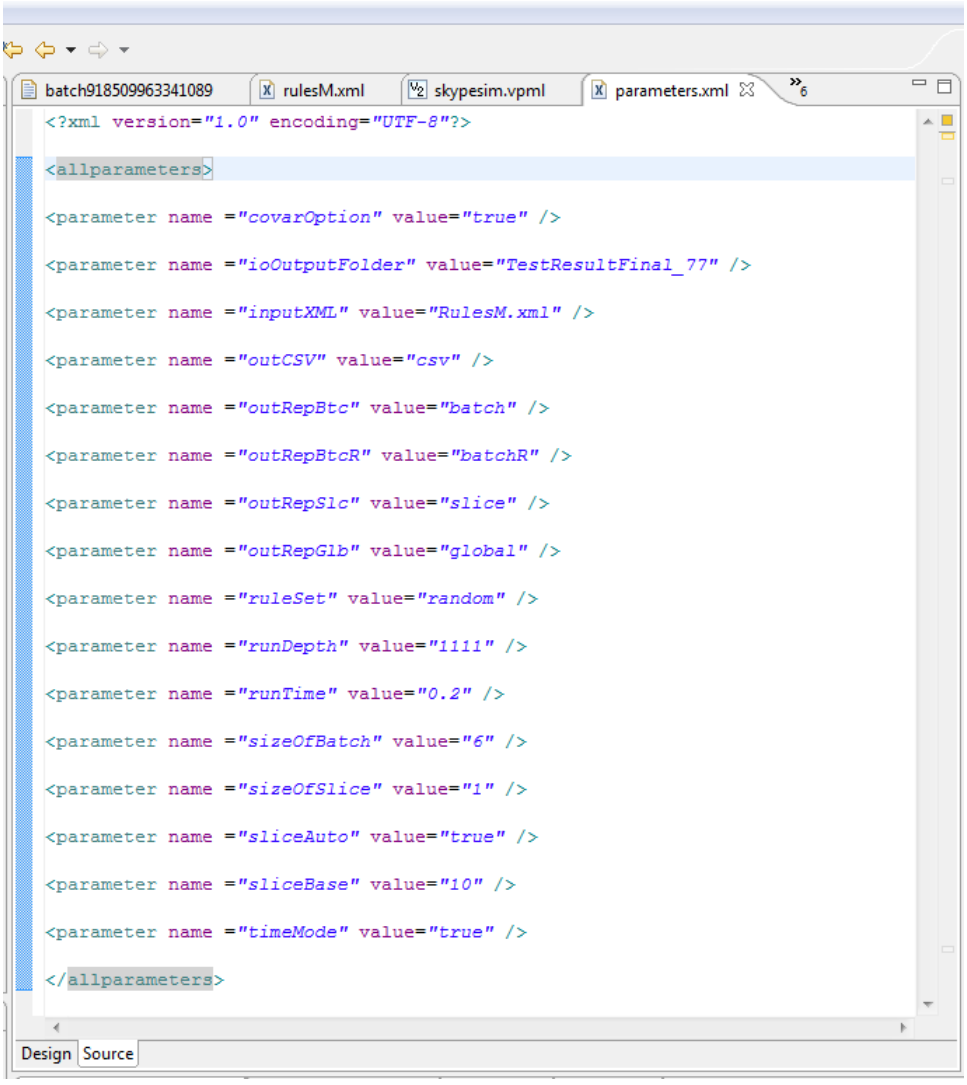

Figure 7.2: Internal parameters

skypesim.vpml | batch684970688206906052out | rulesM.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<allrules>

<ruleset name="random">

<rule name="User_Go_Online_with_Random_Firewall" type="action">
<event name="default" type="exp">
<rate value="3.0"/>
</event>
</rule>

<rule name="Rule_Link_SC_to_SN" type="action">
<event name="default" type="norm">
                <mean value="0.000347" />
                <variance value="2.0" />
</event>
</rule>

<rule name="Rule4_Reconfigure_Link_SC_to_link_back_SN" type="action">
<event name="default" type="norm">
                <mean value="0.000173" />
                <variance value="2.0" />
</event>
</rule>

<rule name="Rule5_Promote_at_Registration_Super_Node" type="action">
<event name="default" type="norm">
                <mean value="0.100" />
                <variance value="2.0" />
</event>
</rule>

<rule name="Rule6_Promote_at_Linked_Super_Node" type="action">
<event name="default" type="norm">
                <mean value="0.100" />
                <variance value="2.0" />
</event>
```

Figure 7.3: XML file with GT rules and probability distribution

(a) The parameter `runDepth`(integer) is used to show the number of steps that we want to run the simulation for. Once this number of events occur in the network the simulation will stop.

(b) The parameter `timeMode`(`boolean`) is used to set the simulator to time mode. In this mode each run is limited by a maximum simulation time, otherwise it is limited by a maximum number of steps.

(c) The parameter `runTime` is used with time mode and it provides the length of the simulation run. It's value is interpreted based on the scale used in the probability distribution. If the rates are assigned based on a per-day basis time, a simulation time of 1 represents 24 hours.

(d) The parameter `sizeOfBatch`(`integer`) specifies the number of runs for each batch of experiment, this parameter specify how many times we want to run the simulation with a single sensitive rule value. This helps to obtain an average value and to find confidence interval.

5. **Content and Organisation of Output**

(a) The `outRepBtc` requests a report on the probe rules, including detailed information of minimum, maximum and averages. This report also provides confidence intervals for each probe rule.

(b) The `outRepBtcR` requests a report on the global statistic of all GT rules, showing the minimum, maximum and average number of matches they had during the simulation.

(c) The `outRepSlc` requests a report for each variation and each slice separately.

(d) The `outRepGlb` requests a report on the global statistics during the simulation, such as if the simulation was running on time mode, the report shows the number of steps that occur.

7.2 This Thesis and GraSS

GraSS has been developed by Polo Torrini during his time on the SENSORIA project at Leicester [162]. The research reported in this thesis has provided a test bed for evaluating and debugging of the tool. We also performed tests in order to validate the scalability of GraSS. Using the time mode we were able to simulate a model for 48 hours on a standard PC with a Core2Duo 2 GHZ processor and 4 GB RAM with 32 bit Windows XP operating system.

During these experiments the tool has managed to simulate the model up to 35000 steps, upon which almost 90% of the memory was in use. The network had 3486 clients and 600 super peers. In other experiments we fixed the number of clients in the model to 1000 and ran the simulation based on steps so as to observe the time it took to complete. The results are presented in Fig. 7.4.

This thesis also contributed sample models including detailed comments and parameter files to enable new users to test the tool.

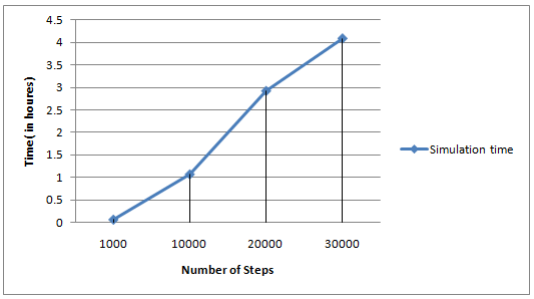

Figure 7.4: Steps Vs time in Grass

7.3 Mapping To VIATRA2

In this section, we are going to convert the type graph and the transformation rules presented in Chapter 6 to executable form in order to be used by GraSS. In the first section, we convert the type graph into a Visual Automated Model Transformations(VIATRA2) model space. We also define the start graph in the model space, along with the parameters for the simulation. In the second part of this chapter, we convert a single GT rule into a VIATRA2 rule. We explain this mapping based on the background presented in Chapter 7. Later in this thesis, the model space and transformation rules in VIATRA2 will be used by GraSS for stochastic simulation and analysis.

7.3.1 VIATRA2 Model Space

Fig. 7.1 shows the model space for the Skype-like protocol. The metamodel is the conversion of the type graph presented in Chapter 6. The model elements SC, SN and ND and associated relations have kept the same names for better understanding. Fig. 7.1 also provides the start graph where the GT rules will be applied. The start graph in Fig. 7.1 shows a single registration server RS, a single super peer SN and a list of registered users. This list will increase as a new

registrations occur, presently the model has 1000 registered users.

7.3.2 The VTCL language

VIATRA2 is a model transformation tool based on graph transformation. Transformation rules in VIATRA2 consist of several constructs that together form an expressive language for developing both model to model transformations and code generators. Graph patterns (GP) define constraints and conditions on models, graph transformation (GT) rules support the definition of elementary model manipulations. The language that is created to implement all these concepts is the Viatra Textual Command Language (VTCL). This language is textual and currently it does not have any graphical editor for development of rules. All the rules needs to be coded.

Graph patterns Graph patterns are the atomic units of model transformations. They represent conditions (or constraints) that have to be fulfilled by a part of the model space in order to execute some manipulation steps on the model. A graph model or part of the model can satisfy a graph pattern, if the pattern could be matched to the subgraph of the model using an *incremental graph pattern matching* technique presented in [164]. The syntax for the definition of patterns is as follows.

```
pattern (name of pattern (formal parameters)) =
{
 Pattern body
}
```

Patterns are defined using the *pattern* keyword. Patterns may have parameters that are declared after the pattern name. The basic pattern body contains model element and relationship definitions. The keyword *neg* identifies a sub-pattern that is embedded into the current one to represent a negative condition for the original pattern. To explain the pattern we consider an example pattern below:

```
1  pattern example(X,S,BW,RG) =
2   {
3
4     SN(X);     // X is variable of type SN//
5     RS(S);
6           neg pattern negcondition(X,Y) =
7         {
8           SC(X);
9         RS(S);
10        RS.register(RG,S,X);
11
12       }
13    find SN_Bandwidth(X,BW);
14           check((toInteger(value(BW)))>256);
15    }
```

Listing 7.1: Example pattern

The negative pattern in the example can be satisfied, if the node X and Y are connected through an edge of type *register*. There are also *check* conditions that are Boolean formulas which must be satisfied in order to make the pattern true. In our example, we check that SC has current bandwidth more than 256 Kbps. Another unique feature of the VTCL pattern language among graph transfor-

mation tools is that negative conditions can be embedded into each other at an arbitrary depth, as negations of negations. In such case a expressiveness of such patterns converges to first order logic [132].

Pattern calls, OR-pattern, recursive pattern In VTCL, a pattern may call another pattern using the keyword *find*. This feature enables to use an existing pattern in a new pattern. A once written pattern will be reused whenever required later in the code. Similarly, alternate bodies for a pattern could be defined by simply creating multiple blocks after the pattern name and parameter definition, and subsequently connecting them with a keyword *or*. In this case the pattern is fulfilled if at least one of the bodies of the pattern is fulfilled. By using pattern call and OR together, recursion can be formulated in which a pattern can call itself. In this scenario one of the bodies contains stop condition whereas the other activates the recursive call. This can be explained with help of the example given below. In this pattern it is ensured that the selected SN has either source edge with node OV or target edge with OV.

```
1  pattern Check_OV_Links_with_SN(Y)=
2   {
3    OV(Z);
4    SN(Y);
5    OV.source(LKI,Z,Y);
6   }
7   or
8   {
9    OV(Z);
10   SN(Y);
11   OV.target(OVT,Z,Y);
```

```
12    }
```

Listing 7.2: Example of Or

Semantics of graph patterns When a graph pattern is activated using the *find* keyword, this activation means that a sub-situation for the free parameter of the specified graph pattern has to be identified such that it satisfies the pattern, in case there is no defined value. If there are bound variables which are passed as parameters, they are considered as additional constraints, and they remain bound throughout the pattern matching process. By default in VIATRA2 the free variable will be substituted by existential quantification, which simply states that only one matching is generated. However, if a variable is universally quantified using the construct *forall*, the matching will be performed on all possible values of the given variable.

7.3.3 Transformation Rules in VIATRA2

As the graph pattern defines the logical conditions or formulas on the model, the manipulation of models is achieved using graph transformation rules. This whole process in VIATRA2 is based on graph patterns for defining the application criteria for transformation steps. The application of a GT rule on a given current graph (model space) replaces an image of its precondition (LHS) pattern with an image of its postcondition (RHS) pattern. A GT rule is composed of a precondition, a postcondition and an optional action part. The syntax of a GT rules in VIATRA2 is given below.

```
1
```

```
gtrule (Name of rule) (in/out (i/o parameters)) =
{
        precondition {
                        pattern definition
                        pattern call
                }
        postcondition {
                        pattern definition
                        pattern call
                }

        action     {
                        pattern definition
                        action pattern call
                 }

}
```

Listing 7.3: Syntax of transformation rules

The VTCL language allow the use of popular notations for construction of transformation rules. The above syntax is a combination of traditional and new specifications for presenting transformation rules in VIATRA2. The precondition is performing the core function of the GT rule. It defines the conditions for the of application of a rule. The execution of a precondition means finding a pattern in the start graph. This represents the LHS of the transformation rule. The postcondition pattern describes what conditions should hold as a result of applying a GT rule. The result of a rule can be identified as the difference between the precondition and the postcondition. Further, the postcondition allows three different

operations on the graph. The first is the preservation of an input parameter of the precondition. The second is deletion, if an element or an input parameter of the postcondition does not appear in the pattern itself, then the matching model element is deleted. The last task is the creation. If a variable appears in the postcondition which is not an input variable of the post condition, then a new model element is created.

The action part to the GT rule can be used without a postcondition to perform the same task as would be performed by the postcondition with additional flexibility of user defined functions or methods. A GT rule can create model elements, delete existing ones and update values of attributes. In this thesis, we are using GT rules with preconditions and actions.

7.3.4 GT Rule Conversion To VIATRA2

In order to explain the process of converting a GT rule into a VIATRA2 rule, we have selected a rule which links an SC with an SN. This rule captures features such as NAC, attribute constraints and updates. We reproduce the rule from Fig. 6.8 in Fig. 7.5, marking the patterns with numbers. These patterns will be converted separately into VIATRA2 pattern. The objective is that they could be reused if another GT rule uses the same patterns. The complete view of the VIATRA2 code is shown in List. B. All other rules have been placed in Appendix B.

```
1 pattern SN_overlay-index_RS(S,Y,RG) =
2   {
3     SN(Y);
4     RS(S);
```

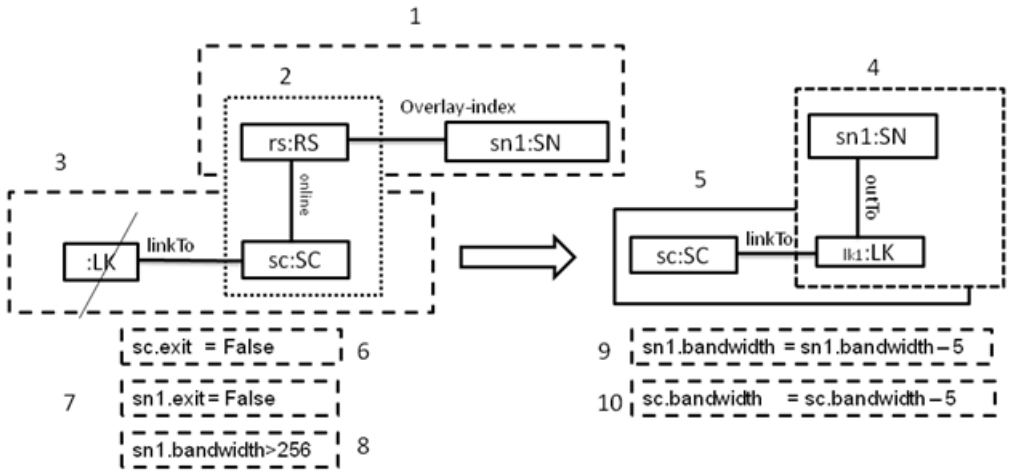

Figure 7.5: GT rule link an SC with SN

```
5     RS.overlay-indexRG,S,Y);
6   }
```

Listing 7.4: SN connected with RS (P.No.1)

The pattern in List. 7.4 finds an SN that is currently connected with a RS through an edge of type RSoverlay.

```
1 pattern SC_go_online(S,X,RG) =
2   {
3   SC(X);
4   RS(S);
5    RS.online(RG,RS,SC);
6    }
```

Listing 7.5: SC is Online (P.No.2)

The pattern in List. 7.5 finds an online SC that is currently connected with RS through an edge *online*.

```
1 pattern test_links(X)=
2  {
```

```
3     SC(X);
4     neg find Link_SN(X);
5    }
6   pattern Link_SN(X)=
7   {
8    LK(L);
9    SC(X);
10   LK.linkto(CO,L,X);
11   }
```

Listing 7.6: SC is not Linked (P.No 3)

The patterns in List. 7.6 are different from the other two. The first pattern calls the other pattern as negative condition. The aim of these two patterns is to make sure that an SC is not currently linked with an intermediate node LK in the current graph. We see that the first pattern calls the second.

```
1 pattern SC_Exit(X,EX) =
2   {
3   SC(X);
4   SkypeNode.Exit(EX);
5   SkypeNode.exit(EXR,X,EX);
6   }
```

Listing 7.7: SC Exit Attribute(P.No 4)

The pattern in List. 7.7 finds the exit attribute of an SC and links variable EX to that attribute. The variable EX can be used to update, compare or change the value of the attribute.

```
1 pattern  SN_Exit(Y,EX) =
2   {
```

```
3   SN(Y);
4   SkypeNode.Exit(EX);
5   SkypeNode.exit(EXR,Y,EX);
6   }
```

Listing 7.8: SN Exit Attribute(P.No 5)

The pattern in List. 7.8 finds the exit attribute of an SN and links variable EX to that attribute.

```
1 pattern SN_Bandwidth(Y,BW)=
2   {
3     SN(Y);
4     SkypeNode.Bandwidth(BW);
5     SkypeNode.bandwidth(BWR,Y,BW);
6   }
```

Listing 7.9: SN Bandwidth Attribute(P.No 6)

The pattern in List. 7.9 finds the bandwidth of the selected SN and links a variable BW to that attribute, which shows the current value of the SN's bandwidth. Similarly, this can be used to update, compare or change the value of the bandwidth of SN.

```
1 patter SC_Bandwidth(X,BW)=
2   {
3     SC(X);
4     SkypeNode.Bandwidth(BW);
5     SkypeNode.bandwidth(BWR,X,BW);
6   }
```

Listing 7.10: SC Bandwidth Attribute(P.No 7)

The pattern in List. 7.10 is used to link BW to an SCs bandwidth. We see that we have repeated the BW in two different patterns. This does not matter as the return value will be stored in the variable which is used to call this pattern.

```
1  gtrule Rule_Link_SC_to_SN() =
2      {
3      precondition pattern lhs(S,X,Y,CRO,RG,BW,SCBW,NL,R)
4      SN(X);     // X is variable of type SN//
5      SC(Y);    // Y is a variable of type SC node //
6      RS(S);    // S is a variable of type RS node //
7      Network(NL); // NL is a variable of type Network node //
8      find SN_overlay-index_RS(S,X,R); // find SN is linked with
          server RS//       P.No. 1
9      find SC_go_online (S,Y,RG);    // call pattern SC go online//
           P.No. 2
10     find SN_Exit(X,EX);  // find exit attribute of SN//          P
          .No. 7
11     find SC_Exit(X,EX1);  // find exit attribute of SC//         P
          .No. 6
12     find SN_Bandwidth(X,BW);    // finds bandwidth attribute and
          current value//  P.No. 8
13     find SC_Bandwidth(Y,SCBW);
14   // finds SC bandwidth and current value which will be used in P
       .No 10 //
15     find test_links(Y); // check if SC is already connected//
                  P.No.
16     check((toInteger(value(BW)))>256);
17     // this check condition make sure that SN bandwidth must be
           greater than 256, this finds
                                                    bandwidth by
```

```
        using P.No 8, while accepting new connection//
18    check(((value(EX)))=False);
19    // this condition use P.No. 7 to find value of exit attribute
        & compare it//
20   check(((value(EX1)))=False);
21 // this condition use P.No. 6 to find value of exit attribute &
     compare it//
22  }  // end of pre-condition//
23
24        action {    // below is post condition of RHS of the
            figure 6.6//
25
26      let LK=undef, CO=undef,OU=undef,NR=undef in seq
27      {
28     new (LK(LK) in NL);  // create  anode LK under the network
          node NK//  P.No. 11
29   new(NL.nodes(NR,NL,LK)); // this make it a child of the network
       //        P.No.11
30   new(LK.linkto(CO,LK,Y));
31   // this establish an edge of type linkto between LK and SC //
          P.No 5
32   new(LK.outTo(OU,LK,X));
33   // this establish an edge of type outTo between LK and SN  //
          P.No. 4
34   delete(RG); // this delete the existing edge between SC and RS
       server// P.No.2
35      setValue(BW,((toInteger(value(BW)))-5));
36   // link cost compensation is deducted from SN//  P.No  9
37      setValue(SCBW,((toInteger(value(SCBW)))-5));
38   // link cost compensation is deducted from SC//  P.No. 10
```

```
39      println("SC connected/Linked  with SN");  // message is
            printed//
40
41          }
42
43                      }          // close action part//
44  } // close GT rule//
```

Listing 7.11: Rule Link an SC with SN

In List. B individual patterns are combined to form a single rule. List. B The number show where each pattern is used.

7.4 Summary

This chapter focussed on the use of GraSS as stochastic simulator. We have discussed the conversion of the type graph of a GTS into a VIATRA2 model space. We have shown transformation rules converted to executable VIATRA2 rules. The complete set of rules is shown in appendix B. The model space in this chapter along with the VIATAR2 transformation rules are used by the GraSS tool for simulation and analysis. This is the focus of the following chapter.

Chapter 8

Stochastic Simulation

In this chapter, we are going to simulate the models and associated variations of the P2P VoIP protocol. This chapter will use the feature tree presented in Chapter 4 along with the model and transformation rules presented in Chapter 6. The simulation is based on the executable forms of the model and transformation rules presented Chapter 6. Based on our case study, first we will assign probability distributions to different GT rules.

In this chapter, we compare different strategies for selection, promotion, load balancing, single and multiple links. As a result, we propose a single protocol for P2P VoIP based on the performance of the design variations tested.

8.1 Probability Distributions For Rules

The aim of this section is to explain how the distributions are assigned to the rules of the graph transformation system in this thesis. One challenge in this case study was to select an appropriate distribution for each rule.

For example, as we don't know exactly how many times a user uses Skype, it can be once in 24 hours orevery 3, 4 or 6 hours. In such a scenario this event can be considered as following an exponential distributions. On the other hand, if we know the average length of time it takes to complete an action, such as a VoIP call, then it can be categorized as a log-normal distribution (positive normal distribution).

Hence, each GT rules presented in Chapter 6 of this thesis needs to be categorized into exponential or log-normal, and then associated with specified scheduled delays. The exponential distribution requires a single parameter of corresponding rate or frequency, whereas the log-normal distribution requires two parameters, i.e., mean and variance. The corresponding rate, mean and variance parameters for the case study were selected based on the Skype statistics, network analysis and reverse engineering data from [15; 53; 97; 134; 150; 173; 182].

Real data was gathered from [150] which was based on one month of analysis, from 17th December 2010 to 17th January 2011. Due to the nature of the P2P protocol no restriction can be imposed on the peers, so most of the rules in this case study will follow exponential distributions while tasks such as a peer linking with a super peer or call establishment are following log-normal distributions. But before presenting values, we briefly explain which distribution suites which feature in the feature tree of Chapter 6, Fig. 6.2.

1. **Register New User**

   The GT rules modelling user registration will be associated with exponential distributions since in the real Skype network there is no exact information on when a user can register. In fact user registration is related to user choice.

2. **User Goes Online**

   A user can go online anytime once they are registered. There are no exact information available that how many time a registered user go online in 24 hour. For example some people may use Skype twice in 24 hour, or even more frequently. Thus, the exponential distribution is the right choice for this feature of the P2P VoIP protocol.

3. **Link with Super Peer**

   Once a user gets online with the server, the next task performed by the protocol is to find a host for the newly online Skype client. Real observation shows that Skype takes between 30 to 50 seconds to find a super peer, link and update the global index. These timings suggests that the GT rule modelling the linking of SC to SN will be associated with a log-normal distribution.

4. **Promotion to Super Peer**

   Promotion does not depend on user choice. Once a peer satisfies the constraint of the bandwidth of 1.5 Mbps, the protocol could randomly select the SC for promotion. Hence, promotion GT rules will be assigned exponential distributions.

5. **Connect Super Peers / SN Overlay Formation**

   After compulsory promotion, the new super peer has to become part of the SN overlay in order to share the global index. This task normally takes up to 50 seconds. Hence, the GT rules modelling this event will be assigned log-normal distributions.

6. **Demotion to Client**

   If a super peer is unable to accept new connections, the protocol takes up to 55 seconds to cancel the super peer role and maintain its client-only role. Hence, GT rules modelling this downgrade event will be assigned log-normal distributions.

7. **Departure of Skype Super Peer / Skype client**

   The departure of peers and super peers both depend on user choice. A user may use Skype for hours or may decide to leave the network in a couple of seconds. The GT rules modelling controlled and uncontrolled departure are assigned exponential distributions. However, once a peer or super peer decides to leave the network in a controlled manner, the dependent peers or SN recover their links in 30 seconds time. Hence, GT rules modelling the recovery are assigned normal distributions.

8. **VoIP Calls**

   This feature of the VoIP protocol consist of three phases. Phase 1 is user's request for VoIP Call. This task depends on when the user wants to make a call to another user. Hence, the GT rules modelling this phase are assigned exponential distributions. Phase 2 consists of call establishment. This task

is performed by the protocol and the protocol takes up to 40 seconds until ringing event. Therefore, the GT rules modelling this phase are assigned log-normal distributions. The last phase is call termination. Real experiments suggest that, on average, VoIP calls are between 3 to 16 minutes. Hence, the choice of a log-normal distribution is the best for the GT rules modelling call termination and completion.

9. **Load Balancing**

   The super peer-based on the local awareness, may promote one of its client to the extended role. The super peer takes up to 50 seconds to promote a local client and transfer the load. Hence, log-normal distributions are assigned to the GT rules modelling this feature.

Tables A.2 and A.1 show some of the core GT rules with heir types of distributions and parameters. The complete list of rules and distributions for all models are given in Appendix A.

8.2 Setup and Parameters

We simulate our model using GraSS [87; 162] presented in Chapter 7 of this thesis.

In GraSS a GTS is represented as a VIATRA2 model as explained in Chapter 7. The VIATRA2 model consists of the model space with the current/start graph and the transformation rules. Moreover, GraSS takes as input an XML file with the definitions of the distributions associated with transformation rules, as

State	Event	Mean μ	Variance σ^2
Link SC to SN	ConnectSCToSN	0.000347	2.5
Re-Link SC	ReconfigureSCtoSN	0.000173	2.0
SN Overlay	SNtoSNOverlay	0.000578	4.5
Control exit Prep.	CtrlExitPrepSc	0.000154	2.0
Control Final Ext SC	CtrlFinalExitSc	0.000154	2.0
Control ext Prep SN	CotExitPrepSN-Delink-SN-S	0.000154	4.0
Control exit Prep SN	CotExitPrepSN-Delink-SN-T	0.000154	4.0
Control exit Prep SN	CotExitPrepSN-Delink-SC	0.000154	4.0
Control Final SN Depart	CtrlFinalSN-Exit	0.000154	4.0
Place Call using STUN	CallUsingSTUNB/WSCs	0.000347	10.00
Place Call using STUN	CallUsingSTUNB/WSNSCs	0.000347	10.00
Place Call STUN Failed	CallWhenSTUNFailed	0.00567	5.00
Call Term.	callTermination	0.033	1.0
Load Balancing	LoadTransferBal	0.000347	2.5
Cancel SN Role	DownToSCfromSN	0.00578	10.0

Table 8.1: GT rule log-normal distribution

Event	Frequency	Mean λ $(hours)$	Mean arrival rate $\lambda^{-}1(hours)$
UserGoOnline	Every 8 hours	3.0	0.33
	Every 6 hours	4.0	0.25
	Every 4.8 hours	5.0	0.20
	Every 4 hours	6.0	0.1666
UserGoOffline	Every 5 hours	4.8	0.208
UserCrash	Every 168 hours	0.142	7.042
UserReqVoIPCall	Every 6 hours	4.0	0.166

Table 8.2: GT rule exponential distribution

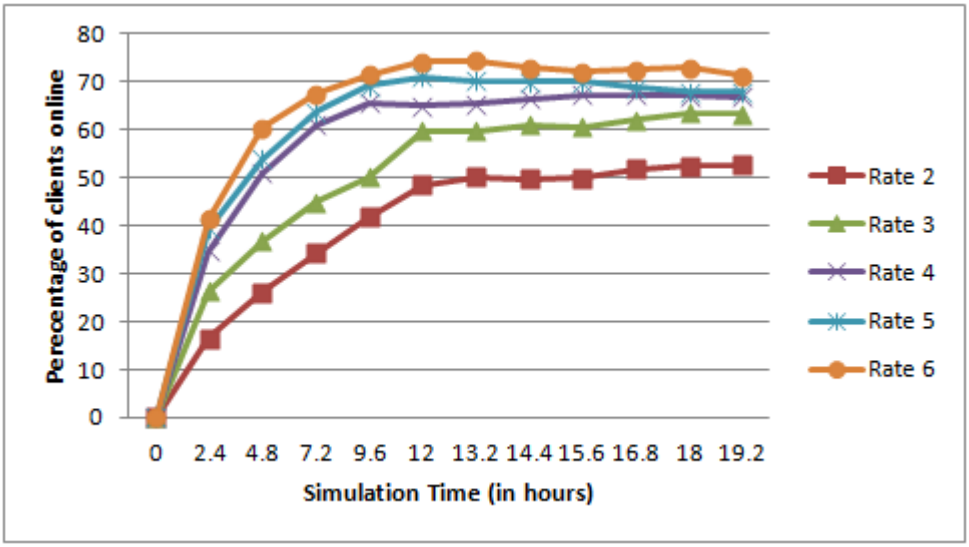

Figure 8.1: Steady state of random SN selection

well as the list of rules with empty post-conditions that are to be used as probes. Additional parameters needed for a simulation run are provided to GraSS as part of the VIATRA2 model.

The simulations in this thesis are based on the time unit of 24 hours with a simulation time of 19.2 hours. We run the simulation with a batch size of 6 runs. The simulation time was selected based on steady state experiments. We simulated all models and observed that after 12 hours simulation time every model was in steady state. Fig.8.1 shows that the model is in a steady state for client joining at rate 2,3,4,5 and 6. The test results for the other models are given in Appendix A.

We evaluate our models based on performance parameters such as

- How many clients are happy with their current selection of super node?
- How many clients are in the model?

- How many are linked with a super peer?
- How many clients are waiting to link with a super peer?
- How many super peers are present in the model?
- How many calls are supported?
- How many clients reconfigured their connections?

All these performance indicators have been modelled as GT rules with empty postcondition in graphical form Chapter 6 and as executable rules in Appendix B. After each simulation step, the tool collects statistics for these probe rules, generates reports giving min, max and average values along with their confidence intervals.

Although the tool supports large numbers of nodes, in this thesis we keep the number of users to 1000. During experiments we have simulated the models upto the size of 10000 users, but observed that after 1000 the averages are nearly the same. We have fixed the rates on the 1000 nodes models such that it produces the same number of averages as those observed in the real Skype network [15; 53; 150; 154]. Further, the models are also evaluated in the presence of random and uniform background traffic.

8.3 Simulation of Super Peer Selection

In this experiment we compared four models using different approaches for connecting clients with super peers. The four different approaches are shown in Fig. 8.2. They are tested by keeping other mandatory features of the protocol

constant varying only the selection strategy. The Exclusive OR (XOR) enables us to select exactly one. In this experiment a registered user can go online and the model randomly decides whether a client is behind firewall or NAT. We test all these four models through variations $\{2, 3, 4, 5, 6\}$ of the rate x for the client to go online. In all models the number of registered users were 1000.

All four models use promotion based on random dynamic protocol and the background traffic is modelled using random traffic rules. Clients depart by crashing or operative exit. SNs can demote if they do not meet the requirements of the role. In all the models VoIP calls are supported both routed via SN and directly.

Each model has been tested by running batches of simulations, varying the rate of GT rule *go online* as shown in at Fig. 6.5 Chapter 6 and VIATRA2 code is in List B.3. Each batch consists of 6 independent runs, each bounded by a time limit of 19.2 hours.

We programamed GraSS to automatically generate independent batches of simulations for each model. We produce confidence intervals based with a confidence level of 95%.

We compare the performance of all approaches with respect to four measurements: the total number of SC nodes in the network, the number of super nodes in the network, the percentage of linked SC nodes, and the ratio of happy peers. The ratio of happy peers is computed as percentage of happy clients divided by the total number of linked clients.

The simulation results are presented as graphs. The results in Fig. 8.3 show the average number of clients in the models. This results show that almost all approaches have 35% - 65% of clients online. At joining rates 2,3 and 4 the average for numbers of clients are close to each other. However, at joining rates

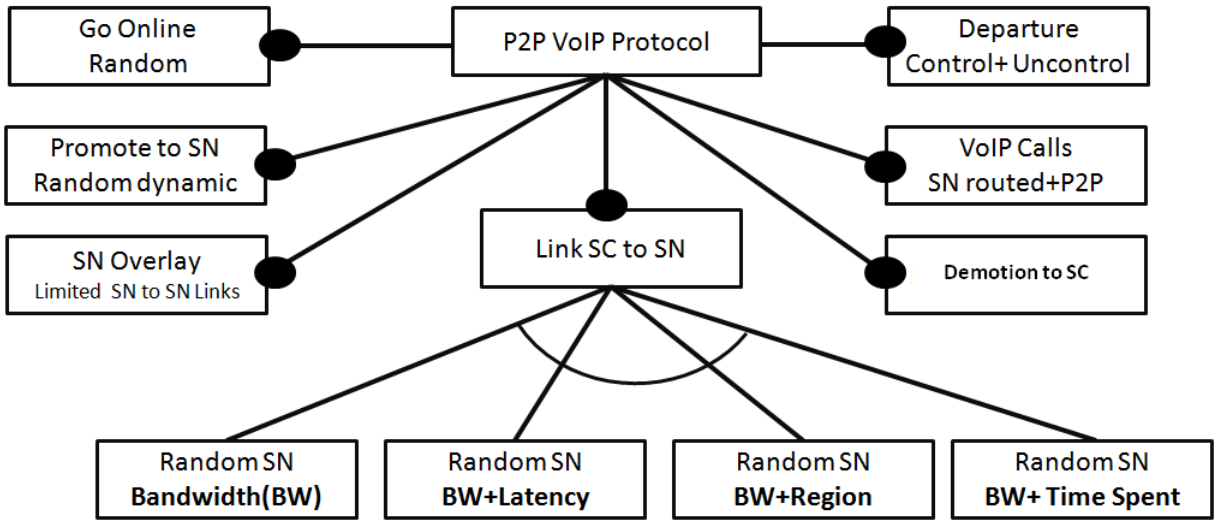

Figure 8.2: Feature tree for SN selection

5 and 6 the latency-based SN selection produces slightly fewer clients.

Fig. 8.4 shows the average numbers of SNs observed along with the average number of SCs.

Fig. 8.5 shows average number of clients online with their firewall enabled. These SCs are not candidate for promotion to super peer and are always dependent on SNs. The observed average shows that all approaches have nearly similar number of SCs with firewall enabled.

Fig. 8.6 shows the average number of SCs linked to SNs. The simulation results show that random linking results in percentages of linked SCs between 56% and 61%. These percentages make this approach superior in terms of linking SCs. Further, time-based SN selection links SCs with percentages between 32% and 42%, making this approach the second best. The simulation results further show that the latency-based selection approach links between 20% to 27% of online clients, placing it on third position. The local SN-based selection results in the

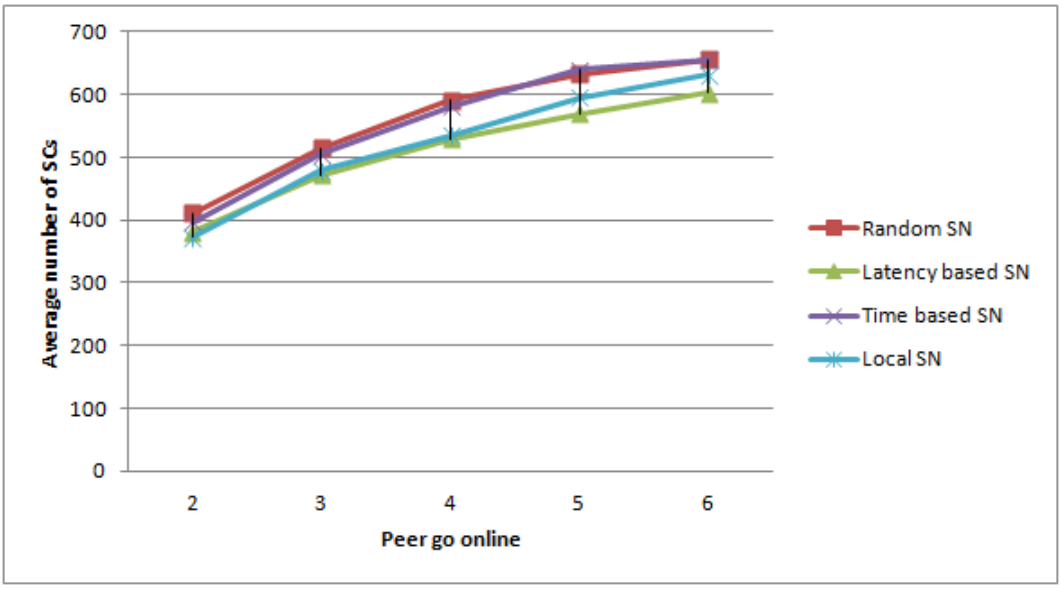

Figure 8.3: Average number of SCs

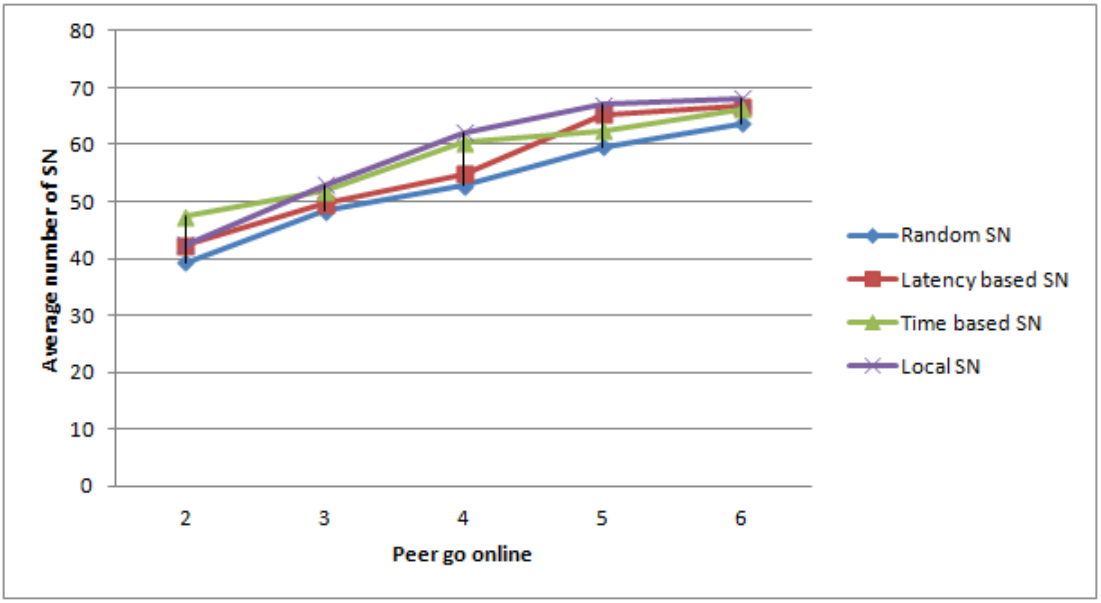

Figure 8.4: Average number of SNs

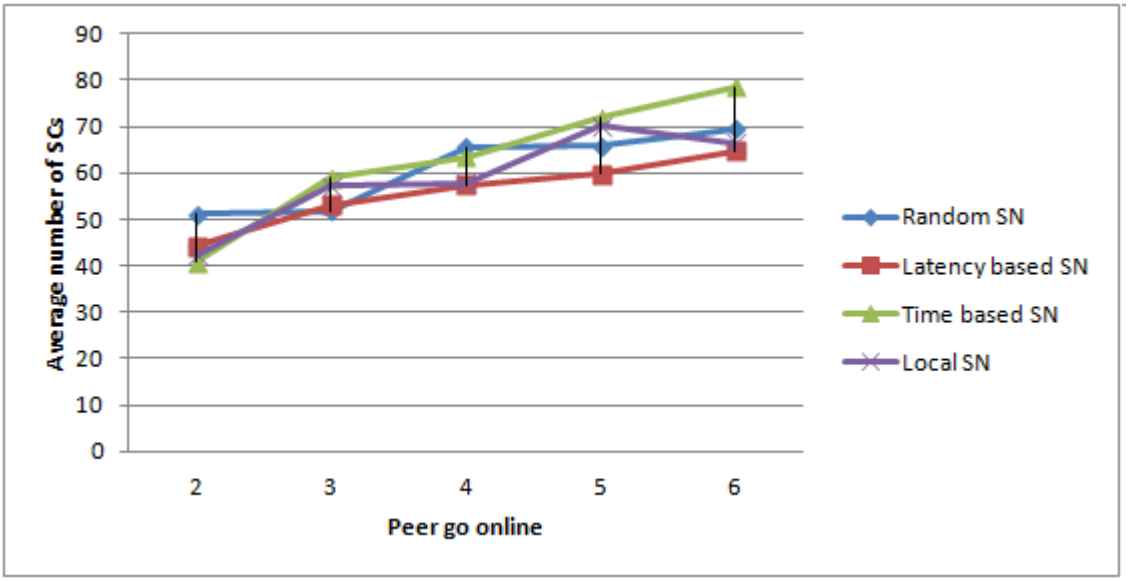

Figure 8.5: Average number of SCs with firewall/NAT

lowest percentages. This is mainly because it is a more complicated approach as it requires a local SN for connection. This slows down the process of linking SCs with SNs. Thus, our first criteria for evaluation of these approaches shows that putting constraints on the selection of SNs results in increases in waiting time. This is one of the reasons that random SN selection has produced results which are the highest among the approaches.

The second evaluation is on the basis of averages of number of happy SCs. As stated earlier an SC said to be happy if its current host SN has 1.5 Mbps of spare bandwidth. Fig. 8.11 shows the observed values for the probe rules that count happy SCs. The results are different as compared to our previous evaluation. They show that local SN selection results in an average percentage of happy SCs between 71% and 77%, which makes this approach superior. The latency based approach stands in second place, with percentages of happy SCs between 61%

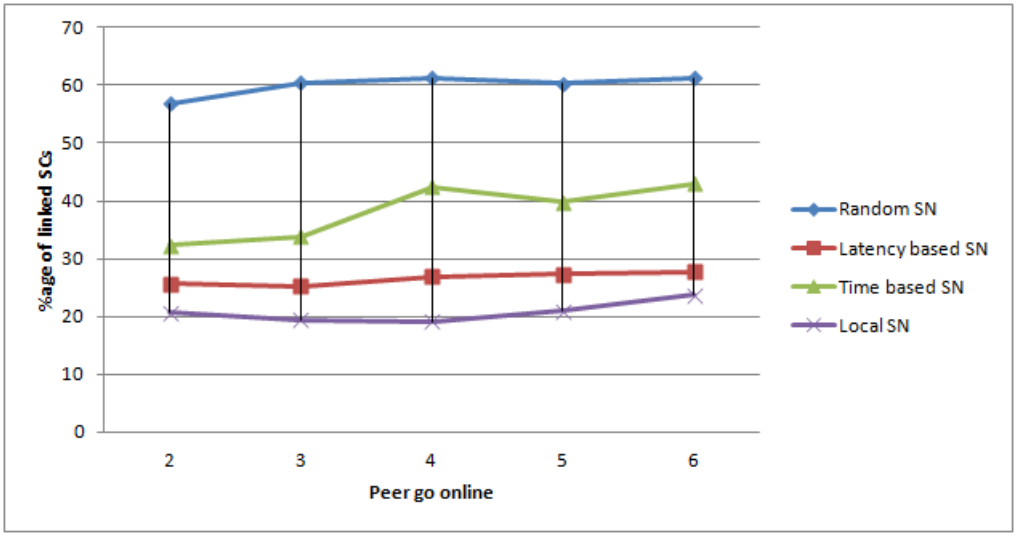

Figure 8.6: Average number of SCs linked with SN

and 63%. At third position is the random SN selection with percentage between 57% and 64%. The lowest percentages result for time-based SN selection, which is due to the fact that an SN that spent more time in the network is more likely to host a greater number of SCs.

We present here confidence intervals for the average numbers of linked SCs with a confidence level of 95%. The tool produce confidence intervals for each of the probe rule. The confidence interval for random SN selection is shown in Fig. 8.7. The figure confirms that lower bounds (LB-) and upper bounds (UB+) are not far away from the average. Similarly, the confidence interval for latency based SN selection is shown in Fig. 8.8. The graph confirms that intervals are close to the average, but wider than the random-based. Further, Fig. 8.9 shows the CI for the time based approach. The graph shows that CI for latency and time-based approach are roughly the same. The last CI in Fig. 8.10 shows local SN based selection. The graph shows that their CI is narrower than in the previous two approaches especially at rate 7.

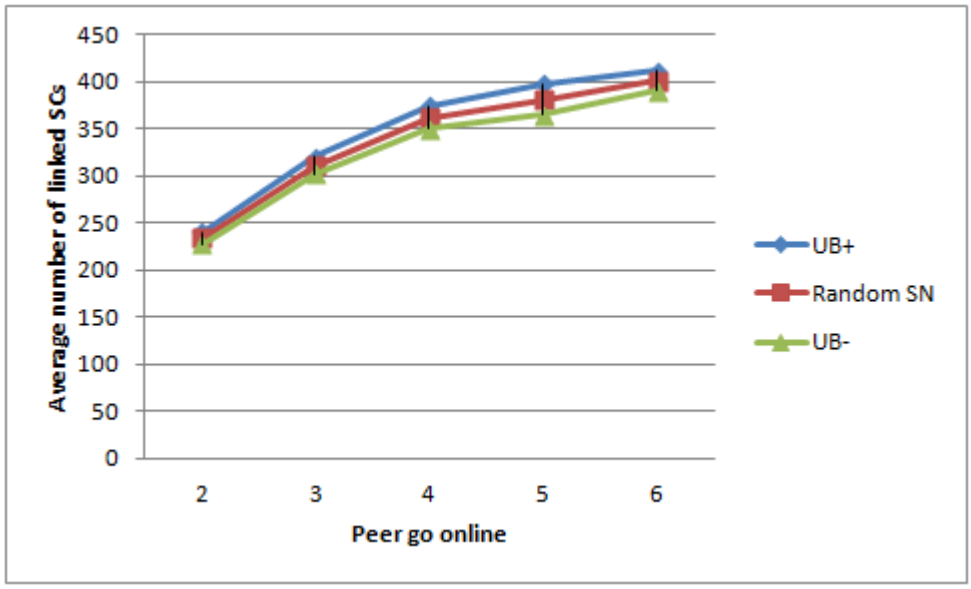

Figure 8.7: Random SN 95% CI for average number of linked SCs

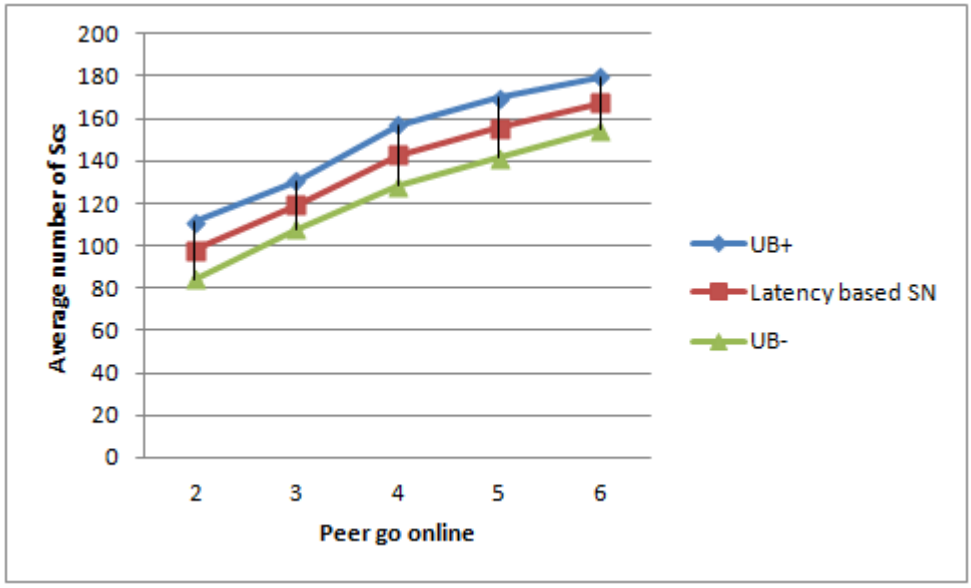

Figure 8.8: Latency based SN 95% CI for average number of linked SCs

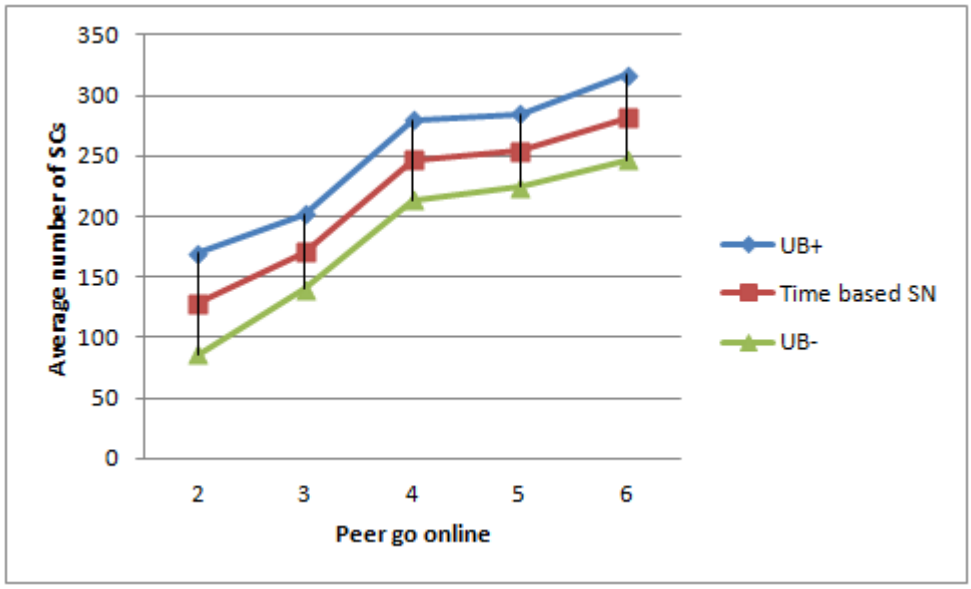

Figure 8.9: Time based SN 95% CI for average number of linked SCs

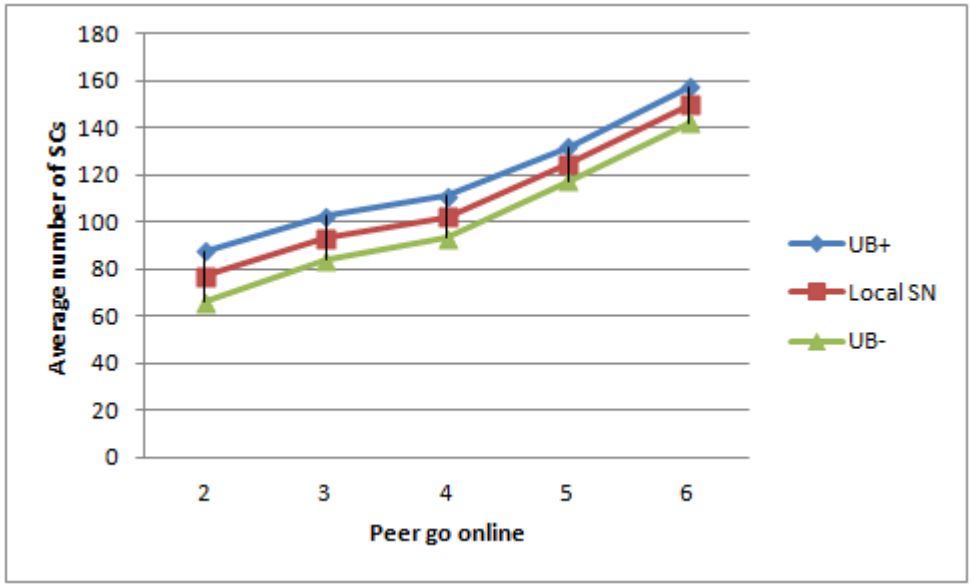

Figure 8.10: Local SN 95% confidence intervals for average number of linked SCs

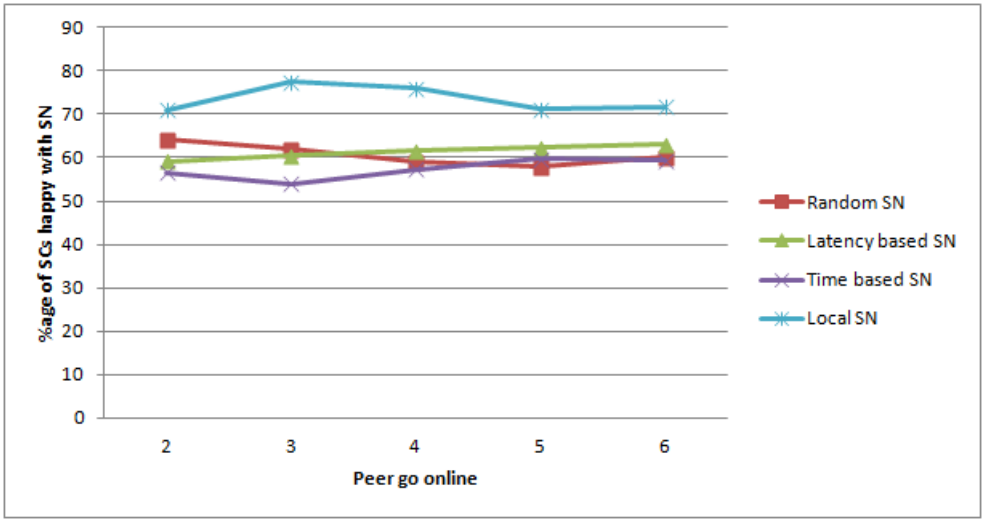

Figure 8.11: Average number of SCs *happy* with SN

Hence, this experiment has shown that random SN selection does reduce the waiting time to select an SN but it can not guarantee the quality of the selected SN. Local and latency based selection slow down the process of finding a suitable SN but make sure that the selected SN is in a position to serve the SC. Further, we observed that SN selection on the basis of the time that the SN has spent online produces the poorest results in terms of happy SCs.

8.4 Simulation of Peer Promotion

Building and maintaining a super peer (SN) based overlay topology is not a simple task. Super peers are selected among the online peers, which may have limited bandwidth available to relay VoIP sessions. For example, the bandwidth limit of a super peer with cable/DSL connection is at most its upstream link bandwidth. Further, as NATs and firewalls are increasingly deployed in the internet, more peers depend on super nodes to obtain VoIP services. The quality of a VoIP

network may degrade when the bandwidth demand of active sessions exceeds the available access capacity of the network. At the same time super peers may be subject to the AS policy constraint [173]. Specifically, due to Internet economics, certain autonomous systems (AS) do implement their own policies [173]. A common example of this policy is the no valley routing policy [44], which states that a customer does not relay traffic for other costumers. Super peers relaying traffic for Skype clients are in violation of this policy [144]. As a result some universities have banned Skype from their campuses [122].

In this experiment we compare and evaluate three different approaches to super peer promotion. They are represented by models and sets of GT rules as discussed in Chapters 4 and 6, referred to as static, dynamic, and need-based protocol. The three different strategies are shown in the feature tree in Fig. 8.12. We see that in all three protocols a registered user can go on-line by two different approaches. In the first approach the model has no control on the number of clients behind firewall. The approach uses a single GT rule to make client go online. The clients with or without firewall decision is made on the basis of a random number. The second approach provides control over users behind firewall using two different rules. One for clients going online with static IP and the other for clients behind firewall. This enables us to see the performance of these approaches by increasing the number of users with firewall and reducing the users with static IPs. Therefore, in this simulation we have a total of 6 models. All models use random selection for linking SCs with SNs. Further, all the other features are the same as in our previous experiment.

We divide this experiment into two steps. First we use the GT rule UserGoOnlinewithRandomFirewall. We test all these models through 4 variations

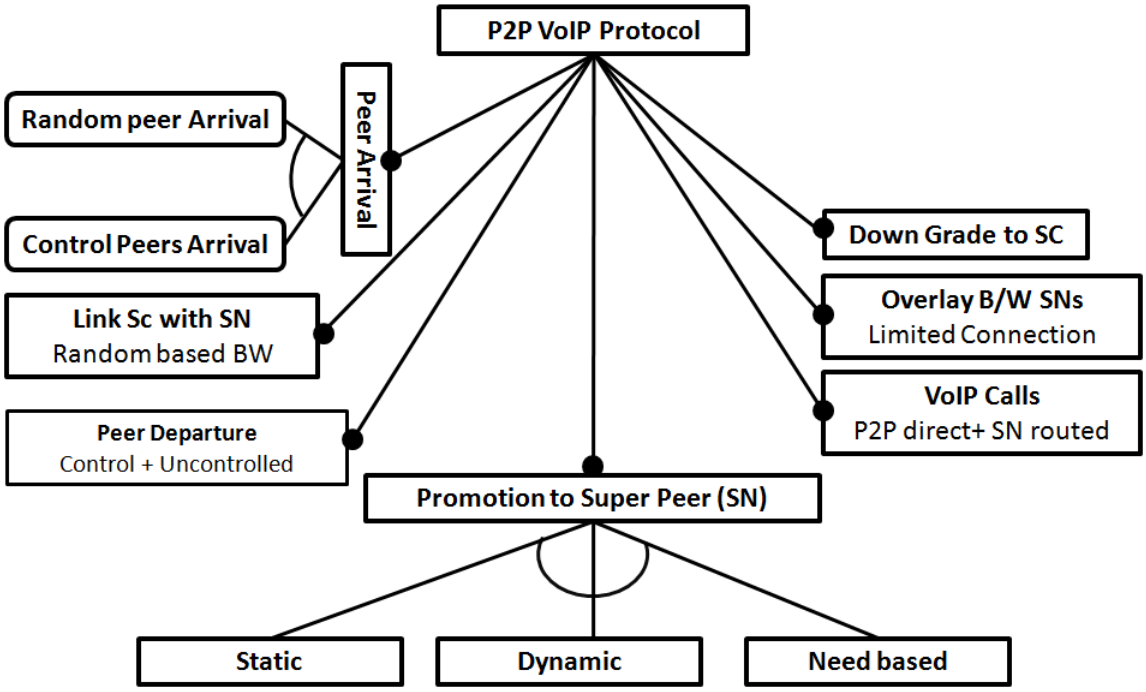

Figure 8.12: Feature tree for super peer promotion

of the rate x for this GT rule UserGoOnlinewithRandomFirewall, using exponential distribution ranging over $\{2, 3, 4, 5, 6\}$. The number of *registered* users is 1000. The second experiment fixes the rate of the GT rule GoOnlinewithFirewallDisabled (y) at $\{4\}$ and we vary the rate of GT rule GoOnlinewithFirewallEnabled (x) over $x \in \{2, 3, 4, 5, 6\}$. This enable us to compare the performance by reducing the number of candidates for super peers and increasing the load. We use four measurements: the total number of SC nodes in the network, the number of super peers in the network, the percentage of linked SC nodes, the ratio of happy peers, and the number of calls completed.

The simulation results in Fig. 8.13 show that all three promotion protocol allow between 400 and 700 clients to go online. Although the dynamic protocol has slightly less clients in the network, the difference is small.

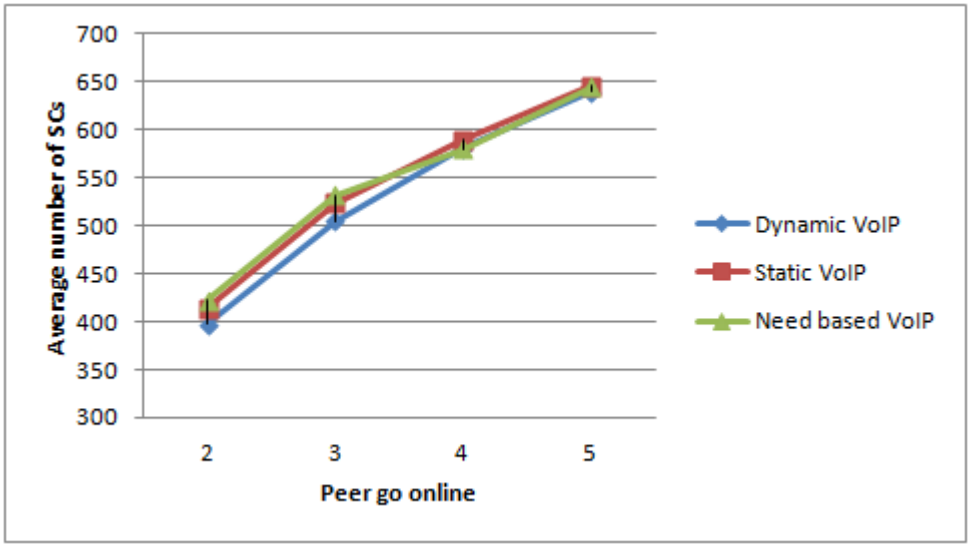

Figure 8.13: Average number of clients

Fig. 8.14 shows the average number of SNs promoted during the simulation.

To compare these models we used a probe rule to count the number of SCs linked and the number of SCs that are happy with their current SN. Fig. 8.15 shows the average number of SCs that are linked with an SN. The graph shows that at joining rate 2, 3 and 4 the need-based protocol and the static protocol are performing at the same level. At joining rate 4 the dynamic protocol results in averaged which are close to other two approaches. However, at joining rate 5 and 6, the need-based protocol and static protocol follow same trend. Further, the results show that the need-based protocol and the static protocol have resulted in well-connected networks.

To further evaluate these approaches, we use our second performance measure of happy SCs. The graph in Fig 8.15 shows that the need-based protocol for super peer promotion is performing better than the other two approaches. Fig. 8.16 shows that at joining rate 4 the number of happy peers is almost the same for all

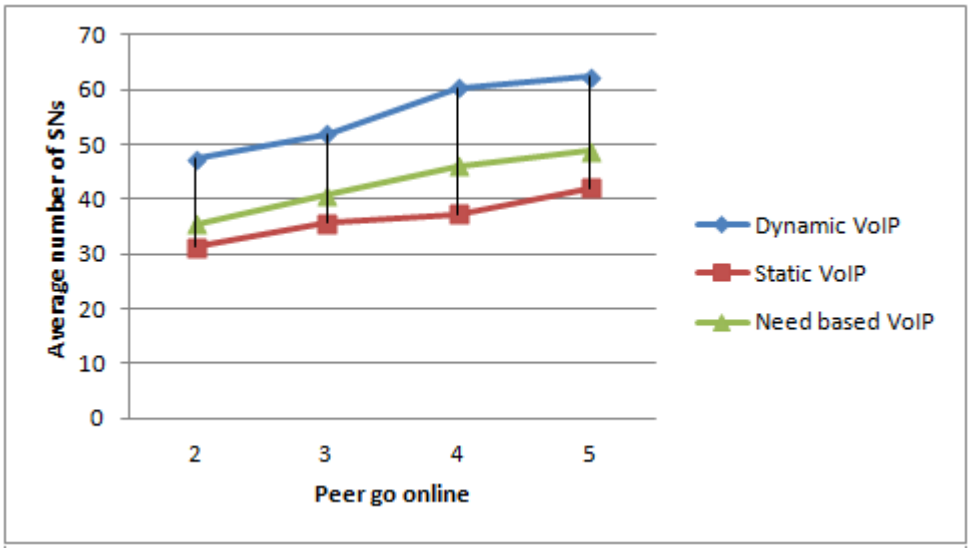

Figure 8.14: Average number of SNs

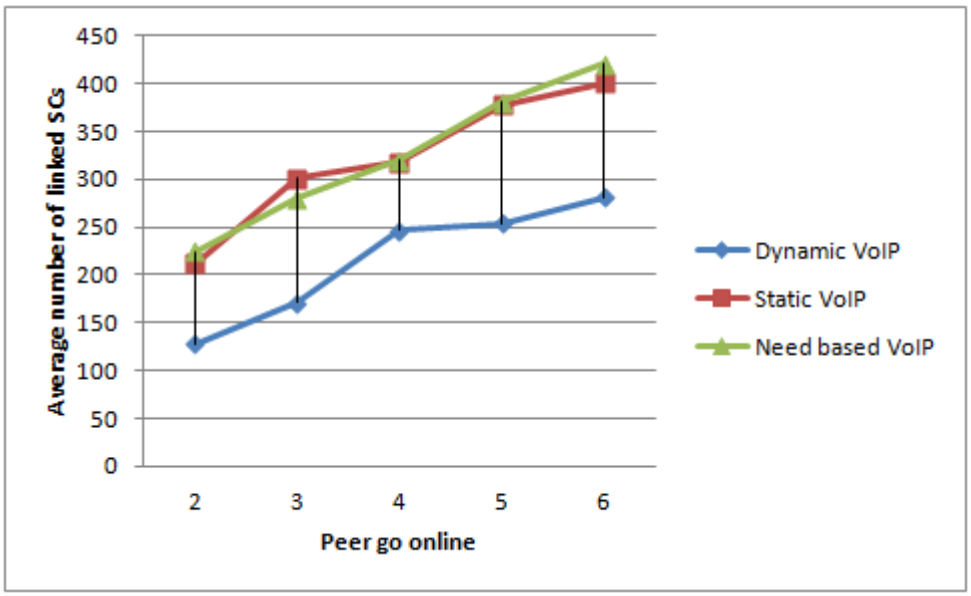

Figure 8.15: Average number of SCs linked with SN

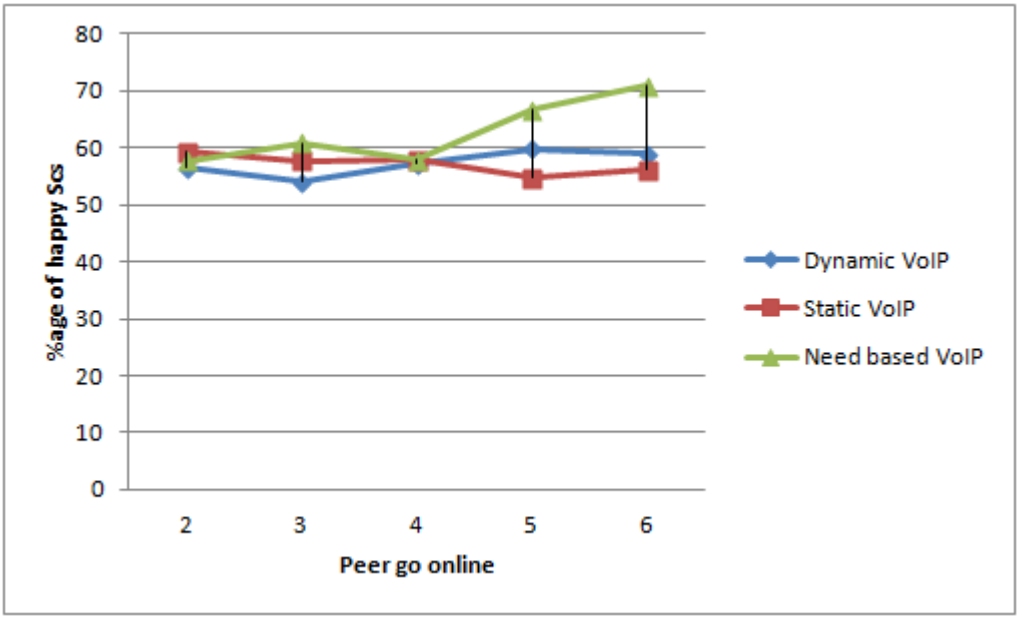

Figure 8.16: Percentage of SCs happy with current SN

approaches, but at rate 5 and 6, as the number of clients in the network increases, the need-based approach produces better results than the other two approaches.

Hence, the need-based and static protocols result in shorter time for SCs to find a host as well as increases the number of happy clients. The first experiment shows that at joining rate 4 all three approaches produce close results. Now we have fixed the rate of the rule GoOnlinewithFirewallDisabled y at $\{4\}$. The aim is to fix the number of candidates for SN and we vary the rates for the rule GoOnlinewithFirewallEnabled x such that $x \in \{2, 3, 4, 5, 6\}$.

Fig. 8.17 shows that all three protocols resulted in similar numbers of clients with firewall enabled. Fig. 8.19 shows that as the number of SCs with firewall enabled increases, the number of SNs reduces. Fig. 8.19 shows the average number of SCs linked in the model. The result confirms that the dynamic protocol has managed to link a good number of SC with SN, thus reducing the waiting time

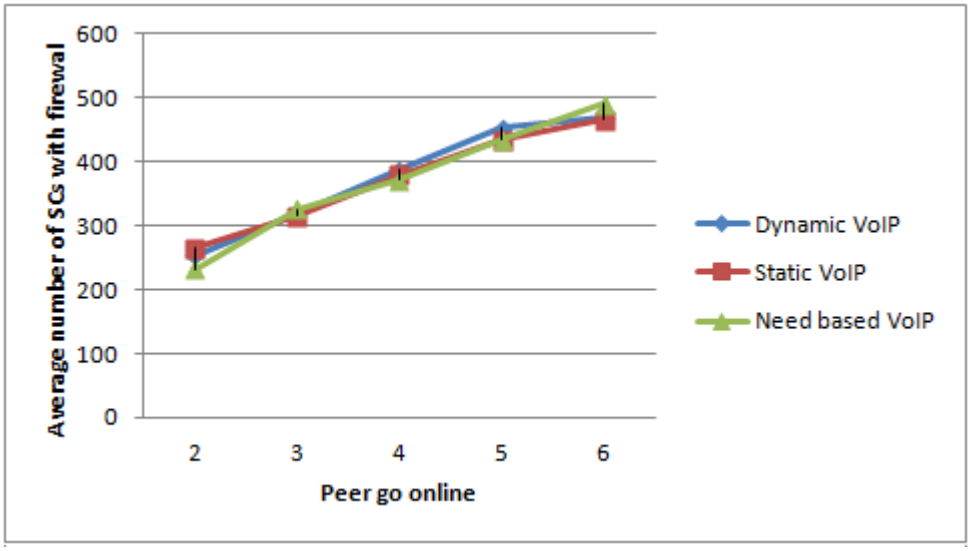

Figure 8.17: Average number of clients with firewall enabled

for SC connection. The need-based protocol also resulted in a good average, compared to the dynamic protocol. However, the difference between these three approaches is small enough at joining rates 3 and 5. Now we will finally evaluate these approaches on the basis of their average numbers of happy SCs. Fig. 8.20 confirms that the need-based protocol is performing much better than the other two approaches when the number SCs behind firewall increases. Although, at the start the static approach is performing better at joining rate 2 and 3, where both rules are running at the same rate, the need-based approach continues to produce better results as the number of SCs behind firewall increase.

The overall conclusion of this experiment is that, as the number of firewalls deployed over the Internet increases and many ISPs are banning the routing for P2P applications, the number of SN candidates will reduce. In such a situation the need-based protocol seems a better choice as it is more economical in its use of super peer candidates.

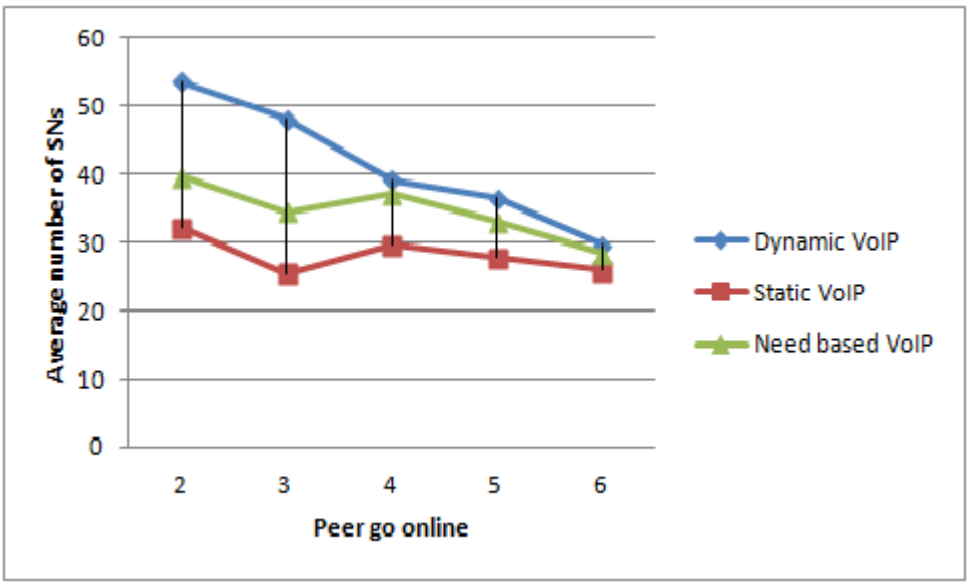

Figure 8.18: Average number of SNs

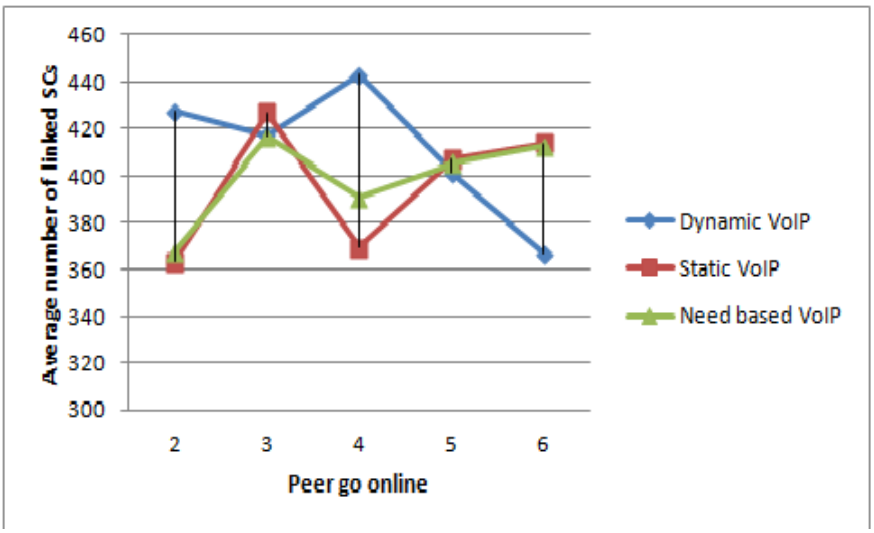

Figure 8.19: Average number of SCs *linked* with SN

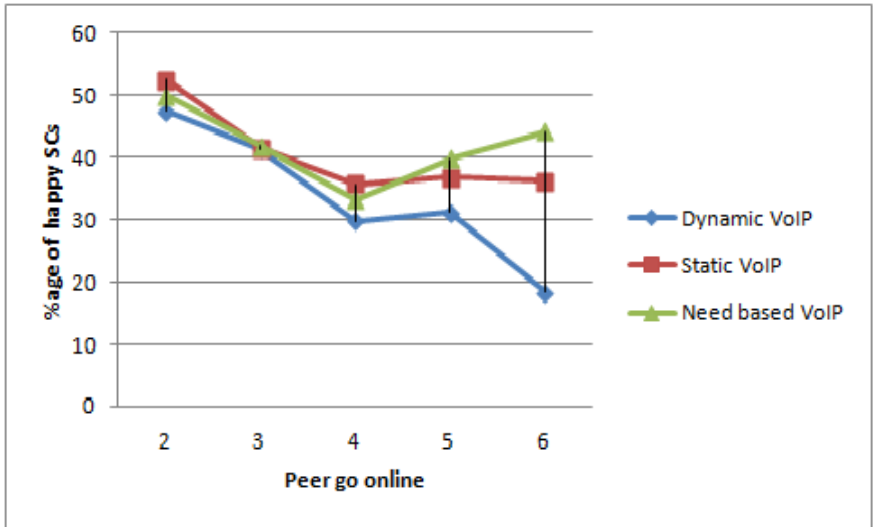

Figure 8.20: Average number of SCs *happy*with SN

8.5 Evaluating Load Balancing

Super peers have been introduced to improve the performance of structured P2P networks. The resulting heterogeneity benefits efficiency without compromising the decentralised nature of P2P networks. However, this only works as long as there are enough super peers and the distribution of clients among them is roughly even. With the increase in the number of users or organisations preventing the use of their clients as super peer, the overall number of candidates for super peer tends to be limited. Thus, selection and load balancing strategies are critical, especially in voice-over-IP (VoIP) networks where poor connectivity results in immediate loss of audio quality.

An optimal strategy for super node promotion and load balancing is difficult to implement in a decentralised system, but there are two obvious local solutions: To move clients from overloaded super nodes to less busy ones, or to promote clients whenever more super nodes are possible. We model two alterna-

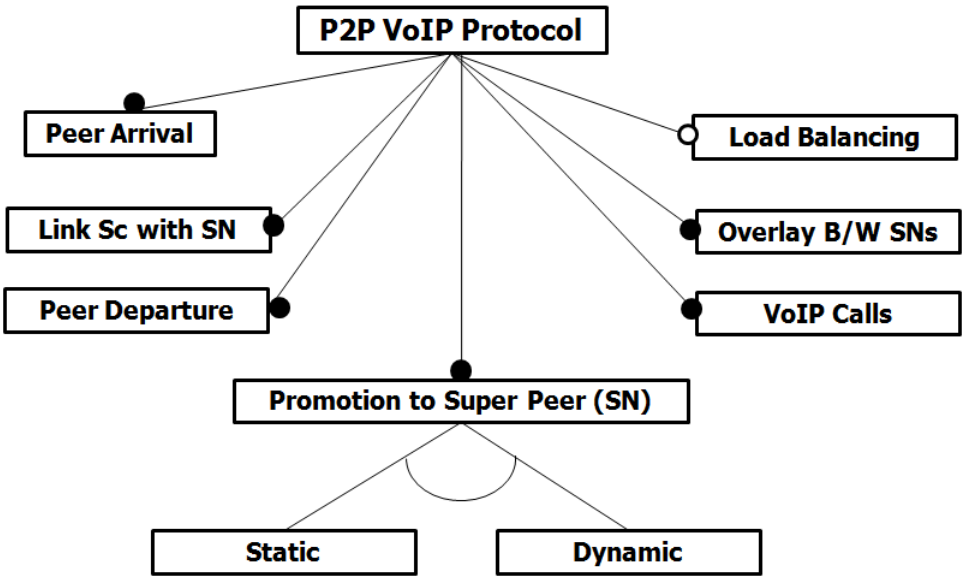

Figure 8.21: Feature tree with static, dynamic and load balancing

tive strategies, one with static super peer selection and load balancing and one based on dynamic selection and promotion, and compare their performance in ensuring client satisfaction.

Fig. 8.21 models the VoIP protocols with two XOR features for peer promotion and an optional feature for load balancing that could be used with both of the promotion strategies. In this experiment we use GT rule UserGoOnlinewithRandomFirewall for clients to go online. We run this experiment with the same parameters as previously and vary the joining rule rates as $x \in \{3, 4, 5, 6\}$.

Fig. 8.22 shows that the dynamic protocol and the static protocol with optional feature of load balancing (LB) result in client populations that go online with averages almost the same. This provides comparable input to the model and enables us to evaluate the output. The average number SNs in both models are presented in Fig. 8.23. The figure shows that at joining rate 3 the average

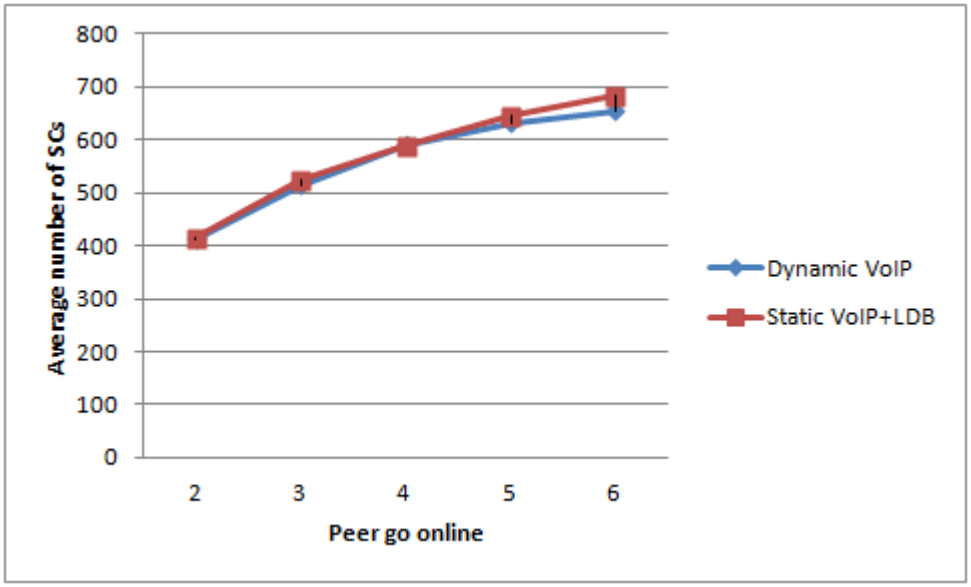

Figure 8.22: Average Number of SCs

numbers of SN are not much different, but at joining rates 5 and 6 this changes significantly.

In order to comment on the performance, we evaluate them on the basis of the numbers of SCs linked and the percentages of happy SCs. Fig. 8.24 shows the average number of SCs linked with SN. The graph shows that both protocols have similar averages at joining rate 2, 3, 5 and 6 . Fig. 8.25 shows the percentages of happy peers in the network. The static protocol with independent load balancing is performing much better compared to the dynamic protocol. The number of happy SCs is stable under increase of the joining rate, which confirms the scalability of the protocol.

This simulation results confirm that large numbers of SNs do not guarantee that the load will be balanced across all the SNs. Therefore, local load transfer decisions can result in a greater number of happy clients.

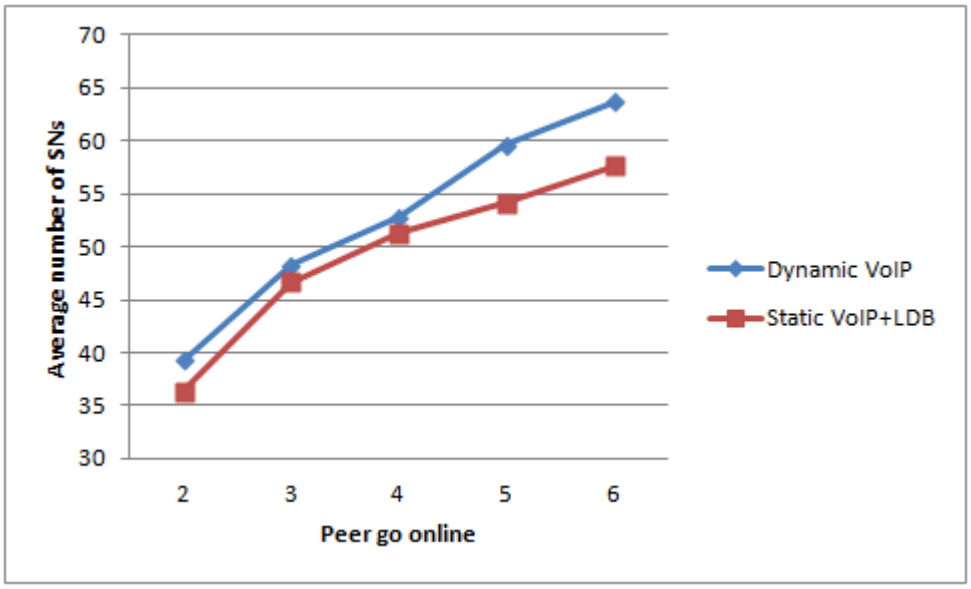

Figure 8.23: Average Number of SNs

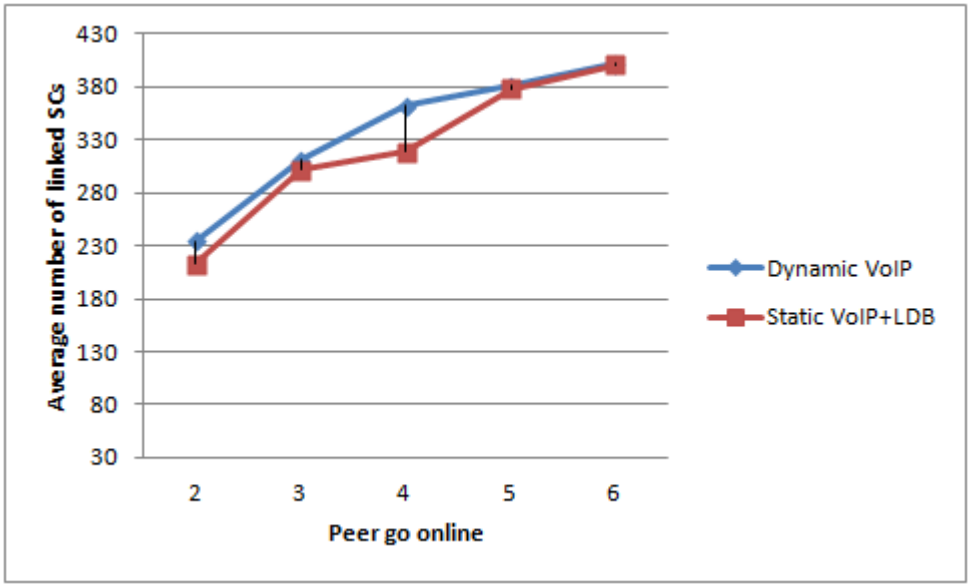

Figure 8.24: Average Number of SCs linked with SNs

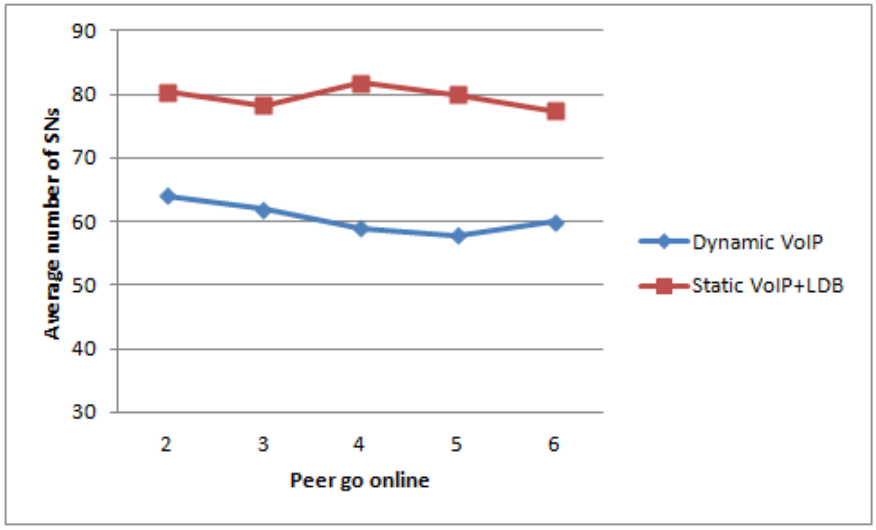

Figure 8.25: Average Number of SCs happy with SNs

8.6 Single Vs Multiple Links

In this simulation experiment we are going to evaluate the effects and benefits of an SC selecting a single or multiple SN. [53; 111; 121; 176] claim that Skype clients keep a single connection to their super peer. [176] suggests that multiple connections of peers with different super peers will help in avoiding a single point of failure. According to their approach, if a super peer leaves the network, all the dependent peers will be disconnected until they connect to another super peer. If a peer is connected to multiple super peers, the departure of a single super peer will not completely disconnect the peer. [118] further comments that a peer must keep multiple connections and must periodically check the status of these connections, so as to notice when a super peer leaves the network. This will help the peer to establish a new connection so that multiplicity is maintained.

The feature tree in Fig. 8.26 represents the VoIP protocol that could either choose to link an SC with a single random SN or multiple random

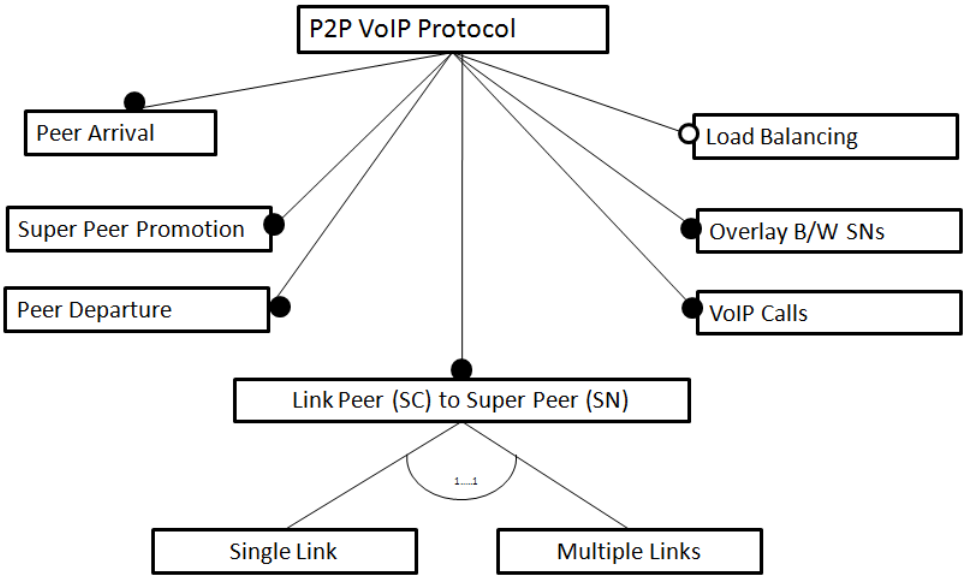

Figure 8.26: Feature tree for VoIP protocol with single/multiple links

SNs. This protocol uses the dynamic version for peer promotion, a single rule UserGoOnlinewithRandomFirewall to allow clients to go online, and random numbers of SCs with firewall.

We simulate both versions of the protocol, called the single-link and multiple-link protocols. We evaluate their performance on the basis of the number of SCs linked with SNs and the average number of happy SCs in the network.

Fig 8.27 shows that both protocols have created a similar pool of online clients. This serves as a uniform input to both models. Fig. 8.28 shows that at joining rate 3, 5 and 7, the numbers of SNs in both models are close in averages. However, at joining rate 4 and 6, the multiple-link protocol is having more SNs than the single-link protocol.

Fig. 8.29 confirms that the single-link protocol has linked a higher number of SCs. However, this is due to the fact that the multiple-link protocol takes two steps to establish a single link and then upgrade to multiple links. So each SC in

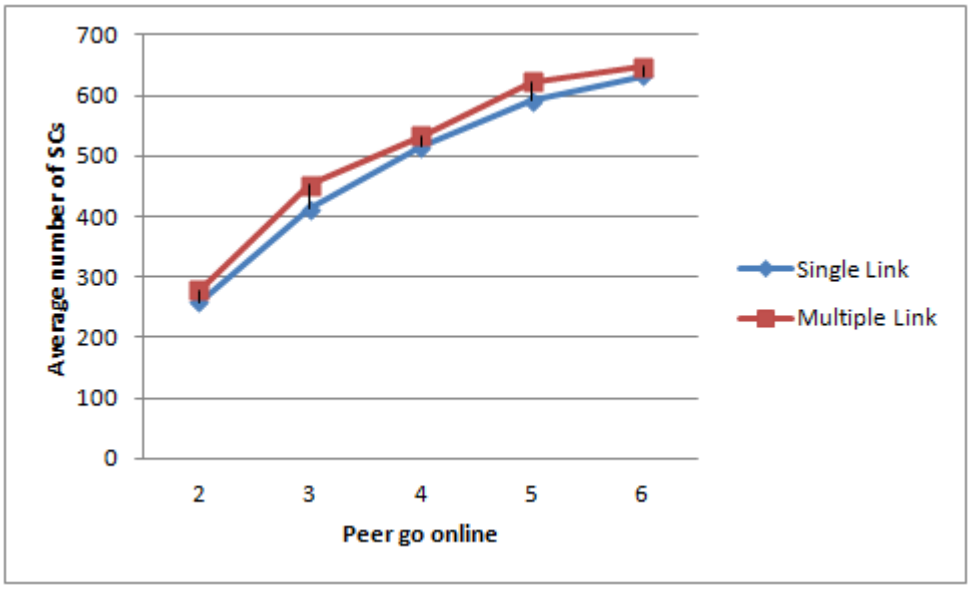

Figure 8.27: Average number of clients

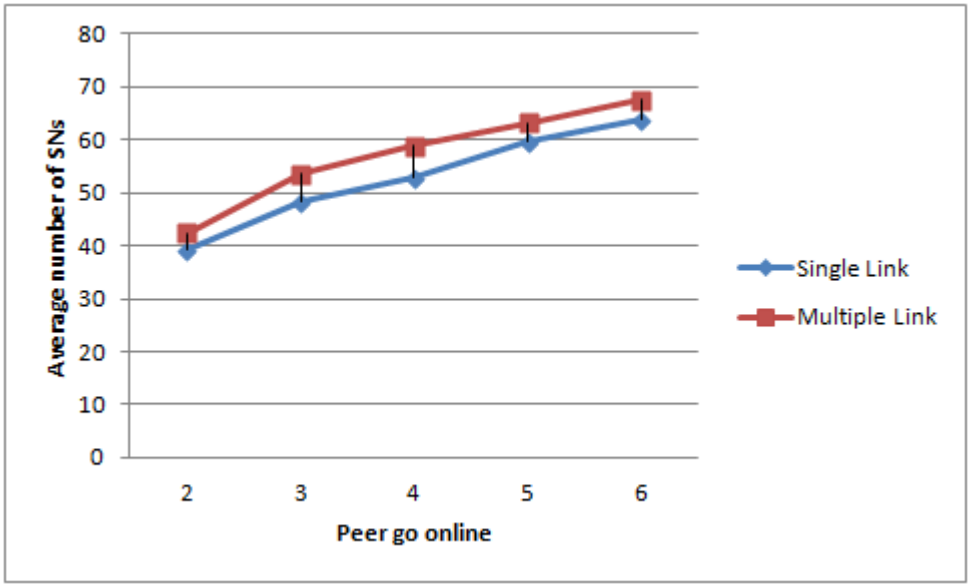

Figure 8.28: Average number of SNs

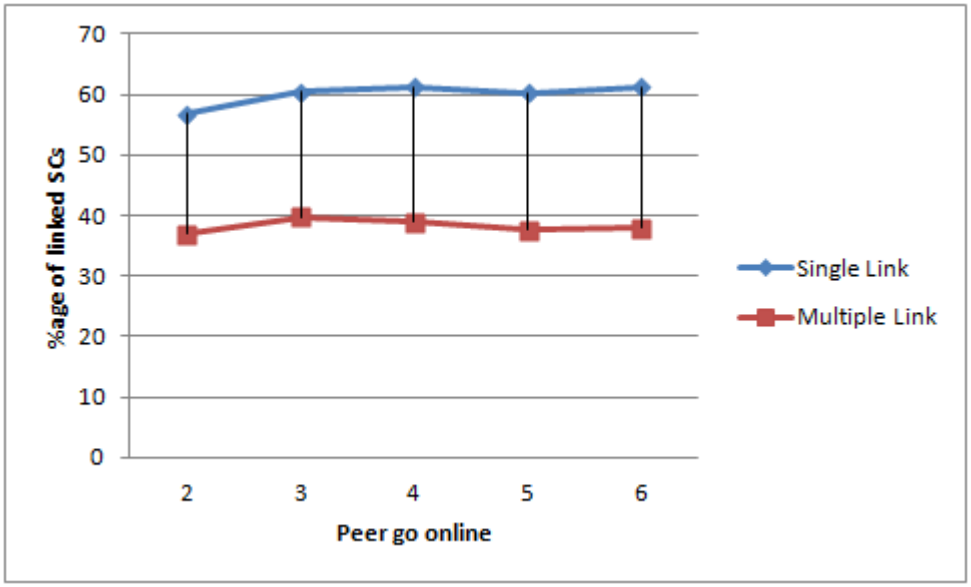

Figure 8.29: Average number of clients linked with SN

the multiple-links protocol is connected twice.

Fig. 8.30 shows that the single link protocol has a higher number of SCs that are happy with their current SN than the multiple-link protocol. Too many connections may result in a congestion of the network as each link costs bandwidth. In the multiple-link protocol, the cost of connections doubles which directly affects the number of happy SCs in the model. However, the multiple-link protocol is more robust in case of reconfiguration as an SC will never be disconnected until both of its SNs leave at the same time.

8.7 A New Protocol for P2P VoIP

Based on the results of the experiment 1 in Section 8.3 we observed that the link local approach has produced a higher number of happy SCs, but the average numbers of linked SCs is lower. This shows that the link local approach increases

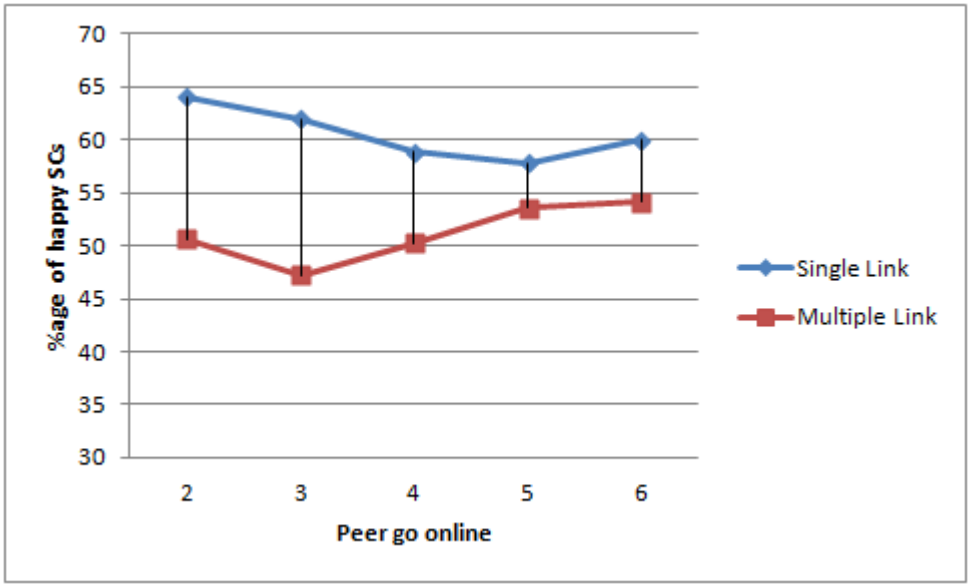

Figure 8.30: Average number of clients happy with SN

the waiting time for connection, which may not be acceptable to users. On the basis of these results we define a new protocol which makes a first attempt to link to a local SN, and if a local SN is not available or could not accept new connections, links an SC to a random SN.

The feature tree in Fig. 8.31 shows three options for SCs linking to a single host SN. The first option links on the basis of the random approach whereas the second approach first tries to link to a local SN, and if not successful will link to a random SN based on bandwidth. The approach links an SC with a local SN. Further, these approaches use the dynamic protocol of SN promotion and a single GT rule lets clients go online, with a random number of SCs behind firewall. We use the same parameters as in previous experiments and vary the joining rule rates $x \in \{3, 4, 5, 6, 7\}$

Fig. 8.32 shows the average number of clients online. This confirms the local approach and the new protocol result in averages that are close. The client

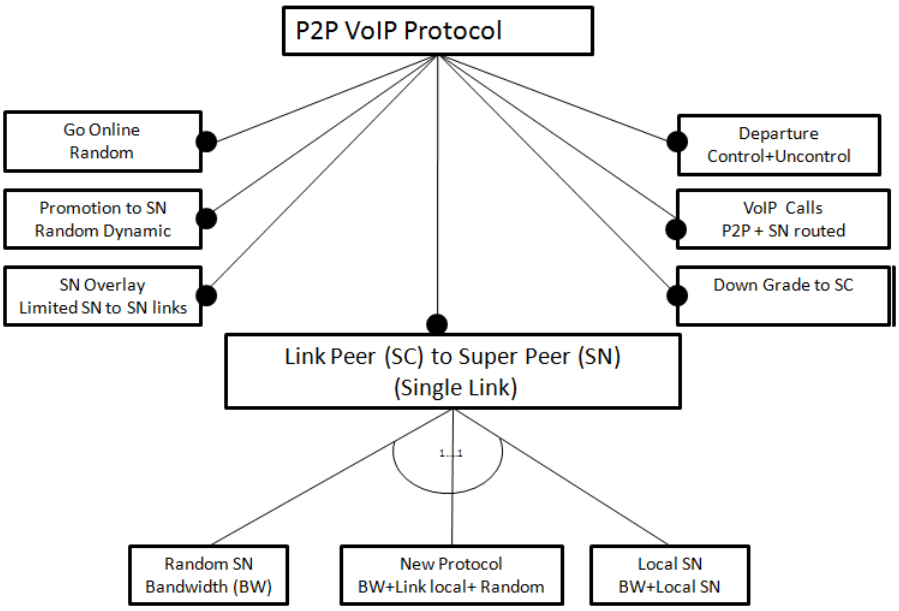

Figure 8.31: Feature tree for new VoIP protocol

population of the random SN approach is also close to the averages of the other two. Fig. 8.33 shows the average number of SNs promoted. The results shows that the new protocol has a lower number of SNs. The link-local and random approach have same number of SNs at joining rate 3. Further at joining rate 4, 5 and 6 the link-local approach has slightly more SNs than the random one.

Fig. 8.34 shows that the random approach is still leading with the highest number of SNs while the new approach stands at second position. The link local approach still has the lowest number of linked SCs. This shows that the new protocol has managed to link a good number of SCs while at the same time benefiting from the possible number of local connections. This is confirmed by the results in Fig. 8.35. The graph shows that the new protocol is not far away from the link local protocol in terms of local connections. This shows the improvement that this new protocol has made with its combination of random and local SN

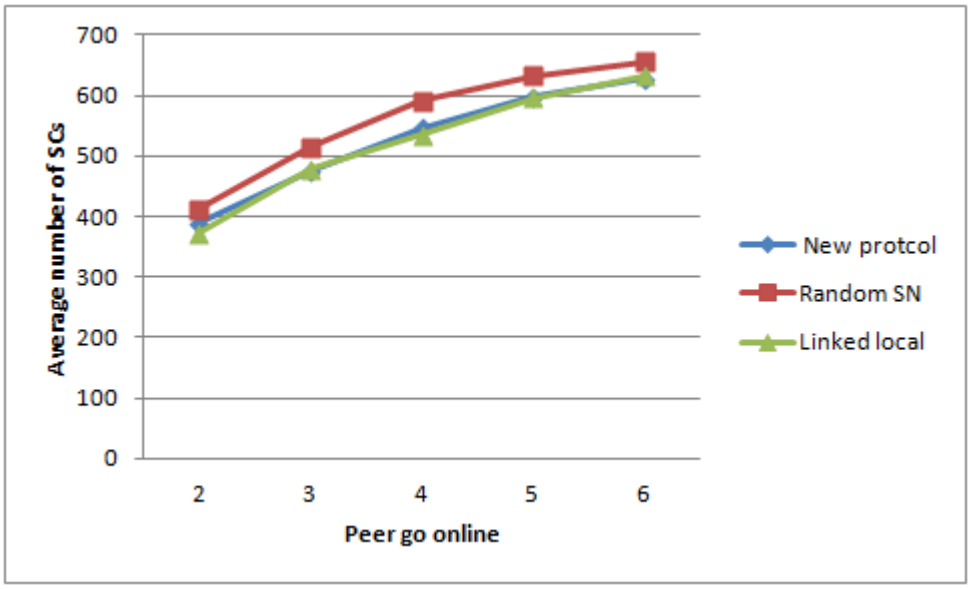

Figure 8.32: Average number of SCs

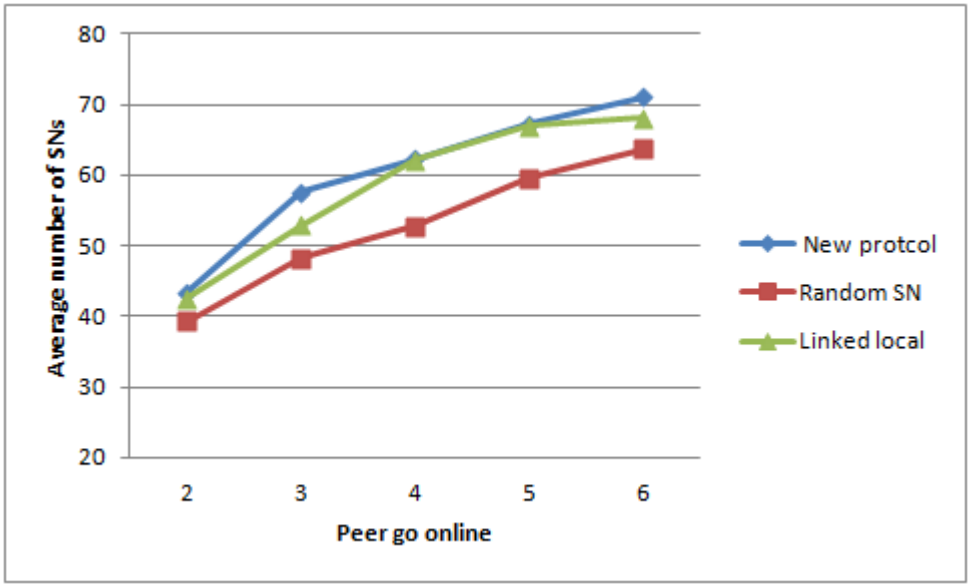

Figure 8.33: Average number of SNs

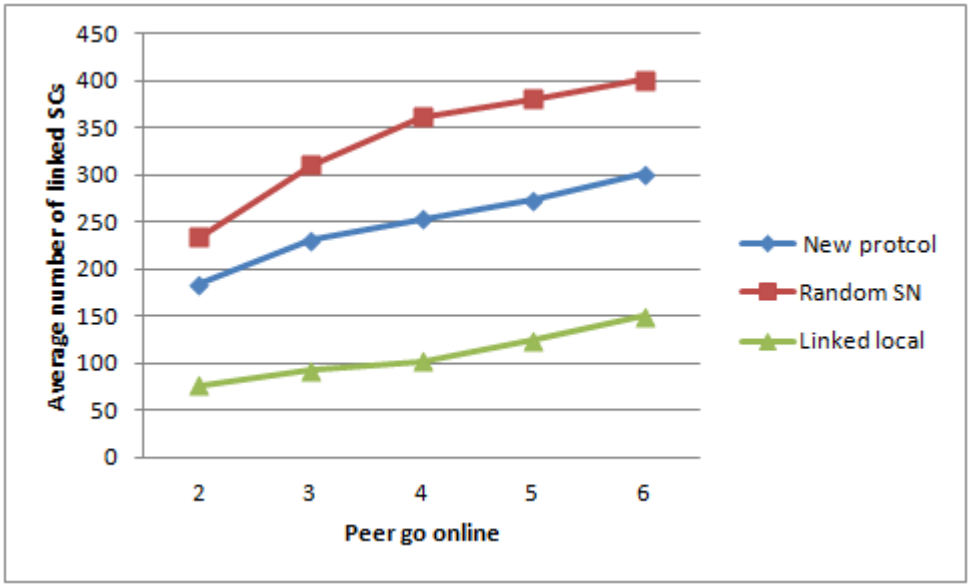

Figure 8.34: Average number of SCs linked with SNs

selection.

The final simulation results in Fig. 8.36 show the average number of happy clients in the model. The graph shows that the new protocol has a percentage of happy SCs between 72% and 68%, which is significantly more than the random SN approach. At the same time this percentage is not much bellow then the link local approach.

Thus, this experiment confirms that by combining the good features of various approaches a new approach can be developed and compared against the parent approaches. Further, this experiment has revealed that combining the features of the random and link local approach has resulted in a new protocol which has overcome the shortcomings of both approaches. The new protocol has now more linked nodes than the local SN, which reduces the waiting time for an SC to find a host, while at the same time showing a percentage of happy SCs which is higher than the random SN approach. Further, the average number of local connections

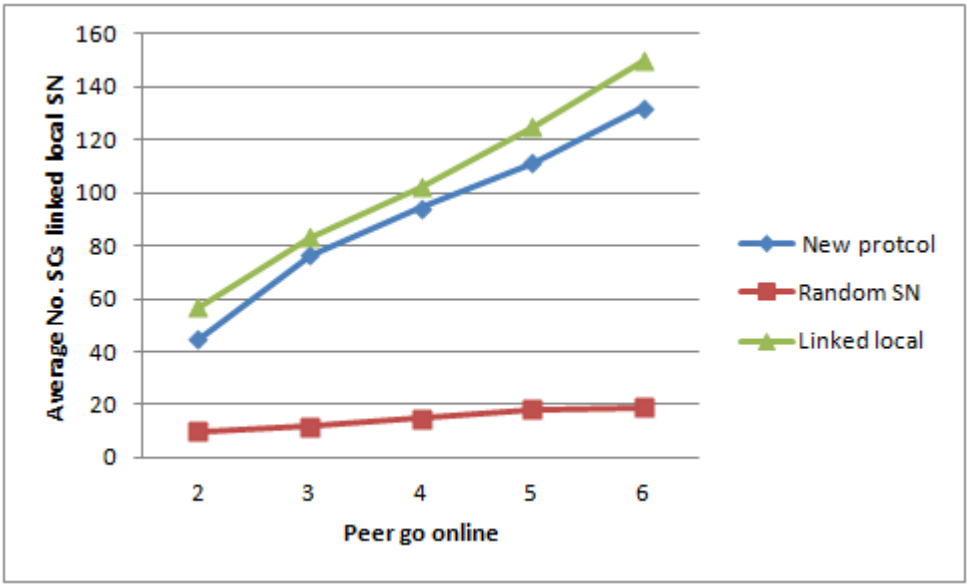

Figure 8.35: Average number of SCs linked with local SNs

in the new protocol has increased with respect to the random SN approach.

8.8 Summary

In this chapter, we have compared various alternative approaches to selection of SN, super peer promotion, super peer selection and load balancing, single or multiple links. Based on simulation results we commented on each of the approach. We measured the performance of each approach in terms of clients successfully linked with SNs and the number of SCs that are currently happy with their host SN. We developed a new protocol based on combining good features of other approaches and simulated the new protocol with the same parameters and environment. We compared the results with those from the parent approaches and observed the progress made.

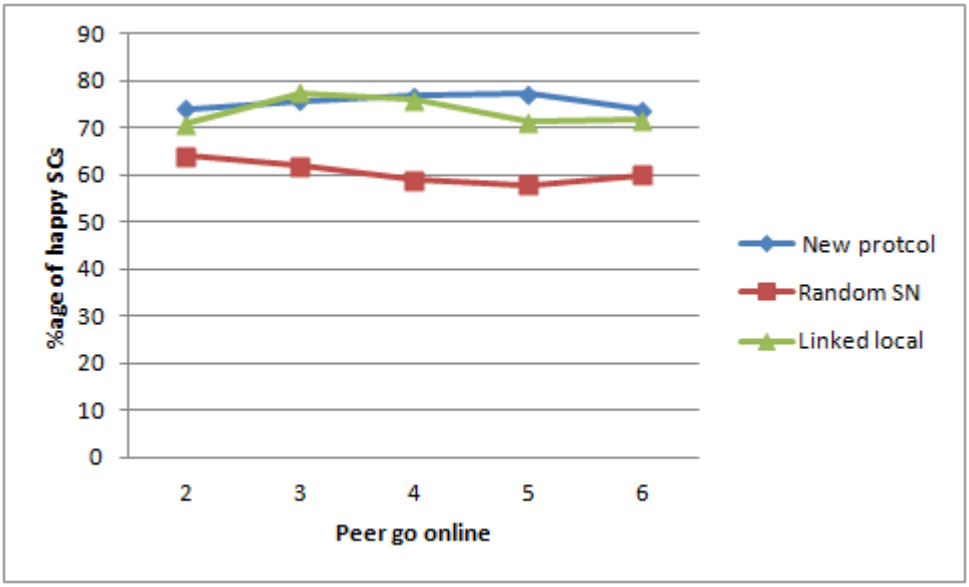

Figure 8.36: Average number of SCs happy with SNs

Chapter 9

Model Validation

In this chapter, we are going to validate the model, using statistics from real Skype network and experimental data. The aim is to build confidence in the model using methods like animation analysis, event validity, face validity and data validation. This chapter also comments on the performance of the alternative approaches. In the last part of this chapter, we discuss and evaluate the possible threats to validity of the simulation results.

9.1 Model Validation and Verification

Verification and validation of the simulation model is one of the core activities in modelling and simulation. There are various methods in the literature for this purpose [10; 139]. In this study, we have used static and dynamic verification techniques in order to make sure that the model has correctly been converted into VIATRA2. The static technique has ensured that model and rules are error free and the right syntax has been followed. The dynamic verification technique has ensured that each rule does the task it was suppose to do.

Similarly for the purpose of model validation, we have used animation, face validation, event validation as well as compared the model with real system data [10; 139]. Animation confirmed the operational behaviour of the model. In order to implement this technique, we have used the GraSS visualizer which displays graphically the operational behavior as the simulation moves forward through time. For example, the operations of peer registering, joining, being promoted, etc. are animated graphically, observing the changes to the network graph.

The face validation technique has been implemented through discussions with experts while presenting papers from this thesis at international conferences. Event validation is the comparison of event occurrences in the simulation with those in the real system. Here, we benefit from the GraSS visualizer, which helps to see the sequences of events. The details of the real system were taken from experimental studies and reverse engineering techniques discussed in Chapter 10 of this thesis.

In order to explain the process of event validation for model validation, we use a UML communication diagram to show the sequence of events in the real

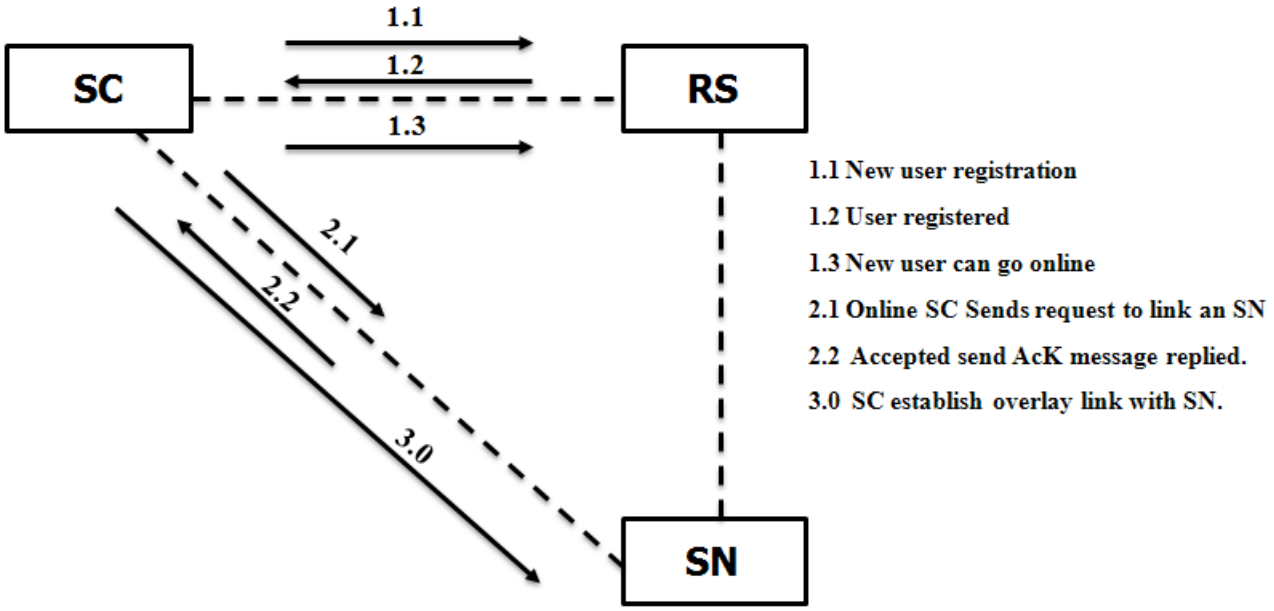

Figure 9.1: UML communication diagram

system. The same sequence is then observed in the simulation model using the step-forward facility in the GraSS visualizer. In Fig. 9.1 a UML communication diagram is used to show the sequence of events to be observed in the simulation model. These tests have confirmed that model event sequence matches those in the real system.

In order to conclude this testing process and confirm the rates and distributions associated with GT rules, we performed a comparison with the output data of the real system. Although collection of real statistics from Skype's proprietary protocol is complex, previous authors [15; 53] not only provided crucial data for model development but have given enough statistics to help to validate the model. Apart from their work, forums like [150; 171] provide Skype statistics with 15 second intervals as well as weekly, monthly and three monthly global

statistics. The analysis in [15] is based on a real study and results show that over a period of 24 hour the average number of clients that join the network varies by 40% of the active registered accounts. Further, real statistic of Skype [150] confirms that an average of 20% to 50% of registered users go online. Due to the time difference, their observed average is 30%, and the all-time peak is 41% (see Chapter 5). Further, the average number of super peers in this period is below 24% of the active population. Similarity, the author in [171] states that in December 2010 there were 1.4 million super peers and 25 million users online, when the Skype network crashed due to non availability of super peers. This figure confirms that almost 6% of the online users were super peers, and still the Skype network crashed. This supports that the Skype super peer population must be over 6% of the online users, in order to make the network stable.

Keeping in mind these averages, we plot the average number of clients that go online during our simulation period of 24 hours. We have fixed the number of registered users at 1000. The simulation results at Fig. 9.2 show the percentage of clients that go online is between 22% to 47%, which confirms the real statistics and previous experimental results.

Fig. 9.3 shows the percentage of the SNs promoted in the course of the simulation which is well over 6% and below 24%. This further confirms that the behaviour of the model matches that of the real system.

9.2 Evaluation of Requirements

In this section, we evaluate our approach based on the requirements in Chapter 1.

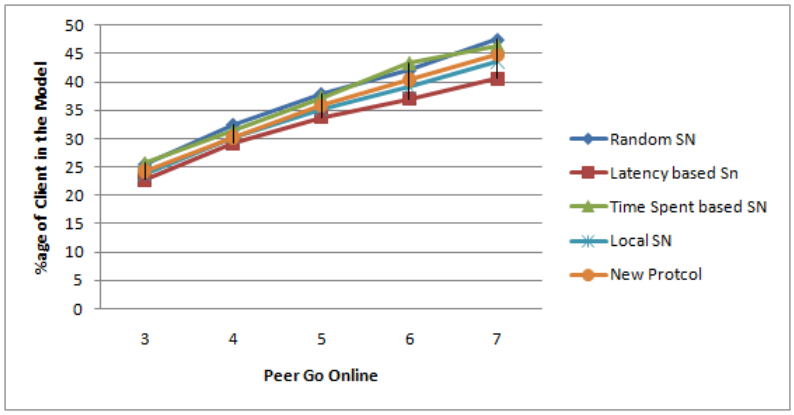

Figure 9.2: Average number of SCs go on-line

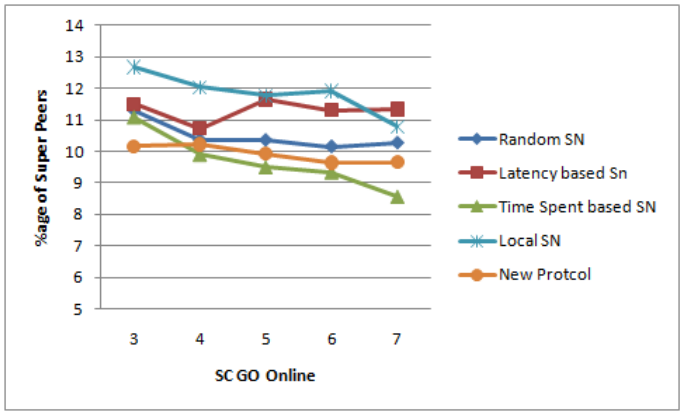

Figure 9.3: Average number of SNs promoted

9.2.1 Peer Promotion and Demotion

Dynamic peer promotion and demotion requires to change the role of the client in the network if its current attributes meet the requirements of the role. For example, Skype clients with sufficient bandwidth can be promoted to super peer and in this role they will serve a number of dependent clients. Similarly, if the client is no longer in a position to serve as super peer, the super peer will be demoted dynamically to the client only. The same concept has been used in the model of this thesis, which dynamically promoted and demotes clients using two GT rules.

Super peer promotion experiments in Section 8.4 evaluated three different approaches. The simulation results confirm that peers were promoted and demoted as required.

9.2.2 Model of Churn

In the model, a Skype client or super peer can join the network once registered. After going online the departure of SCs and SNs is handled in two ways. The first approach implemented follows a cooperative model of churn, in which SCs and SNs can leave the network by informing the host SN in case of SC departure and in case of SN informing both dependent SCs and SNs. This model of churn enables the host, dependent SC and neighbours to reconfigure their topology in time. The second model of churn implements the uncontrolled departure of SCs and SNs. In this model the SCs and SNs leave the network in a selfish manner, and it is up to the dependent peers, to reconfigure their topology to stay connected.

The simulation experiments in Chapter 8 use both version of churn. In order

to confirm that the model used churn, we look into the application of the rules. Experiment 8.3 shows that in the random selection approach, the rule modelling SCs cooperative departure has average applications of 202.500 while the SC uncontrolled rule has 6.677 applications. SN cooperative departure has average applications of 19.500 and uncontrolled applications of 1.500. These results show that SNs are comparatively stable as compared to ordinary clients.

9.2.3 Reconfiguration

As a result of churn we need to reconfigure the topology of P2P networks. In cooperative departure, their current overlay link will remain intact until they reconfigure themselves. However, in the selfish model of churn, the network needs to reconfigure fast enough so that QoS is not affected.

The simulation in Chapter 8 uses the `ReconfigureandlinktoaNewRandomSN`, `RecoverBandwidthduetoCrashingofSC` and `RecoverBandwidthduetoCrashingofSN` GT rule to reconfigure. In order to further confirm this we look into applications of the rules in Experiment 8.3. They show that during random selection of SN at joining rate 3, the GT rule `ReconfigureandlinktoaNewRandomSN` has average applications of 19.33, the GT rule S$BandwidthduetoCrashingofSC$ has 1.167 average applications and the `RecoverBandwidthduetoCrashingofSN` has 1.500 average applications. This confirms that the model has implemented the reconfiguration mechanisms proposed in Chapter 4.4.2.

9.2.4 Direct and Relay Calls

Two types of VoIP calls are supported. The first is direct and the second is the relayed call. The relayed call is a unique feature in the case study of this thesis, which enables an SC behind a firewall to make a VoIP call to another online user. The relayed calls are processed by the super peer in the model. Four types of relayed calls are supported. The first is between two SCs connected to the internet with a firewall, when each has dependency link with a unique SN. This type of relay call involves four types of nodes, consisting of two SNs and two SCs. In the second type of relay call, one of the call participants is connected through firewall and the second is enjoying a static IP connection. In this call only three participants are involved consisting of two SCs and one SN. In third type of call both of the participants are connected to the internet through firewalls, sharing a single host SN. The last type of call is between a firewall-enabled SC and SN.

In the model, GT rule Call application shows P2P direct calls in the model, whereas GT rule RouteCall shows the relayed calls. To confirm we look into the Experiment 8.3 results. They show that at joining rate 3 the average number of calls including direct and relayed is 35.352.

9.2.5 Modelling Latency

Time stamped packet is used to find the latency between an SC requesting connection with an SN. The packet is transmitted by SC and once the SN receives this packet, based on the awareness of its current bandwidth, it can either accept or reject this connection request. If the SN replies, SC uses this packet to find the round trip time taken by the packet, and if this falls within the acceptable

range, a link is established with this SN, otherwise the SN is rejected.

Experiment 8.3 uses this time-stamped packet to implement the selection approach proposed in Chapter 4. The rule application shows that at joining rate 3 in this experiment rule Reject (DnY) has 12.500 applications, which shows the number of rejected connections. In this experiment, GT rule AcK has 193.500 applications, which shows the acceptance messages generated by SN. In this experiment, GT rule reject on latency has 12.500 applications, which shows the number of SNs rejected based on latency.

9.3 Threats to Validity

In this section, we are going to present the possible threats to the validity of the simulation results. The case study in this thesis is subject to threats from the modelling to the encoding stage. Some of the threats we identified are explained in the following subsections.

9.3.1 Start Graph

In order to see the effect of the start graph on the simulation results, we use the super peer selection experiment in Chapter 8. The start graph used in this experiment consists of one super peer and a single registration server. To analyse how the choice of start graph can change the result, we develop a second start graph that consists of five super peers, a single registration server and two online clients. We run the experiment again with all GT rules using the same distributions and rates. The simulation time is 2.4 hours and batch size is 6. Fig. 9.4 shows the two start graphs. We run the simulation at joining rate 3 on both start

Figure 9.4: Two start graphs/ trees

graphs and observe using probe rules the number of SCs, the number of SNs, the number of SCs happy with the current selection of SN, the percentage of happy and linked peers.

The simulation results show that despite the average number of SCs remaining the same, the start graph with more SNs has managed to link slightly more clients. The reason behind this could be that, as the number of SNs increases in the model, the chance of accepting a new connection also increases. The results are shown in Fig. 9.5. Overall difference is small.

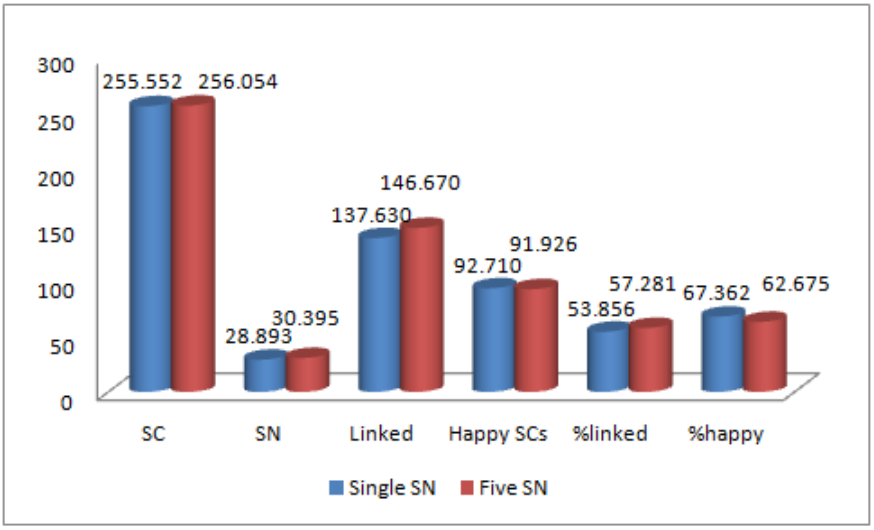

Figure 9.5: Simulation results for two different start graphs

9.3.2 Impact of Reducing Bandwidth Requirements for Super Peers

In order to see the impact of bandwidth requirements on the simulation results, we develop a new set of rules for the SC connection to SN experiment, changing the requirement for promotion from 1.5 Mbps to 1.0 Mbps. These rule use the dynamic version (see Section 8.12). We run the simulation with joining rate 3 and simulation time 2.4 hours. The batch size is 6. We observe the average number of SCs, average number of SNs, average number of linked SCs, Happy SCs, percentage of happy SCs and percentage of linked SCs. The simulation results show that reducing the requirements for promotion has affected the number of SNs in the model. The number has increased at client joining rate 3. The SNs availability has further increased the connection process and has managed to link a large number of clients as compared to Fig. 9.5. The results shown in Fig. 9.6 suggests that beyond the expected improvement, the overall trends remain the

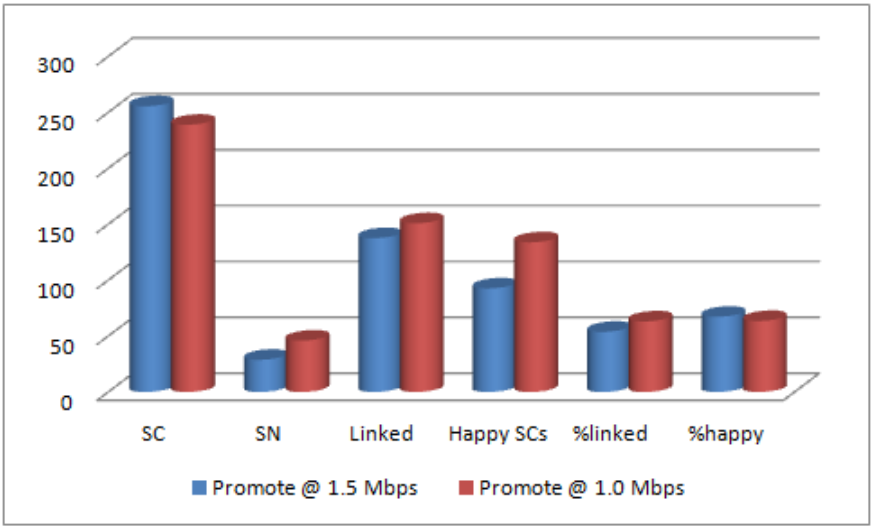

Figure 9.6: Impact of reducing bandwidth constraint

same.

9.3.3 Impact of Simulation Depth

In order to discus the impact of the simulation's maximum depth. We use the random SN selection model with a joining rate of 2. When we run the simulation in time mode for simulation time of 19.2 hours with batch size 6, the simulation experiment managed to run up to a depth of 11062 steps. To analyse the impact, we change this depth to 22124 and run the simulation in step mode. The results in Fig. 9.7 does not show a significant change. This further confirms that running simulation upto 19.2 hours results in a steady state.

9.3.4 Impact of Probability Distribution Selection

This section discusses the impact of probability distribution for GT rules. In order to understand this, we change the probability distribution for the GT rule

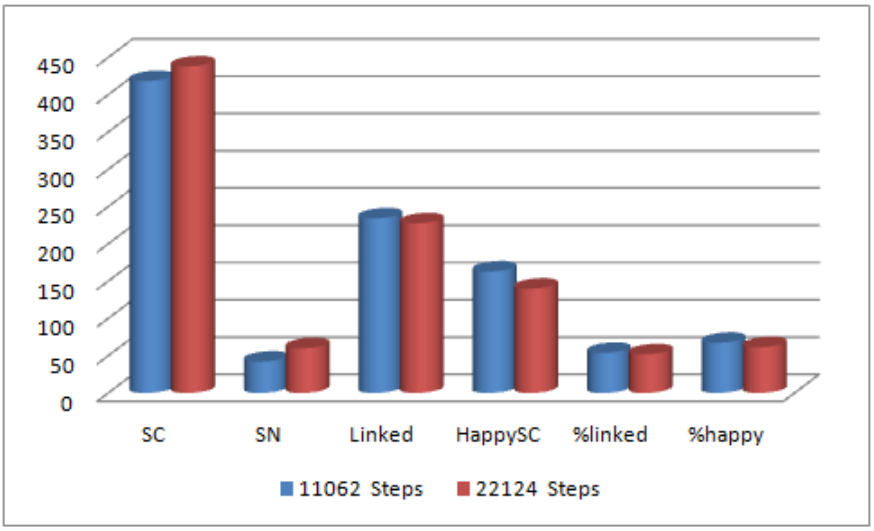

Figure 9.7: Impact of maximum depth of simulation run

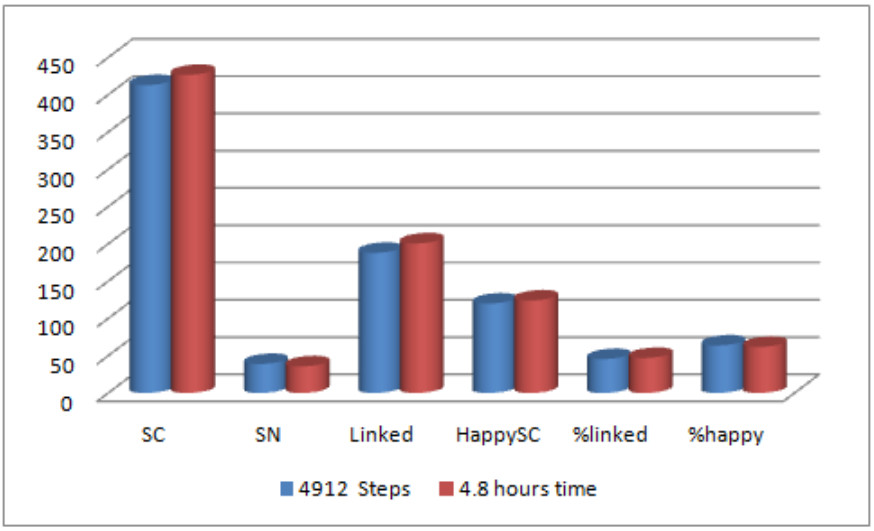

Figure 9.8: Simulation time and steps

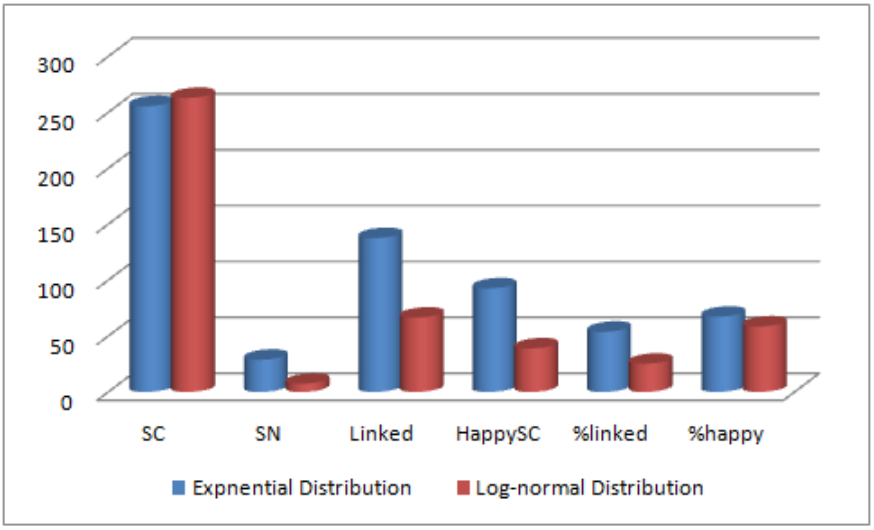

Figure 9.9: Impact of probability distribution

for promotion to super peer. The current exponential probability distribution is replaced by the log-normal distribution with a μ=0.100 and σ^2 = 2.0. We run the simulation for random selection of SN with simulation time of 2.4 hours and batch size 6 and observe the impact in terms of the average number of linked, happy,and total number of SCs and total number of SNs. The simulation results in Fig. 9.9 show significantly changed results. The number of super peers has reduced. This has not only affected the number of linked SCs in the network, but also reduced the percentage of happy SCs. However, the exponential distribution seems to be the right choice as we are not sure when a SC will be selected.

9.3.5 Confidence Intervals

In this thesis as we have used less than 10 runs in each simulation. We use t-distributions to compute the 95% confidence intervals for the parameter of interest. The Fig. 9.10 shows the 95% confidence interval computed for the average

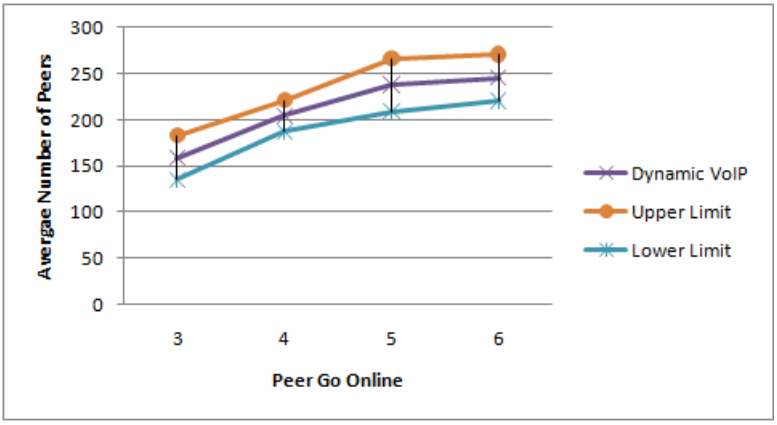

Figure 9.10: 95% Confidence interval

number of peer or clients in the model. The figure shows that the confidence boundaries are not too wide, because in each of the runs the average number of clients online, are close to each other.

9.4 Summary

This chapter presents validation of the model and evaluation of the approach used in this thesis. The validation is performed through techniques like event validation, animation and comparison with data available from the real system. The requirements of the model and approach are evaluated by looking into the GT rule applications in simulation runs.

The last part of this chapter discussed the threats to the validity of the simulation results. The impact of the start graph shows very little effect on the overall results. The impact of reducing bandwidth requirements for promotion show that the number of SNs in the model has increased. The result confirms that this can effect the number of linked and happy SCs in the model. The impact of simulation depth shows that increasing the number steps changes results

significantly. The number of SCs going online has increased along with number of SNs. However, the numbers of happy and linked SCs have changed by a smaller value. The last threat confirms that selection of probability distributions has an impact on the results of the model. We believe that the exponential distribution is the right choice for promotion rules. The 95% confidence interval shows that results of runs are close.

Chapter 10

Comparison to the State of the Art

This chapter starts by discussing existing studies of super peer-based P2P protocols. We categorise this work into studies of protocol architecture, protocol reverse engineering, super peer promotion, super peer selection, peer and super peer dynamics, and the reconfiguration. Due to the large scale and geographic distributions of peers in P2P network, the evaluation of new and existing algorithms is one of the prominent hurdles for researchers and developers. Today simulation is the most widely accepted technique for the evaluation of such systems, in contrast to mathematical modelling and real time monitoring. The literature survey shows that there are a number of P2P overlay simulators developed by various research groups, but many protocols developer opt to use their own custom-built simulators. This raises a number of issues such as reproduction and validation of their results and the comparison of approaches in a similar environment. In this chapter, we discuss some of the current simulation approaches and summarize them against a common set of attributes.

10.1 Super Peer-based P2P

In this section, we are going to study the existing literature on super peer based P2P protocols, categorised according to protocol architecture, super peer promotion, peer dynamics and the reconfiguration.

10.1.1 Skype's Super Peer-based Architecture

Skype claims that it is better than other VoIP software in the market because of its high call completion rate and superior sound quality. As Skype is a proprietary protocol, its exact architecture, is not published. However, to confirm Skype's claims various traffic monitoring and analysis approaches have been used. The authors [15; 17; 53] conducted an experimental study of the Skype traffic with the objective to enhance the understanding and significance of successful P2P VoIP systems, as well as to provide experimental data that may be useful for future design and modelling of such systems. In their experiments they analysed some of the key features of the Skype architecture such as login, NAT and firewall traversal, call establishment, media transfer, codecs, and conferencing. They used three different network setups and conducted their experiments in a black-box manner. The model in Section 6.1 captures the essential features from recommendations [15; 17; 53]. Their results help to validate model as described in Section 9.1

The results in [15; 53; 98] indicate that the structure of the Skype system follows other P2P systems, particularly KaZaA, but there are significant differences in operation. Further, they observed that each online user in the Skype architecture is client unless promoted to the extended role of super peer.

In [176] the authors discuss the super peer-based network in detail in order to fully understand the behaviour. Their work concentrates on fundamental characteristics and performance issues. Their main contribution is a practical guide and a generic procedure for designing super peer based networks. In other words, the super peer-based architecture seems to be working better than other P2P approaches. Further, it was also observed that a client behind NAT or firewall can access and use the network, which makes this application more robust than other P2P applications [98].

The graph model and transformation rules in Chapter 6 are based on the discussion in [15; 53; 98; 176]. The overlay among SNs is based on the recommendations of [176]. The discussion in Section 9.2.2 confirms that super peers are stable as compared to clients. The model supports P2P direct calls between clients connected without firewall and SN-routed calls for clients with firewall. All possible choices of calls are shown in section 6.2.9 and their validation is confirmed in Section 9.2.4.

10.1.2 Protocol Reverse Engineering

The experimental analysis [15; 53; 98] provides an abstract view of the Skype architecture. As the use of Skype increases, network engineers and system engineers feel the need for tools to analyze Skype traffic. One of the reasons behind this need could be the use of user bandwidth by Skype without obtaining prior permission. In order to identify encrypted traffic, one approach Reverse engineering [121]. The core aim of their work is to develop a traffic signature that enables a third party monitoring entity to detect the usage of the Skype application. The

SkypeLogger [8] is one of the tools that uses a traffic signature to identify Skype traffic.

The work reported in this section is used in the model to capture the essential features of the super peer-based architecture. This work has been helpful in identifying the sequence of events in the real network. The tests in Section 9.1 have confirmed that our model event sequence match those in the real system.

10.1.3 Super Peer Promotion and Call Relays

The experimental studies in [15; 53; 98] confirm that super peers are more stable than ordinary clients. The selection of super peers amongst the online clients is a critical decision and the overall performance of the network depends on the quality of the super peers [108]. There are many recommendations ranging from super peer health to time of promotion. The experimental studies in [15; 53] confirm that clients with more bandwidth, memory and CPU power are promoted to super peers. There are other algorithms [35; 111; 175] to enhance performance by promoting peers with a single parameter of fast reliable internet connection. The traffic analysis in [15; 53] states that being super peer is not the choice of the peer's owner, so a network can have as many super peers as possible. However, some approaches [99] suggest that a fixed ratio must be maintained between super peers and peers. Their contribution is based on the idea that super peer should be arranged only when required. The author in [99] proposes promotion to super peer based on bandwidth and time spent in the network. However, [168] suggests promotion based on a parameter referred to as capacity.

The author in [172] studies the VoIP quality of Skype and states that to act as

super peer costs bandwidth. In order to validate his assumption the author conducted large-scale end to end measurements, starting from a systematic analysis of the Skype super peer network. In their work they further investigated performance by applying access capacity constraints and the AS policy constraint. They observed that in the absence of free riding, only 20% of super peers that are located in stub ISPs [89] significantly affect the quality of calls in VoIP sessions. Their work concludes that before building a super peer-based architecture, we need to consider the operational model of the Internet. As stated earlier ordinary clients in super peer-based architectures depend upon super peers. This dependency is limited to global index hosting and query forwarding when a client is connected through static IP based internet connection. However, when the client is behind NAT or firewall, first efforts are made to Travers NAT or firewall by using STUN like protocol but when this attempt is failed then actual calls are also transferred through the super peer [88; 154].

The model and GT rule in Chapter 6 promote a client to super peer if the current bandwidth is more than 1.5 Mbps. Three alternatives proposals for promotion are modelled using the feature diagram in Section 6.2. The same alternatives have been implemented using transformation rules in section 6.2.4. The simulation results in Section 8.4 confirm that the need-based protocol [99] seems to be ideal choice for P2P VoIP as opposed to many super peers [15; 53]. This confirms the analysis of [172].

10.1.4 Peer Linking with Super Peer

In [15; 17; 53] the authors state that ordinary clients after authentication select one of the super peers as host. This host's importance is further increased if the client is behind NAT or firewall. In order to select a super peer [15] the registration server provides a valid list of super peer IPs to clients. The simplest approach is to select a random super peer, but due to limited free bandwidth or distance in the network, quality may be less than optimal.

In [99; 108; 121; 174] the authors suggest that the bandwidth of the super peer shall be used to decide super peer connection. However, their approach seems to fail when the super peer has free bandwidth but the path between the client and the super peer is congested, resulting in latencies or packet delays that are not acceptable for P2P VoIP traffic. To overcome this issue, in [168] the author suggested that bandwidth and round trip time (RTT) shall be used for this decision, while in [180] the authors proposed a delay-aware P2P system (DAPP). Their approach is based on the concept that shortest paths in overlays cannot ensure efficiency in the real physical media. DAAP suggests sending ping-type time stamped messages to get an idea of the actual latency. Similarly, in [9; 99] the authors state that peers from the same physical network provide lower latency then the remote peers in different geographic location as well as different physical networks. All these approaches maximize the quality of service, but are missing one important characteristic, the stability of the super peer. In [108] the authors proposed a super peer-selection strategy based on dynamic capacity and content similarity. The authors validate their approach using simulation and compared it with other approaches concluding that their approach is performing better.

In [99] the author state that bandwidth and time spent by super peers in the network shall be used as decision criteria for linking to super peer.

The approach recommended in [99; 108; 121; 174] is modelled as a random selection of SNs shown as a choice in the feature diagram in Section 6.2. Similarly, the approach recommended in [168; 180] is modelled as the latency based selection and is available to the model as second alternate choice in Section 6.2. The local SN approach [9; 99] is modelled as the third alternative in Section 6.2. The stable SN selection [99] is modelled as the fourth alternate choice in Section 6.2. The simulation results in Section 8.3 confirms that random selection performs better in terms of linking SCs. The other approaches improve the performance but increase the waiting time for connection.

10.1.5 Load Sharing and Super Peer

Load balancing can improve the performance of unstructured P2P networks, but there exist a number of strategies for load balancing in P2P systems. In [102] the authors proposed an algorithm that finds overloaded peer and moves some of their load to less loaded peers. In their algorithm they do not take into account topology or hash tables. They compare and validate their approach using simulation. In order to comment on the performance of their approach they compared it with other two approaches: neighbour load sharing [82; 102] and uniform random load sharing [20; 64; 102].

[187] proposed a load balancing scheme for self-interested P2P networks. They consider that each peer in the network is selfish and is always interested to minimize the query response time for itself rather then considering the global per-

formance. They verify their approach using theoretical proofs and later on used simulation to validate their results.

[30] proposed an approach to restrict the load per super peer by putting a strict limit on the number of peers they can support and compared their approach with others. In [45] load is balanced by selecting super peers, so that is if always uniformly distributed. In order to evaluate their approach they use simulation. They claim that results have shown significance improvements.

In [130] the authors proposed a coordinated load management protocol for P2P systems coupled with a federated grid systems. Users, which include resource owners and consumers, belong to multiple domains. They validate their approach using a simulation.

The approach recommended in [102] is modelled as an optional feature in Section 6.2. The transformation rules in Section 6.2.6 represent this approach. The simulation results in Section 8.5 match the results in [102]. Showing that the number of happy clients increases.

10.1.6 Churn

P2P networks are highly dynamic with peers and super peers joining and leaving the network. One of the core functions in P2P networks is to recover the topology or re-link disconnected peers such that quality is not affected. Different protocols handle reconfiguration using their own approach. Some of the fundamental solutions proposed include keeping as many overlay connections as possible [61]. In this approach stochastic graphs and model checking are used to validate the model. The real Skype studies [15; 17; 53] show that clients try to find a new

super peer as soon as possible if their current connection is lost.

In [116] the authors proposed a new protocol, which uses a backup peer to replace the failed peer. Additionally, their protocol makes sure that the backup node is in a position to support the incoming load.

The model in Section 6, reconfiguration rules in Section 6.2.10 and connection rules in Section 6.2.3 implement the recommendations of [15; 17; 53; 61; 116]. The simulation results in Section 8.6 show that multiple connections can reduce reconfiguration but cost bandwidth which reduces the number of happy clients. The reconfiguration rule validated in Section 9.2.3 shows disconnected clients are reconnected in the model.

In [167] the authors propose two different algorithms for individual peers to self-configure P2P network overlays in various network environments. For a small scale homogeneous network, an optimizing peer selection protocol is proposed. For large scale heterogeneous network a peer-power based selection protocol is proposed. Running two different algorithms as well as peer power computing is a complex task.

In [111] the authors proposed a new generic mechanism for construction of super-peer based P2P networks. Their work gives a detailed strategy for maintenance and reconfiguration. While developing this new protocol, they consider some of the well-known discussions such as in [176]. In their approach nodes exchange probes with randomly selected peers. Based on these reconfigures the topology according to the requirements of the application. They claim that their protocol is extremely efficient as well as robust and able to reconfigure the topology where nodes continuously join and leave the network. They further confirm their reconfiguration algorithm by removing the super peers from the network,

observing the repair process by simulating this protocol using the peerSim simulator.

The model in Section 6 implements reconfiguration mechanisms through the GT rules in Section 6.2.10 which are similar to those proposed in [111]. Although the model does not send probe messages to confirm the status of the link, it uses rules which become enable if the connection is broken. The model in presence of churn (as validated in Section 9.2.2) reconfigures the disconnected clients and super peers as shown in simulation results in appendix A.

10.2 Modelling and Simulation

The analytical approach requires a mathematical model for P2P system, but in most cases, the model is based on simplifying assumptions and presents a very basic view of the real system [113]. The need for simple models makes it difficult to capture a true sense of real world P2P networks [77]. [129] states that this approach has only been successful to model Bit Torrent [27] in the presence of basic model churn.

As an alternative one could consider testing the real systems, but these at large-scale in terms of nodes and geographic distribution, even if we limit our experiment to a sub-area [140]. Sub-area experiments with real systems require significant resources which are costly and may be exposed to node failure or software version control issues. Apart from this, external factors such as cross traffic and changes to the network layer are beyond control and can influence and even change the results. However, despite these reasons some test beds are still used, such the Planet-lab [125].

Analytical approach and real testing are not the first options for research and development in P2P systems. Simulation is one of the prime tools for study, analysis and comparison of P2P approaches and algorithms. The cost of implementation of simulation models is less than for large-scale experiments. Further, if a simulation model is carefully constructed, it can be more realistic than any mathematical model [136]. It is worth mentioning that one can not simply accept the results obtained from a simulation. They are to be validated, and one of the best choices would be to cross-validate the results of different approaches.

The literature review shows a number of simulators [113; 114; 138; 181] P2P algorithms. Most people have used their own custom-built simulators. One of the reason, claimed is the lack of required functionality. Some of them don't have documentation which makes their use almost impossible, while other don't have mechanisms to gather statistical data [113; 114].

In this thesis, we have gone through a number of simulators starting from the popular PeerSim and NS2. We observed that they are short of some of the key functionalities which are required for our model.

10.2.1 Evaluation of Existing Approaches

In this section, we discuss the parameters that are required by the case study for modelling P2P VoIP Skype like protocols. At the end of this chapter, we present Table 10.1 comparing the existing approaches based on those parameters. Some of these are generic and have been proposed in [31; 113; 114; 138; 159] while others are domain-specific. Table 10.1 is developed based on the information in [31; 37; 47; 76; 77; 110; 113; 114; 123; 136; 138; 146; 160; 177; 181; 184].

1. **Simulation Architecture** we distinguish discrete-event and query-cycle architecture.

   Discrete event: Discrete-event system simulation is the modelling of the systems in which the state variables change only at a discrete set of points in time.

   Query-cycle: The simulation loops through each node, processing queries for respective nodes [113].

2. **Model of Churn and Reconfiguration**

   These criteria include features such as how node behaviour is simulated, and what level of churn is implemented and what distributions are associated to peer dynamics. These criteria further evaluate support for failure of communication channels and reconfiguration.

3. **Usability**

   This criteria is related to how easy to use the simulator or modelling approach is, and whether respective support and training material is available. It also evaluates whether the tool provides an API that is easily understandable, and if changes can be made easily. If it makes use of any domain scripting language, how easy it is to learn. What documentation is available and how easy it is to follow.

4. **Scalability**

   This criterion is related to the number of nodes that could be simulated. Skype has more than 20 million online users simultaneously. So it is neces-

sary to evaluate the tool on the maximum number of nodes it can support. It is also important to check how a protocol scales during simulation.

5. **Simulation Model**

   This criterion concentrates on the model used for simulation, whether it uses a static, dynamic, deterministic or stochastic approach. The terminology definition below are reproduced from [4].

   Static: *A static simulation model, sometimes called a Monte Carlo simulation, represents a system at a particular point in time.*

   Dynamic: *A dynamic simulation model represents a system as it changes over a time.*

   Deterministic: *Simulation models that contain no random variables are classified as deterministic. A deterministic models has a known set of inputs that will result in a unique set of outputs.*

   Stochastic *A stochastic simulation model has one or more random variables as inputs.*

   For example, bandwidth or latency, in a deterministic approach has to be specified from the beginning. In a stochastic model, the peer will be assigned bandwidth randomly as it joins the network and similarly latency will increase if the channel is busy and may reduce if the channel is free.

6. **Support**

This criterion reports what applications can be simulated. The literature shows that P2P is nowadays operating in various domain from file sharing to real-time video streaming.

7. **Modelling Level**

   This criterion states at what level the network is modelled and simulated. There can be three levels: network layer, application layer and user layer. This criterion evaluates each approach on the basis of these three layers. The network layer describe how the overlay topology is formed. The application layer tells us what applications can be used on the network formed by the previous layer. The last and most important layer in P2P is the user layer, which tells us whether user behaver such as selfish or controlled exit is supported.

8. **Simulation Statistics**

   The most prominent feature that we need to evaluate is the representation of the results. It means how easy are the various reports in terms of understanding and editing. If we want to add a new query, is the simulator flexible? Often the designers of the simulator only provide a few custom-built reports which may not necessarily be providing the required information. In some cases it just generates a long log file of events and then that file has to be analysed to produce respective results.

9. **Confidence Intervals**

   How does the simulator report confidence intervals in order to check overall results. Further, their simulator should draw confidence interval for each

measurement and provide details of each run and time taken.

10. **System Limitations**

    This shows how the simulator is performing in terms of memory and processing requirements. If the simulator is using memory and processing power excessively, then scalability could be one of the prime issue. In view of today's technology, we have to check whether a tool is able to make use of the cluster in order to reduce execution time and increase scalability.

10.2.2 Overview of P2P Simulators

In this section, we are discussing some of the popular P2P simulators with their pros and cons.

10.2.2.1 PeerSim

PeerSim was funded through the Biology-Inspired Techniques for Self-organization in Dynamic Networks (BISON) [112; 147]. The core objective of the project was to study complex adaptive systems in dynamic networks. This simulator was designed for large scale overlay networks, but with built-in models for OverStat [72; 110], SG-1 [111] and T-Man [73]. PeerSim supports dynamic networks with large numbers of nodes. PeerSim has the capability to simulate both structured and unstructured P2P. The simulator is designed in Java and is extensible by plug-ins. The simulator architecture is based on cycles. The latest releases support discrete events, but this reduces the scalability of the simulator [138; 147].

PeerSim does not support distributed simulation, but various components in

the libraries of the simulator were designed to simulate graph like overlays. This has enabled this simulator to be extended to support other protocols. PeerSim is configured using a single file. The main drawback is the usability of the tool due lack of a GUI. The ouput reports are basic and for detailed statistics additional code is required. Another drawback is that it only provides detailed documentation for the cycle-based, whereas for discrete event-based simulation there is no tutorial or documentation. PeerSim does not model the physical network, and it uses a deterministic model in which parameters such as bandwidth are fixed rather than random.

10.2.2.2 P2PSim

P2PSim is a flow-based simulator written in C++. The mode of its operation is a discrete event simulation and designed to simulate structured overlay only. This multi-threaded simulator supports several UNIX like operating system. However, it does not provide any simulation visualization or GUI support. P2PSim has built-in support for some of the popular protocols such as Chord [151], Accordion [95], Koorde [11], Kelips [55], Tapestry [186] and Kademlia [104]. P2PSim uses a pearl scripts for the generation of graphs. It can simulate up to 3000 nodes using chord protocol. It simulate dynamic networks with nodes joining and departing, as well as node can failure [136].

The biggest disadvantage is the poor usability as almost no documentation exits. It does not support distributed simulation. Further, this tool is limited to structured P2P only. This point removes it from our list, as our case study is based on super peer-based unstructured P2P.

10.2.2.3 3LS (3 Level Simulator)

3LS [160] is a Java based time-stepped unstructured overlay simulator. 3LS uses a central step clock to advance the time in simulation and control events application. It uses three different levels to simulate a network, the physical layer at the bottom, overlay network in the middle, and the application at the top. This architecture makes 3LS able to simulate a number of topologies and application protocols. Communication can only happen between the directly connected levels, so the protocol is responsible for simulating the P2P protocols and associated applications. User input is supplied into the network level through a GUI interface or a file [160]. When the simulation is complete, the results are stored in a files which can be used by the visualization tool.

This simulator does not support dynamic networks and the number of nodes is fixed [136]. Further, it does not implement crashing and scalability is poor i.e fewer than 20 nodes in Gnutella 0.4 [136].

10.2.2.4 Narses

Narses [47] is a discrete event simulator. This simulator is written in Java and specifically designed for large distributed applications. It has managed to overcome the scalability and speed of the most widely used NS-2 by combining packets into flows and thus reduce the overall number of events in the simulation. Narses supports four topology models starting from least accurate to most accurate and the slowest: Naive, NaiveTop, FairTopo and SafeFairTopo. Narses has been successful in modelling and simulating up to 600 nodes [136].

This simulator does not implement dynamic networks and does not have any

implementation of existing popular protocols [136]. The results are basic and for detail statistics the simulator needs to be extended.

10.2.2.5 Neurogrid

Neurogrid [76] is single-threaded discrete event simulator [77] which can simulate both structured and unstructured overlays. The driving force behind the development of this simulator was the comparison between the Neurogrid, Freenet [25] and Gnutella [48] protocols. This tool has higher scalability, and with a machine runing a 32bit operating system and 4 GB working memory it has managed to simulate 300,000 nodes [136]. The work on this simulator is still active, and improvement is in progress in terms of the modelling aspects [80]. NeuroGrid through a Jave applet, facilitates users to specify the number of nodes to simulate, the starting edges for each node, as well as the starting topology and number of searches.

The main drawback of this approach that it does not support churn with node being failure, which is the most pressing requirement of our model. It does support dynamic networks in which nodes can join and leave [136]. Further, it does not support a random bandwidth assignment and traffic.

10.2.2.6 Query Cycle

The Query-Cycle Simulator [141] is designed specifically for P2P file sharing. The core objective of this simulator is to simulate Gnutella-like networks. This simulator use parameters for content distribution, peer behaviour and network.

This simulator has been used to simulate real world networks. Content distribution and peer behaviour are based on respective empirical studies. The

simulator visual interface shows the number of cycles completed, and statistics for each node in terms of download and upload, is provided. The reports are fixed and can not be edited to include additional parameters.

This simulator, due to its mode of operation is not suited for our case study, and does not implement dynamic networks and nod failure [136].

10.2.2.7 GPS

GPS [177] is a discrete-event based simulator. It can simulate both structured and unstructured overlay networks, but with the pre-existing protocol BitTorrent. Since it is message based, it does model the physical network only. It makes use of the tool GT-ITM to generate respective topology [179]. It does not support distributed simulation. GPS supports automatic generation of events based on the BitTorrent Protocol. Generated events are called up at regular intervals during simulation.

The main drawback of the tool is that it is poorly documented. It seems almost impossible to extend the tool to include other protocols. It does support dynamic networks, but the model of churn does not implement node failure. The developers state that churn can be implemented using event script. Scalability is up to 512 [177] nodes and bandwidth computations are based on mathematical equations rather than random number. The other problem with this simulator is usability and maintenance. The group which contributed this simulator is no more working on this tool.

10.2.2.8 OverSim

OverSim is used as a framework with OMNeT++. The physical network model is also derived from OMNet++. At present OverSim only supports three protocols, i.e, Chord, Kademila and GIA overlay network protocols. There do exist documentation for a simple example and a mailing list for support. The documentation shows that OverSim can support up to 100,000 nodes. There are limited publications and the accuracy of the claims can not be confirmed.

This research group is an active group, continuously improving there simulator. As per documentation this simulator has not implemented any dynamic network and model of churn or reconfiguration. The modelling of bandwidth is also as a static parameter rather than random seed.

10.2.2.9 NS-2

NS-2 [117] is a well known simulator in the network research community. It is an event-based simulator for packet level simulation. The simulation is produced using a scripting language called TCL. The TCL scripts are used to define nodes and their associated links with properties such as latency. However, respective protocol is implemented in C++. Until now there is only one P2P, simulation simulation available for NS-2 (Gnutella). NS-2 does not have in-built visualization, but NAM which is a network animator, can be used. NS-2 also supports topology generators such GT-ITM [48] and Brite [117].

This simulator does not support dynamic networks and churn, and all events are timed rather than random.

10.2.2.10 OMNeT++

OMNeT++ is a discrete event simulator environment [136]. It is worth mentioning that it is not strictly a network simulator on its own, but instead it provides an environment for simulation where different models and frameworks could be used along with OMNeT++. All these components are programmed in C++ and then combined using the high-level language. It has strong GUI support and due to its modular architecture models can be easily embedded into new applications. It has a number of contributed models and used in other areas apart from P2P. [136] states that a 1000 node swarming simulation was implemented and later the P2P swarming [155] add-on was added to the library. Multi-tier topologies are supported, and Java interoperability is provided which further increases the usability.

This literature shows that at present it supports only structure P2P. The tool has low scalability because of the packet simulation. There exits no information that churn and dynamic networks are supported.

10.2.2.11 Gnutellasim

GnutellaSim is packet level simulator designed specifically to simulate Gnutella protocol [136]. This simulator provides a detailed model for Gnutella and uses NS-2 as an engine. The literature shows that it could support two protocols, i.e, Gnutella and FreeNet. It can simulate structured as well as unstructured P2P [136].

The simulator is based on NS-2 which does not implement dynamic network and churn. Documentation is also one of the major issues. This simulator can

scale up to 600 nodes [136].

10.2.3 Graph Based Stochastic Simulation

The GraSS operation and architecture has been explained in Chapter 7. In this section we are focussing on the explanation of the evaluation parameters, in order to make a comparison with existing approaches. In GraSS, a GTS is represented as a VIATRA2 model (as explained in chapter 6, consisting of the model space with the current graph and the transformation rules). The model represents discrete event system, where each state is a graph and each event is a graph transformation.

The GraSS is a research tool. The tool has recently been tested for simulations of pure unstructured P2P network. In this thesis the tool has been used for modeling dynamics of super peer-based hybrid P2P architecture. The model of churn used in this thesis consist join, and both selfish and cooperative exit. The reconfiguration model shows that super peer and client can normalize their topology if either choice of churn is used by the participant. Usability at present is poor as it is an experimental tool used only in PhD and MSc research projects. Similarly, scalability is limited to thousands of peers but this depends on the memory availability to the machine running the tool. Support is provided by the VIATRA2 team and researchers who use the tool. The approach has been tested for modelling user behavior using appropriate probability distributions and rates. The last attraction of the approach is the collection of statistics. The use of probe GT rules makes statistics collection easy. Since, probe rules are provided as part of GTS system it does not involve amending or editing the tool output.

The approach is capable to be executed on the cluster which could significantly increase scalability, but this has not yet been tested.

10.2.4 Comparative Evaluation of Approaches

In order to evaluate and justify the approach used in this thesis, first we comment on the use of graph transformation and second part we discuss simulation approach.

The case study in Chapter 5 presents requirements that include dynamic creation, deletion, dynamic upgrade, downgrade and reconfiguration of nodes in the topology. Apart from this the model needs sufficient number of nodes in order evaluate different variations of protocols. Graph transformation allows for creation of new client, deletion, downgrade, upgrade and reconfiguration. The use of stochastic graph transformation makes it easy to study non-functional and functional properties of the system. These stated features are difficult to be modelled using other approaches such as based on automata or Petri-nets.

To select appropriate graph transformation tool, during this thesis we explored a number of tools but most of them were lacking the functionality required. For example the Root [79] tool does not support SPO and with Groove [52] dynamic upgrade, downgrade, failure and deletion are difficult to implement, along with scalability issues. The VIATRA2 through the use of VTCL not only provides means to implement all the requirements but at the same time it shows the scalability required (see section 7.2).

Table 10.1 shows that a number of simulators have been developed either particularly for P2P (e.g., PeerSim, P2PSim, GPS) or extended to support P2P

overlay networks (e.g., NS-2, Narses, DHTSim). No single approach is capable to model and simulate the dynamics of P2P networks like Skype. Table 10.1 shows that most of the simulators are designed to address a specific issue in a protocol. This not only reduces the usability, but comparison of alternative approaches becomes impossible. Example of this is approaches like P2PSim and Overlay Weaver. These two are limited to DHT-based systems, making them unsuitable for simulating unstructured overlays. Another major issue in existing approaches is scalability, which is a core component of P2P network evaluation. Scalability differs widely with a number of simulators incapable of practically handling more than a few hundred nodes.

Table 10.1 shows that although GraSS has poor usability it meets all the requirements of modelling dynamics and reconfiguration of P2P VoIP. The use of GT rules for modelling transformations and probe rules for collection of statistics enable the tool to simulate alternate solutions and comment on their performance. This will help the developer of the protocol to test and compare their algorithms against existing solutions. This approach further helps to mix and match features of protocols and evaluate their performance against the parent protocols.

Table 10.1. Evaluation of Modelling and Simulation Approaches

S.No	Approach	S.A		SR.A			Reconfiguration	Churn	Usability			Scalability		Model		Statistics			Application			M.Level			S.L		Conf. Interval
		Discrete Event	Query Cycle	Structure	Understructure	Hybrid (SN)			Poor	Good	Excellent	nodes >= 1000	Nodes<1000	Deterministic	Stochastic	Poor	Good	Excellent	File-Sharing	VoIP	Other	Physical Layer	Application Layer	User Behaviour	Stand Alone	Grid/Cluster	
1	PeerSim	✓	✓	X	✓	✓	X	✓	X	✓	X	✓	X	✓	X	✓	X	X	✓	X	X	X	✓	X	✓	X	X
2	P2PSim	✓	X	✓	X	X	X	✓	✓	X	X	✓	X	✓	X	✓	X	X	✓	X	X	X	✓	✓	✓	X	X
3	3LS	✓	X	X	✓	X	X	X	✓	X	X	X	✓	✓	X	✓	X	X	✓	X	X	X	✓	X	✓	X	X
4	Narses	✓	X	✓	✓	X	X	X	✓	X	X	X	✓	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	X
5	NeuroGrid	✓	X	X	✓	X	X	✓	X	✓	X	✓	X	✓	X	X	✓	X	✓	X	X	X	✓	X	✓	X	X
6	Query Cycle	X	✓	X	✓	X	X	X	✓	X	X	X	✓	✓	X	✓	X	X	✓	X	X	X	✓	X	✓	X	X
7	GPS	✓	X	✓	✓	X	X	✓	✓	X	X	X	✓	✓	X	✓	X	X	✓	X	X	X	✓	X	✓	X	X
8	DHTSim	✓	X	✓	X	X	X	X	X	✓	X	X	X	✓	X	✓	X	X	✓	X	X	X	✓	X	✓	X	X
9	Overlay Weaver	✓	X	✓	X	X	X	X	✓	X	X	✓	X	✓	X	✓	X	X	✓	X	X	X	✓	X	✓	X	X
10	PlanetSim	✓	X	✓	✓	X	X	X	X	✓	X	✓	X	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	X
11	OverSim	✓	X	✓	✓	X	X	X	✓	X	X	✓	X	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	X
12	NS-2	✓	X	✓	✓	X	X	X	X	✓	X	X	✓	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	✓	X
12	OMneT++	✓	X	✓	✓	X	X	X	X	✓	X	X	✓	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	X
13	GnutellaSim	✓	X	X	✓	X	X	X	✓	X	X	X	✓	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	X
14	GraSS	✓	X	X	✓	✓	✓	✓	✓	X	X	X	✓	✓	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	X

10.3 Summary

In this chapter, we have reviewed the literature regarding super peer based P2P networks. The study shows that a number of approaches have been proposed ranging from super peer promotion based on bandwidth to promotion based on stability and locality. We also studied the dynamics in super peer architecture resulting from user behaviour or crashing of systems. Further, the need for re-configuration in these protocols has been justified through the studies in the literature. In the second part of this chapter we studied some of the parameters that could be used to evaluate existing approaches of P2P simulators. We examined existing simulation approaches and commented on their strength and weaknesses. We also evaluated these approaches based on the requirements of our case study and reflected the results in a table.

Chapter 11

Conclusions

This chapter presents the conclusions of the work in this thesis. We first list the achievements of the thesis and then describe future work.

11.1 Summary of Contributions

We proposed a new approach to the modelling and simulation of P2P network reconfigurations using graph transformation, a visual rule based formalism. Through investigation of a relevant case study applications and existing design alternatives, we first sketch the general structure of a P2P VoIP protocol and then follow the design variations by means of a feature tree. In this way, a rich set of mandatory and optional features have been identified and modelled (Chapter 4). The feature diagram presents the mandatory feature that are essential for the operation of P2P VoIP protocol and optional features such as load balancing that could help the core functions to stabilize the network.

The approach used in this thesis consists of a metamodel for super peer based P2P VoIP networks and transformation rules (Chapter 6) describing their operation. We have converted the metamodel into a VIATRA2 model space and the rules into VIATRA2 transformation rules (Chapter 7 and Appendix B). The aim was to convert the model and transformation rules into an executable form for stochastic simulation.

We have evaluated the alternative solutions to super peer selection, connection, promotion, and load balancing using stochastic simulation. The simulation results evaluate alternative approaches based on parameters such as the number of clients happy with their current selection of super node, the number of clients in the model, the number SNs in the model, the number of clients linked with super peers, the number of clients waiting for connection, the number of calls supported, and the number of clients reconfigured. This evaluation enables the designer to look into various alternatives in a single model and compare their

performance (Chapter 8). Based on the simulation results, a custom protocol has been proposed, which performs better then the parent protocols (Chapter 8).

We validated the model using statistics from the real Skype network and experimental data to build confidence in the model using methods like animation analysis, event validity, face validity and data validation. In the last part of Chapter 9, we discussed and evaluated the possible threats to the validity of the simulation results. We show that change in the start graph has less effect on the number of happy clients in the model. However, reducing bandwidth requirements for promotion affects the number of linked and happy clients..

11.2 Future Work

In this section, possibilities for extending this work in the future are discussed. Future work has two possible directions, i.e, model expansion and tool enhancement. The model expansion aspect is within the scope of this thesis while tool enhancement goes beyond its scope.

The model could be extended including support for reconfiguration based on quality. The approach enables the client to check the quality of the existing super peer and if SN does not meet the requirements, the clients should be able to select a new super peer. Further, if the client behind a firewall observes unacceptable response from a super peer, the clients should be able to change the connection dynamically without disturbing the existing VoIP call. Another possibility is to allow reconfiguration based on jitter, packet loss or echo in the VoIP call.

In this thesis the modelling of Skype client connected through mobile devices is not addressed. As these devices are subject to QoS issues related to network

speed as well as mobility, this could be another possibility to explore the modelling of P2P VoIP.

There also exist a possibility to explore other tools using the model presented in this thesis. This will help to identify how much effort is required to export as well as encode the model space and transformation rules of VIATRA2 into other tools. Similarly, if a model is exported successfully, there exist a possibility to compare the results which can help to comment on the performance of the tools.

There are a few directions for enhancement of the GraSS and VIATRA2 tools. In this thesis we observed that conversion of GT ruls written in graphical notation to the VIATRA2 VTCL language requires great care as there is no mechanism to check for logical error. The first recommendation is for VITARA2 tool to support automated conversion of graphically represented GT rules into the VTCL language.

As currently GraSS only supports exponential distribution and log-normal distribution, however, to further enhance the tool performance, it should support other probability distributions. This will enable the tool to be used in other domains. At present the results are written into different text files, so visualization of the results could be another direction for enhancement. The last recommendation is to improve the scalability of the tool and improve its GUI for visualization.

As future plan I am considering to use the approach presented in this thesis for modelling and simulation of population migration. The study of literature shows that traditional model of immigration was based on the archaeological evidence for culture change. This model was later on proved to be less practical as change in culture could be resulted either through trade or influx of small ruling elite with considerably less or no impact on the gene pool. Migration is

one of the core entities in population change and it is considered the most difficult component to estimate. Natural population have complex demographic histories and their respective sizes and ranges change over time, but these changes do leave signature on their genetic compositions. In order to get some realistic information, biological data will do help to uncover the truth of population migration by examining various genetic variations. Previous studies in UK have successfully used genetic variation ranging from examine blood group to serum protein and iso-enzymes. Recent results for Anglo-Saxon mass migration used y-chromosome as evidence, as this mitochondrial genome is useful sources as they provide detailed information regarding genealogies. The proposed model would represent DNA sequence through an attribute and they would support random generation of the DNA sequence. The model will further support exact matching and part matching of the DNA sequence. This would help to collect statistics regarding sequence matching in population. The flexibility of this approach would be that we can simulate foreword as well as reverse simulation.

As future plan I am also considering the to study the dynamics email based social network, in which individuals establish, maintain and allow links through contact-lists and emails. In social networks the model is based on the common neighbours as well as classical preferential attachment theory. Thus, enabling a deeper understanding of the topology of the inter-node connection. By using the approach of this thesis a model could be developed and stochastic simulation using GraSS can be used to simulate and validate the model by simulating real-world social networks with convincing realism.

Appendix A

Results and Probability Distributions

A.1 Stead State Experiments Results

The Fig.A.1 shows the simulation time required for latency based SN selection model to achieve the steady state. Similarly the Fig.A.2 and Fig.A.3 shows the simulation required for time based SN selection and local SN selection respectively.

A.2 GT Rules Probability Distributions and Rates

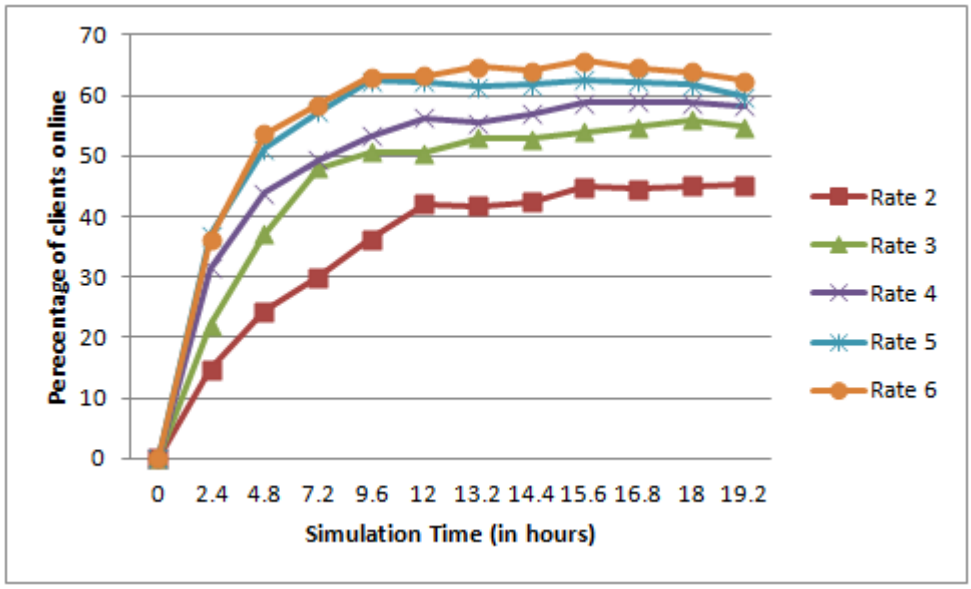

Figure A.1: Steady state latency based SN selection

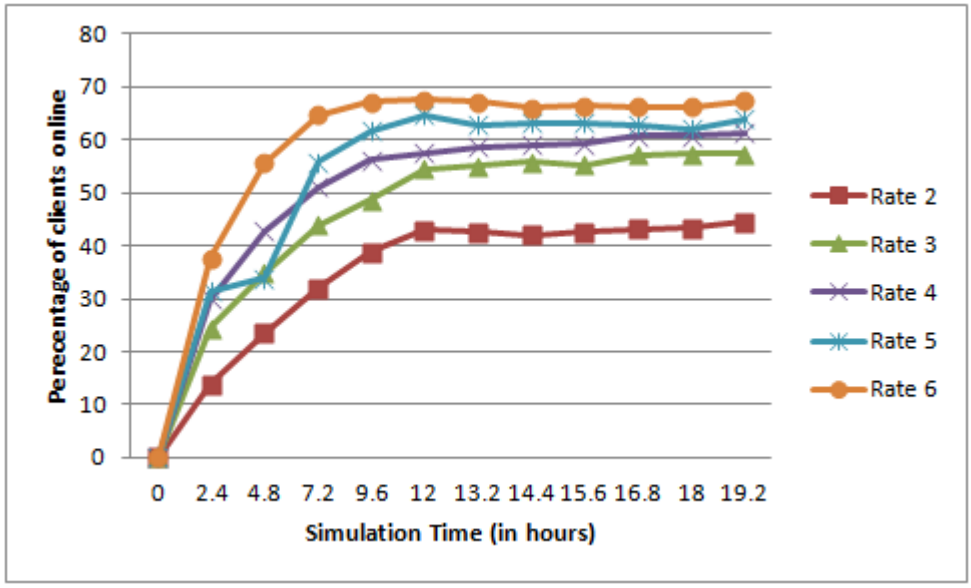

Figure A.2: Steady state time based SN selection

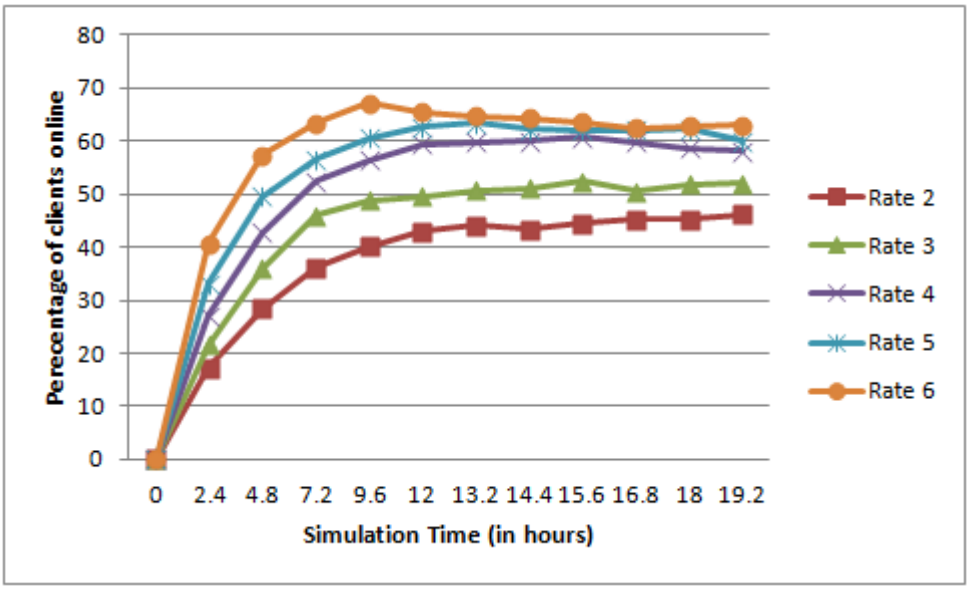

Figure A.3: Steady state local SN selection

Event	Mean μ	Variance σ^2
Rule$_{L}ink_{S}C_{t}o_{S}N$	0.000347	2.0
Reconfigure$_{L}ink_{S}C_{t}o_{l}ink_{b}ack_{S}N$	0.000173	2.0
Rule7$_{c}onnect_{S}N_{S}N$	0.000578	10
Rule9$_{D}own_{t}o_{S}C$	0.000173	10.0
Rule10$_{C}ontroled_{E}xit_{P}reperation_{S}C$	0.000154	2.0
Rule10$_{C}ontroled_{F}inal_{E}xit_{S}C$	0.000154	2.0
P2P$_{C}all_{w}ith_{f}irewaal_{e}nabled_{P}lace_{c}all_{a}nd_{R}oute$	0.000520	10.0
Place$_{c}all_{a}nd_{R}oute_{L}ocal$	0.000578	10.0
Place$_{c}all_{S}C_{t}o_{S}C$	0.000462	10.0
Place$_{c}all_{S}C_{t}o_{S}N$	0.000347	10.0
Place$_{c}all_{S}N_{t}o_{S}N$	0.000347	10.0
Terminate$_{r}outed_{C}all_{S}C_{S}N_{S}N_{S}C$	0.033	1.0
Terminate$_{r}outed_{C}all_{S}C_{S}N_{S}C$	0.033	1.0
Terminate$_{S}N_{t}o_{S}N$	0.033	1.0
Terminate$_{S}C_{t}o_{S}N$	0.033	1.0
Terminate$_{S}C_{t}o_{S}C$	0.033	1.0

Table A.1: GT rule log-normal distribution

Event	Mean λ ($hours$)
Rule5$_{P}romote_{a}t_{R}egistration_{S}uper_{N}ode$	3.0
Rule6$_{P}romote_{a}t_{L}inked_{S}uper_{N}ode$	1.0
ule6$_{P}romote_{a}t_{N}eed_{b}ased$	3.0
Rule10$_{C}ontrolled_{E}xit_{o}f_{S}C$	12
Rule11$_{U}nControled_{_{E}xit_{S}C}$	0.542
Rule12$_{C}ontroled_{E}xit_{S}N_{S}elected$	8
Rule13$_{d}eparture_{o}f_{S}N_{U}ncontrolled$	0.142
Requiest$_{c}all_{S}C_{t}o_{S}C_{b}ehind_{f}irewall$	4.0
Requiest$_{c}all_{S}C_{w}ith_{l}ive_{I}P$	4.0
Requiest$_{c}all_{S}N$	4.0
Random$_{S}C_{T}raffic$	48.0
Random$_{S}C_{T}raffic_{f}ir$	48.0
Random$_{S}N_{T}raffic$	48.0

Table A.2: GT rule exponential distribution

Appendix B

VIATRA2 Rules

B.1 Graph Pattern Used in this Thesis

In this section we present the executable forms of the rules presented in Chapter 6 We list all patterns here, except those which we already presented in the previous sections. Further, in this section we are going to present only the single link choice of SC to SN.

```
pattern Register_New(S,N,RG) =
  {
    ND(N);
    RS(S);
    RS.register(RG,S,N);
  }
 Explanation:
 This pattern register a new user with RS.
```

```
pattern RS_rg_SN(S,Y,RG) =
  {
    SN(Y);
    RS(S);
    RS.RSoverlay(RG,S,Y);

  }
 Explanation:
 This pattern registers SN with RS.
```

```
pattern SC_Chronos(X,CRO) =
```

```
  {
    SC(X);
    SkypeNode.Chronos(CRO);
    SkypeNode.chronos(CRR,X,CRO);
  }

 Explanation:
 This pattern access the chronos attribute of the SC.
```

```
pattern SN_Chronos(Y,CRO) =
  {
    SN(Y);
    SkypeNode.Chronos(CRO);
    SkypeNode.chronos(CRR,SN,CRO);
  }


 Explanation:
 This pattern access the chronos attribute of the SN.
```

```
pattern SC_Exit(X,EX) =
  {
    SC(X);
    SkypeNode.Exit(EX);
    SkypeNode.exit(EXR,X,EX);
  }

 Explanation:
 This pattern access the exit attribute of the SC.
```

```
pattern SN_Exit(Y,EX) =
  {
    SN(Y);
    SkypeNode.Exit(EX);
    SkypeNode.exit(EXR,Y,EX);
  }
 Explanation:
 This pattern access the exit attribute of SN.
```

```
pattern SN_Bandwidth(Y,BW)=
  {
    SN(Y);
    SkypeNode.Bandwidth(BW);
    SkypeNode.bandwidth(BWR,Y,BW);
  }
 Explanation:
 This pattern access the bandwidth attribute of the SN.
```

```
pattern SC_Bandwidth(X,BW)=
```

```
  {
    SC(SC);
    SkypeNode.Bandwidth(BW);
  SkypeNode.bandwidth(BWR,X,BW);
  }
 Explanation:
 This pattern access the bandwidth attribute of the SC.
```

```
pattern check_SC_with_RL(N,L,RLC)=
 {
  RL(L);
  ND(N);
    RL.ofline(RLC,L,N);
 }

 Explanation:
 This pattern confirms that ND is currently offline.
```

```
pattern SC_Online(S,X,RG) =
  {
    SC(X);
    RS(S);
    RS.online(RG,S,X);
  }
Explanation:
This pattern confirms that SC is having online edge with
RS server.
```

```
pattern test_links(X)=
     {
      SC(X);
      neg find Link_SN(X);
     }
Explanation:
This pattern calls other pattern.
```

```
pattern Link_SN(X)=
    {
     LK(K);
     SC(X);
     LK.linkto(CO,K,X);
    }
Explanation:
This pattern confirms that SC is linked with LK through
edge linkTo.
```

```
pattern Packet_SE_check(X)=
  {
    SC(X);
```

```
    Packet(PK);
    Packet.sender(SE,PK,X);
  }
Explanation:
This pattern confirms that SC has an edge labelled sender
with packet PK.
```

```
pattern Packet_RE_check(X)=
  {
    SC(X);
    Packet(PK);
    Packet.receiver(SE,PK,X);
  }
Explanation:
This pattern confirms that SC is having edge labelled
receiver with the packet PK.
```

```
pattern SN_Registerd_with_RS(S,Y) =
  {
    SN(S);
    RS(Y);
    RS.RSoverlay(RG,S,Y);
  }
Explanation:
This confirms that SN is having edge labelled RSoverlay
with server RS.
```

```
pattern Packet_reciver_Check(Packet,Y,RE)=
{
  Packet(Packet);
  SN(Y);
  Packet.receiver(RE,Packet,SN);
}
Explanation:
This pattern confirms that SN is having edge labelled
receiver with packet PK.
```

```
pattern Packet_sender_Check(Packet,X,SE)=
{
  Packet(Packet);
  SC();
  Packet.sender(SE,Packet,X);
}
Explanation:
This pattern confirms that SN is having edge labelled
sender with packet PK.
```

```
pattern Packet_at_Check(Packet,Y,AT)=
{
```

```
  Packet(Packet);
  SN(Y);
  Packet.at(AT,Packet,SN);
}
Explanation:
This pattern confirms current location of the packet with
SN through and edge labelled at.
```

```
pattern Packet_Content(Packet,CONT)=
  {
    Packet(Packet);
    Packet.Content(CONT);
    Packet.content(CI,Packet,CONT);
    }
Explanation:
This pattern access the content attribute of the packet.
```

```
pattern Packet_Chronos(P,CO)=
  {
    Packet(P);
    Packet.Chronos(CO);
    Packet.chronos(CI,Packet,CO);
    }
Explanation:
This pattern access the chromos attribute of the packet.
```

```
pattern Constraint(SN)=
  {
    SN(Y);
    find SN_Bandwidth(Y,BW);
    check((toInteger(value(BW)))<256);
  }
  or
  {
    SN(Y);
    find SN_Exit(Y,EX);
    check((value(EX))=="True");
  }
Explanation:
This pattern use or gate to combine to conditions. If
either condition is true pattern will results true. It
first confirm that SN bandwidth and later it check exit
attribute.
```

```
pattern Packet_at_Check_SC(Packet,SC)=
{
  Packet(Packet);
  SC(X);
  Packet.at(AT,Packet,X);
```

```
}
Explanation:
This pattern confirms the current location of the packet
with SC through an edge labelled at.
```

```
pattern SC_firewall(X,FIRW) =
  {
    SC(X);
    SC.Firewall(FIRW);
    SC.firewall(FIRWR,X,FIRW);
  }
Explanation:
This pattern confirms the current location of the packet
with SC through an edge labelled at.
```

```
pattern SC_firewall(X,FIRW) =
  {
    SC(X);
    SC.Firewall(FIRW);
    SC.firewall(FIRWR,X,FIRW);
  }
Explanation:
This pattern access the firewall attribute of the SC.
```

```
pattern SC_Linked_SN(X,Y,K)=
    {
      SC(X);
      SN(Y);
      LK(K);
      LK.linkto(LT,K,X);
      LK.outTo(LO,K,Y);
    }
Explanation:
This pattern confirms that SC and SN are linked through
node LK.
```

```
pattern Check_Links_with_SN(Y)=
   {
    LK(K);
    SN(Y);
    LK.outTo(LKI,K,Y);
   }
Explanation:
This pattern confirms that LK is linked with SN through
an edge outTo.
```

```
pattern Check_OV_Links_with_SN(Y)=
   {
    OV(O);
```

```
    SN(Y);
    OV.source(LKI,O,Y);
   }
   or
   {
    OV(O);
    SN(Y);
    OV.target(OVT,O,Y);
   }
Explanation:
This pattern confirms that SN is having overlay connection
through node OV either as source or target node.
```

```
pattern SN_Target_Check(Y,O,OVT)=
    {
      SN(Y);
      OV(O);
      OV.target(OVT,O,Y);
    }
Explanation:
This pattern confirms that SN is having target link with
overlay link OV.
```

```
pattern SN_Source(Y,O)=
   {
    SN(Y);
    OV(O);
    OV.source(OVI,O,Y);
   }
Explanation:
This pattern confirms that SN is having source edge with
overlay link OV.
```

```
   pattern Test(O)=
   {
    OV(O);
    find target(O,Y);
   }
   pattern target(OV,Y)=
   {
    OV(O);
    SN(Y);
    OV.target(OVI,O,Y);
   }
Explanation:
This pattern confirms OV is having link with SN in
the current graph as source.
```

```
   pattern SN_Target(Y,O)=
```

```
   {
    SN(Y);
    OV(O);
    OV.target(OVI,O,Y);
   }

Explanation:
This pattern confirms that SN is having target edge
with overlay link OV.
```

```
   pattern Test_Target(O)=
   {
    OV(O);
    find Source(O,Y);
   }
   pattern Source(O,Y)=
   {
    OV(O);
    SN(Y);
    OV.source(OVI,O,Y);
   }
Explanation:
This pattern confirms OV is having link with SN in
the current graph as target.
```

```
   pattern Test_Target(O)=
   {
    OV(O);
    find Source(O,Y);
   }
   pattern Source(O,Y)=
   {
    OV(O);
    SN(Y);
    OV.source(OVI,O,Y);
   }
Explanation:
This pattern confirms OV is having link with SN in
the current graph as target.
```

```
   pattern SC_Source(K,X)=
  {
    LK(K);
    SC(X);
    LK.linkto(LKR,K,X);
  }
Explanation:
This pattern confirms that link LK is having edge
linkTo with SC.
```

```
   pattern SN_SC_Target(K,Y)=
  {
    LK(K);
    SN(Y);
    LK.outTo(LKR,K,Y);
  }
Explanation:
This pattern confirms that link LK is having edge
labelled outTo with SN.
```

```
   pattern SN_SC_Target(K,Y)=
  {
    LK(K);
    SN(Y);
    LK.outTo(LKR,K,Y);
  }
Explanation:
This pattern confirms that SC and SN are linked
through node LK via edges outTo and linkTo.
```

```
   pattern SN_SC_Target(LKR,K,Y)=
  {
    LK(K);
    SN(Y);
    LK.outTo(LKR,K,Y);
  }
Explanation:
This pattern confirms that SN and LK are linked
through an edge outTo.
```

```
  pattern Check_SC_Source(X)=
  {
    LK(K);
    SC(X);
    LK.linkto(LKR,K,X);
  }
Explanation:
This pattern confirms that SC and LK are linked
through an edge linkTo.
```

```
  pattern SC_loaction(X,LOC) =
  {
    SC(X);
    SkypeNode.Location(LOC);
  SkypeNode.location(LOCRR,X,LOC);
  }
Explanation:
This pattern access the location attribute of the SC.
```

```
  pattern Check_OV(Y,Y1,O)=
    {
      SN(Y);
      SN(Y1);
      OV(OV);
      OV.source(OVS,OV,Y);
      OV.target(OVT,OV,Y1);
    }
Explanation:
This pattern confirms that SN of type Y1 and Y are linked
in the overlay network.
```

```
  pattern SN_location(Y,LOC)=
{
  SN(Y);
  SkypeNode.Location(LOC);
  SkypeNode.location(LOCR,Y,LOC);
}
Explanation:
This pattern access the location attribute of the SN.
```

```
 pattern OV_Source(O,Y)=
    {
      SN(Y);
      OV(O);
      OV.source(OVR,O,Y);
    }
Explanation:
This pattern confirms that SN and OV are linked
through an edge source.
```

```
 pattern OV_Target(O,Y)=
    {
      SN(Y);
      OV(O);
      OV.target(OVT,O,Y);
    }
Explanation:
This pattern confirms that SN and OV are linked
through an edge target.
```

```
 pattern Check_OV_Links(Y)=
    {
      SN(Y);
      find OV_Source(O,Y);
      find OV_Target(O,Y);
    }

Explanation:
```

```
This pattern find the SN is having source or target
edges with OV.
```

```
pattern Check_LK_Links(Y)=
    {
      SN(Y);
      LK(K);
      find LK_Check(Y,K);
    }
Explanation:
This pattern calls another pattern to check linking
between SN and LK.
```

```
pattern LK_Check(Y,K)=
    {
      LK(K);
      SN(Y);
      LK.outTo(LKR,K,Y);
    }

Explanation:
This pattern confirms that LK and SN are linked
through an edge.
```

```
pattern check_for_existing_call(X)=
     {
      SC(X);
      RouteCall(R);
      RouteCall.caller(CAI,R,X);

     }
     or
     {
      SC(SX);
      RouteCall(R);
      RouteCall.callee(CAI,R,X);
     }
Explanation:
This pattern confirm that SC is VoIP call either as
caller or callee.
```

```
pattern check_for_call(X)=
     {
      SC(X);
      Call(C);
      Call.caller(CAI,C,X);

     }
     or
```

```
     {
      SC(C);
      Call(C);
      Call.callee(CAII,C,X);
     }

Explanation:
This pattern confirms that SC is in direct P2P VoIP
call either as caller or callee.
```

```
pattern existing_req(X)=
     {
      SC(X);
      CallReq(Q);
      CallReq.initiate(CAT,Q,X);
     }
Explanation:
This pattern confirms that SC has initiated VoIP
call request.
```

```
pattern check_requiest_for_call_SN(Y,Q)=
     {
      SN(Y);
      CallReq(Q);
      CallReq.initiate(CTI,Q,Y);
     }
Explanation:
This pattern confirms that SN has initiated VoIP
call request.
```

```
pattern call_check(Y,Y1,C)=
    {
      Call(C);
      SN(Y1);
      SN(Y);
      Call.caller(CAI,C,Y);
      Call.callee(CAII,C,Y1);
    }
    or
    {
      Call(C);
      SN(Y1);
      SN(Y);
      Call.caller(CAI,C,Y1);
      Call.callee(CAII,C,Y);
    }
Explanation:
This pattern confirms that SN and SN1 are in VoIP
call.
```

```
pattern check_for_call_SN(Y)=
     {
      SN(Y);
      Call(C);
      Call.caller(CAI,C,Y);

     }
     or
     {
      SN(Y);
      Call(C);
      Call.callee(CAII,C,Y);
     }
Explanation:
This pattern confirms that SN is already in call
either as caller or callee.
```

```
pattern Find_routed_call(X,X1,Y,Y1,R)=
   {
          SN(Y);
              SN(Y1);
              SC(X);
              SC(X1);
              RouteCall(R);
      RouteCall.caller(PKS,R,X);
      RouteCall.callee(PKR,R,X1);
      RouteCall.route(SNR,R,Y);
      RouteCall.route(SNRR,R,Y1);

   }
   or
   {
          SN(Y);
              SN(Y1);
              SC(X);
              SC(X1);
              RouteCall(R);
      RouteCall.caller(PKS,R,X1);
      RouteCall.callee(PKR,R,X);
      RouteCall.route(SNR,R,Y);
      RouteCall.route(SNRR,R,Y1);

   }
Explanation:
This pattern finds multiple SN routed VoIP calls
between SCs.
```

```
Pattern
Find_routed_call_SC_SN_SC(X,X1,Y,R)=
```

```
   {
          SN(Y);
                SC(X);
             SC(X1);
             RouteCall(R);
      RouteCall.caller(PKS,R,X);
      RouteCall.callee(PKR,R,X1);
      RouteCall.route(SNR,R,Y);


   }
   or
   {

          SN(Y);
                SC(X);
             SC(X1);
             RouteCall(R);
      RouteCall.caller(PKS,R,X1);
      RouteCall.callee(PKR,R,X);
      RouteCall.route(SNR,R,Y);
        }
Explanation:
This pattern find direct VoIP call between SC and SN
```

```
pattern call_check_1C(X,X1,C)=
    {
      Call(C);
      SC(X);
      SC(X1);
      Call.caller(CAI,C,X);
      Call.callee(CAII,C,X1);
    }
    or
    {
      Call(C);
      SC(X);
      SC(X1);
      Call.caller(CAI,C,X1);
      Call.callee(CAII,C,X);
    }
Explanation:
This pattern finds direct VoIP call between SC and
SC1.
```

```
pattern SN_SN(Y,Y1)=
    {
      SN(Y);
      SN(Y1);
```

```
      OV(O);
      OV.source(OVI,O,Y);
      OV.target(OVIT,O,Y1);

    }
Explanation:
This pattern confirms that SN and SN1 are linked in
the overlay.
```

```
pattern SN_SN_OVI(Y,Y1,O)=
    {
      SN(Y);
      SN(Y1);
      SN.overlayPath(OVI,Y,Y1);

    }
Explanation:
This pattern confirms that there is an edge OVI
between SN and SN1.
```

```
pattern SN_SN_Test(Y,Y1)=
    {
      SN(Y);
      SN(Y1);
      find SN_SN(Y,Y1);

    }

Explanation:
This pattern confirms that two SNs are linked in the
SN overlay.
```

```
pattern OV_Path(Y)=
    {
      SN(Y);
      find SN_SN(Y,Y1);
      find SN_SN(Y,Y2);
      find SN_SN(Y,Y3);
      find SN_SN(Y,Y4);
    }
Explanation:
This pattern confirms that SN in linked with other
four SNs in the overlay network.
```

B.2 VIATRA2 Interpretation of The GT Rules used in this Thesis

In this section now we present the all the rules in VIATRA2 Code, The rules are presented in the same order as that in graphical form. The purpose of this section is to give the reader how the graphical notation of the GT rules are mapped into VIATRA2 rules.

B.2.1 Registration of New Users

The rule in Fig. 6.3 is mapped to VIATRA2 rule in List. B.1

```
1  gtrule Registeration_of_New_User() =
2      {
3              precondition pattern lhs(L) =
4              {
5                RL(L);
6
7              }
8              action {
9               let
10                  N1=undef,
11                      KK=undef
12             in seq {
13                  new (ND(N1) in L);// create client inside the
                       network
14                  rename(ND1,"ND1");
15                  new(RL.ofline(KK,L,N1));
16                }
17           println("New Skype User Registered");
18              }
19     }
```

Listing B.1: Registration of New User

B.2.2 Registration Successful

The rule in Fig. 6.4 is mapped to VIATRA2 rule in List. B.2

```
1  gtrule Rule_Sucessful_Registeration() =
2      {
3              precondition pattern lhs(S,L,RG,N) =
```

```
4               {
5                 RS(S);
6                 RL(L);
7                 ND(N);
8                 find Register_New(S,N,RG);
9
10              }
11              action {
12                move(N,L);
13                delete(RG);
14               let
15                KK=undef
16                  in seq {
17                  rename(N,"ND1");
18                  new(RL.ofline(KK,L,N));
19                 }
20                println("New Skype User successfully Registered");
21              }
22    }
```

Listing B.2: Successful registration

User Go Online with Random Firewall

The rule at Fig. 6.5 is mapped to VIATRA2 rule at List. B.3

```
1   gtrule User_Go_Online() =
2     {
3               precondition pattern lhs(N,S,L,RLC,NK) =
4               {
5                 RS(S);
6                 RL(L);
7                 ND(N);
8                 find check_SC_with_RL(N,L,RLC);
9                 // this checks if user is already registerd//
10                Network(NK);
11              }
12  action {
13        move(ND,Network);
14  new(instanceOf(ND,DSM.metamodel.SkypeSim.SkypeSimEditor.SC));
15  delete(instanceOf(ND,DSM.metamodel.SkypeSim.SkypeSimEditor.ND));
16            delete(RLC);
17            rename(N,"sc");
18                    let
19                      K=undef,
20                      KR=undef,
21                      B=undef,
22                      BR=undef,
23                      CH=undef,
24                      CHR=undef,
25                      NR=undef,
```

```
                    RG=undef,
                    EX=undef,
                    EXR=undef,
                    LOC=undef,
                    LR=undef
            in seq {

              new(SC.Firewall(K)in N);
// create attribute firewall in the SC
            new(SC.firewall(KR,N,K));
// create relation between the SC and attribute
                new(SkypeNode.Bandwidth(B) in N);
                setValue(B,skypesim.random(56,2000));
// use random function to assign bandwidth
                new(SkypeNode.bandwidth(BR,N,B));
                new(SkypeNode.Chronos(CH) in N);
                setValue(CH,systime());
// time Now //
                new(SkypeNode.chronos(CHR,N,CH));
                new(SkypeNode.Location(LOC) in N);
                new(SkypeNode.location(LR,N,LOC));
                setValue(LOC,skypesim.random(1,24));
                new(SkypeNode.Exit(EX) in N);
                setValue(EX,"False");
// set default Exit to false
                new(SkypeNode.exit(EXR,N,EX));
              new(Network.nodes(NR,NK,N));
              new(RS.online(RG,S,N));
// set firewall enabled and disabled based on bandwidth
              if (((toInteger(value(B)))>256))
               setValue(K,"False");
               else
                setValue(K,"True");
                println("User go Online ");

            }
            }

    }
```

Listing B.3: User go online with random firewall

Rule User Go Online with Firewall Disabled

The rule at Fig. 6.6 is mapped to VIATRA2 rule at List. B.4

```
gtrule User_Go_Online_with__Firewall_diabled() =
    {
            precondition pattern lhs(N,S,L,RLC,NK) =
            {
              RS(S);
```

```
6                RL(L);
7                ND(N);
8                find check_SC_with_RL(N,L,RLC);
9                // this checks if user is already registered//
10               Network(Network);
11             }
12             action {
13                move(N,NK);
14                new(instanceOf(N,DSM.metamodel.SkypeSim.
                     SkypeSimEditor.SC));
15                delete(instanceOf(N,DSM.metamodel.SkypeSim.
                     SkypeSimEditor.ND));
16           delete(RLC);
17           rename(ND,"sc");
18                  let
19                    K=undef,
20                    KR=undef,
21                    B=undef,
22                    BR=undef,
23                    CH=undef,
24                    CHR=undef,
25                    NR=undef,
26                    RG=undef,
27                    EX=undef,
28                    EXR=undef,
29                    LOC=undef,
30                    LR=undef
31             in seq {
32               new(SC.Firewall(K)in N);
                        new(SC.firewall(KR,N,K));
33                setValue(K,"False");
34                new(SkypeNode.Bandwidth(B) in N);
35                setValue(B,skypesim.random(256,2000));
                                                  new(SkypeNode.
                     bandwidth(BR,N,B));
36                new(SkypeNode.Chronos(CH) in ND);
37                setValue(CH,systime());
                              new(SkypeNode.chronos(CHR,N,CH));
38                new(SkypeNode.Location(LOC) in N);
39                new(SkypeNode.location(LR,N,LOC));
40                setValue(LOC,skypesim.random(1,24));
41                new(SkypeNode.Exit(EX) in N);
42                setValue(EX,"False");
43                new(SkypeNode.exit(EXR,N,EX));
44              new(Network.nodes(NR,Network,N));
45              new(RS.online(RG,S,N));
46            println("User go Online with firewall disabled ");
47
48             }
```

```
49              }
50
51      }
```

Listing B.4: Users go online with firewall disabled

User Go Online with Firewall Enabled

The rule at Fig. 6.7 is mapped to VIATRA2 rule at List. B.5

```
1  gtrule User_Go_Online_with__Firewall_enabled() =
2      {
3              precondition pattern lhs(N,RS,RL,RLC,Network) =
4              {
5                RS(RS);
6                RL(RL);
7                ND(N);
8                find check_SC_with_RL(N,RL,RLC);
9                // this checks if user is already registerd//
10               Network(Network);
11             }
12             action {
13              move(N,Network);
14            new(instanceOf(N,DSM.metamodel.SkypeSim.SkypeSimEditor
                 .SC));                    delete(instanceOf(N,DSM.
                 metamodel.SkypeSim.SkypeSimEditor.ND));
15          delete(RLC);
16          rename(N,"sc");
17                 let
18                   K=undef,
19                   KR=undef,
20                   B=undef,
21                   BR=undef,
22                   CH=undef,
23                   CHR=undef,
24                   NR=undef,
25                   RG=undef,
26                   EX=undef,
27                   EXR=undef,
28                   LOC=undef,
29                   LR=undef
30             in seq {
31
32               new(SC.Firewall(K)in N);
                       new(SC.firewall(KR,N,K));
33               setValue(K,"True");
34               new(SkypeNode.Bandwidth(B) in N);
35               setValue(B,skypesim.random(128,1000));
                                                  new(SkypeNode.
                   bandwidth(BR,N,B));
36               new(SkypeNode.Chronos(CH) in N);
```

```
37                  setValue(CH,systime());
                                    new(SkypeNode.chronos(CHR,N,CH));
38                  new(SkypeNode.Location(LOC) in N);
39                  new(SkypeNode.location(LR,N,LOC));
40                  setValue(LOC,skypesim.random(1,24));
41                  new(SkypeNode.Exit(EX) in N);
42                  setValue(EX,"False");// set default Exit to false
43                  new(SkypeNode.exit(EXR,N,EX));
44                new(Network.nodes(NR,Nk,N));
45                new(RS.online(RG,S,N));
46                println("User go Online with firewall disabled ");
47
48              }
49              }
50    }
```

Listing B.5: User go online with firewall enabled

B.2.3 Selection of SN by SC

B.2.3.1 Random Selection of SN on Bandwidth

The rule at Fig. 6.8 is mapped to VIATRA2 rule at List. B, as this rule is already presented in this chapter so we are not repeating it.

B.2.3.2 Selection Based on Bandwidth and Latency

- Create and Send Time Stamped Packet

The rule in Fig. 6.9 is mapped to VIATRA2 rule in List. B.6

```
1  gtrule Rule_Create_and_Send_time_Stamp_Packet() =
2      {
3              precondition pattern lhs(X,Y,CRO) =
4              {
5                SN(Y);
6                SC(X);
7                find SC_Chronos(X,CRO);
8                find SN_Registerd_with_RS(S,Y);
9                    neg find Link_SN(X);
10                   find SN_Bandwidth(Y,BW);
11                   neg find Packet_SE_check(X);
12                   neg find Packet_RE_check(X);
13                   find SN_Exit(Y,SNEX);
14               find SC_Exit(X,SCEX)
15               check((value(SCEX))=="False");
16               check((value(SNEX))=="False");
17               check((toInteger(value(BW)))>256);
18
```

```
19              }
20
21              action {
22                  let Packet=undef,
23      PC=undef,
24      CRR=undef,
25      PCO=undef,
26      PCR=undef,
27      SE=undef,
28      RE=undef,
29      AH=undef
30        in seq {
31    new(Packet(P)in Y); //Create Packet in SC
32    new(Packet.Chronos(PC)in P); //Create a chronos attribute
         new(Packet.chronos(CRR,P,PC));          new(Packet.Content(
         PCO)in Packet);
33    new(Packet.content(PCR,P,PCO));
34    new(Packet.sender(SE,P,X));
35    new(Packet.receiver(RE,P,Y));
36    new(Packet.at(AH,Packet,SN));
37              setValue(CRO,systime());
38              setValue(PC,value(CRO));
39              setValue(PCO,"Ping");
40              println("Ping Packet Created");
41              }
42              }
43  }
```

Listing B.6: Create and Send time stamped ping packet

- Reply Packet with AcK

The rule in Fig. 6.10 is mapped to VIATRA2 rule in List. B.7

```
1   gtrule Reply_Ping_Packet_with_ACK() =
2       {
3               precondition pattern lhs(P,CONT,Y,X,SE,RE,AT,CO) =
4               {
5                 Packet(P);
6                 find Packet_reciver_Check(P,Y,RE);
7                 find Packet_sender_Check(Pt,X,SE);
8                 find Packet_at_Check(P,Y,AT);
9                 find Packet_Content(P,CONT);
10                find SC_Chronos(X,CRO);
11                find Packet_Chronos(P,CO);
12                find SN_Exit(Y,SNEX);
13                find SC_Exit(Y,SCEX)
14                    find SN_Bandwidth(Y,BW);
15                    find SC_Bandwidth(X,SCBW);
16                check((toInteger(value(BW)))>256);
17                check((value(SCEX))=="False");
```

```
18                check((value(SNEX))=="False");
19                    check((value(CONT))=="Ping");
20            }
21
22            action {
23            let NSE=undef,NRE=undef,NAT=undef in seq {
24            new (Packet.sender(NSE,P,Y));
25            new(Packet.receiver(NRE,P,X));
26            new(Packet.at(NAT,P,X));
27            move(Packet,SC);
28            }
29            delete (SE);
30            delete (RE);
31            delete (AT);
32            setValue(CONT,"AcK");
33            setValue(CO,systime());
34            println("Ping Replied Back");
35
36            }}
```

Listing B.7: Return reply packet with AcK

- Reply Packet with DnY

The rule in Fig. 6.11 is mapped to VIATRA2 rule in List. B.8

```
1  gtrule Rule3d_Revers_DnY_Stamp_Packet_to_SN() =
2      {
3              precondition pattern lhs(P,CONT,Y,X,SE,RE,AT,CO) =
4              {
5                SN(Y);
6                SC(X);
7                    Packet(P);
8                find Packet_reciver_Check(P,Y,RE);
9                find Packet_sender_Check(P,X,SE);
10               find Packet_at_Check(P,Y,AT);
11               find Packet_Content(P,CONT);
12               find SC_Chronos(SC,CRO);
13               find Packet_Chronos(P,CO);
14               find SN_Exit(Y,EX);
15                   find SN_Bandwidth(Y,BW);
16                   find SC_Bandwidth(X,SCBW);
17                   check((value(CONT))=="Ping");
18                   find Constraint(Y);
19
20             }
21
22             action {
23             let NSE=undef,NRE=undef,NAT=undef in seq {
24             new (Packet.sender(NSE,P,Y));
25             new(Packet.receiver(NRE,P,X));
```

```
26              new(Packet.at(NAT,P,SC));
27              move(Packet,X);
28              }
29              delete (SE);
30              delete (RE);
31              delete (AT);
32              setValue(CONT,"DnY");
33              setValue(CO,systime());
34              println("Ping Replied Back");
35
36              }}
```

Listing B.8: Return reply packet with DnY

- Links SC with SN Based on Latency

The rule in Fig. 6.12 is mapped to VIATRA2 rule in List. B.9

```
1  gtrule Rule3_Link_SC_to_SN() =
2      {
3              precondition pattern lhs(X,Y,RG,NK,P,CONT,PCO,CRO,BW,
                   SCBW) =
4              {
5                SN(Y);
6                SC(X);
7                Network(NK;
8                Packet(P);
9                find Packet_at_Check_SC(P,X);
10               find SC_Online(RS,SC,RG);
11               find Packet_Content(P,CONT);
12               find Packet_Chronos(P,PCO);
13               find SC_Chronos(X,CRO);
14               find SN_Bandwidth(Y,BW);
15               find SC_Bandwidth(Y,SCBW);
16                   neg find Link_SN(X);
17               check((value(CONT))=="AcK");
18             check(((toInteger(value(PCO)))- (toInteger(value(CRO)
                   )))<= 800);
19             }
20
21             action {
22
23                   setValue(CRO,systime());
24                   let LK=undef, CO=undef,OU=undef,NR=undef in seq
                        {
25                 new (LK(LK) in Network);
26                 new(Network.nodes(NR,Nk,LK));
27                 new(LK.linkto(CO,LK,X));
28                 new(LK.outTo(OU,LK,Y));
29                 delete(RG);
30         setValue(BW,((toInteger(value(BW)))-4));
```

```
31          setValue(SCBW,((toInteger(value(SCBW)))-4));
32            delete(Packet);
33        println("connected/Linked  and round trip time taken is ");
34              }
35
36              }
37      }
```

Listing B.9: Links SC with SN Based on Latency

- Reject SN Based on Latency

The rule in Fig. 6.13 is mapped to VIATRA2 rule in List. B.10

```
1  gtrule Reject_SN_based_on_Latency() =
2      {
3              precondition pattern lhs(S,X,RG,P,CONT,PCO,CRO,BW,
                   SCBW) =
4              {
5
6                SC(X);
7                RS(S);
8                Packet(P);
9                find Packet_at_Check_SC(P,SC);
10               find SC_Online(S,X,RG);
11               find Packet_Content(P,CONT);
12               find Packet_Chronos(P,PCO);
13               find SC_Chronos(X,CRO);
14             find SN_Bandwidth(Y,BW);
15             find SC_Bandwidth(X,SCBW);
16                    neg find Link_SN(X);
17               check((value(CONT))=="AcK");
18             check(((toInteger(value(PCO)))- (toInteger(value(CRO)
                   )))>= 300);
19             }
20
21             action {
22
23       delete(P);
24     println("Reject SN as Latency is More than required");
25             }
26
27             }
```

Listing B.10: Reject SN based on latency

- Reject SN Based on Return of DnY Packet

The rule in Fig. 6.14 is mapped to VIATRA2 rule in List. B.11

```
1  gtrule Reject_SN_based_on_DnY() =
2      {
3              precondition pattern lhs(S,X,RG,P,CONT,PCO,CRO,BW,
                   SCBW) =
```

```
4               {
5
6                 SC(X);
7                 RS(S);
8                 Packet(P);
9                 find Packet_at_Check_SC(P,X);
10                find SC_Online(S,X,RG);
11                find Packet_Content(P,CONT);
12                find SC_Chronos(X,CRO);
13              find SC_Bandwidth(X,SCBW);
14                    neg find Link_SN(X);
15                check((value(CONT))=="Dny");
16
17              }
18
19              action {
20
21            delete(P);
22              println("Reject SN DnY packet Received");
23              }
24
25              }
```

Listing B.11: Delete reply packets try new SN

B.2.3.3 SC linked with SN on Bandwidth and Locality (region)

The rule in Fig. 6.15 is mapped to VIATRA2 rule in List. B.12

```
1  gtrule Rule3_Link_SC_to_SN_Region_Based() =
2      {
3               precondition pattern lhs(S,X,Y,CRO,RG,BW,SCBW,NK) =
4               {
5                 SN(Y);
6                 SC(X);
7                 Network(NK);
8                 RS(S);
9                 find SC_Online(S,X,RG);
10                find SC_Chronos(X,CRO);
11                find SN_Exit(Y,EX);
12                  find SN_Bandwidth(Y,BW);
13                  find SC_Bandwidth(X,SCBW);
14                 // neg find Link_SN(X);//
15                  find test_links(X);
16
17                check((value(EX))=="False");
18                // this part is for region based
19                find SC_Location(X,LOC);
20                find SN_Location(Y,LOCN);
```

```
21                check(((value(LOC)))== ((value(LOCN))));
22                check((toInteger(value(BW)))>256);
23
24              }
25
26              action {
27                  setValue(CRO,systime());
28                    let LK=undef, CO=undef,OU=undef,NR=undef in seq
                          {
29                  new (LK(LK) in Network);
30                  new(Network.nodes(NR,NK,LK));
31                  new(LK.linkto(CO,LK,X));
32                  new(LK.outTo(OU,LK,Y));
33                  delete(RG);
34              setValue(BW,((toInteger(value(BW)))-4));
35              setValue(SCBW,((toInteger(value(SCBW)))-4));
36              println("connected/Linked ");
37              }
38
39
40      }}
```

Listing B.12: Link SC to an SN from same region

B.2.3.4 Selection of SN based Bandwidth and Time spent in Network

The rule in Fig. 6.17 is mapped to VIATRA2 rule in List. B.13

```
1  gtrule Rule_Link_SC_to_SN() =
2      {
3              precondition pattern lhs(S,X,Y,CRO,RG,BW,SCBW,NK) =
4              {
5                SN(Y);
6                SC(X);
7                RS(S);
8                Network(NK);
9                find SC_Online(S,X,RG);
10               find SN_Chronos(Y,CRO);
11               find SN_Exit(Y,SNEX);
12               find SC_Exit(X,SCEX);
13                 find SN_Bandwidth(Y,BW);
14                 find SC_Bandwidth(X,SCBW);
15                 find test_links(X);
16               check((toInteger(value(BW)))>256);
17               check((toInteger(value(CRO)))>100);
18               check((value(SNEX))=="False");
19               check((value(SCEX))=="False");
20
21             }
```

```
22
23              action {
24                  setValue(CRO,systime());
25                    let LK=undef, CO=undef,OU=undef,NR=undef in seq
                          {
26                  new (LK(LK) in Network);
27                  new(Network.nodes(NR,NK,LK));
28                  new(LK.linkto(CO,LK,X));
29                  new(LK.outTo(OU,LK,Y));
30                  delete(RG);
31          setValue(BW,((toInteger(value(BW)))-4));
32          setValue(SCBW,((toInteger(value(SCBW)))-4));
33           println("SC connected/Linked WIth SN");
34              }
35
36      }}
```

Listing B.13: Link SC to an SN based on the current bandwidth of SN and stability

B.2.4 Promotion to Super Peer

B.2.4.1 Static Protocol for Promotion of SC to SN

The rule in Fig. 6.28 is mapped to VIATRA2 rule in List. B.14

```
1  gtrule Rule5_Promote_at_Registration_Super_Node() =
2      {
3              precondition pattern lhs(X,S,RG,BW) =
4              {
5
6                SC(X);
7                RS(S);
8                find SC_firewall(X,FIRW);
9                    find SC_Online(S,SC,RG);
10               find SC_Bandwidth(X,BW);
11               check((value(FIRW))=="False");
12               check((toInteger(value(BW)))>= 1500);
13
14             }
15             action
16       {
17         new(instanceOf(SC,DSM.metamodel.SkypeSim.SkypeSimEditor.
               SN));
18         let RG1=undef  in seq{
19         new (RS.RSoverlay(RG1,S,X));
20               delete(instanceOf(SC,DSM.metamodel.SkypeSim.
                    SkypeSimEditor.SC));
21         delete(RG);
```

```
22
23          }
24          println("Promoted to Supere Node at registration");
25
26              }
27      }
```

Listing B.14: SC is promoted to SN as soon as SC goes online

B.2.4.2 Dynamic Protocol for Promotion of SC to SN

The rule in Fig. 6.29 is mapped to VIATRA2 rule in List. B.15

```
1  gtrule Rule6_Promote_at_Linked_Super_Node() =
2      {
3              precondition pattern lhs(X,S,K,BW,BW1) =
4              {
5
6                SC(X);
7                RS(S);
8                SN(Y);
9                LK(L);
10               //neg find check_existing(S);
11               find SC_Linked_SN(X,Y,L);
12               find SC_firewall(X,FIRW);
13               find SN_Bandwidth(y,BW1);
14               find SC_Bandwidth(X,BW);
15               check((value(FIRW))=="False");
16               check((toInteger(value(BW)))>= 1500);
17
18
19
20             }
21             action
22       {
23
24         new(instanceOf(SC,DSM.metamodel.SkypeSim.SkypeSimEditor.
               SN));
25         let RG1=undef  in seq{
26         new (RS.RSoverlay(RG1,S,X));
27         rename(SC,"sn");
28         delete(instanceOf(X,DSM.metamodel.SkypeSim.SkypeSimEditor
               .SC));
29         delete(L);
30         setValue(BW1,((toInteger(value(BW)))+5));
31         }
32         println(" promoted at linked stage Super Node");
33
34       }
```

```
35      }
```

Listing B.15: SC is promoted to SN while SC is linked with SN

B.2.4.3 Need Based Protocol for Promotion of SC to SN

The rule in Fig. 6.30 is mapped to VIATRA2 rule in List. B.16

```
1  gtrule Rule6_Promote_at_Need_based() =
2      {
3              precondition pattern lhs(X,S,L,BW,BW1) =
4              {
5                SC(X);
6                RS(S);
7                SN(Y);
8                LK(L);
9                find SC_Linked_SN(X,Y,L);
10               find SC_firewall(X,FIRW);
11               find SN_Bandwidth(Y,BW1);
12               find SC_Bandwidth(X,BW);
13               check((value(FIRW))=="False");
14               check((toInteger(value(BW)))>= 1500);
15               check((toInteger(value(BW1)))<= 1000);
16
17             }
18             action
19       {
20               new(instanceOf(SC,DSM.metamodel.SkypeSim.
                     SkypeSimEditor.SN));
21         let RG1=undef  in seq{
22         new (RS.RSoverlay(RG1,S,X));
23         rename(SC,"sn");
24         delete(instanceOf(X,DSM.metamodel.SkypeSim.SkypeSimEditor
               .SC));
25         delete(LK);
26         setValue(BW1,((toInteger(value(BW)))+5));
27         }
28         println(" promoted at need based");
29
30       }}    }
```

Listing B.16: SSC is promoted to SN based on the need

B.2.5 Overlay Network among SNs

The rule in Fig. 6.31 is mapped to VIATRA2 rule in List. B.17

```
1  gtrule Rule7_connect_SN_SN() =
2      {
```

```
3               precondition pattern lhs(Y,Y1,BW,BWR,Network) =
4               {
5                 RS(S);
6                 SN(Y);
7                 SN(Y1);
8                 Network(NK);
9                 find SN_Exit(Y1,EX);
10                find SN_Exit(Y,EX1);
11                find SN_Bandwidth(y,BW);
12                find SN_Bandwidth(y1,BWR);
13                neg find SN_SN_Test(Y,Y1);
14                neg find SN_SN_Test(Y1,Y);
15                neg find OV_Path(y);
16                check((toInteger(value(BW)))>256);
17                check((toInteger(value(BWR)))>256);
18                check((value(EX))=="False");
19                check((value(EX1))=="False");
20              }
21              action
22        {
23          let O=undef, CO=undef,OU=undef,NR=undef in seq{
24                  new (OV(O) in NK);
25                  new(Network.nodes(NR,NK,O));
26                  new(OV.source(CO,O,Y));
27                  new(OV.target(OU,O,Y1));
28                    setValue(BW,((toInteger(value(BW)))-5));
29              setValue(BWR,((toInteger(value(BWR)))-5));
30                println("Connected  "+fqn(SN)+"with"+fqn(SN));
31              }
32      }}
```

Listing B.17: Overlay network among SNs

B.2.6 Demotion of Super Peer

The rule in Fig. 6.33 is mapped to VIATRA2 rule in List. B.18

```
1  gtrule Rule9_Down_to_SC() =
2      {
3               precondition pattern lhs(Y,BW,RG,S) =
4               {
5                 RS(S);
6                 SN(Y);
7                 find RS_rg_SN(S,Y,RG);
8                 find SN_Bandwidth(Y,BW);
9               neg find Check_Links_with_SN(Y);
10              neg find Check_OV_Links_with_SN(Y);
11              check((toInteger(value(BW)))<256);
12              }
```

```
13              action
14        {
15          new (instanceOf(SN,DSM.metamodel.SkypeSim.SkypeSimEditor.
                SC));
16          delete (instanceOf(SN,DSM.metamodel.SkypeSim.
                SkypeSimEditor.SN));
17          delete(RG);
18          let REG=undef in new(RS.register(REG,S,Y));
19          println("Down to Ordinary SC ");
20              }
21      }
```

Listing B.18: Departures/ Chrun of SC and SN

B.2.7 Departure of SN and SC

B.2.7.1 Control Departure of SC

- Control Departure Selection of SC

The rule in Fig. 6.34 is mapped to VIATRA2 rule in List. B.19

```
1  gtrule Rule10_Controlled_Exit_of_SC() =
2      {
3              precondition pattern lhs(X,EX) =
4              {
5                  SC(X);
6                  find SC_Exit(X,EX);
7                  check((value(EX))=="False");
8
9              }
10
11         action
12         {
13           setValue(EX,"True");
14
15                 println("SC About to Exit from Network");
16             }
17     }
```

Listing B.19: SC informs that SC is leaving network

- Control Departure Preparation of SC

The rule in Fig. 6.35 is mapped to VIATRA2 rule in List. B.20

```
1  gtrule Rule10_Controled_Exit_Preperation_SC() =
2      {
3              precondition pattern lhs(Y,X,BW,BWR,L,S) =
4              {
5                // Controlled Exit of Client from Network
6
```

```
7                 SN(Y);
8               SC(X);
9               LK(L);
10              RS(S);
11              find SC_Exit(X,EX);
12              find RS_rg_SN(S,Y,RG);
13                find SN_Bandwidth(Y,BW);
14              find SC_Bandwidth(X,BWR);
15              find Check_SC_SN_Link(X,Y,LK);
16                check((value(EX))=="True");
17
18              }
19              action
20        {
21
22          let RG=undef in new (RS.register(RG,S,X));
23          delete(L);
24          setValue(BW,((toInteger(value(BW)))+5));
25          setValue(BWR,((toInteger(value(BWR)))+5));
26          println("SC disconnected from SN due to SC Exit");
27              }
28      }
```

Listing B.20: SC inform SN and break their common link

- Control Final Departure SC

The rule in Fig. 6.36 is mapped to VIATRA2 rule in List. B.21

```
1
2   gtrule Rule10_Controled_Final_Exit_SC() =
3       {
4               precondition pattern lhs(X,EX,RL) =
5               {
6                   SC(SC);
7                   RL(RL);// Registerd client list or offline client
                        list
8                   find SC_Exit(X,EX);
9                   neg find Check_SC_Source(SC);
10                  check((value(EX))=="True");
11
12
13              }
14
15
16        action
17        {
18        let KK=undef in seq{
19          new(RL.ofline(KK,RL,X));
20          move(SC,RL);
21    new(instanceOf(SC,DSM.metamodel.SkypeSim.SkypeSimEditor.ND));
```

```
22          println("SC goes offline");
23              }
24      }  }
```

Listing B.21: SC goes offline in a network

Uncontrol Departure of SC

The rule in Fig. 6.37 is mapped to VIATRA2 rule in List. B.22

```
1  gtrule Rule11_UnControled__Exit_SC() =
2      {
3              precondition pattern lhs(X,RL) =
4              {
5                  SC(X);
6                  RL(RL);
7              }
8
9        action
10       {
11         delete(X);// to be disscuessed with reiko
12         let N=undef,KK=undef  in seq {
13                 new (ND(N) in RL);// create client inside the
                       network
14                 new(RL.ofline(KK,RL,ND));
15       }
16                println("SC goes offline from Network ");
17             }
18     }
```

Listing B.22: SC goes offline without informing SN

B.2.7.2 Control Departure of SN

Control Departure Selection of SN

The rule in Fig. 6.38 is mapped to VIATRA2 rule in List. B.23

```
1  gtrule Rule12_Controled_Exit_SN_Selected() =
2      {
3              precondition pattern lhs(Y,EX) =
4              {
5
6                SN(Y);
7                find SN_Exit(Y,EX);
8              check((value(EX))=="False");
9              }
10
11
12       action
13       {
14         setValue(EX,"True");
```

```
15                println("SN selected for controlled Exit");
16            }
17    }
```

Listing B.23: SN is leaving the network soon

Control Departure Preparation of SN through De-linking SCs
The rule in Fig. 6.39 is mapped to VIATRA2 rule in List. B.24

```
1  gtrule Rule_Control_Deparure_Preperation_SN_De_linking_SCs() =
2      {
3              precondition pattern lhs(Y,X,BW,BWR,L,S) =
4              {
5
6                SN(Y);
7              SC(X);
8              LK(L);
9              RS(S);
10             find SN_Exit(Y,EX);
11             find RS_rg_SN(S,Y,RG);
12               find SN_Bandwidth(Y,BW);
13             find SC_Bandwidth(X,BWR);
14             find Check_SC_SN_Link(X,Y,LK);
15               check((value(EX))=="True");
16
17             }
18             action
19       {
20
21         let RG=undef in new (RS.register(RG,RS,SC));
22         delete(L);
23         setValue(BW,((toInteger(value(BW)))+4));
24         setValue(BWR,((toInteger(value(BWR)))+4));
25     println("SC disconnected from SN due to SN Controlled Exit");
26             }
27     }
```

Listing B.24: SN Links between SC and SN is removed

Control Departure Preparation of SN through De-linking SNs
The rule in Fig. 6.40 is mapped to VIATRA2 rule in List. B.25

```
1  gtrule Rule_Control_Deparure_Preperation_SN_De_linking_SNs_Source
       () =
2      {
3
4              precondition pattern lhs(Y,BW,BWR,O,EX) =
5              {
6                SN(Y);
7              SN(Y1);
8              OV(O);
```

```
9               find SN_Exit(Y,EX);
10              find SN_Bandwidth(Y,BW);
11              find SN_Bandwidth(Y1,BWR);
12              find Check_OV(Y,Y1,OV);
13              check((value(EX))=="True");
14
15              }
16              action
17        {
18          delete(O);
19          setValue(BWR,((toInteger(value(BWR)))+4));
20          setValue(BW,((toInteger(value(BW)))+4));
21  println("Edge between SN and SN1 removed Due to Controlled Exit
        of SN ");
22              }
23      }
```

Listing B.25: SN informs other SNs in the overlay network and retract connections

Control Final Departure SN

The rule in Fig. 6.41 is mapped to VIATRA2 rule in List. B.27

```
1   gtrule Control_Final_Departure_of_SN() =
2       {
3               precondition pattern lhs(SN,RL) =
4               {
5
6                   RL(RL);
7                 SN(Y);
8                 find SN_Exit(Y,EX);
9                 neg find Check_OV_Links(Y);
10                neg find Check_LK_Links(Y);
11              check((value(EX))=="True");
12
13              }
14        action
15        {
16
17
18    new (instanceOf(SN,DSM.metamodel.SkypeSim.SkypeSimEditor.ND));
19    delete (instanceOf(SN,DSM.metamodel.SkypeSim.SkypeSimEditor.SN)
          );
20          let KK=undef in seq{
21          new(RL.ofline(KK,RL,Y));
22          move(Y,RL);
23          }
24
25    println("SN Departed from Network Through Proper exit");
26              }   }
```

Listing B.26: SN finally go offline

Uncontrol Departure of SN

The rule in Fig. 6.42 is mapped to VIATRA2 rule in List. B.27

```
gtrule Rule13_departure_of_SN_Uncontrolled() =
    {
            precondition pattern lhs(Y) =
            {

              SN(Y);
              RL(RL);

            }

      action
      {
        delete(Y);
        let N=undef,KK=undef  in seq {
                new (ND(N1) in RL);
                new(RL.ofline(KK,RL,N));
          }
 println("SN Departed from Network due to Uncontrolled behaviour"
    );
            }
    }
```

Listing B.27: SN leave the network without informing either SN or SC

B.2.8 VoIP Calls

B.2.8.1 Call Request Initiated and Placement

Call Request by SC behind Firewall/NAT

The rule in Fig. 6.43 is mapped to VIATRA2 rule in List. B.28

```
gtrule Requiest_call_SC_to_SC_behind_firewall() =
    {
            precondition pattern lhs(NK,X) =
            {
              SC(X);
              Network(NK);
              neg find check_for_call(X);
              neg find existing_req(X);
              find SC_firewall(X,FIRW);
              LK(L);
              find SC_Source(L,X);
              check((value(FIRW))=="True");

            }

```

```
16              action {
17                  let PK=undef, PKS=undef in seq{
18                  new (CallReq(PK) in Network);
19                  new (CallReq.initiate(PKS,PK,X));
20                      println("Call Request Generated");
21
22          }
23            }
24
25      }
```

Listing B.28: SC behind firewall initiate request for VoIP call

Call Request by SC having Static IP

The rule in Fig. 6.44 is mapped to VIATRA2 rule in List. B.29

```
1  gtrule Requiest_call_SC_with_live_IP() =
2      {
3              precondition pattern lhs(NK,X) =
4              {
5                SC(X);
6                Network(NK);
7                neg find check_for_call(X);
8                neg find existing_req(X);
9                find SC_firewall(X,FIRW);
10               LK(L);
11               find SC_Source(L,X);
12               check((value(FIRW))=="False");
13
14             }
15
16             action {
17                 let PK=undef, PKS=undef in seq{
18                 new (CallReq(PK) in NK);
19                 new (CallReq.initiate(PKS,PK,X));
20                     println("Call Request Generated");
21
22         }
23           }
24
25     }
```

Listing B.29: SC with live internet connection initiate request for VoIP call

- SN initiate P2P VoIP call

The rule in Fig. 6.51 is mapped to VIATRA2 rule in List. B.30

```
1  gtrule Requiest_call_SN() =
2      {
3              precondition pattern lhs(NK,Y) =
4              {
```

```
5                 SN(Y);
6                 Network(NK);
7                 neg find check_for_call_SN(Y);
8                 neg find existing_req_SN(Y);
9               }
10
11              action {
12                  let PK=undef, PKS=undef in seq{
13                  new (CallReq(PK) in NK);
14                  new (CallReq.initiate(PKS,PK,Y));
15
16                      println("Call Request Generated by SN");
17
18            }
19              }
20
21      }
```

Listing B.30: SN initiate P2P VoIP call

B.2.8.2 Call Placement

P2P Call between SCs routed through Multiple SNs

The rule in Fig. 6.46 is mapped to VIATRA2 rule in List. B.31

```
1  gtrule P2P_Call_with_firewaal_enabled_Place_call_and_Route() =
2      {
3               precondition pattern lhs(X,X1,Y,Y1,SCBW,SCBW1,SNBW,
                    SNBW1,L,L1,NK,Q) =
4               {
5                 SN(Y);
6                 SN(Y1);
7                 SC(X);
8                 SC(X1);
9                 LK(L);
10                LK(L1);
11                Network(NK);
12                find LK_LinkTo(L,X);
13                find LK_OutTo(L,Y);
14                find LK_LinkTo(L1,X1);
15                find LK_OutTo(1,Y1);
16                find SC_Bandwidth(X,SCBW);
17                find SC_Bandwidth(X1,SCBW1);
18                find SN_Bandwidth(Y,SNBW);
19                find SN_Bandwidth(Y1,SNBW1);
20                find SC_firewall(X,FIRW);
21                find SC_firewall(X1,FIRW1);
22                neg  find check_for_existing_call(X);
23                  neg  find check_for_existing_call(X1);
```

```
                CallReq(CallReq);
              find check_requiest_for_call(X,Q);
              neg find existing_req(X1);
              check((value(FIRW))=="True");
              check((value(FIRW))=="True");
            }

            action {
            let PK=undef, PKS=undef,PKR=undef,SNR=undef,SNRR=
                undef in seq{
                 new (RouteCall(PK) in NK);
                 new (RouteCall.caller(PKS,PK,X));
                 new (RouteCall.callee(PKR,PK,X1));
                 new (RouteCall.route(SNR,PK,Y));
                 new (RouteCall.route(SNRR,PK,Y1));
                 setValue(SCBW,((toInteger(value(SCBW)))-80));
                 setValue(SCBW1,((toInteger(value(SCBW1)))-80));
                 setValue(SNBW,((toInteger(value(SNBW)))-80));
                 setValue(SNBW1,((toInteger(value(SNBW1)))-80));
                     println("Call Generated between SC and SC1
                        through SN");
           delete(CallReq);
            }
    }

    }
```

Listing B.31: Caller and Callee both are behind firewall

P2P Call between SCs routed through Single SN

The rule in Fig. 6.47 is mapped to VIATRA2 rule in List. B.32

```
gtrule Place_call_and_Route_Local() =
    {
            precondition pattern lhs(X,X1,Y,SCBW,SCBW1,SNBW,LK,
               LK1,NK,Q) =
            {
              SN(Y);
              SC(X);
              SC(X1);
              LK(L);
              LK(L1);
              Network(NK);
              find LK_LinkTo(L,X);
              find LK_OutTo(L,Y);
              find LK_LinkTo(L1,X1);
              find LK_OutTo(L1,Y);
              find SC_Bandwidth(X,SCBW);
              find SC_Bandwidth(X1,SCBW1);
              find SN_Bandwidth(Y,SNBW);
```

```
18                check((toInteger(value(SCBW)))<256);
19                    check((toInteger(value(SCBW1)))<256);
20                neg  find check_for_existing_call(X);
21                    neg  find check_for_existing_call(X1);
22                    CallReq(Q);
23                find check_requiest_for_call(X,Q);
24                neg find existing_req(X1);
25             }
26
27             action {
28                 let PK=undef, PKS=undef,PKR=undef,SNR=undef in
                       seq{
29                 new (RouteCall(PK) in NK);
30                 new (RouteCall.caller(PKS,PK,X));
31                 new (RouteCall.callee(PKR,PK,X1));
32                 new (RouteCall.route(SNR,PK,Y));
33                 setValue(SCBW,((toInteger(value(SCBW)))-80));
34                 setValue(SCBW1,((toInteger(value(SCBW1)))-80));
35                 setValue(SNBW,((toInteger(value(SNBW)))-80));
36       println("Call Generated between SC and SC1 through SN
             Locallâ );
37               delete(Q);
38
39             }
40
41
42     }
43
44     }
```

Listing B.32: Caller and Callee both are behind firewall

P2P Direct Call between SCs

The rule in Fig. 6.49 is mapped to VIATRA2 rule in List. 6.49

```
1  gtrule Place_call_SC_to_SC() =
2      {
3              precondition pattern lhs(L,L1,X1,X,SCBW,SCBW1,NK,Q) =
4              {
5                SC(X);
6                SC(X1);
7                LK(L);
8                LK(L1);
9                find SC_Source(L,X);
10               find SC_Source(L1,X1);
11               CallReq(Q);
12                 find SC_firewall(X,FIRW);
13                 find SC_firewall(X,FIRW1);
14               find check_requiest_for_call(X,Q);
15               neg find existing_req(X1);
```

```
16                Network(NK);
17                find SC_Bandwidth(X,SCBW);
18                find SC_Bandwidth(X1,SCBW1);
19                neg find check_for_call(X);
20                neg find check_for_call(X1);
21                check((value(FIRW))=="False");
22                check((value(FIRW1))=="False");
23
24
25             }
26
27             action {
28                 let PK=undef, PKS=undef,PKR=undef in seq{
29                 new (Call(PK) in NK);
30                 new (Call.caller(PKS,PK,X));
31                 new (Call.callee(PKR,PK,X1));
32                 setValue(SCBW,((toInteger(value(SCBW)))-80));
33                 setValue(SCBW1,((toInteger(value(SCBW1)))-80));
34                 delete (CallReq);
35    println("Call Generated between SC and SC1 ANd both are having
          live IP");
36
37            }
38             }
39     }
```

Listing B.33: SC to SC direct P2P VoIP call

P2P Direct Call between SC and SN

The rule in Fig. 6.50 is mapped to VIATRA2 rule in List. B.34

```
1  gtrule Place_call_SC_to_SN() =
2      {
3              precondition pattern lhs(L,Y,X,SCBW,SNBW1,NK,Q) =
4              {
5                SC(X);
6                SN(Y);
7                LK(L);
8        CallReq(Q);
9                find SC_Source(L,X);
10               Network(N);
11               find SC_Bandwidth(X,SCBW);
12               find SN_Bandwidth(Y,SNBW1);
13               find SC_firewall(X,FIRW);
14               neg find check_for_call(X);
15               neg find check_for_call_SN(Y);
16               check((toInteger(value(SCBW)))>256);
17                   check((toInteger(value(SNBW1)))>256);
18               find check_requiest_for_call(X,Q);
19               neg find existing_req_SN(Y);
```

```
20                check((value(FIRW))=="False");
21
22            }
23
24            action {
25                let PK=undef, PKS=undef,PKR=undef in seq{
26                new (Call(PK) in NK);
27                new (Call.caller(PKS,PK,Y));
28                new (Call.callee(PKR,PK,X));
29                setValue(SCBW,((toInteger(value(SCBW)))-80));
30                setValue(SNBW1,((toInteger(value(SNBW1)))-80));
31                    println("Call Generated between SN and SC1");
32                    delete(Q);
33
34          }
35            }
36    }
```

Listing B.34: SC to SN direct P2P VoIP call

- P2P Direct Call between SN and SN

The rule in Fig. 6.51 is mapped to VIATRA2 rule in List. B.35

```
1  gtrule Place_call_SN_to_SN() =
2      {
3              precondition pattern lhs(Y,Y1,SNBW,SNBW1,NK,Q) =
4              {
5                SN(Y);
6                SN(Y1);
7                Network(NK);
8                CallReq(Q);
9                find check_requiest_for_call_SN(Y,Q);
10               neg find existing_req_SN(Y1);
11               find SN_Bandwidth(Y,SNBW);
12               find SN_Bandwidth(Y1,SNBW1);
13                 neg find check_for_call_SN(Y);
14                 neg find check_for_call_SN(Y1);
15               check((toInteger(value(SNBW)))>256);
16                 check((toInteger(value(SNBW1)))>256);
17
18
19             }
20
21             action {
22                 let PK=undef, PKS=undef,PKR=undef in seq{
23                 new (Call(PK) in NK);
24                 new (Call.caller(PKS,PK,Y));
25                 new (Call.callee(PKR,PK,Y1));
26                 setValue(SNBW,((toInteger(value(SNBW)))-80));
27                 setValue(SNBW1,((toInteger(value(SNBW1)))-80));
```

```
28                      println("Call Generated between SN and SN");
29                      delete(Q);
30
31          }
32            }
33
34      }
```

Listing B.35: SN to SN P2P VoIP call

B.2.8.3 VoIP Call Terminations

- P2P Direct VoIP call Termination Between SCs

The rule in Fig. 6.52 is mapped to VIATRA2 rule in List. B.36

```
1  gtrule Terminate_SC_to_SC() =
2      {
3              precondition pattern lhs(C,X1,X,SNBW,SNBW1,NK) =
4              {
5                SC(X);
6                SC(X1);
7                Call(C);
8                Network(NK);
9                find SC_Bandwidth(X,SNBW);
10               find SC_Bandwidth(X1,SNBW1);
11               find call_check_1C(X,X1,Call);
12                 }
13
14             action {
15                 delete(C);
16                 setValue(SNBW,((toInteger(value(SNBW)))+80));
17                 setValue(SNBW1,((toInteger(value(SNBW1)))+80));
18                     println("Call terminated between SC and SC");
19
20          }
21
22 }
```

Listing B.36: Terminate SC to SC P2P direct VoIP call

- P2P Direct VoIP call Termination Between SC and SN

The rule in Fig. 6.53 is mapped to VIATRA2 rule in List. B.37

```
1  gtrule Terminate_SC_to_SN() =
2      {
3              precondition pattern lhs(C,Y,X,SNBW,SNBW1,NK) =
4              {
5                SN(Y);
6                SC(X);
7                Network(NK);
```

```
8                 find SN_Bandwidth(Y,SNBW);
9                 find SC_Bandwidth(X,SNBW1);
10                find call_check_SC(Y,X,Call);
11                          }
12
13              action {
14                  delete(C);
15                  setValue(SNBW,((toInteger(value(SNBW)))+80));
16                  setValue(SNBW1,((toInteger(value(SNBW1)))+80));
17                          println("Call terminated between SC and
                               SN");
18
19          }
20
21      }
```

Listing B.37: Terminate SC to SN P2P direct VoIP call

- P2P Direct VoIP call Termination between SNs

The rule in Fig. 6.55 is mapped to VIATRA2 rule in List. B.38

```
1  gtrule Terminate_SN_to_SN() =
2      {
3              precondition pattern lhs(C,Y,Y1,SNBW,SNBW1,NK) =
4              {
5                SN(Y);
6                SN(Y1);
7                Network(NK;
8                find SN_Bandwidth(Y,SNBW);
9                find SN_Bandwidth(Y1,SNBW1);
10               find call_check(Y,Y1,C);
11                }
12
13             action {
14                 delete(C);
15                 setValue(SNBW,((toInteger(value(SNBW)))+80));
16                 setValue(SNBW1,((toInteger(value(SNBW1)))+80));
17                       println("Call terminated between SN and SN"
                            );
18
19          }
20
21      }
```

Listing B.38: Terminate SN to SN P2P direct VoIP call

- P2P Routed VoIP Call Termination Established through Single SN

The rule in Fig. 6.56 is mapped to VIATRA2 rule in List. B.39

```
1  gtrule Terminate_routed_Call_SC_SN_SC() =
```

```
2     {
3             precondition pattern lhs(X,X1,Y,SCBW,SCBW1,SNBW,R) =
4             {
5               SN(Y);
6               SC(X);
7               SC(X1);
8               RouteCall(R);
9               find Find_routed_call_SC_SN_SC(X,X1,Y,R);
10              find SC_Bandwidth(X,SCBW);
11              find SC_Bandwidth(X1,SCBW1);
12              find SN_Bandwidth(Y,SNBW);
13                        }
14                        action {
15              delete (R);
16                setValue(SCBW,((toInteger(value(SCBW)))+80));
17                setValue(SCBW1,((toInteger(value(SCBW1)))+80));
18                setValue(SNBW,((toInteger(value(SNBW)))+80))
19
20              println("Routed call terminated Between SC to SN
                    and SC");
21
22            }
23
24    }
```

Listing B.39: Terminate SC to SC single SN routed VoIP call

- P2P Routed VoIP Call Termination Established through Multiple SNs

The rule in Fig. 6.57 is mapped to VIATRA2 rule in List. B.40

```
1  gtrule Terminate_routed_Call_SC_SN_SN_SC() =
2     {
3             precondition pattern lhs(X,X1,Y,Y1,SCBW,SCBW1,SNBW,
                 SNBW1,R) =
4             {
5               SN(Y);
6               SN(Y1);
7               SC(X);
8               SC(X1);
9               RouteCall(R);
10              find Find_routed_call(X,X1,Y,Y1,R);
11              find SC_Bandwidth(X,SCBW);
12              find SC_Bandwidth(X1,SCBW1);
13              find SN_Bandwidth(Y,SNBW);
14              find SN_Bandwidth(Y1,SNBW1);
15
16            }
17
18            action {
```

```
19                delete (R);
20                  setValue(SCBW,((toInteger(value(SCBW)))+80));
21                  setValue(SCBW1,((toInteger(value(SCBW1)))+80));
22                  setValue(SNBW,((toInteger(value(SNBW)))+80));
23                  setValue(SNBW1,((toInteger(value(SNBW1)))+80));
24              println("Routed call terminated Between SC to SN SN
                    and SC");
25
26              }
27
28
29      }
```

Listing B.40: Terminate SC to SC Multiple SNs routed VoIP call

B.2.9 Background Traffic Generation

SN Transmitting and Receiving other Traffic The rule in Fig. 6.66 is mapped to VIATRA2 rule in List. B.41

```
1  gtrule Random_SN_Traffic() =
2      {
3              precondition pattern lhs(Y,BW) =
4              {
5
6                SN(Y);
7                find SN_Exit(Y,EX);
8                find SN_Bandwidth(Y,BW);
9                check((toInteger(value(BW)))>= 256);
10               check((value(EX))=="False");
11             }
12
13             action {
14               setValue(BW,skypesim.random(256,2000));
15               println("Background Traffic Created/removed");
16
17             }
18     }
```

Listing B.41: Random Other Traffic at SN

- SC Transmitting and Receiving other Traffic (behind firewall The rule in Fig. 6.67 is mapped to VIATRA2 rule in List. B.42

```
1  gtrule Random_SC_Traffic() =
2      {
3              precondition pattern lhs(L,X,BW) =
4              {
5
6                LK(L);
```

```
7                 SC(X);
8                 find Link_SC_LK(X,L);
9       find SC_firewall(X,FIRW);
10                find SC_Bandwidth(X,BW);
11                check((toInteger(value(BW)))>= 256);
12                    check((value(FIRW))=="True");
13
14              }
15
16              action {
17                setValue(BW,skypesim.random(256,1000));
18                println("Background Traffic Created/removed");
19              }
20      }
```

Listing B.42: Random other traffic at SC

- SC Transmitting and Receiving other Traffic

The rule in Fig. 6.68 is mapped to VIATRA2 rule in List. B.43

```
1  gtrule Random_SC_Traffic() =
2      {
3              precondition pattern lhs(L,X,BW) =
4              {
5
6                LK(L);
7                SC(X);
8                find Link_SC_LK(X,L);
9            find SC_firewall(X,FIRW);
10               find SC_Bandwidth(X,BW);
11               check((toInteger(value(BW)))>= 256);
12                 check((value(FIRW))=="False");
13             }
14
15             action {
16               setValue(BW,skypesim.random(256,2000));
17               println("Background Traffic Created/removed");
18             }
19     }
```

Listing B.43: Random Other Traffic at SC

- SN Transmitting and Receiving Uniform other Traffic

The rule in Fig. 6.69 and Fig. 6.70 is mapped to VIATRA2 rule in List. B.44 and List. B.45.

```
1  gtrule Rule16_Increment_SN_Bandwidth() =
2      {
3              precondition pattern lhs(Y,BW) =
4              {
5                SN(Y);
```

```
6                   find SN_Bandwidth(Y,BW);
7                   check((toInteger(value(BW)))>256);
8               }
9
10              action {
11              setValue(BW,((toInteger(value(BW)))+80));
12              println("Bandwidth Increased ");
13              }
14
15      }
```

Listing B.44: uniform reduce bandwidth

```
1   gtrule Rule16_Decrement_SN_Bandwidth() =
2       {
3               precondition pattern lhs(Y,BW) =
4               {
5                 SN(Y);
6                   find SN_Bandwidth(Y,BW);
7                   check((toInteger(value(BW)))>256);
8               }
9
10              action {
11
12              setValue(BW,((toInteger(value(BW)))-80));
13
14              println("Bandwidth reduced due to traffic load");
15              }
16
17      }
```

Listing B.45: Uniform increase bandwidth

- SC Transmitting and Receiving Uniform other Traffic

The rule in Fig. 6.71 and Fig. 6.72 is mapped to VIATRA2 rules in List. B.46 and List. B.47.

```
1   gtrule Rule16_Decrement_SC_Bandwidth() =
2       {
3               precondition pattern lhs(X,BW) =
4               {
5                 SC(X);
6                   find SC_Bandwidth(X,BW);
7                   check((toInteger(value(BW)))>256);
8
9               }
10
11              action {
12
13              setValue(BW,((toInteger(value(BW)))-80));
```

```
14              println("Bandwidth reduced due to traffic laod");
15              }
16      }
```

Listing B.46: Uniform reduce bandwidth

```
1
2  gtrule Rule16_increment_SC_Bandwidth() =
3      {
4              precondition pattern lhs(X,BW) =
5              {
6                SC(X);
7                  find SC_Bandwidth(X,BW);
8                  check((toInteger(value(BW)))>256);
9
10             }
11
12             action {
13
14             setValue(BW,((toInteger(value(BW)))+80));
15
16             println("Bandwidth reduced due to traffic laod");
17             }
18     }
```

Listing B.47: Uniform increase bandwidth

B.2.10 Reconfiguration Rules

To Link to a New SN when Existing SN leaves Network
- Reconfigure and link to a New Random SN based on bandwidth

The rule in Fig. 6.59 is mapped to VIATRA2 rule in List. B.49.

```
1  gtrule Rule4_Reconfigure_Link_SC_to_link_back_SN() =
2      {
3              precondition pattern lhs(L,Y,X,BW) =
4              {
5
6                LK(L);
7                SN(Y);
8                SC(X);
9                find SN_Exit(Y,EX);
10               find SN_Bandwidth(Y,BW);
11               find Link_SC_LK(X,LK);
12               neg find Link_SN_LK(Y,LK);
13               neg find Test_SN_LK(LK);
14               check((toInteger(value(BW)))>256);
15               check((value(EX))=="False");
16             }
```

```
17
18              action {
19                let OU=undef in seq{
20                new(LK.outTo(OU,LK,Y));
21                setValue(BW,((toInteger(value(BW)))-5));
22                }
23                println("Re-linked Back to New Super Node");
24
25              }
26      }
```

Listing B.48: Reconfigure and link SC to a new SN

- Reconfigure and link to a New Random SN based on Bandwidth and Latency

The rule in Fig. 6.60 is mapped to VIATRA2 rule in List. B.49.

```
1  gtrule Rule4_Reconfigure_Link_SC_to_link_back_SN_latency() =
2      {
3              precondition pattern lhs(S,L,Y,X,BW,CRO) =
4              {
5
6                LK(L);
7                SN(Y);
8                SC(X);
9                RS(S);
10               find SN_Exit(Y,EX);
11               find SN_Bandwidth(Y,BW);
12               find SC_Chronos(X,CRO);
13               find Link_SC_LK(X,LK);
14               neg find Packet_SE_check(X);
15                 neg find Packet_RE_check(X);
16               neg find Link_SN_LK(Y,L);
17               neg find Test_SN_LK(L);
18               check((toInteger(value(BW)))>256);
19               check((value(EX))=="False");
20             }
21
22             action {
23               delete(L);
24         let RG=undef in new (RS.online(RG,S,X));
25               let P=undef,
26             PC=undef,
27             CRR=undef,
28             PCO=undef,
29             PCR=undef,
30             SE=undef,
31             RE=undef,
32             AH=undef
33             in seq {
```

```
34              new(Packet(P)in Y);
35              //Create Packet in SC
36              new(Packet.Chronos(PC)in P);                    //Create
                   a chronos attribute in Packet
37              new(Packet.chronos(CRR,P,PC));
38          //Create a relation between packet and Chronos entity
39              new(Packet.Content(PCO)in P);
40          //Create a content attribute in Packet
41              new(Packet.content(PCR,P,PCO));
42        //Create a relation between packet and content entity
43              new(Packet.sender(SE,P,X));
44        // Create a relation
45              new(Packet.receiver(RE,P,Y));
46        // Create a relation
47              new(Packet.at(AH,Pt,Y));
48        // Create a relation
49              setValue(CRO,systime());
50              setValue(PC,value(CRO));
51              setValue(PCO,"Ping");
52                println("Ping Packet Created and Transmitted ");
53
54              }
55      }   }
```

Listing B.49: Reconfigure and send a time stamped packet to a new SN

- Reconfigure and Link to a New SN based on Bandwidth and Locality

The rule in Fig. 6.61 is mapped to VIATRA2 rule in List. B.50.

```
1  gtrule Rule4_Reconfigure_Link_SC_to_link_back_a_SN_based() =
2      {
3              precondition pattern lhs(L,Y,X,BW) =
4              {
5
6                LK(L);
7                SN(Y);
8                SC(X);
9                find SN_Exit(Y,EX);
10               find SN_Bandwidth(Y,BW);
11               find Link_SC_LK(X,LK);
12               neg find Link_SN_LK(Y,LK);
13               neg find Test_SN_LK(L);
14               check((toInteger(value(BW)))>256);
15               check((value(EX))=="False");
16
17               //this part is for region based
18                 find SC_Location(X,LOC);
19               find SN_Location(Y,LON);
20               check((toInteger(value(LOC)))== (toInteger(value(
```

```
                    LON))));
21              }
22
23              action {
24                let OU=undef in seq{
25                new(LK.outTo(OU,LK,Y));
26                setValue(BW,((toInteger(value(BW)))-5));
27                }
28                println("Re-linked Back to New Super Node");
29
30              }
31      }
```

Listing B.50: Reconfigure and re-link to a new SN based on bandwidth and region

- Reconfigure and link to a New SN based on bandwidth and time spent in the network

The rule in Fig. 6.62 is mapped to VIATRA2 rule in List. B.52.

```
1  gtrule Rule4_Reconfigure_Link_SC_to_link_back_SN() =
2      {
3              precondition pattern lhs(L,Y,X,BW) =
4              {
5                  LK(L);
6                SN(Y);
7                SC(X);
8                find SN_Exit(Y,EX);
9                find SN_Bandwidth(Y,BW);
10               find SN_Chronos(Y,CRO);
11               find Link_SC_LK(X,LK);
12               neg find Link_SN_LK(Y,LK);
13               neg find Test_SN_LK(L);
14               check((toInteger(value(BW)))>256);
15               check((toInteger(value(CRO)))>100);
16               check((value(EX))=="False");
17             }
18
19             action {
20               let OU=undef in seq{
21               new(LK.outTo(OU,LK,Y));
22               setValue(BW,((toInteger(value(BW)))-5));
23               }
24               println("Re-linked Back to New Super Node");
25
26             }
27     }
```

Listing B.51: Reconfigure and re-link to a new SN based on bandwidth and time spent in network

- Reconfigure and Recover Bandwidth due to Crashing of SN as Target in Overlay

The rule in Fig. 6.63 is mapped to VIATRA2 rule in List. B.52.

```
gtrule Rule14_Recovery_SN_Target_SN1() =
    {
            precondition pattern lhs(BW,Y,O) =
            {

              OV(O);
              SN(Y);
              find SN_Bandwidth(Y,BW);
              find OV_Target(O,Y);
              neg find OV_Test_Source(O);

            }

            action {
              delete(O);
              setValue(BW,((toInteger(value(BW)))+5));

              println("Bandwidthe Recovered as Target SN Departed
                  ");

            }
    }
```

Listing B.52: Reconfigure and remove overlay source edge

- Reconfigure and recover bandwidth due to crashing of SN as Source in Overlay

The rule in Fig. 6.64 is mapped to VIATRA2 rule in List. B.53.

```
gtrule Rule14_Recovery_SN_Source_SN1() =
    {
            precondition pattern lhs(BW,Y,O) =
            {

              OV(O);
              SN(SN);
              find SN_Bandwidth(Y,BW);
              find OV_Source(O,Y);
              neg find OV_Test_Target(O);

            }

            action {
              delete(O);
              setValue(BW,((toInteger(value(BW)))+5));

```

```
18                println("Bandwidthe Recovered as Target SN Departed
                      ");
19
20              }
21      }
```

Listing B.53: Reconfigure and remove overlay source edge

- Reconfigure and Recover Bandwidth due to Crashing of SC

The rule in Fig. 6.65 is mapped to VIATRA2 rule in List. B.54.

```
1  gtrule Rule14_Recovery_SN_OutTO_SC() =
2      {
3              precondition pattern lhs(BW,Y,L) =
4              {
5
6                LK(L);
7                SN(Y);
8                find SN_Bandwidth(Y,BW);
9                find LK_OutTo(L,Y);
10               neg find LK_test_LinkTo(L);
11
12             }
13
14             action {
15               delete(L);
16               setValue(BW,((toInteger(value(BW)))+5));
17                 println("Bandwidthe Recovered as Linked  SN
                       Departed from Network");
18
19             }
20     }
21  \end{lstlisting }
22
23 \subsection{Load Balancing by Reconfiguring}
24 The rule at Fig.~\ref{LoadBalancing} is mapped to VIATRA2 rule at
       List.~\ref{VTrule53}.
25 \begin{lstlisting}[label=VTrule53, caption=Reconfigure and
      transfer from overloaded]
26 gtrule Load_Balancing_by_reconfigeration() =
27     {
28             precondition pattern lhs(Y1,L,Y,X,BW,LKR,BW1) =
29             {
30
31               LK(L);
32               SN(Y);
33               SC(X);
34               SN(Y1);
35               find SN_Exit(Y,EX);
36               find SN_Exit(Y1,EX1);
```

```
37                find SN_Bandwidth(Y,BW);
38                find SN_Bandwidth(Y1,BW1);
39                find Link_SC_LK(X,LK);
40                find Existing_Link(Y,L,LKR);
41                check((toInteger(value(BW)))<800);
42                check((toInteger(value(BW)))>1500);
43                check((value(EX))=="False");
44                check((value(EX1))=="False");
45
46             }
47
48             action {
49
50                delete(LKR);
51                let OU=undef in seq{
52                new(LK.outTo(OU,LK,Y1));
53                setValue(BW1,((toInteger(value(BW)))-5));
54                setValue(BW,((toInteger(value(BW)))+5));
55                }
56                println("Move SC from overloaded to under loaded");
57
58             }
59    }
```

Listing B.54: Reconfigure and remove linking edge with SC

B.2.11 Rule for Collection of Statistics during Simulation

- Number of SCs

The rule in Fig. 6.73 is mapped to VIATRA2 rule in List. B.55.

```
1  gtrule scCount() =
2      {
3                precondition pattern lhs(X) = { SC(X); }
4                action {println("found node: "+fqn(X));}
5      }
```

Listing B.55: Probe rule to count the number of SCs

-Number of SNs

The rule in Fig. 6.74 is mapped to VIATRA2 rule in List. B.56.

```
1  gtrule snCount() =
2      {
3                precondition pattern lhs(X) = { SN(X); }
4                action {println("found node: "+fqn(X));}
5      }
```

Listing B.56: Probe rule to count number of SNs

- Number of SCs linked with SN

The rule in Fig. 6.75 is mapped to VIATRA2 rule in List. B.57.

```
gtrule Rule19_SC_connected_SN() =
    {
            precondition pattern lhs(X,L,Y) =
            {
              SN(Y);
              SC(X);
              LK(L);
              find LK_LinkTo(L,X);
              find LK_OutTo(L,Y);

            }

            action {
                 println("SC connected with SN");
            }
    }
```

Listing B.57: Probe rule to count the number of linked SCs

- Number of SCs Happy with current SN

The rule in Fig. 6.76 is mapped to VIATRA2 rule in List. B.58.

```
gtrule Rule17_SC_Happy_with_SN_Bandwidth() =
    {
            precondition pattern lhs(X,L,Y,BW) =
            {
              SN(Y);
              SC(X);
              LK(L);
              find LK_LinkTo(L,X);
              find LK_OutTo(L,Y);
                  find SN_Bandwidth(Y,BW);
                  check((toInteger(value(BW)))>1000);
            }

            action {
            println("SC is happy with existing SN");
            }

    }
```

Listing B.58: Probe rule to count the number of happy SCs

- Number of SCs re-linked with new SN

The rule in Fig. 6.77 is mapped to VIATRA2 rule in List. B.59.

```
gtrule Rule20_Reconfigure_Link_SC() =
    {
```

```
3               precondition pattern lhs(L,X) =
4               {
5
6                 LK(L);
7                 SC(X);
8                 find Link_SC_LK(X,L);
9                 neg find Test_SN_LK(L);
10
11                          }
12
13              action {
14
15                println("Re-linked Back to New Super Node");
16
17              }
18      }
```

Listing B.59: Probe rule to count the number reconfigured SCs

-Number of P2P Direct Calls SCs

The rule in Fig. 6.78 is mapped to VIATRA2 rule in List. B.60.

```
1  gtrule Prob_Rule_call_placed() =
2      {
3               precondition pattern lhs(C) =
4               {
5               Call(C);
6               find caller_checka(C);
7               find callee_checka(C);
8               }
9               action
10       {
11       println("Call placed");
12              }
```

Listing B.60: Probe rule to count the number P2P direct calls

-Number of SN Routed Calls

The rule in Fig. 6.79 is mapped to VIATRA2 rule in List. B.61.

```
1  gtrule Prob_Rule_Route_calls() =
2      {
3               precondition pattern lhs(R) =
4               {
5
6
7                 RouteCall(R);
8                 find Routed_caller_checka(R);
9                   find Routed_callee_checka(R);
10
11
```

```
12              }
13              action
14        {
15
16          println("Average Number of Routed Calls");
17              }
18      }
```

Listing B.61: Probe rule to count the number P2P routed calls

-Number of P2P Direct Calls between SCs through STUN

```
1
2  gtrule Place_call_and_Route_Local_WITH_STN() =
3      {
4              precondition pattern lhs(X,X1,SCBW,SCBW1,L,L1,NK,Q) =
5              {
6
7                SC(X);
8                SC(X1);
9                LK(L);
10               LK(L1);
11               Network(NK);
12               find LK_LinkTo(L,X);
13               find LK_LinkTo(L1,X1);
14               find LK_OutTo(L1,Y);
15               find SC_firewall(X,FIRW);
16               find SC_firewall(X1,FIRW1);
17               find SC_Bandwidth(X,SCBW);
18               find SC_Bandwidth(X1,SCBW1);
19               neg  find check_for_existing_call(X);
20                 neg  find check_for_existing_call(X1);
21                 CallReq(Q);
22               find check_requiest_for_call(X,Q);
23               neg find existing_req(X1);
24               check((value(FIRW))=="False");
25               check((value(FIRW1))=="True");
26             }
27
28             action {
29                 let PK=undef, PKS=undef,PKR=undef in seq{
30                 new (Call(PK) in NK);
31                 new (Call.caller(PKS,PK,X));
32                 new (Call.callee(PKR,PK,X1));
33                 setValue(SCBW,((toInteger(value(SCBW)))-80));
34                 setValue(SCBW1,((toInteger(value(SCBW1)))-80));
35                     println("Call Generated between SC and SC1
                           through STN");
36             delete(Q);
37             }
```

```
38      }
39
40      }
```

Listing B.62: Probe rule to count the number P2P routed calls

-Number of P2P Direct Calls between firewall /Nat enabled SC and SN

```
1
2
3   gtrule Place_call_SC_to_SN_Rev() =
4     {
5             precondition pattern lhs(L,Y,X,SCBW,SNBW1,NK,Q) =
6             {
7               SC(X);
8               SN(Y);
9               LK(L);
10              find SC_Source(L,X);
11              Network(NK);
12              find SC_Bandwidth(X,SCBW);
13              find SN_Bandwidth(Y,SNBW1);
14              find SC_firewall(X,FIRW);
15              neg find check_for_call(X);
16              neg find check_for_call_SN(Y);
17              find SC_firewall(X,FIRW);
18              CallReq(Q);
19              find check_requiest_for_call_SN(Y,Q);
20              neg find existing_req(X);
21              check((value(FIRW))=="True");
22              check((toInteger(value(SCBW)))>256);
23                check((toInteger(value(SNBW1)))>256);
24
25            }
26
27            action {
28                let PK=undef, PKS=undef,PKR=undef in seq{
29                new (Call(PK) in Network);
30                new (Call.caller(PKS,PK,Y));
31                new (Call.callee(PKR,PK,X));
32                setValue(SCBW,((toInteger(value(SCBW)))-80));
33                setValue(SNBW1,((toInteger(value(SNBW1)))-80));
34                    println("Call Generated between SN and SC1
                          through STN");
35                delete(Q);
36
37          }
38            }
39    }
```

Listing B.63: Probe rule to count the number P2P routed calls

References

[1] O. Abboud, A. Kovacevic, and Graffi. Underlay awareness in P2P systems: Techniques and challenges. *IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, 2009. 93, 96, 100

[2] D. Adami, C. Callegari, S. Giordano, M. Pagano, and T. Pepe. A real-time algorithm for Skype traffic detection and classification. In *Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces*, NEW2AN '09 and ruSMART '09, pages 168–179. Springer-Verlag, 2009. 79

[3] Akamai. Akamai. Website, accessed on 30^{th} December 2010, 2010. `http://www.akamai.com`. 45

[4] Gerard Nourry Alan Gib, Jean-Claud St-Jacues and Tim Johnson. A comparison of deterministic vs stochastic simulation models for assessing adaptive information management techniques over disadvantaged tactical communication networks. In *Proceeding 7th International Command and Control Research and Technology Symposium*, (7thICCRTS), 2002. 226

[5] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris.

Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35:131–145, October 2001. 45

[6] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45:56–61, November 2002. 1, 46, 50

[7] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36:335–371, December 2004. 39

[8] Epokh UIC audit. A network hooking application. Technical report, UIC audit, 2006. 217

[9] N. Ganguly1 B. Mitra, S. Gghosi and F. Peruni. Stability analysis of peer-to-peer networks against churn. *PRAMANA (Journal of physics)*, 2008. 66, 71, 96, 219, 220

[10] O. Balci B. Whitner. Guidelines for selecting and using simulation model verification techniques. In *Proceedings of the 21st conference on Winter simulation*, 1989. 199

[11] Lakhoo R. Baker, M. Peer-to-peer simulators. Technical Report 2007-1, ACET University of Reading, 2007. 229

[12] R. Bardohl, H. Ehrig, J. de Lara, O. Runge, G. Taentzer, and I. Weinhold. Node type inheritance concept for typed graph transformation. Technical Report 2003-19, Technical University of Berlin, 2003. 17, 19, 20

[13] R. Bardohl, H. Ehrig, J. de Lara, and G. Taentzer. Integrating meta-modelling aspects with graph transformation for efficient visual language definition and model manipulation. In M. Wermelinger and T. Margaria-Steffen, editors, *Proc. 7th Int. Conference on Fundamental Approaches to Software Engineering, FASE 2004*. 17, 19

[14] David Barkai. *Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net*. Intel Press, 2001. 39, 42

[15] S. A. Baset and H. G. Schulzrinne. An analysis of the Skype peer-to-peer internet telephony protocol. In *Proceedings 25th IEEE International Conference on Computer Communications*, pages 1 –11, April 2006. 25, 60, 64, 65, 66, 69, 71, 78, 79, 80, 81, 82, 83, 84, 96, 162, 168, 200, 201, 215, 216, 217, 218, 219, 221, 222

[16] Gábor Bergmann, András Ökrös, István Ráth, Dániel Varró, and Gergely Varró. Incremental pattern matching in the VIATRA model transformation system. In *Proceedings of the third international workshop on Graph and model transformations*, GRaMoT '08, pages 25–32, New York, NY, USA, 2008. ACM. 29, 141

[17] Dennis Bergström. An analysis of Skype VoIP application for use in a corporate environment. Technical Report 1.3-2004, CISSP, 2004. 215, 219, 221, 222

[18] Tom Berson. Skype cryptosystem overview, Anagram Laboratories. Website, Accessed on 10th December, 2010, 2010. http://www.iacr.org/conferences/asiacrypt2005/rump/Berson_AC05_Rump.pdf. 78, 79

[19] BitTorrent. Bittorrent. Website, Accessed on 30^{th} November, 2010. http://en.wikipedia.org/wiki/BitTorrent. 47

[20] J. E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Pract. Exper.*, 2:289–313, November 1990. 220

[21] BOINC. Berkeley open infrastructure for network computing. Website, accessed on 30th November, 2010. http://boinc.berkeley.edu/. 51

[22] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. Revealing Skype traffic: when randomness plays with you. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 37–48, New York, NY, USA, 2007. ACM. 79

[23] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems.* Kluwer, 2008. 26

[24] Hui Min Chong and H.S. Matthews. Comparative analysis of traditional telephone and voice-over-internet protocol (VoIP) systems. In *Electronics and the Environment, 2004. Conference Record. 2004 IEEE International Symposium on*, pages 106 – 111, 2004. 58, 59, 60

[25] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: a distributed anonymous information storage and retrieval system. In *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*, pages 46–66, New York, NY, USA, 2001. Springer-Verlag New York, Inc. 51, 231

[26] ILBC codec. Ilbc codec, accessed on 14th January, 2011. Website, 2011. http://www.vocal.com/data_sheets/ilbc.pdf. 79

[27] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003. 43, 223

[28] D.E. Comer. *Internetworking with TCP/IP*. Prentice Hall, Englewood Cliffs, 1995. 60, 61

[29] N. Krishnan D. Brookshier, D. Govoni and J. C. Soto. *JXTA: Java P2P Programming*. Sams Publishing, Indianapolis, 2002. 51

[30] M. Ruhul D. Karger. New algorithms for load balancing in peer-to-peer systems. In *Proceedings of the annual IRIS student workshop, Massachusetts Institute of Technology*, 2003. 65, 69, 221

[31] X. Chen D. Zhang and H. Yang. State of the art and challenges on peer-to-peer network simulators. In *proceedings of the 5th International Conference on Wireless Communication, Networking and Mobile Computing (WiCom 09), China*. 7, 224

[32] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with chord, a distributed lookup service. In *Proc. of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pages 71–76, 2001. 2

[33] Vasilios Darlagiannis. Overlay network mechanisms for peer-to-peer systems. Master's thesis, Technischen Universitat Darmstadt, 2005. 2, 3

[34] Kingshuk Dasgupta and Ananth Chandramouli. Peer-to-peer computing in the enterprise. Technical Report USA/0102/ID/MP/7.5K, Infosys Technologies, 2001. 43, 65

[35] Kevin Regan David Hadaller and Tyrel Russell. Necessity of supernodes survey. Technical Report 2005-1, Department of Computer Science, University of Toronto. 67, 217

[36] Definitions. Static IP address/Dynamic IP address. Website, 2011. http://searchwindevelopment.techtarget.com/definition/static-IP-address-dynamic-IP-address. 64

[37] DHTSim. Dhtsim, accessed on 14th January, 2011. Website, 2005. http://www.informatics.sussex.ac.uk/users/ianw/teach/dist-sys/. 224

[38] distributed.net. distributed.net, accessed on 17th January, 2011. Website, 2011. http://en.wikipedia.org/wiki/EDonkey_network. 47

[39] eDonkey. edonkey, accessed on 14th January, 2011. Website. http://en.wikipedia.org/wiki/EDonkey_network. 47

[40] Anthony Mosco Edwin Mier, David Mier. Assessing Skype network impact, published on Network World. December 2005. Website, 2005. http://www.networkworld.com/reviews/2005/121205-skype-test.html. 77

[41] H. Ehrig, M. Pfender, and H. J. Schneider. Graph grammars: an algebraic approach. In *14th Annual IEEE Symposium on Switching and Automata Theory*, pages 167–180. IEEE, 1973. 16

[42] Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer. Fundamental Theory for Typed Attributed Graph Transformation. In *Proceedings of International Conference Graph Transformation*, ICGT, pages 161–177, 2004. 21

[43] Dollimore J. Couloris G. and Kindberg T. *Distributed systems, concepts and design.* Addison Welsley, 2001. 38

[44] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9:681–692, December 2001. 177

[45] Lu Gao and Peng Min. Optimal super peer selection based on load balancing for P2P file-sharing system. In *Proceedings of the 2009 International Joint Conference on Artificial Intelligence*, JCAI '09, pages 92–95, Washington, DC, USA, 2009. IEEE Computer Society. 221

[46] S. L GARFINKEL. VoIP and Skype security, accessed on 14th January, 2011. Website, 2005. http://www1.cs.columbia.edu/~salman/skype/OSI_Skype6.pdf. 76, 78, 79

[47] Thomas J. Giuli and Mary Baker. Narses: A scalable flow-based network simulator. *Computing Research Repository (CoRR)*, cs.PF/0211024, 2002. 224, 230

[48] Gnutella. Gnutella, accessed on 14th January, 2011. Website, 2005. http://en.wikipedia.org/wiki/Gnutella. 1, 51, 231

[49] B. Goode. Voice over Internet protocol (VoIP). *Proceedings of the IEEE, Volume 90, Pages 1495 - 1517*, 2002. 60

[50] P. GRADWELL. *VoIP Discussion at the BCS Specialist Internet Group*. British Computer Society: London. http://www.bcs.org/content/ConWebDoc/3556. 57

[51] Grokster. Grokster ltd, accessed on 14th January, 2011. Website, 2011. http://www.grokster.com. 49

[52] GROOVE. Graphs for object-oriented verification. Website, accessed on 13th May, 2011, 2011. http://user.cs.tu-berlin.de/~sjurack/rootshttp://groove.sourceforge.net/groove-index.html. 236

[53] Saikat Guha, Neil Daswani, and Ravi Jain. An experimental study of the Skype peer-to-peer voip system. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, (IPTPS'06), Santa Barbara, USA. 3, 65, 66, 69, 71, 78, 79, 80, 81, 82, 85, 96, 162, 168, 188, 200, 215, 216, 217, 218, 219, 221, 222

[54] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. *SIGOPS Oper. Syst. Rev.*, 37:314–329, October 2003. 1

[55] Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, and Robbert van Renesse. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of 2nd International Workshop on Peer-to-Peer Systems*, 2003. 229

[56] U. Parange G. Taentzer H. Ehrig, K. Ehrig. *Fundamentals of Algebric Graph Transformation*. Springer, 2005. 16, 26

[57] A. Habel, R. Heckel, and G. Taentzer. Graph grammars with negative application conditions. *Fundamenta Informaticae*, 26(3-4):287–313, 1996. 25

[58] K.; Chujo T.; Yang X. Hamada, T.; Chujo. Peer-to-peer traffic in metro networks: analysis, modeling, and policies. In *Network Operations and Management Symposium (NOMS) 2004*. 1

[59] Bert Hayes. Skype: A practical security analysis, published at INFOSEC reading room. Website, 2008. http://www.sans.org/reading_room/whitepapers/voip/skype-practical-security-analysis_32918. 77, 78

[60] Qinxia (Alice) He. Analysing the characteristics of VoIP traffic. Master's thesis, College of Graduate Studies and Research,University of Saskatchewan, 2007. 60

[61] R. Heckel. Stochastic analysis of graph transformation systems: A case study in P2P networks. In *Proc. Intl. Colloquium on Theoretical Aspects of Computing*, (ICTAC05). Springer-Verlag, 2005. 6, 69, 221, 222

[62] R. Heckel, J.M. Küster, and G. Taentzer. Confluence of typed attributed graph transformation systems. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. 1st Int. Conference on Graph Transformation, ICGT 02*, volume 2505 of *LNCS*, pages 161–176. Springer, 2002. 21

[63] Reiko Heckel and Paolo Torrini. Stochastic modelling and simulation of mobile systems. In Gregor Engels, Claus Lewerentz, Wilhelm Schäfer,

Andy Schürr, and Bernhard Westfechtel, editors, *Graph Transformations and Model-Driven Engineering*, volume 5765 of *Lecture Notes in Computer Science*, pages 87–101. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-17322-65. 30

[64] S. H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *J. Parallel Distrib. Comput.*, 10:160–166, September 1990. 220

[65] Carl Kesselman Ian Foster. *The Grid 2: Blueprint for a New Computing Infrastructure*. The Elsevier Series in Grid Computing. 2002. 3

[66] Global Index. Global index (gi). Website, 2003. http://www.skype.com/skype_p2pexplained.html. 83

[67] Global IP and Sound. Global IP and Sound Ltd, accessed on 14th January, 2011. Website, 2011. http://www.packetizer.com/codecs/ilbc/iLBC.WP.pdf. 79

[68] iSAC codec. iSAC codec, accessed on 14th January, 2011. Website, 2011. http://www.vocal.com/data_sheets/ilbc.pdf. 79

[69] ITU-T. *Recommendations G.114*. One way transmission time, ITU, 1996. 59

[70] C. Huitema J. Rosenberg, J. Weinberger and R. Mahy. Stun: Simple traversal of user datagram protocol (udp) through network address translators (nats). Website, 2003. RFC 3489, IETF“. 82

[71] J.H. James, Bing Chen, and L. Garrison. Implementing VoIP:A voice transmission performance progress report. *Communications Magazine, IEEE*, 42(7):36 – 41, July 2004. 59

[72] Márk Jelasity and Alberto Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, ICDCS '04, pages 102–109, Washington, DC, USA, 2004. IEEE Computer Society. 228

[73] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. T-man: Gossip-based fast overlay topology construction. *Comput. Netw.*, 53:2321–2339, August 2009. 228

[74] Li. Chun Ji. Computation in peer-to-peer networks. Technical Report SA2004-1, Department of Computer Science, University of Saskatchewan, Canada. 72

[75] Wenyu Jiang, K. Koguchi, and H. Schulzrinne. QoS evaluation of VoIP endpoints. In *Communications, 2003. ICC '03. IEEE International Conference on*, pages 1917 – 1921 vol.3, May 2003. 59

[76] Sam Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Revised Papers from the NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, pages 202–214, London, UK, UK, 2002. Springer-Verlag. 224, 231

[77] Sam Joseph. An extendible open source P2P simulator. *P2PJournal*, 2003. 223, 224, 231

[78] H.Hakonen J.Smed, T.Kaukoranta. Networking and multiplayer computer games-the story so far. *International Journal of Intellegent Games and Simulation*, 2003. 3

[79] Stefan Jurack. Roots. Website, accessed on 13th May, 2011, 2011. http://user.cs.tu-berlin.de/~sjurack/roots. 236

[80] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, (WWW '03), pages 640–651, New York, NY, USA, 2003. ACM. 231

[81] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report. 63

[82] David R. Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pages 36–43, New York, NY, USA, 2004. ACM. 220

[83] KaZaA. Kazaa, accessed on 14th January, 2011. Website, 2011. http://www.KaZaA.com. 42, 49, 51, 76

[84] Ram Keralapura, Nina Taft, Chen-Nee Chuah, and Gianluca Iannaccone. Can ISPs take the heat from overlay networks. In *Proceedings Hot Topics in Networks*, HotNets, 2004. 46

[85] Ajab Khan and Reiko Heckel. Model-based stochastic simulation of super peer promotion in P2P VoIP using graph transformation. In *Proceedings*

of the International Conference on Data Communication, Seville, Spain, DCNET2011, 2011. 6, 67, 108

[86] Ajab Khan, Reiko Heckel, Paolo Torrini, and István Ráth. Model-based stochastic simulation of P2P VoIP using graph transformation system. In *Proceedings of the 17th International Conference on Analytical and Stochastic Modeling Techniques and Applications*, (ASMTA'10, Walls, UK), year = 2010, pages = 204-217,. 6

[87] Ajab Khan, Paolo Torrini, and Reiko Heckel. Model-based simulation of VoIP network reconfigurations using graph transformation systems. *Post Proceedings of ICGT-DS ECEASST*, 16, 2009. 3, 6, 30, 141, 165

[88] Wookyun Kho, S.A. Baset, and H. Schulzrinne. Skype relay calls: Measurements and experiments. In *Proceedings The 27th Conference on Computer Communications INFOCOM Workshops 2008, IEEE*, pages 1 –6, April 2008. 218

[89] A.A. Kist and R.J. Harris. Cost efficient overflow routing for outbound isp traffic. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, pages 876 – 882 Vol.2, June-1 July 2004. 218

[90] Samuel Korpi. Internet telephony - security issues in Skype, Helsinki University of Technology. Website, 2006. http://www.tml.tkk.fi/Publications/C/21/Korpi_ready.pdf. 77

[91] P. Kosiuczenko and G. Lajios. Simulation of generalised semi-Markov pro-

cesses based on graph transformation systems. *Electronic Notes in Theoretical Computer Science*, 175:73–86, 2007. 26, 30

[92] Martin A. Landers. An overview of peer-to-peer network topologies. Technical Report 1/2004, Technical University of Munich. 47, 48, 49

[93] Pierre L'Ecuyer, Lakhdar Meliani, and Jean Vaucher. SSJ: a framework for stochastic simulation in Java. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, WSC '02, pages 234–242. Winter Simulation Conference, 2002. 141

[94] Bo Arne Leuf. *Peer to Peer: Collaboration and Sharing over the Internet*. Addison Welsley, 2002. 38, 49

[95] Jinyang Li, Jeremy Stribling, Robert Morris, and M. Frans Kaashoek. Bandwidth-efficient management of DHT routing tables. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 99–114, Berkeley, CA, USA, 2005. USENIX Association. 229

[96] Jian Liang, Rakesh Kumar, and Keith W. Ross. The KaZaA overlay: A measurement study. *Computer Networks Journal (Elsevier)*, 2005. 1, 76, 93

[97] Skype limited. *Skype: Guide for Network Administrators*. Skype, 2006",. 7, 69, 162

[98] Gao Lisha and Luo Junzhou. Performance analysis of a P2P-based VoIP software. In *Proceedings of the Advanced Int'l Conference on Telecom-*

munications and Int'l Conference on Internet and Web Applications and Services, AICT-ICIW '06, pages 11–, Washington, DC, USA, 2006. IEEE Computer Society. 215, 216, 217

[99] V. Lo, Dayi Zhou, Yuhong Liu, C. GauthierDickey, and Jun Li. Scalable supernode selection in peer-to-peer overlay networks. In *Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on*, pages 18 – 25, july 2005. 65, 66, 67, 70, 102, 217, 218, 219, 220

[100] Skype Ltd. Cnet,most downloads, accessed on 10th January, 2011. Website, 2011. http://www.download.com. 76

[101] Skype Ltd. Skype company, accessed on 10th January, 2011. Website, 2011. http://www.skype.com/company. 76

[102] K. Shen M. Zhong and J. Seiferas. Dynamic load balancing in unstructured peer-to-peer networks: Finding hotspots, eliminating them. unpublished manuscript, http://www.cs.rochester.edu/u/zhong/papershotspots.pdf. 220, 221

[103] Priyanka Mahalanabis Matthew Benford, Michael Boyce and Steve Webb. Peer-to-peer instant messaging (P2P IM). http://www.cc.gatech.edu/~webb/P2P_IM.htm. 52

[104] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, 2002. Springer-Verlag. 229

[105] Merriam-Webster. Online dictionary, accessed on 14th January, 2011. Website, 2011. http://www.m-w.com. 38

[106] Microsoft. Introduction to windows peer-to-peer networking. Technical report, Microsoft, Tech, 2003. http://technet.microsoft.com/en-us/library/bb457079.aspx. 40

[107] D. S. Milojicic. Peer-to-peer computing. Technical Report HPL-2002-57R1, HP Ltd. 1, 38, 39, 46, 47, 48

[108] Su-Hong Min, Joanne Holliday, and Dong-Sub Cho. Optimal super-peer selection for large-scale P2P system. In *Proceedings of the 2006 International Conference on Hybrid Information Technology - Volume 02*, ICHIT '06, pages 588–593, Washington, DC, USA, 2006. IEEE Computer Society. 217, 219, 220

[109] Sándor Molnár and Marcell Perényi. On the identification and analysis of skype traffic. *Int. J. Commun. Syst.*, 24:94–117, January 2011. 81, 82, 83

[110] Alberto Montresor. Robust aggregation protocols for large-scale overlay networks. In *In Proceedings of the 2004 International Conference on Dependable Systems and Networks*, (DSNâ04). 224, 228

[111] Alberto Montresor. A robust protocol for building superpeer overlay topologies. In *Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, P2P '04, pages 202–209, Washington, DC, USA, 2004. IEEE Computer Society. 2, 3, 65, 66, 67, 70, 188, 217, 222, 223, 228

[112] Caro G. D. Heegaard P. E Montresor, A. Bison project deliverable number D11, accessed on 10th January, 2011, 2011. http://www.cs.unibo.it/bison/deliverables/D11.pdf. 228

[113] S. Naicken, A. Basu, B. Livingston, S. Rodhetbhai, and I. Wakeman. Towards yet another peer-to-peer simulator. In *Proceedings of The Fourth International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs), Ilkley, UK*, 2006. 6, 7, 223, 224, 225

[114] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.*, 37:95–98, March 2007. 7, 224

[115] Napster. Napster, accessed on 14th January, 2011. Website, 2011. http://http://www.napster.com. 42, 46, 51

[116] Rafael R. Obelheiro and Joni da Silva Fraga. Overlay network topology reconfiguration in byzantine settings. In *PRDC '07: Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing*, volume 0, pages 155–162, Washington, DC, USA, 2007. IEEE Computer Society. 222

[117] University of Southern California. The network simulator-NS2. Website, 2003. http://www.isi.edu/nsnam/ns/ns-topogen.html. 233

[118] Dj Oneil, Hun Jeong, Kang Jinoh, and Kim Donghyong Kwon. Transport layer identification of P2P super nodes. accessed on 13th January,2011,http://www-users.cs.umn.edu/~jinohkim/docs/supernode.pdf,. 65, 66, 102, 188

[119] Andy Oram. *Peer-to-peer: Harnessing the power of disruptive technologies*. Reilly, 2001. 39, 40

[120] Andy Oram. From p2p to web services: Addressing and coordination. Technical report, 2004. http://www.xml.com/pub/a/2004/04/07/p2p-ws.html. 40

[121] F. Desclaux P. Biondi. Silver needle in the Skype. Technical Report DCR/STI/C2006, EADS Corporate Research Centre, Suresnes, France. 65, 66, 79, 188, 216, 219, 220

[122] R. Paul. More universities banning Skype. Website, 2006. accessed on 10th January,2011http://arstechnica.com/news.ars/post/20060924-7814.html. 177

[123] Pedro, Carles Pairot, Ruben Mondejar, Jordi Pujol, Helio Tejedor, and Robert Rallo. Planetsim: A new overlay network simulation framework. *Software Engineering and Middleware*, 2005. 224

[124] Colin Perkins, Orion Hodson, and Vicky Hardman. A survey of packet-loss recovery techniques for streaming audio. *IEEE Network*, 12:40–48, 1998. 60

[125] Planetlab. Planetlab, accessed on 14th May, 2011. Website, 2011. http://www.planetlab.org. 223

[126] Thomas Porter. *The perils of deep packet inspection*. SecurityFocus, 2005. 58

[127] B. Aditya Prakash, Hanghang Tong, Nicholas Valler, Michalis Faloutsos, and Christos Faloutsos. Virus propagation on time-varying networks: theory and immunization algorithms. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, ECML PKDD'10, pages 99–114, Berlin, Heidelberg, 2010. Springer-Verlag. 51

[128] Sanjay Udani Princy Mehta. *Voice over IP sound good on the Internet.* IEEE Potentials Magazine, 2001. 59

[129] Dongyu Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, pages 367–378, New York, NY, USA, 2004. ACM. 223

[130] Aaron Harwood Rajiv Ranjan and Rajkumar Buyya. Coordinated load management in peer-to-peer coupled federated grid systems. *Journal of Supercomputing, Special Issue on Advances in Network and Parallel Computing*, 2010. 221

[131] Bharat Rao. Skype: Leading the VOIP revolution. Website, 2004. accessed on 17th May, 2011http://www.cashflowec.com/files1/skype.case.pdf. 76, 77, 78

[132] Arend Rensink. Representing first-order logic using graphs. In *International Conference on Graph Transformations (ICGT), volume 3256 of Lecture Notes in Computer Science*, pages 319–335. Springer, 2004. 151

[133] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association. 45

[134] Dario Rossi, Marco Mellia, and Michela Meo. Understanding Skype signaling. *Comput. Netw.*, 53:130–140, February 2009. 71, 79, 162

[135] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, pages 329–350, London, UK, 2001. Springer-Verlag. 2

[136] V.S. Dixit Rupali Bhardwaj, Anil Kr. Upadhyay. An overview on tools for peer to peer network simulation. *Int'l Journal of Computer Applications*, 2010. 29, 224, 229, 230, 231, 232, 234, 235

[137] M. Ilyas S. A.Ahson. *VoIP Handbook*. CRC Press, 2009. 57, 79

[138] B. Livingston S. Naiken, A. Basu and S. Rodhetbhai. A survey of peer-to-peer network simulators. In *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool,UK*. 7, 224, 228

[139] R. G. Sargent. Verifying and validating simulation models. In *Proceedings of the 28th conference on Winter simulation, IEEE Computer Society, Washington*, 1996. 199

[140] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and

analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Syst.*, 9:170–184, August 2003. 223

[141] Mario Schlosser and Sepandar Kamvar. Simulating a file-sharing p2p network. Technical Report SU-2002, Stanford University, 2002. 231

[142] Detlef Schoder and Kai Fischbach. Peer-to-peer prospects. *Commun. ACM*, 46:27–29, February 2003. 41

[143] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *Peer-to-Peer Computing, IEEE International Conference on*, 0:0101, 2001. 43

[144] Srinivasan Seetharaman and Mostafa Ammar. Characterizing and mitigating inter-domain policy violations in overlay routes. In *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 259–268, Washington, DC, USA, 2006. IEEE Computer Society. 177

[145] Clay Shirky. What is p2p... and what isn't. Technical Report 2001, OReilly Network. 40

[146] Kazuyuki Shudo, Yoshio Tanaka, and Satoshi Sekiguchi. Overlay weaver: An overlay construction toolkit. *Comput. Commun.*, 31:402–412, February 2008. 224

[147] PeerSim P2P Simulator. Peersim p2p simulator. Website, accessed on 13th May, 2011, 2011. http://peersim.sourceforge.net. 228

[148] Atul Singh. *Self- Organizing Topology Adaptation in Peer-to-Peer Networks.* PhD thesis, Trinity College, University of Dublin, 2007. 40, 41

[149] Skype. Skype Limited, accessed on 14th January, 2011. Website, 2005. http://www.skype.com. 1, 43, 45, 52, 61

[150] Skype Statitics. Aaytch llc,accessed on 17th january,2011. Website, 2011. http://aaytch.com/pages/borderless. 84, 162, 168, 200, 201

[151] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications,Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 149–160, New York, NY, USA, 2001. ACM. 229

[152] S. Street. *IP Telephony: The End of the World as We Know It?* CAUSE/-EFFECT, 1999. 58

[153] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM. 38, 65, 71

[154] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting Skype-Relayed traffic. 79, 168, 218

[155] P2P Swarming. P2P swarming protocol simulation. Website accessed on 15th May, 2011, 2011. 234

[156] MathWave Technologies. Mathwave: Data analysis simulation. Website accessed on 12th May, 2011, 2011. http://www.mathwave.com. 27

[157] AnyFirewall Technology. Nat traversal for voip and internet communications using stun, turn and ice. Website, 2011. ://www.voiptraversal.com/. 82

[158] Sebastian Thöne. *A Style-Based Modeling and Refinement Technique with Graph Transformations*. PhD thesis, Faculty of Computer Science, Electrical Engineering, and Mathematics, University of Paderborn, 2005. 21, 22

[159] N. S. Ting. A generic peer-to-peer network simulators. In *Proceedings of the 2003-2004 Grad Symposium Computer Science, Department, University of Saskatchewan. April 7-8,2003.* 7, 224

[160] Nyik San Ting and Ralph Deters. 3LS:a peer-to-peer network simulator. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, P2P '03, pages 212–, Washington, DC, USA, 2003. IEEE Computer Society. 224, 230

[161] Nyik San Ting and Ralph Deters. A generic peer-to-peer network simulator. Technical report, Department of Computer Science, University of Saskatchewan, 2003. 4, 65

[162] Paolo Torrini, Reiko Heckel, and István Ráth. Stochastic Simulation of Graph Transformation Systems. In David Rosenblum and Gabriele Taentzer, editors, *Fundamental Approaches to Software Engineering*, volume 6013 of *Lecture Notes in Computer Science*, chapter 11, pages 154–157.

Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010. 29, 141, 142, 147, 165

[163] UDP. *User Datagram Protocol*. Accessed On 1st December,2010 http://en.wikipedia.org/wiki/User_Datagram_Protocol, 2010. 60, 61

[164] Dániel Varró. *Automated Model Transformations for the Analysis of IT Systems*. PhD thesis, Budapest University of Technology and Economics, Department of Measurement and Information Systems, May 2004. 149

[165] SNOW A. McGivern M. HOWARD C. VARSHNEY, U. *Voice over IP*, volume 45. Communications of the ACM, 2002. 59

[166] H. Wang. Skype voip service- architecture and comparison. 65, 83, 84

[167] James Z. Wang and Matti Vanninen. Self-configuration protocols for P2P networks. *Web Intelli. and Agent Sys.*, 4:61–76, January 2006. 222

[168] Shuling Wang, Shoubao Yang, Kai Shen, and Lihua Xie. A super node selecting mechanism based on AHP. In *Grid and Cooperative Computing, 2008. GCC '08. Seventh International Conference on*, pages 403 –406, oct. 2008. 65, 66, 67, 217, 219, 220

[169] Wikipedia. History of the internet. Website, accessed on 30th September, 2011, 2005. http://en.wikipedia.org/wiki/History_of_the_Internet. 1

[170] Steffen Wolf. *Optimization Problems in Self-Organizing Networks*. Logos Verlaog Berlin GmbH, 2010. 70

[171] Phil Wolff. 1.4 Million Skype supernodes crashed , http://http://skypejournal.com/blog/?s=skype+crashed, accessed on 10th october, 2011. 2010. 85, 200, 201

[172] H. Xie and Y. R. A measurement-based study of the Skype peer-to-peer VoIP performance. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems, Bellevue*, IPTPS, 2007. 217, 218

[173] Y. R. Hayyong Xie. A measurment-based study of the skype peer-to-pee voip performance. Technical report, Yale University, 2007. 3, 96, 162, 177

[174] Z. Xu and Y. Hu. SBARC: A supernode based peer-to-peer file sharing system. In *8th IEEE International Symposium on Computers and Communication (ISCC 2003)*, Kemer - Antalaya, Turkey, June 2003. 219, 220

[175] Zhiyong Xu and Yiming Hu. Sbarc: A supernode based peer-to-peer file sharing system. In *In Eighth IEEE International Symposium on Computers and Communications, Kemer-Antalya*, 2003. 67, 217

[176] Beverly Yang and Hector Garcia-m. Designing a super-peer network. In *Proceedings of the 19th Int. Conf. on Data Engineering (ICDE), Bangalore, India*, 2003. 65, 66, 102, 188, 216, 222

[177] W. Yang and N. Abu-Ghazaleh. GPS: a general peer-to-peer simulator and its use for modeling BitTorrent. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, pages 425 – 432, Sept. 2005. 224, 232

[178] Yanfeng Yu, Dadi Liu, Jian Li, and Changxiang Shen. Traffic identification and overlay measurement of Skype. In *Computational Intelligence and Security, 2006 International Conference on*, volume 2, pages 1043 –1048, November 2006. 80, 81, 82, 83

[179] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation.*, volume 2 of *INFOCOM '96*, pages 594 –602 vol.2, March 1996. 232

[180] Da-lu Zhang and Chen Lin. Efficient delay aware peer-to-peer overlay network. In Wenfei Fan, Zhaohui Wu, and Jun Yang, editors, *Advances in Web-Age Information Management*, volume 3739 of *Lecture Notes in Computer Science*, pages 682–687. Springer Berlin / Heidelberg, 2005. 10.1007/1156395264. 66, 219, 220

[181] Dengyi Zhang, Xuhui Chen, and Hongyun Yang. State of the art and challenges on peer-to-peer simulators. In *Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing*, WiCOM'09, Beijing, China. 2009. 224

[182] Dongyan Zhang, Chao Zheng, Hongli Zhang, and Hongliang Yu. Identification and analysis of skype peer-to-peer traffic. In *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, ICIW '10, pages 200–206, Washington, DC, USA, 2010. IEEE Computer Society. 81, 162

[183] F.P. Zhang, O.W.W. Yang, and B. Cheng. Performance evaluation of jit-

ter management algorithms. In *Proceedings of the Canadian Conference Electrical and Computer Engineering, 2001*, Pages 1011 -1016, Vol.2. 2001. 59

[184] Yan Zhang, Wei Wang, and Shunying Lü. Simulating trust overlay in P2P networks. In *Proceedings of the 7th international conference on Computational Science, Part I: ICCS*, ICCS '07, Beijing, China. 2007. 224

[185] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Anthony D. Joseph, and John D. Kubiatowicz. Exploiting routing redundancy via structured peer-to-peer overlays. In *11th IEEE International Conference on Network Protocols*, (ICNP'03, Atlanta, Georgia, USA. 2003. 45

[186] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location. Technical Report UCB/CSD-01-1141, University of California at Berkeley, 2001. 2, 229

[187] Xuan Zhou and Wolfgang Nejdl. Priority based load balancing in a self-interested P2P network. In *Proceedings of the 2005/2006 International Conference on Databases, Information Systems, and peer-to-peer Computing*, DBISP2P'05/06, Trondheim, Norway. 2005. 65, 69, 220