A Lightweight Technique for Extracting Software Architecture and Performance Models from Traces.

by

Tauseef A. Israr

A thesis submitted to the Faculty of Graduate Studies
in partial fulfillment of the requirements
for the degree of

Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6

April 12, 2001

The undersigned recommend to the Faculty of Graduate Studies and Research acceptance of the thesis

A Lightweight Technique for Extracting Software Architecture and Performance Models from Traces.

Submitted by Tauseef A. Israr
in partial fulfillment of the requirements
for the degree of Masters of Engineering

Chair, Department of Systems and Computer Engineering

Thesis Supervisor

Carleton University

April 12, 2001

Abstract

Performance models of software designs can give early warnings of problems such as resource saturation or excessive delays. However models are seldom use d because of the considerable effort needed to construct them. Software Architecture and Model Extraction (SAME) is a lightweight model building technique that extracts communication patterns from executable designs to develop a Layered Queuing Network model in an automated fashion. It is a formal, traceable model building process. The transformation follows a series of well -defined transformation steps, from input domain, (an executable software design or the implementation of software itself) to output domain, a Layered Queuing Network (LQN) Performance model. The SAME technique is appropriate for a message passing distributed system where tasks interact by point -to-point communication. With SAME, the performance analyst can focus on the principles of software performance analysis rather than model building.

Acknowledgement

At last, I have done it. I must say that the last few years have been sensational.

Firstly, I would like to thank Allah, the Almighty for everything. I am not sure what part of me He likes for He has given me more than what I deserve. I am certain that this is all due to the prayers of my grand parents.

To my parents who never ever doubted my abilities, have always given me hope and all the love in the world and made me what I am today, I will always love you. To my brothers, who are always there for me whenever I need them most, esp. Junaid bhai whose love, support and guidance is exemplary.

To my beloved wife, Arooj “Oojee”, thank you for making me the luckiest person in the world. I will always love you and your everlasting support and love in last few years is second to none.

To my mentor, Prof. Murray Woodside, I remembered stepping into your office a few years back, a little shaken, scared and lost. At that time I needed hope and a lot of guidance. I must say, if it hasn’t been for you, I would not have been writing this today. You believed in me and gave me a chance, when even I doubted myself. You are more than just a good teacher to me; you are the one who helped me realize my destiny.

And to my dearest friend Khalid, you are the finest example of friendship. Thank you for everything.

Imran Bhai and Naheed Baji, you suggested me to do the masters in software engineering and after two years, here I am saying, THANK YOU.

Table of Contents

List of Figures

List of Tables

Table of Notations

Designation	**Definition**
R(H)	Root of tree H.
Proc(A)	Process ***a***, whose activation is represented by vertex A.
A > B	Vertex A is predecessor of vertex B. Vertex A can be of higher level than P(B), as long as there is a direct path from A to B.
B < A	Vertex B is a descendant of vertex A.
A <> B	Vertex A is either a sibling of B, or A is neither a predecessor nor a descendant of vertex B, and vice versa.
P(X)	Parent of vertex X.
C{X}	A set of children vertices of Vertex X.
L{H}	A set of leaf nodes for tree H.
PA{A}	Set of previous activations of process ***a***. Note that all activations of process ***a***, have the same label, A.
AA(A)	An active activation for process ***a***. There is only one active activation allowed in this study.
Inert{H}	A set of inert vertices for tree H.
A	An inert vertex for process ***a***.
NIC{A}	A set of Non-Immediate Candidate vertices of proc(A). An activation of a process, which may no longer be a replying server to client proc(A) in a synchronous message.
IC(A)	An Immediate Candidate vertex. An activation of a process, which may be a replying server to client proc(A) in a synchronous message.

Chapter 1: Introduction

1.1 Need for Performance Models

The development of software systems around the globe is plagued by tight deadlines and aggressive schedules. In performance critical systems, it is desirable to predict performance problems well in advance, when they are easy to fix and dealt with. The traditional methodology of software development has a great impact on such systems as it has relegated the performance aspects of design towards the end of the software cycle, after the system is operational. Too often the development decisions are made relatively blind to their performance impact. As the project matures, performance problems sometime snowball and result in degenerating of development into an “interactive fire-drill” between waiting customers, systems engineers and development groups [Strosnider96]. Early evaluation of performance problems points to some kind of a performance model of a system, since software does not yet exists.

Performance models provide data for the quantitative assessment of the performance characteristics of software systems as they are developed. Performance modeling of early design can reduce the risk of performance-related failures by giving early warnings of problems. Performance models provide performance predictions under varying environmental conditions or design alternatives and these conditions can be used to detect problems. Performance models contains the specification of activities together with their cost in terms of execution times and frequency of their execution for system’s responses. Solving the performance models yields the total execution time for each system’s responses together with the utilization of the processors in the system. If the design does not meet requirements it is changed, and the performance model is regenerated and resolved. This cycle continues until the requirements are met. Through creating performance models of a system, designers build performance into the system rather than (try to) add it later after the system is developed.

1.2 Obstacles to Developing a Performance Model

Performance models have many uses; however creating a correct model with important specifications is not everyone's cup of tea. Analysts while creating a model have to review software descriptions, such as CASE models, design documents, source code design description or implementation source code. Important end-to-end system behavior, the involved software components, device usage by each component, the interaction between components are identified and this information is converted into an appropriate model format such as a simulation model, a queuing network model, or a petri-net model. This large quantity of information becomes unwieldy even for a moderate sized system. Introduction of object oriented (OO) methodology has even more complicated the process by polymorphism, inheritance and class structures. Consequently models become expensive to develop and validate. Clearly some sort of automation is needed to develop better and accurate models efficiently and easily.

1.3 Background

Curtis Hrischuk proposed the technique called Trace-based Load Characterization (TLC) [Hrischuk99]. This technique is based on traces thus the name "Trace-based". Traces have low cost because they can be automatically generated from instrumentation. Traces reveal the dynamic details of design, which are sometimes difficult to determine through design documentation or code inspection but are valuable to performance analysis. The dynamic details include tasks (or processes) involved in dynamically bound interactions, data dependant branching and the involvement of polymorphism and inheritance hierarchy of object oriented (OO) systems.

Although TLC does automate the process of developing models through traces, it has its own shortcomings. The technique is based on matching templates to event graphs to recognize event patterns. This is not an efficient way of detecting and creating models due to processing involved and the templates are limited to the imagination of the author. TLC uses a special kind of a trace called angio-trace [Hrischuk95]. It is based on the idea of a dye to capture forks, joins and multiple concurrent threads and thus it requires more information than a simple event trace.

El-Sayed in [El-Sayed99] invented a technique he called Model Builder to develop performance models through traces out of SDL design tool. These traces have very limited information regarding the message passed in component interactions, e.g. the information regarding the sender of a message is not clearly defined and dyes were missing, which were reconstructed. Some assumptions are made to simplify the problem. For example, it is assumed that the queues are FIFO with no priorities and thus this limits the ability of this technique to create models beyond this criterion.

Both of these previous techniques have drawbacks. TLC is specifically based on the concept of angio-trace, which is not widely understood and Model Builder in El-Sayed's research is based on assumptions which significantly narrows the type of systems analyzed.

The goal of this research is to develop a technique that uses generic traces. It should cover almost all types of systems without narrowing the type of systems to be analyzed as in [ElSyed99] and also at the same time does not rely just on angio-traces to create performance models. Thus it can be used for analyzing systems not only in early design stage, but also systems already in existence.

1.4 Introduction to SAME

The technique presented in this research is called Systems Architecture and Model Extraction (SAME) technique. SAME can be applied anywhere in software development cycle, to any abstraction. It is based on execution traces, which can be obtained from a variety of sources, as early as an executable form of any CASE tool such as ObjecTime Developer Toolset to an actual execution log of a system.

SAME is flexible enough to handle both non-distributed systems and as well as loosely coupled distributed systems. A distributed system is composed of geographically dispersed, heterogeneous hardware with scheduled, concurrent software objects. The components communicate solely by messages each of which are recorded in a trace in chronological order. The execution of a distributed operation recorded in a trace will be called a scenario. SAME assumes that a finite set of scenarios can capture all the important behavioral aspect of the software system.

1.5 The model building strategy of SAME

The goal of this study is to develop automation to build models. In order to do so, several guiding principles have to be followed, as listed below:

1. The actual design implementation in CASE tool or the operational software are the sources of information to ensure that the input data is reliable.
2. Each transformation step in the model building process must be deterministic and preserve the necessary properties of the input data.
3. To avoid manual intervention transformation process must be sufficiently formal.

SAME is a formal, traceable model building process. The transformation follows a series of well -defined transformation steps, from input domain, (in this case an executable software design or the implementation of software itself) to output domain, a Layered Queuing Network (LQN) Performance model. Figure 1.1 il lustrates the model building process with formal transformations. SAME consists of four steps, which are illustrated in Figure 1.1:

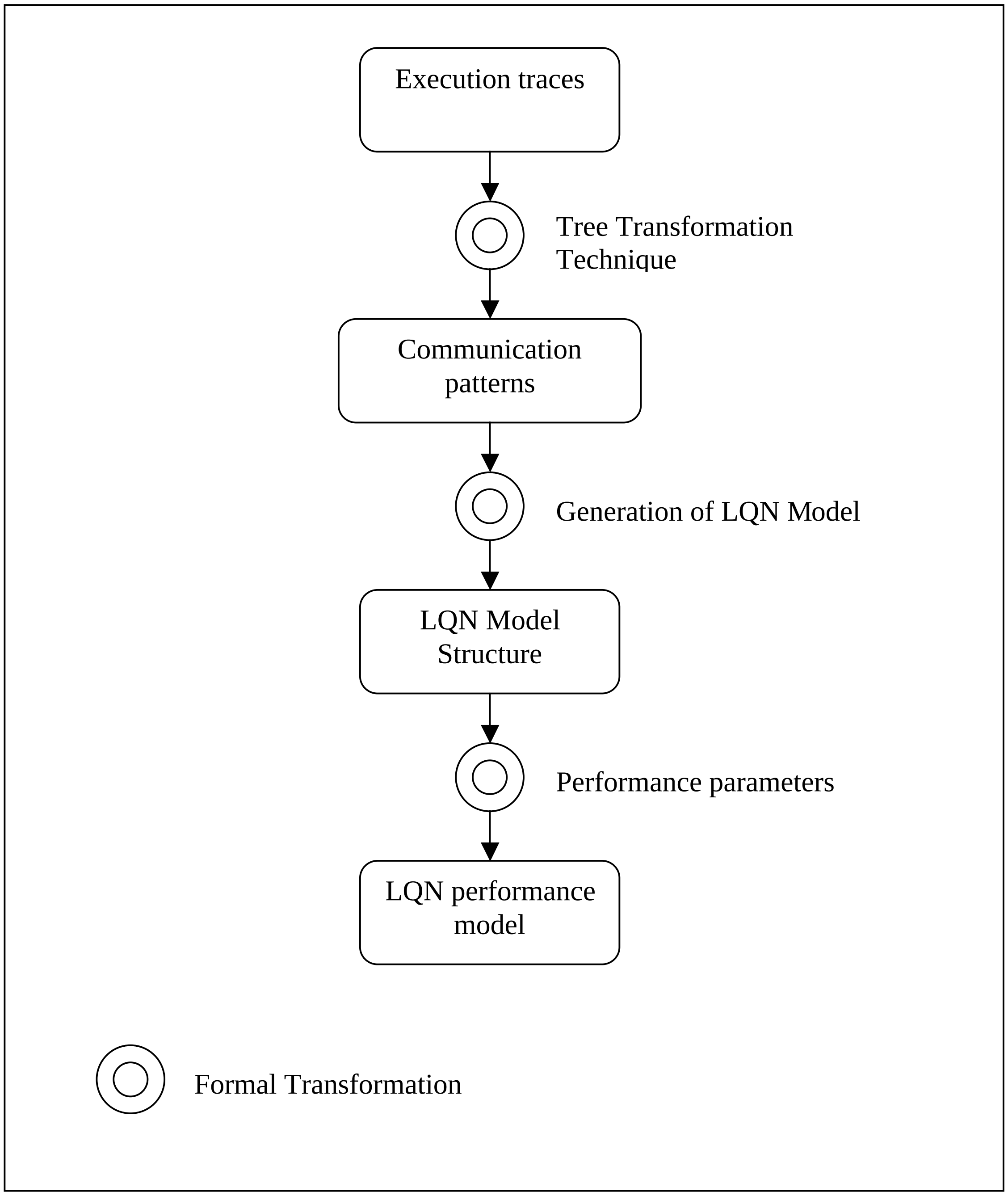


Figure 1.1 Steps involved in Software Architecture and Model Extraction Technique

1.5.1 Obtaining Execution Traces

An execution trace records every operation during execution of a scenario in chronological order. Contrary to the case of previous research, there is no need for angio-tracing. In order for SAME to produce a complete pe rformance model, it requires that a set of traces covers all scenarios of the distributed function. A timestamp is attached to every event recorded in a trace to maintain its chronological order. To obtain the complete behavior of system, the traces are merged together while still maintaining the chronological order of events. This gives the complete behavior of the system in one trace. In case of distributed systems, correct chronological ordering is achieved through the synchronization of clocks. If the clocks get skewed, corrective measures are taken to correct the problem.

1.5.2 Extraction of Communication Patterns

Processes, tasks or actors interact with each other using three interaction protocols (also known as communication patterns). In t his study, communication patterns are behavioral patterns rather than the design patterns. They are as follows:

1. Asynchronous Communication Pattern
2. Synchronous Communication Pattern
3. Forwarding Communication Pattern

These patterns are fully explained in s ections 3.3.1, 3.3.2 and 3.3.3. The heart of SAME model building technique is the detection of these communication patterns in a given trace. Communication patterns are detected through a tree transformation technique, which is fully explained in Chapter 4.

Communication patterns give the precedence relationship between processes, which gives enough information to obtain the hierarchical structure of a system. This information is extremely valuable as it validates the implementation of design.

1.5.3 Generation of LQN model structure

Communication patterns extracted from a trace serves as input for generating the Layered Queuing Network (LQN) model structure. LQN model is an extended form of queuing network model, which designates the processes (o r tasks) in a hierarchical layered order and is explained in more detail in section 2.2. LQN models structures are incomplete version of complete LQN models as they lack information regarding resource consumptions.

1.5.4 Generation of complete LQN model

In this step resource functions are added to LQN model structure. For each operation, a resource consumption value (for CPU consumption, storage operations, and any other operations of the process to carry out the execution step) must be made available from a repository of resource functions. Addition of resource functions is not part of this thesis and was previously achieved by Stefan Bayarov in [Bayarov99].

By solving this model, performance measures such as process service times, throughput of jobs, contention delays and response times are obtained which are compared against system requirements.

1.6 Contributions

There are several contributions of this research, which are as follows:

1. A complete algorithm for finding the communication patterns in a trace, described in chapter 4.
2. A complete algorithm for the development of LQN performance model from communication patterns, in chapter 4.
3. The validation of the algorithm, in chapter 5.
4. The analysis of substantial telephone case study, in chapter 6.
5. The development of a general trace-based model building technique.

6. The implementation of both algorithms in Java programming language

1.7 Dissertation Overview

This thesis is organized as follows. Chapter 2.0 gives background in performance engineering and related work. Chapter 3.0 presents the perspective and serves as motivation to chapter 4, which is the explanation of tree transformation algorithm. Chapter 5 proves the validity of the tree transformation algorithm. Chapter 6 is a case study and chapter 7 is the conclusion.

Chapter 2: SPE and Models

This chapter gives an overall view of Software Performance Engineering (SPE), the role of performance modeling as one of the approaches to SPE, and the problem of building performance models.

Software Performance Engineering (SPE) is defined as a set of methods for constructing software systems in such a way that they meet some defined performance objectives [Smith90]. Performance refers to throughput or response time as seen by the end user of a software system. Software performance engineering is used to enhance performance or address performance issues throughout the lifetime of a computer system. However performance engineering done in early design stages avoids developers wasting significant time in the implementation of a software system. Different approaches to SPE applied over the span of software development cycle are briefly mentioned in the next section, leading towards software performance modeling.

2.1 Different Approaches to SPE

Software performance engineering may use a combination of three techniques;

1. Fine-tuning
2. Performance measurement
3. Performance models.

The use of these techniques is heavily dependant on the architecture of the computer system being analyzed

2.1.1 Fine-tuning

Fine-tuning is applied to already existing systems and is performed by someone, who is knowledgeable of the design and the implementation of the system. Fine -tuning is relatively faster than the other SPE techniques, but is only good for small amount of performance enhancement and can only be successfully applied to tightly coupled non - distributed systems. Smith refers to this technique as the “fix it later” approach

[Smith90]. The obvious drawback to this approach is that it is extremely difficult to change any major components or re-design the system, once it is implemented and performance difference is almost unsubstantial.

2.1.2 Performance Measurements

A second approach in solving performance-related problems is by measuring performance metrics [Fenton97]. Metrics are obtained through experiments or normal use of a system. One of the advantages of this approach is that it gives the testing engineer exact, accurate and reliable measures of the system. It may also reveal the bottleneck of the system.

However, this approach has its own drawbacks. As in the fix-it-later approach, a system under test must be operational in order to make any measurements and corrections. This may prove to be highly unfeasible as it takes considerable time in human hours and corrective measures may include redesigning of the system or changing its one or more major components. Also, there may be a problem in choosing a live or synthetic workload and to be sure that either one proves to be representation of its typical operating condition [Jain91].

2.1.3 Performance Modeling

The third approach to SPE is by constructing performance models. A model should emphasize key aspects of a system by combining behavioral and structural aspects of workload elements in terms of resource usage. Using this information, performance predictions can be developed which can be analyzed for problems for several system configurations and workloads.

There are several advantages of doing SPE through performance models, one being that performance models can be used in early stages of a system design, to pinpoint and rectify key problem areas. This illuminates the need of redesigning the software system after it is operational. This is especially important in hard -real time systems where all operations should meet strict performance requirements.

A modeling technique called Rate Monotonic Analysis (RMA) was introduced as a simple way to assert the schedulability of a set of tasks [Liu73]. Though limited to the simplest ca se of multiprogramming (strictly non -periodic activities), RMA allows analysts to predict with mathematical certitude that each task in a set would or would not meet its timing requirements. In RMA priorities of the tasks are assigned monotonically in the order of their rate, with the highest rate having the highest priority. This technique would eventually become the foundation of later schedulability assessment techniques, in real-time systems.

Another modeling technique is Queuing Network Models (QNMs) as mentioned in [Jane91]. QNM captures the most important features of actual systems, for example many independent devices with queues and jobs moving from one device to the next. Experience shows that performance issues are much more sensitive to para meters such as mean service time per job at device, or mean number of visits per job to a device, than to many of the details of policies and mechanisms throughout the operating system (which are difficult to represent concisely). In QNMs, the assumptions of the analysis are realistic. General service time distribution can be handled at many devices; load -dependant devices can be modeled; multiple classes of jobs can be accommodated.

There are two ways of solving queuing network models, analytical and si mulation. Analytical solving produces performance results through solving a model logically and produces accurate results given the correct input. The models solved through simulations give simulated results, however simulation is versatile and solves mo dels, which cannot be solved easily analytically.

Although QNM is a powerful technique to solve performance problems, there are certain things that QNMs cannot do quickly and easily [Smith90]. There are limits in the ability of QNMs to represent executio n behavior that can be solved with efficient, exact analytical methods. In computer systems, though, jobs may have simultaneous resource possession. For example, a job may be executing on the CPU, issue a request to pre-fetch data from disk, and then con tinue its CPU execution while the disk request executed in parallel. Additionally special measures are needed for passive resources: resources required for processing but do not actively provide service. Memory is the classic example. Programs require m emory to execute on the CPU, but memory does not process jobs. Other passive resources include locks for exclusive access to data or files, message passed between processes, and so forth. Other computer system characteristics that present problems for QNMs are as follows:

- Routing to queue servers that depends on data characteristics or the state of the computer system.
- Job execution that varies over the life cycle of the job (called job phases),
- Jobs that fork into multiple processes, then later join and continue processing,
- Certain combinations of queue -scheduling disciplines and service time distributions.

These drawbacks of queuing network models are some what overcome by an extended version of queuing network model, the Layered Queuing Network (LQN) model.

2.2 Layered Queuing Network Model

Layered queuing modeling is a new adaptation of extended queuing models for software systems proposed by Woodside et al. [Rolia95, Woodside89 and Woodside95]. It is capable of modeling most of the features that ar e important from performance point of view such as multi -threaded processes, devices, locks, communication and so on. LQN model is the target performance model for this research.

LQN models represents software resources in a natural way so that they are part of framework and thus approximations do not have to be developed for every system. The model is closely linked to software descriptions and provides a transparent representation of software architecture, which makes models easy to develop and understand. The model is well suited for systems with parallel processes running on a multiprocessor or on a network, such as a client-server system.

Software and hardware objects are represented as tasks in LQN models, which may execute concurrently. They are three categories of tasks in LQN models.

a. **Client Task**: Only sends requests, and are used to model input sources such as users and so on.
b. **Active Server Tasks**: These tasks can receive requests and as well as make their own requests to other tasks.
c. **Pure Se rver Tasks**: These tasks are used to model hardware devices such as processors or disks. They receive requests from other task but cannot make their own requests.

A task accepts a service request at an entry. Requests for service are characterized as entry-to-entry interactions using a communication protocol. An entry may correspond to an actual task communication port or a message type that identifies a particular service by a task. LQN represents three communication protocol types mentioned as following:

1. Asynchronous: A task that requests a service does not wait for the reply of the request. No reply message is needed. An example of this interaction is illustrated in Figure 2.1-a.

2. Synchronous, RPC or Rendezvous: A task that requests a service waits for a reply to its request and becomes ready to run after it has received its reply. This is illustrated in figure 2.1-b.

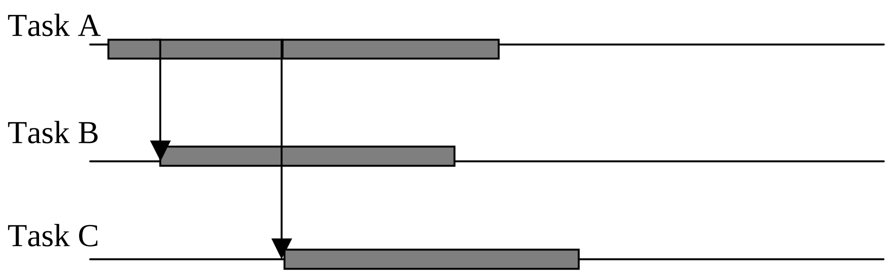


Figure 2.1-a: Asynchronous interactions

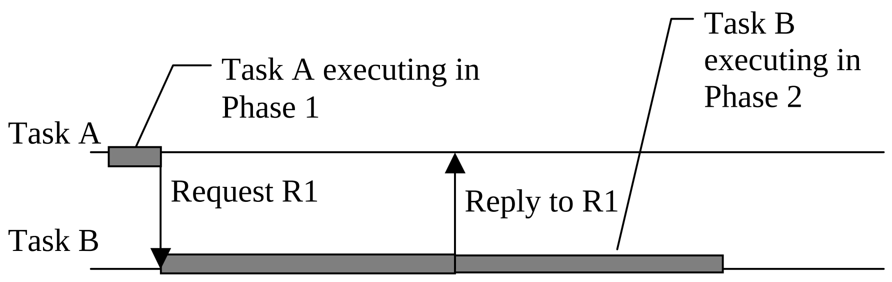


Figure 2.1-b: A synchronous interaction

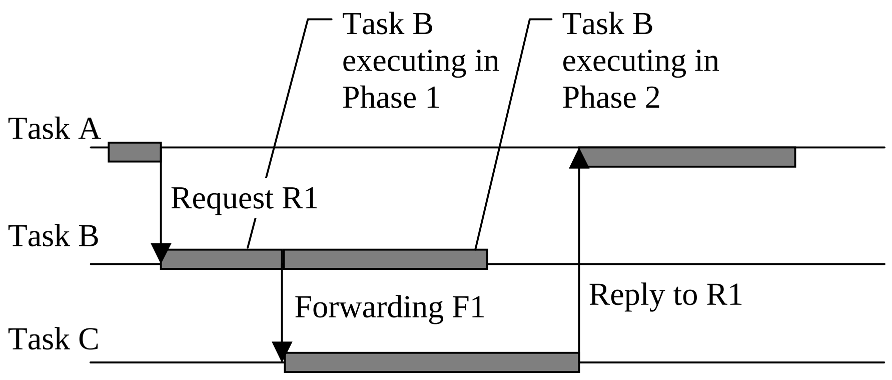


Figure 2.1-c: A forwarding interaction

Figure 2.1: Interaction Patterns

3. Forwarding: A forwarding request is a special case of an asynchronous request. It is illustrated in Figure 2.1-c. It occurs when the responding task of a RPC request asynchronously sends the request to another responding process; not to the initiating blocked task. Each responding process can continue to forward the request further to other responding processes until the last responding process in the series sends a reply directly to the blocked thread. This type of behavior occurs when a task acts as a manager by sendin g requests to a set of worker tasks [Gentleman81].

There may be two phases of execution in the synchronous protocol type. The first phase consists of the activities between the acceptance of a synchronous message and its consequent reply (or forwarding). All activities that occur after the reply (or forwarding), but before the acceptance of a new request, are labeled as a second phase. The second phase identifies resource contention between the initiating task and the continuing responding task [Woodside89].

An example of an LQN model is shown in Figure 2.2. Tasks are shown as parallelograms and requests from one task to another are made from and to service entries, which are ports or addresses of particular services offered by a task. An entry executes activities with precedence relationships, and activities have resource demands and can make requests to other tasks. For each activity, a resource consumption value (for CPU consumption, storage operations, and any other operations of the process to carry out the execution step) must be made available from a repository of resource functions. Resource functions will not be discussed in this thesis.

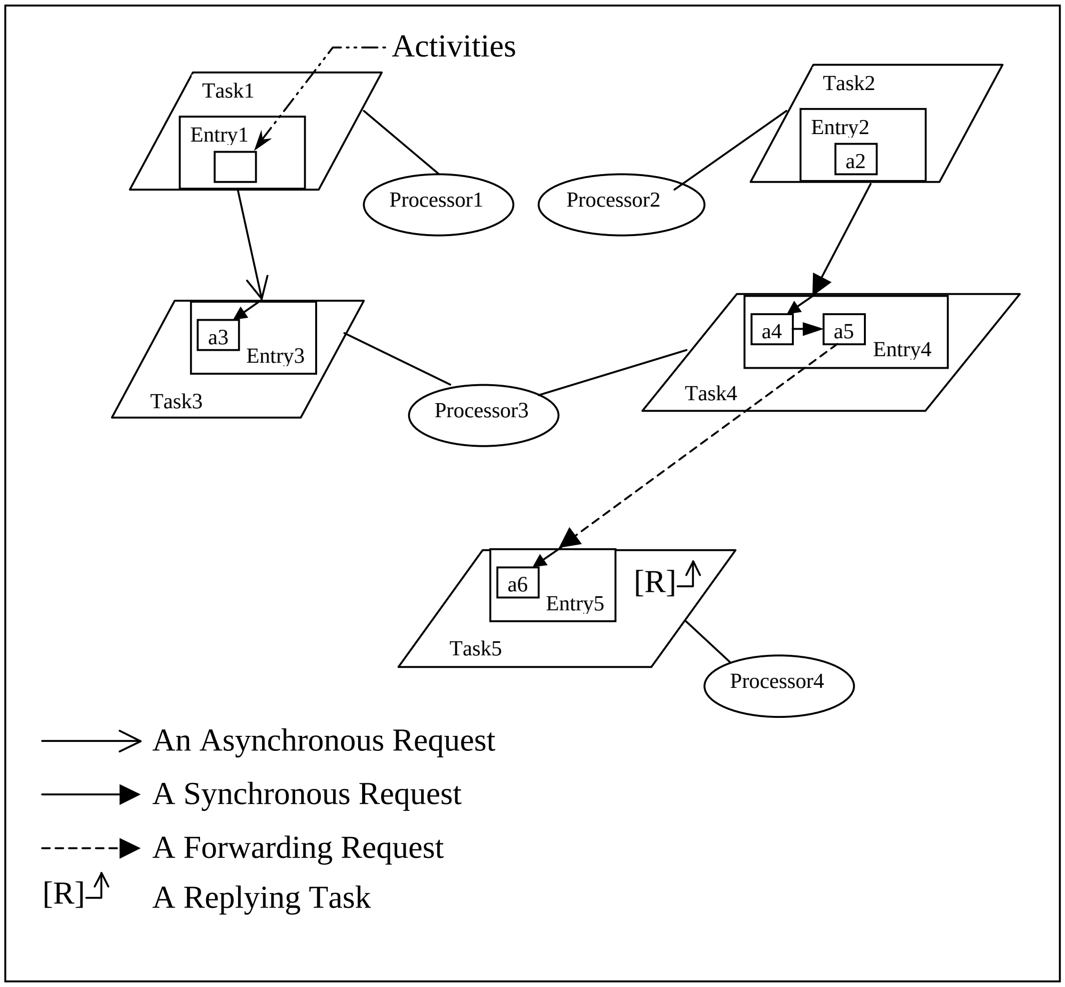


Figure 2.2: An example of an LQN model

2.3 Different approaches to building performance models

There are several different ways to construct a performance model, depending on the information provided about the system,

Connie Smith proposed a systematic way to use quantitative methods to asses requirements, design and hardware altern atives, starting early in the life -cycle while a wide range of options exists, and continuing through the life -cycle [Smith90]. The method derives a model of software execution patterns (called an execution graph) from the design, constructs a second model to solve for performance predictions, and then uses the predictions to guide the modification of the design.

C. Scratchley [Scratchley99] described an approach called PERFECT, which evaluates the feasibility of proposed software concurrency architectures for a set of scenarios and a set of quality -of-service requirements. The method first specifies and captures the scenarios using Use Case Maps [Buhr96], which represent paths of execution against a background of the software components that execute them, and annotates the quality-of-service requirements on the scenarios. Then the method allocates sub -paths in the specification to processes and decides whether each process will be single or multi - threaded. Finally, constructing and simulating a virtual impl ementation of the system, which conforms to the specified behavior and the specified concurrency architecture, performs an evaluation.

D. Menasce and H. Gomaa proposed a methodology based on a software performance engineering language, CLISSPE [Menasce98, 99]. Use Cases were developed and mapped to a performance modeling specification using the language. A compiler for CLISSPE generates an analytic performance model for the system. The compiler from the system specification derives Service demand parameter s at servers, storage boxes, and networks. Detailed models of DBMS query optimizers allow the compiler to estimate the number of I/Os and CPU time for SQL statements.

2.4 Trace Based Approach to Generate LQN Models

This section will discuss earlier work done on automated generation of LQN models from system design prototypes.

Distributed systems are composed of graphically dispersed, heterogeneous hardware with scheduled, concurrent software components. These systems differ somewhat from the classical p arallel or concurrent system models such as described in [schwarz94]. First tasks are resources because simultaneous distributed operations share them. Secondly, a task’s lifetime can extend beyond that of a distributed operation. The set of tasks is al so dynamic where tasks may be added or removed. Thirdly, task execution follows a cycle, beginning when a service request is accepted and ending when the service request is satisfied.

Software performance models of distributed operations describe tasks and their interactions because they affect queuing delays, parallelism, and resource contention. For example, a heavily used task can queue arriving requests and can even become a bottleneck [Neilson95]. To characterize the involved tasks, their individual activities, and their interactions with each other, the concept of an angio-trace is introduced in [Hrischuk95a].

2.4.1 Angio-traces

An angio-trace characterizes a distributed operation independent of other simultaneous distributed operations. The name angio-trace is derived by analogy from an angiogram. An angiogram is a visualization of individual's blood flow that is produced by injecting a radio opaque-dye into the blood stream and taking the x-ray of the dye dispersion. Similarly an angio-trace assigns a dye to each distributed operation so that they can be distinguished. Also, angio-trace uses a new type of logical clock to characterize potential and operation causality, independent of environmental factor such as scheduling. Since it does not require a global system clock or clock synchronization mechanisms, it is quite suitable for monitoring a distributed system. Angio-trace allows events to interleave so that multiple angio-traces can be recorded simultaneously. The angio-trace event format also records information with each user defined and communication event that captures their cause and affect relationships.

There are three generations of angio-trace. The first generation angio-trace (AT1) is introduced in [Hrischuk95]. It is further evolved in its second generation (AT2) in [Hrischuk99] and third generation angio-trace (AT3) is mentioned in [El-Sayed99]. Angio-trace used in this thesis is AT1 since it is a part of Simulation RTS mechanism of ObjecTime Developer Toolset. A sample of an angio-trace obtained from ObjecTime Developer Toolset is illustrated in Figure 2.3.

```
thread Sendingrequest.1.1 {
174863 Client send %Snd;
174864 Server1 state PowerIdle/CollectDigits;
174864 Server1 receive %Snd;
174868 Server1 send %Snd;
174868 Server1 spawn SendingCallrequest.1.1.1;
174871 Server1 send %Display;
174871 Server1 spawn SendingCallrequest.1.1.2;
174874 Server1 state PowerIdle/SettingUp;
};
thread SendingCallrequest.1.1.1 {
174869 Server2 receive %Snd;
174876 Server2 send %Snd;
174878 Server2 state Initialized;
174876 Server3 state S1;
174876 Server3 receive %Snd;
174885 Server3 send %Snd;
174885 Server4 state S1;
174885 Server4 receive %Snd;
174899 Server4 send %CallSetup;
174913 Server5 state Ready;
174913 Server5 receive %CallSetup;
174917 Server5 send %SetupReq;
174917 Server6 state Null0;
174917 Server6 receive %SetupReq;
}

thread SendingCallrequest.1.1.2 {
174871 Client receive %Display;
};
```

Figure 2.3: Sample angio-trace output

2.4.2 Trace-based Load Characterization Technique

This section will discuss the process to generate LQN model from angio-traces, as defined in the technique Trace-based Load Characterization (TLC) in [Hrischuk99a].

TLC generates a layered queuing network performance model that is well suited to modeling a message passing distributed system because traces reveal the dynamic details of a design that are difficult to determine from a source code or documentation information but are important to performance analysis. A distributed operation is a set of coordinated interactions between shared system server tasks and user specific tasks. An execution of a distributed operation recorded in a trace is called a *scenario*. To apply TLC it must be assumed that a finite set of scenarios can capture all the important behavioral characteristics of the distributed operations and a pre-requisite for TLC is some executable form of the design, which corresponds to the final design. It can be a very abstract executable CASE tool model or a code prototype. At a minimum, the executable design must represent some sort of task architecture and include the coarse behavior of each task. Figure 2.4 illustrates the steps involved in generating a LQN performance model from an angio-trace.

In the first step of TLC, the angio-trace records the causal flow of a distributed operation's execution by recording a timestamp with each event and then ordering the events by the time stamps.

The event ordering forms a graph called TOEG (Task and Operation Event Graph). The TOEG is a node-labeled, directed, binary, acyclic graph whose structure is based on the causal relationships between the events in the same distributed operation. Each trace event produces a labeled node in the TOEG. The arcs in the TOEG, which represent cause and effect relationships, are deduced from the event timestamp information.

The core of the automated workload characterization and model development is the algorithm for reducing a TOEG to an LQN sub-model, by transforming one graph to another. It uses a rule based graph analysis approach [Hayes-Roth78]. The rules have a predecessor part and a consequent part. The predecessor part of the rule is a graph fragment that completely matches an interaction pattern in the TOEG, called interaction template. The consequent component is a sequence of modeling operations to develop parts of LQN sub -model. The order of matching of templates is governed by a control algorithm.

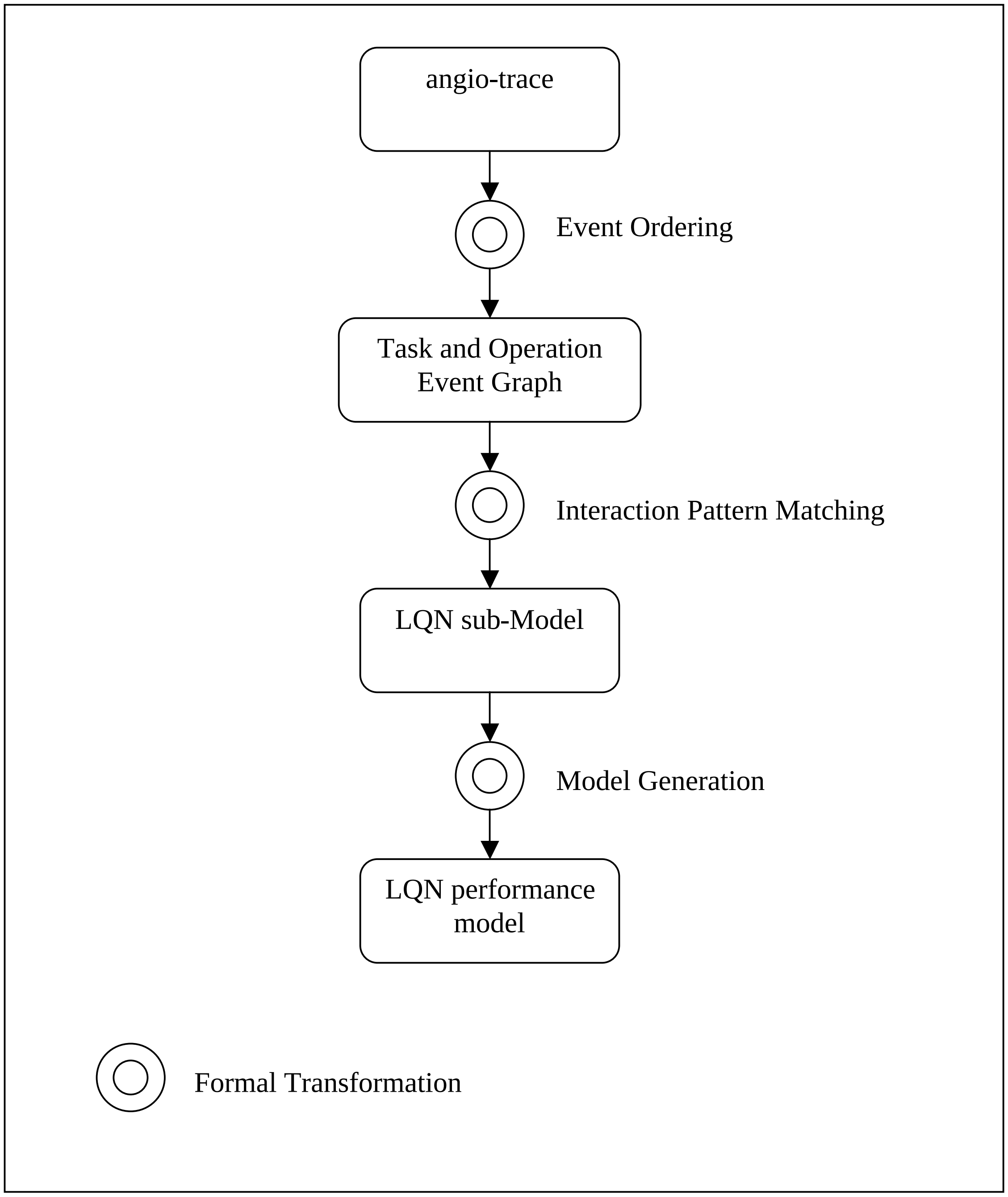


Figure 2.4: Steps involved in Trace-based Load Characterization Technique

Chapter 3: Concepts

The Software Architecture and Model Extraction (SAME) Technique can be used to analyze a wide range of systems. It is flexible to handle both homogenous non-distributed and as well as loosely coupled heterogeneous distributed systems. However there are certain assumptions, which must be made in order for SAME to be applicable. This chapter will explain the notion of task, trace, interaction patterns and assumptions that surrounds them. In the last section, the notion of interaction trees is introduced which are basic to detecting communication patterns.

3.1 Tasks and Actors

A task or an actor is a unit of software program. Each task has a host device, which is utilized when a request is processed. Host devices are not limited to just processors; they can be printers, disk devices or logical devices such as mutual exclusion locks and so on. Two or more tasks can share the same host device for example different processes utilizing a processor. In case of Object Oriented (OO) software programs, a task can simply be a static or dynamic object. Tasks communicate with each other by sending and receiving signals called messages.

In the context of this study, a task must have a unique id; such as processes running on Unix platform have unique process identification so that a tracing tool should be able to identify these tasks. If there are different threads of execution of the same task, each thread should be identified uniquely. A task has a single queue for messages asking for services; however replies to synchronous requests do not go to the queue, as there can be a second mailbox for the reply.

3.2 Traces

A trace is a record of sequence of events. Tracing a program is used to understand how the program executes. An event is described as a significant occurrence or happening within a system [Narain96].

In the context of this study, events recorded in a trace should have the following fields:

1. Task id:

In order for SAME technique to be applicable, it requires that events should be recorded with tasks with unique ids.

2. Timestamp:

In case of non -distributed concurrent systems, events are recorded in interleaved fashion, in chronological order. SAME requires that in case if events are not recorded in a chronological order, they should have a timestam p attached to them. This becomes extremely important in case of distributed systems. To overcome such a problem, the author suggests that all clocks on distributed nodes should be closely synchronized hence introducing a notion of a global clock in a dis tributed system. The synchronization of clocks can be monitored regularly. This can be achieved by each node sending messages to each other with their send times embedded in the message and the receiving node checking to see if the receive event does ind eed occur after the send event. If there is a little discrepancy, event ordering can be adjusted after the trace is recorded.

3. Message id:

An event should contain a message identification for the purpose of determining which task sends what message if send and receive events are recorded separately.

4. Event type:

It is important that an event should be recorded with a type of event. There are 3 basic types of events considered in this thesis.

   1. Send event,
   2. Receive event, and
   3. Default: non-interaction event

The assumptions of tracing used in this study are:

1. A trace must captures all necessary fields mentioned above.
2. Events are recorded by a data collection system that does not miss any events (e.g., an instrumentation system [Waheed95])
3. Message communication is reliable but messages can be delivered in an arbitrary order.

4. Monitoring does not change the order of events of the distributed operation.

A typical example of a trace is given in Figure 3.1.

```
Trace1 {
0       User send %start
1       Serv1 receive %start
200     Serv1 send %fetch
201     DiskHandler receive %fetch
400     DiskHandler send %complete
401     Serv1 receive %complete
1000    Serv1 send %complete
1001    User receive %complete
}
```

Figure 3.1: An example of a trace

3.3 Communication Patterns

Tasks communicate with each other through messages using different interaction protocols, also known as communication patterns. Extraction of these patterns from an execution trace is extremely important in performance analysis of software systems, as they provide clues for determining resource contention and blocking delays. As described in section 2.2, the following three communication patterns are considered:

1. Synchronous communication pattern.
2. Asynchronous communication pattern
3. Forwarding communication pattern

Following sections describe these patterns in detail.

3.3.1 Synchronous communication pattern

Synchronous communication pattern is found in client-server systems and in implementation environments such as Ada, V, Remote Procedure Call Systems, in Transputer systems, and in specification technique such as CSP, CCS and LOTOS.

In a synchronous communication, task that initiates a request is called a "client" and it is said to request a rendezvous. It gets blocked in the "rendezvous delay" until it gets the reply. The receiving or "server" task is said to accept the request, and executes two or more phases. The first phase is a service phase, which starts at the acceptance of the request and ends at its reply. The second phase is an autonomous phase in which the server task acts completely on its own, after being launched by the first phase. Figure 3.2 illustrates a complete synchronous communication pattern between two tasks.

There are certain advantages and disadvantages to synchronous message communication. One of the advantages being that the attention of both client and server must be maintained for the duration of message exchange. This attention is required to preserve message integrity [Narain96]. The major disadvantage from a performance point of view is when a client requests a service and the server is busy, the request has to be queued until the server is available, and the client remains blocked until it gets the reply back. This sort of blocking is a major performance concern and thus it is necessary to detect such a pattern in an execution log.

For messages to be called "synchronous", certain conditions and assumptions are used to define them more accurately and to facilitate their detection in a trace. They are as follows:

1. A client will remain blocked and will not perform any activities until it receives a reply from the server for its request.

2. Any message passing between any two tasks, in which the client seems inactive during the time it receives a message (supposedly a reply) will be considered a synchronous message, and not two separate unrelated messages, given that following conditions are met.
    a. Client task may not perform any operation before it receives the "reply" of the message.

b. The replying task is not reactivated (launched) by some other task and the reply is in the continuation of the "request" it received early from the client task.

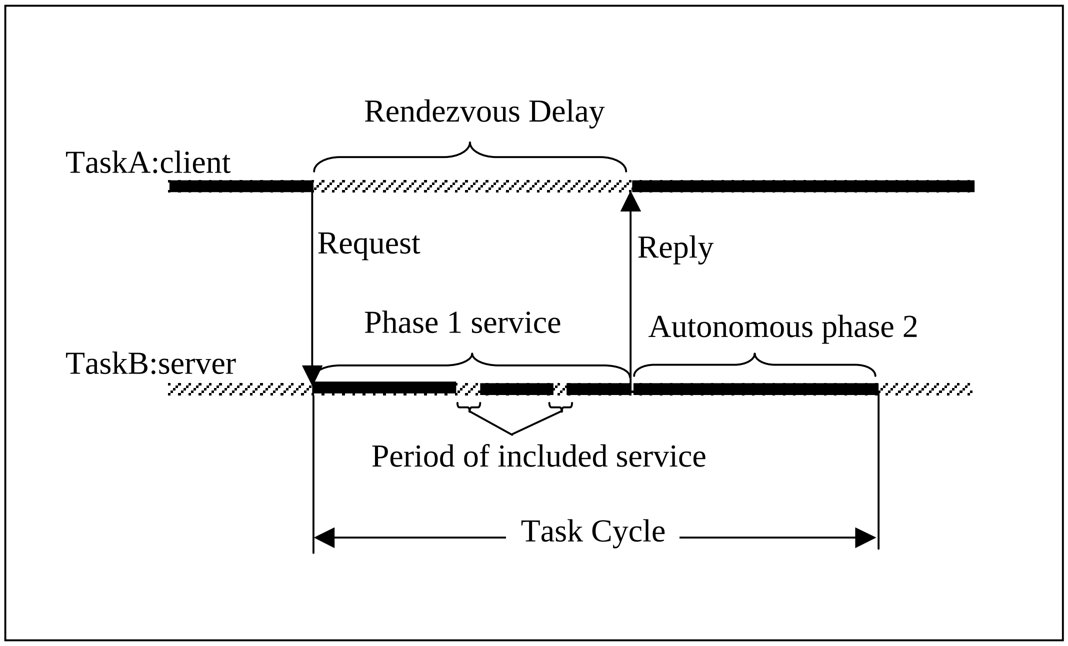


Figure 3.2: Illustration of complete synchronous communication pattern

3. The server can autonomously work in phase 2 after sending the reply.
4. A synchronous pattern may have multiple nested synchronous, asynchronous or forwarding patterns.

3.3.2 Asynchronous Communication Pattern

An asynchronous communication occurs when a task " client" sends a message to another task, "server", and there is no reply. There is no blocking or synchronization required between tasks.

When communication between tasks is asynchronous, it is possible and entirely acceptable that the server not be re ady to accept messages when the transmission occurs. The message is queued, and will be serviced when the server is ready. It is however extremely important that message communication is reliable so the service request does not gets lost. Figure 3.3 ill ustrates an example of an asynchronous message communication. Here TaskA does not wait for the reply from TaskB and after sending a message. Although asynchronous communication is not as reliable as synchronous communication, it has quite a few performance benefits. One of the obvious performance plus is that the client is never blocked. This gives the program the ability to perform concurrently. Asynchronous message communication is a foundation for constructing concurrent computing environments" [Nar ain96]. For an interaction to be recognized as an asynchronous communication pattern, the following conditions should be met.

1. A client should not be blocked waiting for the reply from the server. It can autonomously carry-on and perform other operations.
2. There is no phase 2 for the server task.

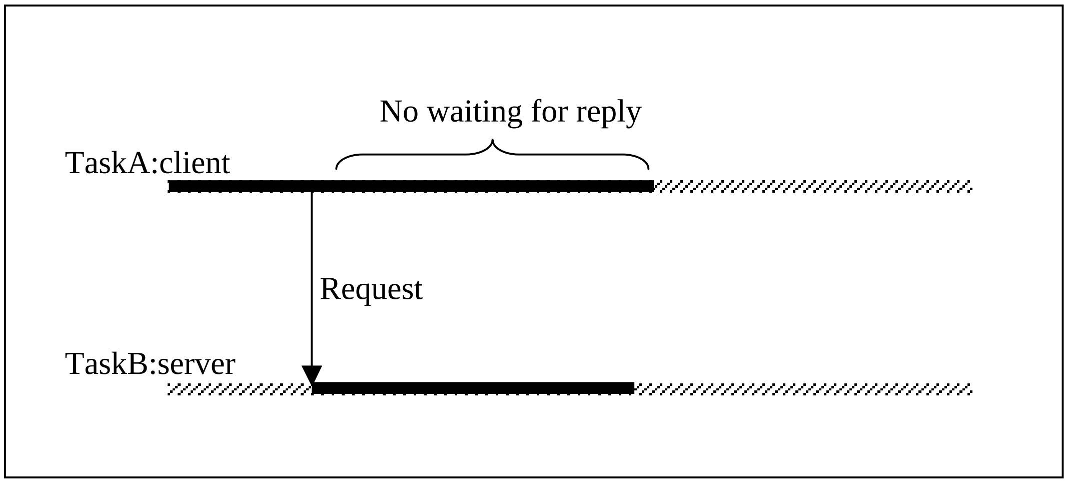


Figure 3.3: Illustration of an asynchronous communication pattern

3.3.3 Forwarding Communication Pattern

A forwarding communication pattern occurs when a client sends a request to a server and waits for the reply. The server, instead of replying to the client "forwards" the message to another server, which may itself either reply to the client or again forwards the request to a different server. The server, which forwards a request, is cal led a "forwarding server". In a forwarding communication pattern, there may be two or more forwarding servers involved. The server which replies to the client is called a "replying server".

For a forwarding communication pattern to be called as such, it should have the following properties:

1. The client remains blocked until it receives a reply from a forwarding server.
2. There can be more than one, but a finite number of forwarding servers.

3. The replying server may have phase 2 of service after replying to the client.
4. A forwarding server after it has forwarded is independent and can perform any unrelated operations.
5. In a forwarding pattern, a service may not be requested twice of the same server.
6. Like a synchronous communication pattern, a forwarding patter n may have multiple nested synchronous, asynchronous and forwarding patterns.

Figure 3.4 illustrates a simple 3 -server forwarding communication pattern. Here a client TaskA sends a request to server TaskB. TaskA is blocked until it receives a reply. TaskB did some processing on the request and forwards the request to TaskC, a lower level server. TaskC processed the request and sent a reply to client, TaskA. It is interesting to note that after forwarding a message to TaskC, TaskB is no longer part of this forwarding communication pattern and was free to perform an autonomous unrelated operation. Also, according to one of the properties mentioned above, TaskB may not be asked for service twice with in the same forwarding pattern. The replying task, Ta skC, may perform some autonomous operation in phase 2 of its service after sending a reply back to TaskA.

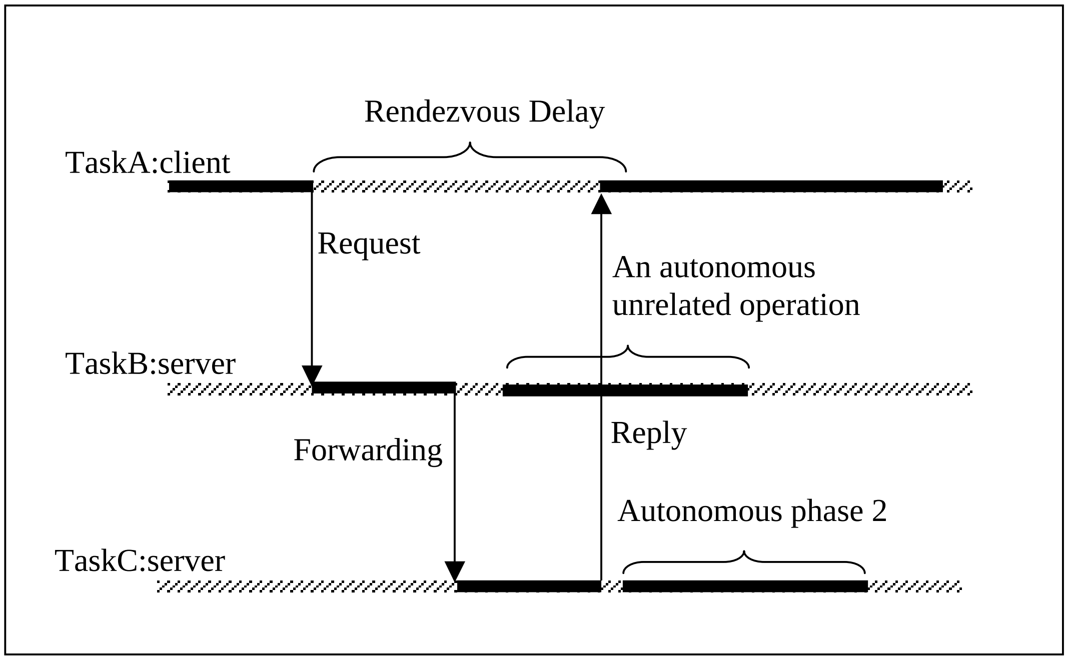


Figure 3.4: Illustration of a simple forwarding communication pattern

3.4 Interaction Trees

In this thesis, communication patterns are detected through Interaction Tree Transformation (ITT) technique. This section introduces the notion of interaction trees and gives definitions of their components. These definitions are summarized in Table 3.1

Generally speaking, a *tree* is defined as a data structure accessed beginning at the root vertex. A root vertex is a distinguished initial or fundamental item of the tree. The root is an only vertex, which has no parent. Each vertex is either a leaf or an interior node. An interior node has one or more child nodes and is called the parent of its child nodes. Contrary to a physical tree, the root is usually depicted at the top of the structure, and the leaves are depicted at the bottom. Figure 3.5 illustrates an example of a typical tree.

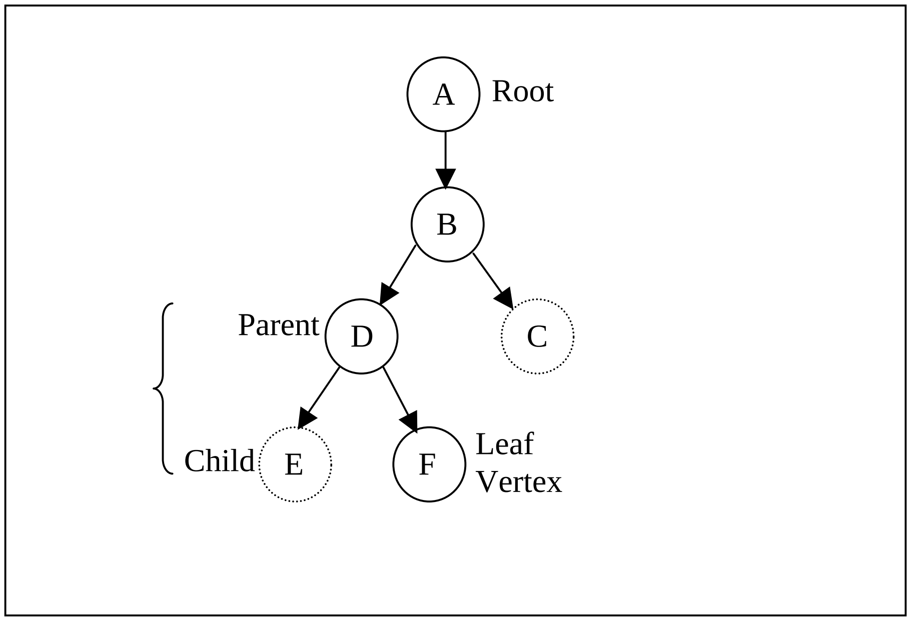


Figure 3.5: Illustration of a typical tree, H.

In the context of this study, an interaction tree represents a part of a scenario, a sub - scenario. It is created and transformed as events unfold in a tr ace. Vertices in a tree represent activations of processes. The arcs in trees represents the messages passed between these processes. Since each tree represents a sub -scenario, which is started with a single message, root vertex will always have out degree 1.

Vertex A, in tree H, in Figure 3.5, represents the activation of process proc(A) and similarly, vertices B, C, D, E and F represents the activation of processes proc(B), proc(C), proc(D), proc(E) and proc(F) respectively. We will express ordering of vertices as follows: vertex A > B, B > D and D > E. Vertices C and D are on the same level and are therefore called siblings. C is neither greater than nor less than D. The relationship between them is denoted by C<>D.

The root of a tree *tree_id*, is denoted by R(*tree_id*). In Figure 3.5, A = R(H). Since a tree may have one or more leaves, they combined, form a set of leaves, L{*tree_id*}. In tree H, vertices C, E and F $\in$ L{H}. In any given tree, since there can be only one parent to a child vertex, it is represented by P(*child_id*), however, a parent can have multiple children vertices, which belong to a set, C{*parent_id*}. In tree H, vertex D = P(C), whereas, vertex E $\in$ C{D}.

In any given trace, a process may have one or more activations. A new activation occurs when a process receives a request from another process, or when it invokes some other process, which is not a cause effect of a previous request, it received earlier. In case of a new activation, all the Previous Activations, PA{*activation_id*}, of the process become *inert* and only the new activation will be *active*. Therefore, there can be only one Active Activation, AA(*activation_id*), of a process at any given time.

A node of a tree may be active or inert. An inert node represents an activation, which has been superceded by a latest activation of the same process. There can only be one active node for each process representing the latest activation.

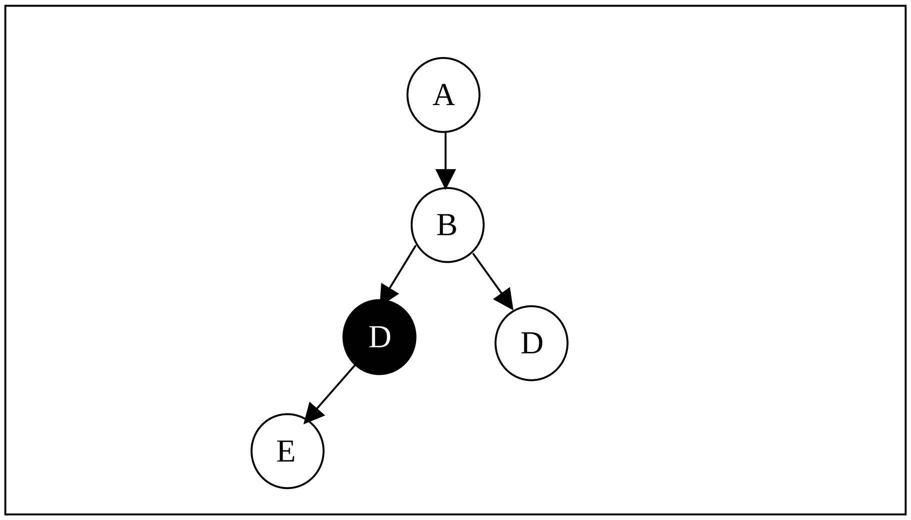


Figure 3.6: Notation of inert vertex D **in tree G**

Graphically, a dark circle, as illustrated in Figure 3.6, represents an inert vertex. An inert vertex is written in bold, **D**, where **D**∈ PA{D} and also **D** ∈ *inert*{G}. The active vertices are shown as non -dark regular circles. Vertices A, B, D and E in tree G are active vertices.

Designation	**Definition**
R(H)	Root of tree H.
Proc(A)	Process ***a***, whose activation is represented by vertex A.
A > B	Vertex A is predecessor of vertex B. Vertex A can be of higher level than P(B), as long as there is a direct path from A to B.
B < A	Vertex B is a descendant of vertex A.
A <> B	Vertex A is either a sibling of B, or A is neither a predecessor nor a descendant of vertex B, and vice versa.
P(X)	Parent of vertex X.
C{X}	A set of children vertices of Vertex X.
L{H}	A set of leaf nodes for tree H.
PA{A}	Set of previous activations of process ***a***. Note that all activations of process ***a***, have the same label, A.
AA(A)	An active activation for process ***a***. There is only one active activation allowed in this study.
Inert{H}	A set of inert vertices for tree H.
A	An inert vertex for process ***a***.

Table3.1: Definitions and notations for tree components.

Chapter 4: Tree Transformation Technique

SAME extracts communication patterns through interaction tree transformations. This chapter explains in detail the algorithm of transforming interaction trees in order to detect communication patterns. It also explains the processing involved in the extraction of a performance model from communication patterns.

4.1 Concept of an interaction tree

A rooted tree represents a communication sequence between processes. The root of the tree represents an activation of a process, which starts a non -causal spontaneous interaction. Other vertices of the tree represent activation of processes, which are involved in this sequence. The hierarchy of a tree maintains the relationship of the processes in sequence.

A tree is developed step by step as each event in the trace is processed. For example trees in Figure 4.1 represent the dynamics of creating a tree from a c ommunication trace given below.

First, Process ***A*** sends a message to Process ***B***;
then, Process ***B*** sends a message to Process ***C***;
then, Process ***B*** sends a message to Process ***D***.

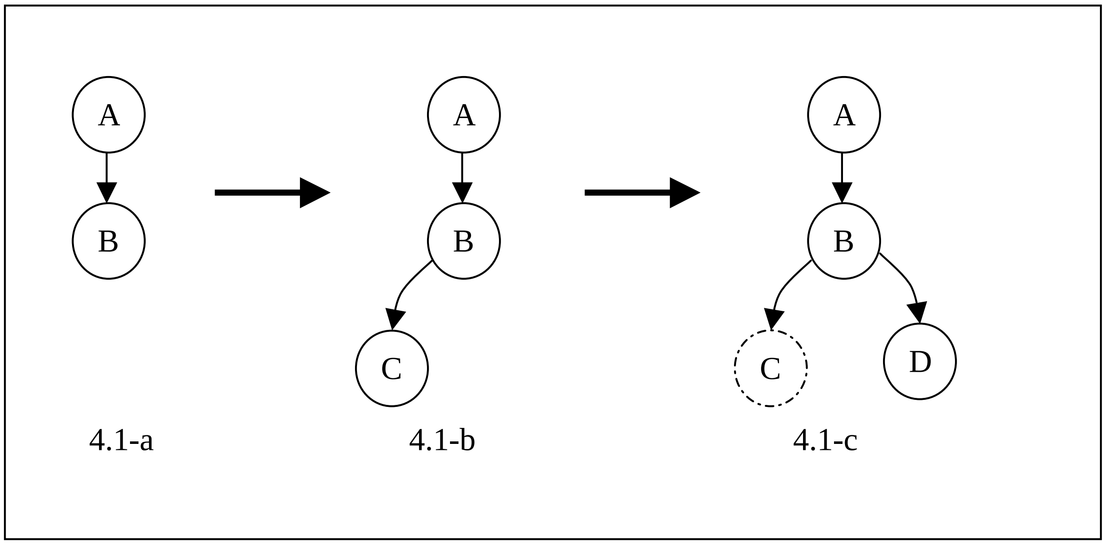


Figure 4.1: Dynamics of developing a tree.

In Figure 4.1, vertices A, B, C and D represent the activation of processes ***A***, ***B***, ***C*** and ***D***. A is the root of the tree and is denoted by R(H). Since each tree is initiated by a single message, the degree of the root is always one. Vertices C and D belong to a set of leaf vertices L(H). Here vertices A > B > C and A > B > D. Since vertices C and D are siblings, they are neither less than or greater than each other, C <> D. Vertex B is a child of vertex A, B ∈ C{A} and vertex A is a parent of vertex B, A = P{B}.

When a child ve rtex is added to a parent vertex, the child vertex is called an Immediate Candidate (IC). The reason why it is called an IC is because the process whose activation it represents has a potential of replying to the request it received earlier from the proce ss whose activation is represented by its parent vertex.

If a vertex has multiple children, the child added last is an IC and the remaining are Non - Immediate Candidate. The reason why they are called NICs is because according to assumptions in section 3.3.1, the processes, whose activation they represent may not be a part of a synchronous pattern with the process whose activation is represented by its parent vertex.

In Figure 4.1-c, vertex C is a NIC and is represented by the dashed circle. On contr ary, vertex D is an IC, an Immediate Candidate. Process ***D*** may send a reply to process ***B*** to complete a synchronous pattern, whereas Process ***C*** may not do so. However, both processes can take part in the completion of a forwarding pattern.

4.2 Multiple trees

In section 4.1, a trace is studied which represents only one communication sequence. When there are more than one communication sequences, multiple trees are required. Following is a trace that involves two communication sequences. The events are lis ted in chronological order.

1. Process ***A*** sends a message to Process ***B***;
2. Process ***B*** sends a message to Process ***C***;
3. Process ***B*** sends a message to Process ***D***;
4. Process ***E*** sends a message to Process ***B***.

Till event 3 where Process ***B*** sends a message to process ***D***, the trace is the same as discussed in section 4.1. The corresponding tree “G” can be seen in figure 4.2-a. In event 4 where process ***E*** sends a message to process ***B***, a new communication sequence is started. This is because there was no activation of process ***E*** represented in any available trees.

To represent this new communication sequence, a new tree “H” is created with E =R(H) and B $\in$ C{E}, please refer to Figure 4.2-b. Since the activation of process ***B*** in tree “G” has been superseded by a new activation in tree “H”, it is labeled inert.

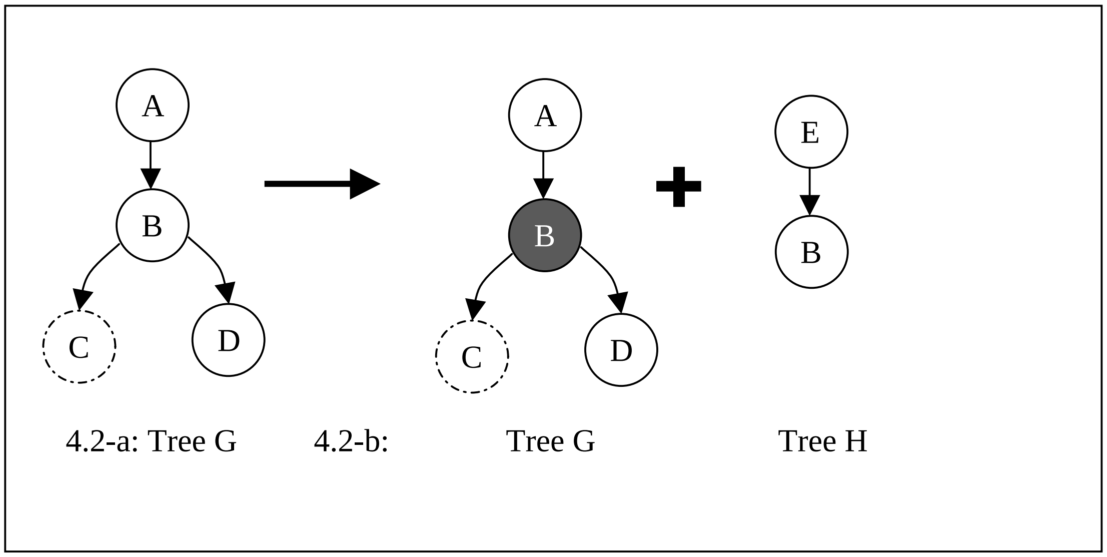


Figure 4.2: Multiple trees with active and inert vertices

4.3 Complete Algorithm

This section provides a complete algorithm of SAME technique for the detection of communication patterns. Since, each event is processed individually, it is matched against a set of comprehensive cases. There are 19 cases in total and trees are transformed differently in each case.

For an interaction event where any process proc(B) receives a message from any other process proc(A) at time $t = t_n$, the location of active vertices B and A in interaction trees determine the fate of the production of communication patterns. The first 16 of 19 cases illustrate tree transformation and production of communication patterns based on the locations of vertices A and B in interaction trees. Cases 17 and 18 covers non interaction patterns. The last case deals with the situation when a trace simply ends and there are still candidate trees existing. Three communication patterns are produced in the following format.

1. **Synchronous Communication Pattern**

   For a synchronous communication pattern between two processes, proc(x) and prox(y), the format is:

   Sync[proc(x), proc(y)], t_i, t_f

   Where,
   t_i = initial time. Time, at which the communication pattern started.
   t_f = final time. Time, at which the communication pattern is completed.

2. **Asynchronous Communication Pattern**

   For an asynchronous communication pattern between two processes, proc(x) and proc(y), the format is:

   Async[proc(x), proc(y)], t_i

   Where,
   ti = time the asynchronous interaction took place.

3. **Forwarding Communication Pattern**

   For a forwarding communication initiating from proc(x) and terminating at proc(y), the format is:

   Fwd n [proc(x), proc(a_1), proc(a_2),.., proc(a_{n-2}),proc(y)], t_i, t_{F1}, t_{F2},..,t_{Fn-2},t_f

   Where,

n = number of processes involved in a forwarding communication chain. The value of n should always be greater than or equal to 3.

proc(a1)..proc(a_{n-1}) = processes which forwards the request.

t_i = initial time. Time, at which the communication pattern started.

T_{Fj} = jth Forwarding time. It is the time where the request is forwarded fo r the jth time.

t_f = final time. Time, at which the communication pattern is completed.

Following is the complete algorithm for the SAME technique for detection of communication patterns. Tree 4.1 contains the algorithm for cases 1 -9, in which vertices A and B are located in no or different interaction trees. The function "Cleanup(*Vertex x*)" mentioned in the algorithm below is fully explained in section 4.3.1.

Cases & actions	**A $\notin$ any tree**	**For some tree J, A = R(J)**	**For some tree J, A < R(J)**
B $\notin$ any tree	**<u>Case1</u>** **Action:** 1. Create a new tree H. A = R(H) & B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** **None**	**<u>Case2</u>** **Action:** 1. x = IC(A) 2. Remove A from tree J with production 1. 3. Cleanup(x) 4. Create a new tree H. A = R(H) & B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(A),proc(x)], $t_{arc(A,x)}$	**<u>Case3</u>** **Action:** 1. Add new vertex B to vertex A. B = IC(A). $t_{arc(A,B)} = t_n$ **Production:** **None**
For some tree G, B = R(G)	**<u>Case 4</u>** **Action:** 1. x = IC(B) 2. Remove B from tree G with a production 1. 3. Cleanup(x) 4. Create a new tree H. A= R(H) & B= IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(B),proc(x)], $t_{arc(B,x)}$	**<u>Case 5</u>** **Action:** 1. x = IC(B), y = IC(A) 2. Remove B from tree G with a production 1. 3. Remove A from tree J with production 2. 3. Cleanup(x) and Cleanup(y) 4. Create a new tree H. A = R(H) & B = IC(A) **Production:** 1. Async[proc(B),proc(x)], $t_{arc(B,x)}$ 2. Async[proc(A),proc(y)], $t_{arc(A,y)}$	**<u>Case 6</u>** **Action:** 1. x = IC(B) 2. Remove B from tree G with a production 1. 3. Cleanup(x) 4. Add new vertex B to vertex A. B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(B),proc(x)], $t_{arc(B,x)}$
For some tree G, B < R(G)	**<u>Case 7</u>** **Action:** 1. If B $\notin$ L{G}, make B inert. else if B $\in$ L{G}, x = P(B), remove B from tree G with a production 1. 2. Create a new tree H. A= R(H) & B= IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(x),proc(B)], $t_{arc(x,B)}$	**<u>Case 8</u>** **Action:** 1, 2 and 3 same as Case2 4. If B $\notin$ L{G}, make B inert. else if B $\in$ L{G}, y = P(B), remove B from tree G with a production 2. 5. Create a new tree H. A= R(H) & B = IC(A)), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(A),proc(x)], $t_{arc(A,x)}$ 2. Async[proc(y),proc(B)], $t_{arc(y,B)}$	**<u>Case9</u>** **Action** 1. If B $\notin$ L{G}, make B inert. else if B $\in$ L{G}, x = P(B), remove B from tree G with a production 1. } 2. Add new vertex B to vertex A. B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(x),proc(B)], $t_{arc(x,B)}$

Table4.1: Initial 9 cases where active vertices A or B are either in no trees or in different trees.

Cases 1-9 in table 4.1 deals with the transformation of trees when active vertices A and B are in different trees or in no trees. When vertices A and B are in the same tree, say tree G, A > B, or A < B, or A <> B. Cases, where vertex A > B or A<> B are described in Table 4.2.

Cases & actions	**A = R(G)**	**L{G} ≤ A < R(G)**
B ∈ L{G}	<u>**Case10**</u> **Action:** 1. x = P(B), 2. Remove B from tree G with a production 1. 3. y = IC(A) 4. Remove A from tree G with a production 2. 5. Cleanup(y) 6. Create a new tree H. A = R(H) & B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(x),proc(B)], $t_{arc(x,B)}$ 2. Async[proc(A),proc(y)], $t_{arc(A,y)}$	<u>**Case11**</u> **Action:** 1. x = P(B), 2. Remove B from tree G with a production 1. 3. Add new vertex B to vertex A. B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** 1. Async[proc(x),proc(B)], $t_{arc(x,B)}$
B ∉ L(G)	<u>**Case 12**</u> **Action:** 1. Make B inert. 2. y = IC(A) 3. Remove A from tree G with a production 2. 4. Cleanup(y) 5. Create a new tree H. A = R(H) & B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** 2. Async[proc(A),proc(y)], $t_{arc(A,y)}$	<u>**Case 13**</u> **Action:** 1. Make B inert. 3. Add new vertex B to vertex A. B = IC(A), $t_{arc(A,B)} = t_n$ **Production:** none

Table 4.2: Cases 10-13, where active vertices A > B or A <> B in tree G

Synchronous and forwarding patterns are detected when the vertex representing the activation of a receiving process, in this case vertex B > the vertex representing the activation of a sending process, vertex A, in same interaction tree. In cases 10 -13, vertex B is either < A or <> A, therefore there is no production of synchronous and forwarding patterns. Following cases 14, 15 and 16 which deals with the situation where vertex A < B.

Case 14: $A \in$ IC{B}

In this case, vertex A is an immediate candidate o f vertex B and now since proc(B) receives a message from proc(A) at time t_n, a synchronous communication pattern is detected between proc(B) and proc(A). Here proc(B) is the client which requested for the service from proc(A) and waited for the reply. Th e time span of the synchronous pattern is between $t_{arc(B,A)}$ and t_n.

Action:

1. Graph G is transformed as described in section 4.3.2.

Production:

1. Sync[proc(B),proc(A)], $t_{arc(B,A)}$, t_n

Case 15: $A \in$ NIC{B}

In this case vertex A is not an immediate candidate of vertex B. There is no synchronous pattern produced because proc(B) did not wait for the reply from proc(A).

Action:

1. Vertex B is made inert.
2. Add new vertex B to vertex A. B = IC(A), $t_{arc(A,B)} = t_n$

Production:

None

<u>Case 16: A < C{B}</u>

Here vertex A is located below the level of the children of vertex B, which results in the detection of a forwarding communication pattern. The vertices in the path from vertex B to A represents the activation of processes involved in the forwarding chain.

Action:

1. Graph G is transformed as described in section 4.3.3

Production:

1. Fwd j [proc(B), proc(a_1), proc(a_2),.., proc(aj_{-2}),proc(A)], $t_{arc(B,a1)}$, $t_{arc(B,a1)}$,…, $t_{arc(aj-1,A)}$,t_n

   Where,

   j = number of processes involved in a forwarding communication chain.

   proc(a1)..proc(aj$_{-1}$) = processes which forwards the request.

All cases above deal with situations where tree transformation occurs when a process interacts with another process. Case 17 and 18 deals with a situation where a tree H is transformed containing any vertex A, when process proc(A) does not interact with another process but performs a non-interaction operation.

<u>Case 17: A = R(H)</u>

Since proc(A) does not wait for the reply, an asynchronous message is produced between proc(A) and proc(y), where y = IC(A).

Action:

1. Production 1
2. Remove vertex A from tree H.
3. cleanup(y)

Production:

1. Async[proc(A),proc(y)], $t_{arc(A,y)}$

<u>Case 18: A < R(H)</u>

Action:

1. IC(A) becomes the member of NIC{A}.

Production:

None

<u>Case 19: Trace Termination</u>

When an execution trace terminates, it is possible that there are still some unsolved trees remaining. Since there are no more events to process, it means that no more synchronous and forwarding patterns can be detected. SAME assumes that remaining interactions which could not be a part of a synchronous or a forwarding pattern are asynchronous messages. Therefore SAME processes the remaining trees by producing all asynchronous messages between the processes, whose activations are represented by vertices remaining in the trees.

4.3.1 Function Cleanup(*vertex y*)

Function Cleanup is used when the root vertex of trace R(*tree_id*) is removed and IC(R(*tree_id*)), becomes the new R(*tree_id*) . The input parameter to the function is IC(R(*tree_id*)) vertex. The function is as follows:

```
Function Cleanup(vertex y)
{
  Case 1.0:     y is an inert vertex
  {             for (i = 0, i < size of c{y}, i++)
                        {produce asynchronous message between proc(y) and proc($c_i${y})
                                remove $c_i${y} from y
      cleanup($c_i${y})
                        }//end of for statement
remove y from tree H.
    }//end of Case 1.0
    Case 2.0: y is an active vertex
    {                   for (i = 0, i < size of NICs{y}, i ++)
                        {         produce asynchronous message between proc(y) and
proc(NICi{y})
                                  remove NICi{y} from y
                                  cleanup(NICi{y})
                        }//end of for statement

                        Case 2.1: IC{y} exists
                        {         y = R(new tree G)
}//end of Case 2.1

Case 2.2: IC{y} does not exist
{   remove y from tree H
}//end of Case 2.2
    }//end of Case 2.0
}//end of function cleanup
```

An example:

Figure 4.3 illustrates tree H with the root vertex A. Vertex A represents the activation of a process proc(A). If proc(A) does not wait for the reply and sends a message to proc(X) at time t_n, the tree H is transformed in the following way.

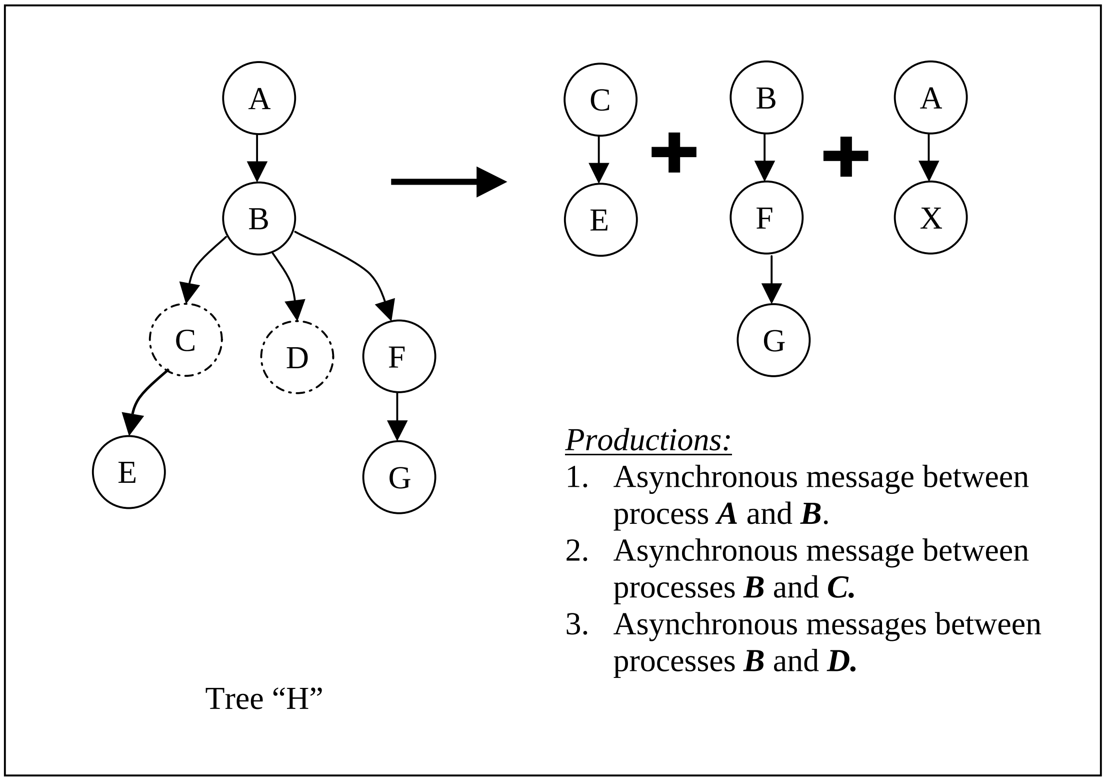
Action:

1. $y \in$ IC{x}
2. Production 1
3. cleanup(y)
4. Create a new tree G, A = R(G) and X = IC(A)

Production:

Asynch[proc(A), proc(y)], $t_{arc(A,y)}$

According to the method stated above, the vertex A will be removed from tree H and an asynchronous message is produced between process proc(A) and proc(B). If vertex B has degree >= 1, it becomes new R(H), otherwise it is also removed. All the NIC children of B are removed from vertex B, since the root vertex should only have degree 1. Asynchronous messages are produced between process proc(B) and processes proc(NICs(B)). Only the IC child, if any, remains as the child of vertex B.



Productions:

1. Asynchronous message between process ***A*** and ***B***.
2. Asynchronous message between processes ***B*** and ***C.***
3. Asynchronous messages between processes ***B*** and ***D.***

Figure 4.3: Tree transformation using function cleanup.

4.3.2 Tree transformation after detection of a synchronous message

Figure 4.4 illustrates tree H before a pr oc(B) sends a message to proc(A) at time $t = t_n$. Since Vertex B = IC{A}, it is a completion of a synchronous message between processes proc(A) and proc(B). The method by which, tree H is transformed is as follows:

<u>Pre-requisites:</u>

1. A = IC{B}
2. Both A and B are active vertices

<u>Method:</u>

```
Produce a synchronous message between proc(B) and proc(A)
for (i=0,i< size of C{B},i++)
{       Produce an asynchronous message between processes proc(B) and proc($C_i${B})
    Remove $C_i${B} from B
        cleanup($C_i${B})
}
```

Remove vertex B from tree H. In case where A = R(H), tree H is no longer required is is removed..

Since both vertices A and B are active vertices and B = IC{A}, a synchronous message is produced between processes proc(A) and proc(B) after process proc(B) sends a message to p rocess proc(A). Vertices which are < B are separated with a production of asynchronous messages between processes proc(B) and proc(C{B}), and under goes function cleanup. Vertex B is removed from tree H. Since Vertex A is R(H), it too is removed.

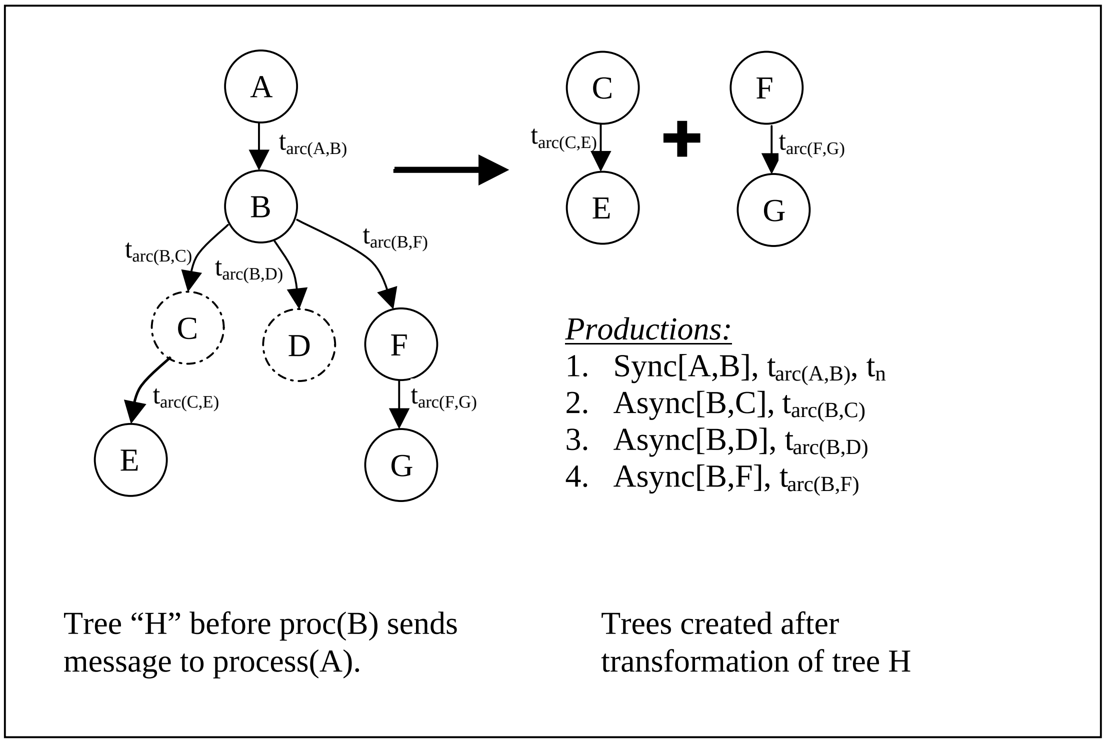

A
B
C
D
E
F
G
$t_{arc(A,B)}$
$t_{arc(B,C)}$
$t_{arc(B,D)}$
$t_{arc(B,F)}$
$t_{arc(C,E)}$
$t_{arc(F,G)}$
C
E
F
G
$t_{arc(C,E)}$
$t_{arc(F,G)}$
Productions:
1. Sync[A,B], $t_{arc(A,B)}$, t_n
2. Async[B,C], $t_{arc(B,C)}$
3. Async[B,D], $t_{arc(B,D)}$
4. Async[B,F], $t_{arc(B,F)}$
Tree "H" before proc(B) sends message to process(A).
Trees created after transformation of tree H

Figure 4.4: The transformation of a tree H after a synchronous message is detected between processes proc(A) and proc(B)

4.3.3 Tree transformation after the detection of a forwarding message

Figure 4.5 illustrates the transformation of tree H after the detection of a forwarding message. Here, instead of process proc(B) sending a message to proc(A), proc(E) sends a message to proc(A) at time t_n. The method by which the tree is transformed is as follows:

Let $x = A$, $y = E$, $z_0 = B$, $z_1 = C$

$z_0 \ldots z_1$ are vertices that are in path between vertices A to E.

Pre-requisites:

1. $y < x$,
2. $x \neq p\{y\}$, vertex x is not a direct parent of vertex y because that may result in a synchronous message instead of a forwarding message.
3. Both x and y are active vertices.

Method:

Produce a forwarding message between processes proc(x),proc(z_0), proc(z_1) and proc(y).

```
For (i = 0, i < number of vertices between x and y, i++)
{       Remove $z_i$ from p{$z_i$}
        for (n=0, n <number of c{$z_i$}, n++)
        {       if $c_n${$z_i$} is not in path between x and y
                {       produce an asynchronous message between proc($z_i$) and
proc($c_n${$z_i$})
          remove $c_n$($z_i$)
          cleanup($c_n$($z_i$)
                }
        }//end of nested for statement
}//end of for statement

Remove y from p{y}
for (i=0,i< size of c{y},i++)
{       Produce an asynchronous message between processes proc(y) and proc($c_i${y})
   Remove $c_i${y} from y
        cleanup($c_i${y})
}
if x = R(H)
{       Remove x from tree H.
}
```

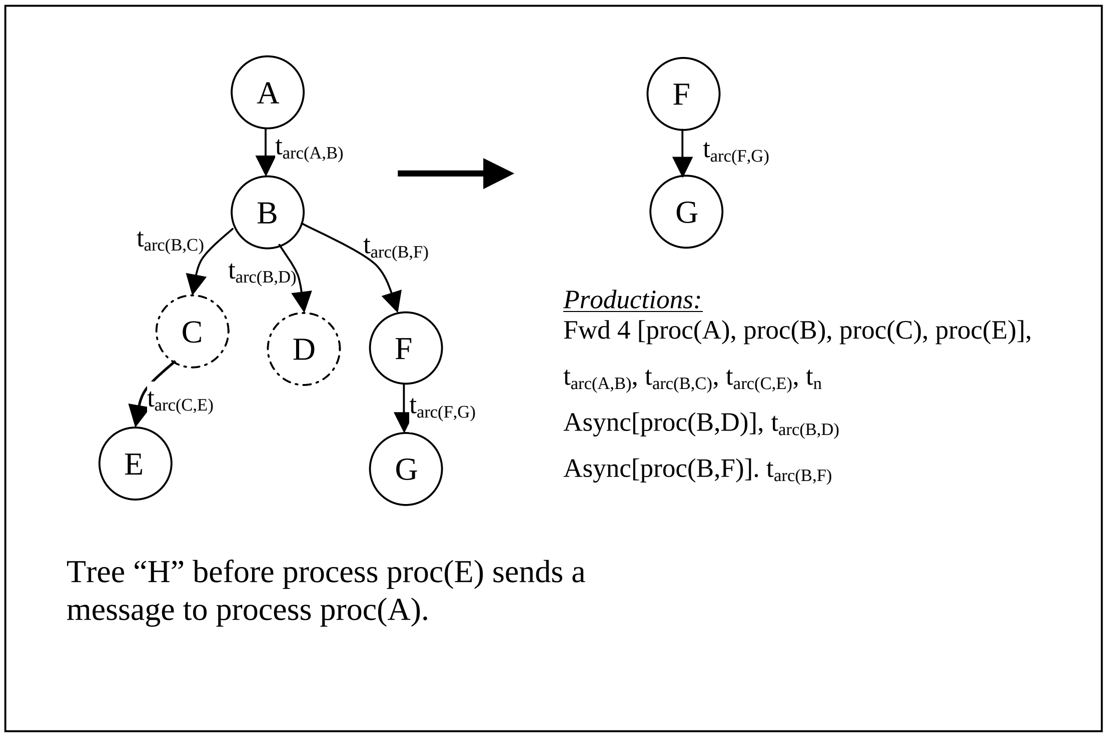

A
$t_{arc(A,B)}$
B
$t_{arc(B,C)}$
$t_{arc(B,D)}$
$t_{arc(B,F)}$
C
D
F
$t_{arc(C,E)}$
$t_{arc(F,G)}$
E
G
F
$t_{arc(F,G)}$
G
Productions:
Fwd 4 [proc(A), proc(B), proc(C), proc(E)],
$t_{arc(A,B)}$, $t_{arc(B,C)}$, $t_{arc(C,E)}$, t_n
Async[proc(B,D)], $t_{arc(B,D)}$
Async[proc(B,F)]. $t_{arc(B,F)}$
Tree "H" before process proc(E) sends a message to process proc(A).

Figure 4.5: The transformation of a tree H after a forwarding message is detected between processes proc(A), proc(B), proc(C) and proc(E).

4.4 The 2nd Pass of SAME technique

This section explains in detail the processing involved in the extraction of a LQN performance model from communication patterns.

The 2nd Pass technique takes a list of communication pattern as its input. From this it extracts all processes involved in a system and generates corresponding the tasks in the LQN model.

Tasks in LQN performance model make and accept service requests at entries. Requests for services are characterized as entry -to-entry interaction using one of the three communication protocols. A new entry is always added to task when it accepts a service requests, whether it is a synchronous, an asynchronous or a forwarding request. In the context of LQN performance model, a s ervice request is referred to as a call. The task services the request and if the call is synchronous or forwarding, it either replies back or forwards it to another task. Before replying or forwarding, a task may decide to send calls to other tasks. Th ese are called nested calls. Also, after replying or forwarding, a task may spontaneously perform operations, which starts its second phase of execution.

In the context of this study, where there are disjoint sets of interactions, it is imperative to know when a task accepts, replies back or forwards a request in LQN model domain, so that calls can be added to entries, in appropriate phases. To achieve this, an extended notion of LQN model entry is used. It contains two new fields:

$t_i[E_1]$ = time at which task receives a new request and an entry E_1 is created and
$t_f[E_1]$ = time at which entry E_1 replies to the request, or forwards it to other entry. $t_f[E_1]$ indicates the end of its phase 1 of execution.

The communication pattern list is processed in th e ascending order of their starting time. Following is the algorithm for generating LQN performance model from a list of sorted communication patterns.

4.4.1 The Algorithm

This section gives detail algorithm of generating a performance model from communication patterns. The algorithm is divided into three cases, one for each communication protocol.

Case 1. Synchronous Interaction between times $[t_1,t_2]$

Let,
T_i = task that initiate a request and
T_r = replying task

Case 1.1: No entries in task T_i or $t_1 < t_i$ for all E in T_i

1. Create and add a new entry E_1 in task T_i, where $t_i[E_1] = t_1$

2. Create and add a new entry E_2 in task T_r, where $t_i[E_2] = t_1$
3. Add a synchronous call from entry E_1 to E_2, where $t_f[E_2] = t_2$.

Case 1.2: Entries exists in task T_i

1. Get E_1, such that $t_i[E_1] = \max\{ t_i \mid t_i < t_1\}$
2. Create and add a new entry E_2 in task T_r, where $t_i[E_2] = t_1$
3. If $t_1 < t_f[E_1]$, add a synchronous call in phase 1 of E_1, to E_2
4. If $t_1 > t_f[E_1]$, add a synchronous call in phase 2 of E_1, to E_2
5. $t_f[E_2] = t_2$

Case 2: Asynchronous interaction at time t_1

Let,

T_i = task that initiate a request and

T_r = receiving task

Case 2.1: No entries in task T_i or $t_1 < t_i$ for all E in T_i

1. Create and add a new entry E_1 in task T_i, where $t_i[E_1] = t_1$
2. Create and add a new entry E_2 in task T_r, where $t_i[E_2] = t_1$
3. Add an asynchronous call from entry E_1 to E_2, where $t_f[E_2] = \infty$,

Case 2.2: Entries exist in task T_i

1. Get E_1, such that $t_i[E_1] = \max\{ t_i \mid t_i < t_1\}$
2. Create and add a new entry E_2 in task T_r, where $t_i[E_2] = t_1$
3. If $t_1 < t_f[E_1]$, add an asynchronous call in phase 1 of E_1, to E_2,
4. $t_f[E_2] = \infty$
5. If $t_1 > t_f[E_1]$, add an asynchronous call in phase 2 of E_1, to E_2
6. $t_f[E_2] = \infty$,

Case 3: Forwarding interaction at times $[t_1, t_{f1}, t_{f2}, .., t_{fn}, t_r]$

Let,

T_i = task that initiate a request and

T_{fj} = j^{th} forwarding task with the total of n forwarding tasks.

T_r = replying task

n = Number of forwarding tasks.

Case 3.1 No entries in task T_i or $t_1 < t_i$ for all E in T_i

```
1. Add a new entry $E_1$ in task $T_i$,
2. $t_i[E_1] = t_1$
3. for each j in (0, n-1):
     a. Add a new entry $E_{fj}$ in task $T_{fj}$
     b. $t_i[E_{fj}] = t_{fj-1}$
4. Add a new entry $E_2$ in task $T_r$,
5. $t_i[E_2] = t_{fn}$
6. Add a synchronous call from entry $E_1$ to $E_{f1}$
7. $t_f[E_{f1}] = t_{f1}$
8. for each j in (0, n-1):
     a. Add a forwarding call from entry $E_{fj}$ to $E_{fj+1}$
     b. $t_f[E_{fj}] = t_{fj}$
9. Add a forwarding call from entry $E_{fn}$ to $E_r$
10. $t_f[E_{fn}] = t_{fn}$
11. $t_f[E_r] = t_r$
```

Case 3.2: Entries exists in task T_i

```
1. Get $E_1$, such that $t_i[E_1] = \max\{ t_i \mid t_i < t_1\}$
2. for each j in (0, n-1):
     a. Add a new entry $E_{fj}$ in task $T_{fj}$
     b. $t_i[E_{fj}] = t_{fj-1}$
3. Add a new entry $E_2$ in task $T_r$
4. $t_i[E_2] = t_{fn}$
5. If $t_1 < t_f[E_1]$, add a synchronous call in phase 1 of $E_1$, to $E_{F1}$
6. $t_f[E_{f1}] = t_{f1}$
6. If $t_1 > t_f[E_1]$, add a synchronous call in phase 2 of $E_1$, to $E_{F1}$
7. $t_f[E_{f1}] = t_{f1}$
8. for each j in (0, n-1):
```

   a. Add a forwarding call from entry E_{fj} to E_{fj+1}
   b. $t_f[E_{fj}] = t_{fj}$

9. Add a forwarding call from entry E_{fn} to E_r
10. $t_f[E_{fn}] = t_{fn}$
11. $t_f[E_r] = t_r$

4.5 Implementation

Both the tree transformation and 2nd pass of SAME technique is implemented in Java programming language.

Application SAME1 is the implementation of tree transformation technique. The input of SAME1 is an execution trace in ASCII file format. It outputs synchronous, asynchronous and forwarding interactions in a format similar to one found in section 4.3.

Application SAME2 is the implementation of an 2nd Pass aspect of SAME technique. The input of SAME2 program is an ASCII file containing the interactions. It outputs an LQN performance model.

Chapter 5: Validation

This chapter demonstrates that the algorithms described in chapter 4 do indeed cover all the interaction cases as claimed. It describes test scenarios in timel ine diagrams, and a step-by-step walk through is done at every event to help broaden the understanding of the tree transformation technique. Also programs SAME 1 and SAME2 are applied to corresponding traces to demonstrate automated extraction of communic ation patterns and performance models.

There are a total of 6 tests, which are divided in three categories. Tests 1, 2 and 3 contain examples of simple synchronous, asynchronous and forwarding patterns. Tests 4 and 5 contain examples of nested communic ation patterns and Test 6 contains an example of a scenario representing concurrent systems. It is assumed that all processes are inactive prior to the start of a scenario and any vertex represents an active vertex unless stated otherwise.

5.1 Test 1: Synchronous Communication Pattern

The time-line diagram in Figure 5.1 shows two processes, proc(A) and proc(B) with two interactions between them, one at time t = 10 and the second at time t = 100. The generation and transformation of interaction tree is explained as follows:

At time t = 10:

Event description:

proc(B) receives a message from proc(A).

SAME action:

There are no trees containing active vertices A or B and under the case 1 in chapter 4, a new tree H is generated, where A = R(H) and B = IC(A). The arc between the vertices is labeled with the time, t = 10.

At time t = 100:

Event description:

proc(A) receives a message from proc(B). Since between time t = 10 and t = 100, proc(A) did not perform any activity, it was assumed blocked.

SAME action:

Since B = IC(A), under the case 14, a synchronous message is detected between proc(A) and proc(B). The dashed arc between vertices B and A, labeled t = 100, is not really a part of a tree but rather illustrates that a pattern is detected. The Tree H is no longer required after the detection of a pattern and is removed.

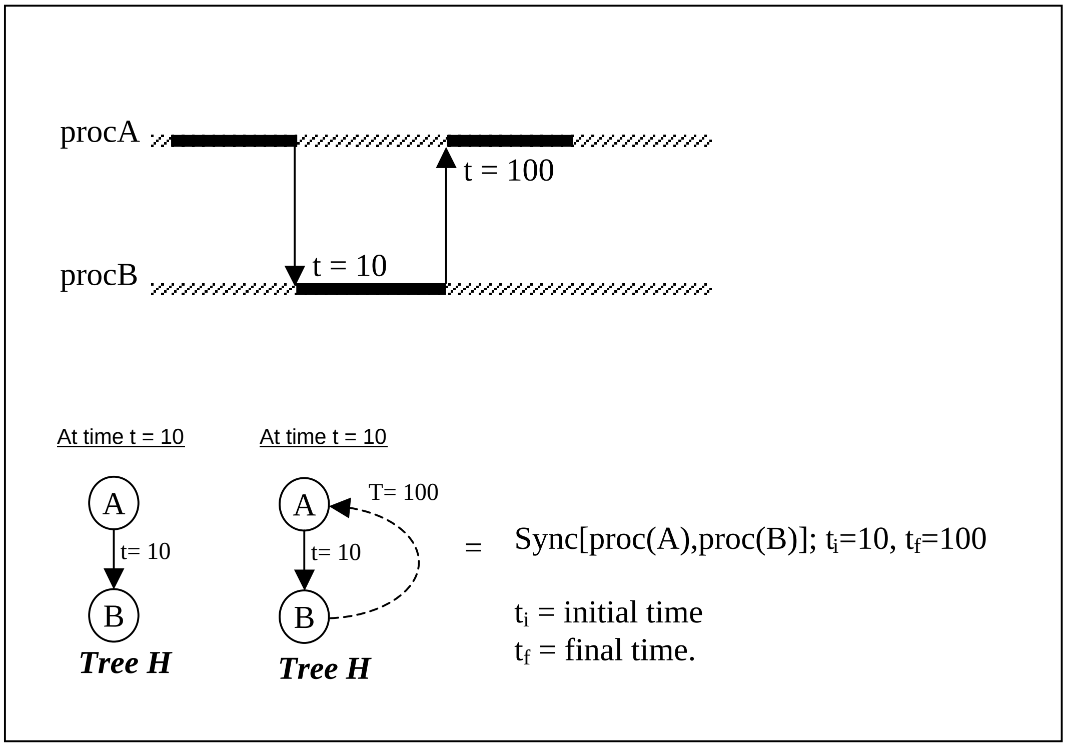


Figure 5.1: Example containing a simple synchronous communication pattern.

Programs SAME1 and SAME2 are applied to corresponding execution trace given below. The output of SAME1 program, which extracts communication patterns, is shown in Figure 5.2. The graphical representation of the output of SAME2 program, which generates the LQN performance model, is shown in Figure 5.3.

```
Trace1{
10 procA send %request1
10 procB receive %request1
100 procB send %complete
100 procA receive %complete
}
```

```
S procA procB 10 100
```

Figure 5.2: The output of SAME1 program

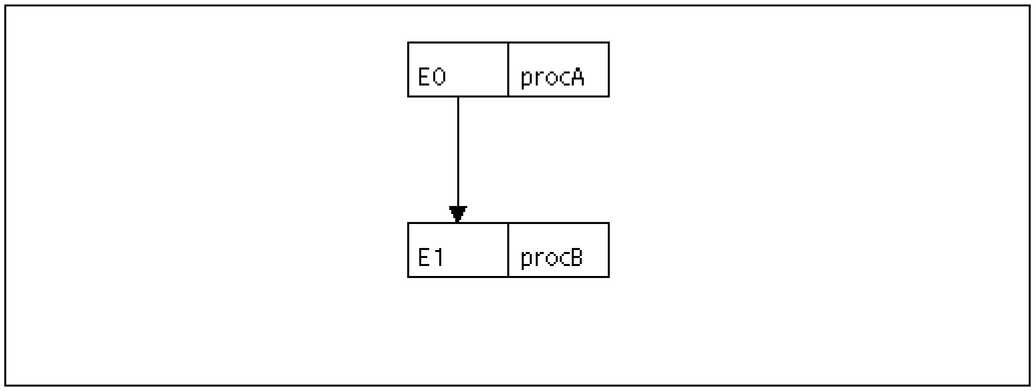


Figure 5.3: The graphical representation of LQN performance model

5.2 Test 2: Asynchronous Communication Pattern

The time -line diagram in Figure 5.4 illustrates two processes, proc(A) and proc(B). There is only one interaction between them at time t = 10, proc(A) sends a message to proc(B). The scenario ends at time t = 100 and since there is no reply to the requ est of proc(A), the interaction is considered asynchronous. The asynchronous pattern is detected as follows:

<u>At time t = 10</u>

Event Description:

Proc(B) receives a request from proc(A) and it services it.

SAME Action:

A tree H is created with vertices A = R(H) and B = IC(A) under case 1 in chapter 4. The arc joining vertices A and B is labeled with t = 10, time the message is received by process proc(B).

<u>At time t = 100</u>

Event Description:

The scenario ends and Proc(A) did not receive any reply to the request it had sent earlier at time t = 10.

SAME Action:

Under case 19 in chapter 4, the interaction between processes proc(A) and proc(B) is considered an asynchronous interaction.

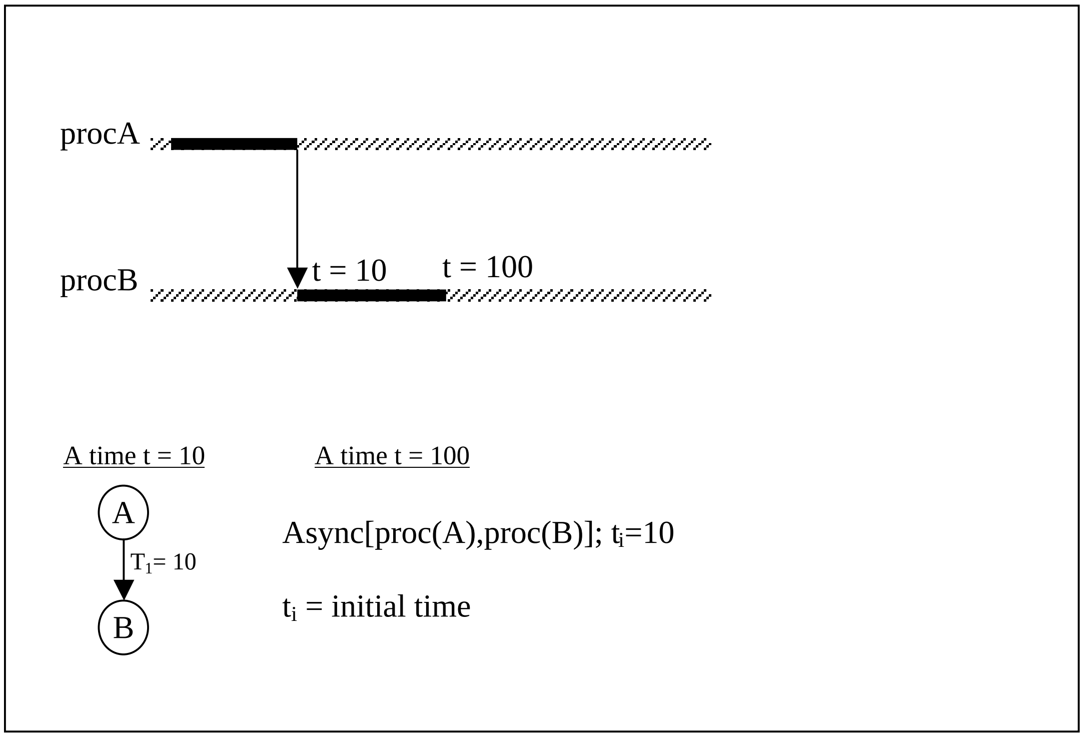


Figure 5.4: Example containing a simple asynchronous communication pattern.

Below is the corresponding trace for the scenario described in Figure 5.4. It is processed by SAME1 program and output of which is processed by SAME2 program to generate a performance model. The output of SAME1 program is shown in Figure 5.5 and the graphical representation of the output of SAME2 program is shown in Figure 5.6.

```
Trace2{
10 procA send %request1
10 procB receive %request1
}
```

```
A procA procB 10
```

Figure 5.5: The output of SAME1 program

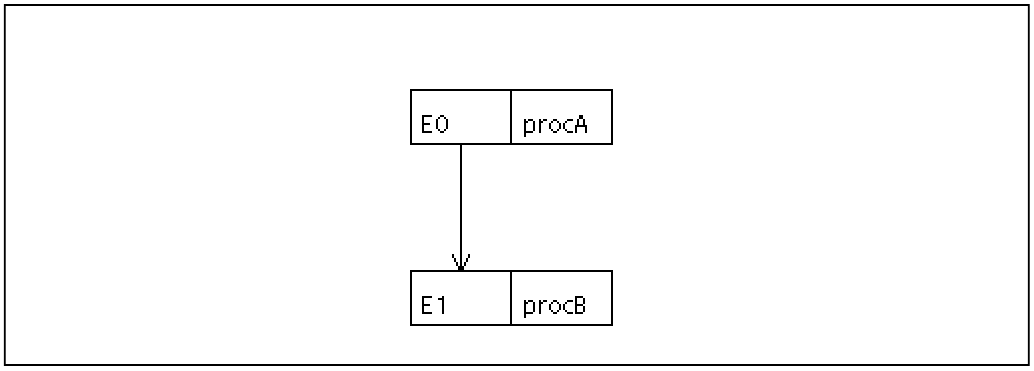


Figure 5.6: The graphical representation of LQN performance model

5.3 Test 3: Forwarding Communication Pattern

The time-line diagram in Figure 5.7 shows an example of a simple forwarding pattern between processes proc(A), proc(B) and proc(C). The pattern is detected as follows:

At time t = 10:

Event Description:

proc(B) receives a message from proc(A).

SAME Action:

A tree H is created with A= R(H) and B = IC(A) as under the case 1 in chapter 4.

<u>At time t = 100:</u>

Event Description:

proc(C) receives a message from proc(B).

SAME Action:

Since there exists a tree with active vertex B, vertex C is attached adjacent to vertex B, such that C = IC(B). The arc between vertices B and C is labeled with time t = 100.

<u>At time t = 150</u>

Event Description:

proc(A) receives a message from proc(C).

SAME Action:

In tree H, vertex A > C and as under case 16 in chapter 4, a forwarding communication pattern is detected between processes proc(A), proc(B) and proc(C) as shown in Figure 5.7

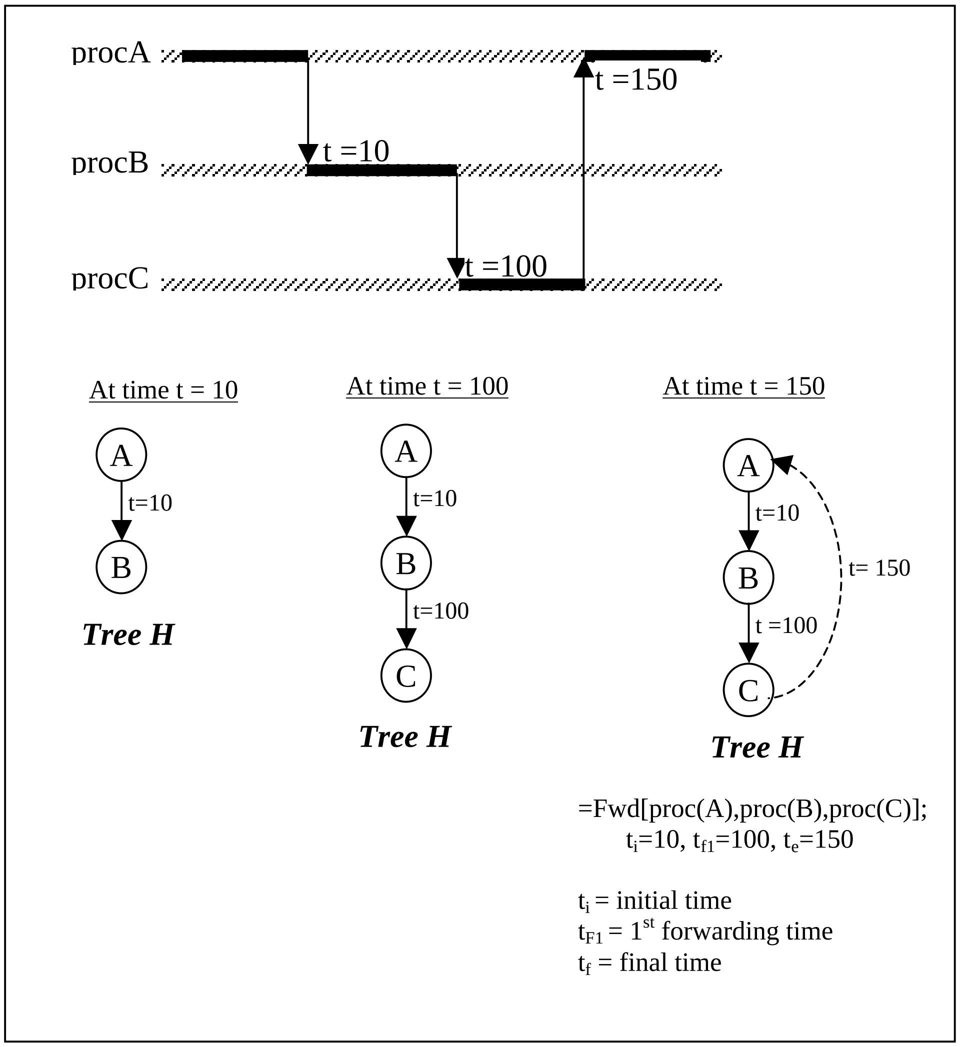


Figure 5.7: Example of a Forwarding Pattern

The trace corresponding the scenario is given below. SAME1 program is applied to it and the output is shown in Figure 5.8. The communication patterns are processed with SAME2 program and the graphical representation of the output, the LQN model is shown in Figure 5.9.

```
Trace3{
10 procA send %request1
10 procB receive %request1
100 procB send %request2
100 procC receive %request2
150 procC send %complete
150 procA receive %complete
}
```

```
F 3 procA procB procC 10 100 150
```

Figure 5.8: The output of SAME1 program

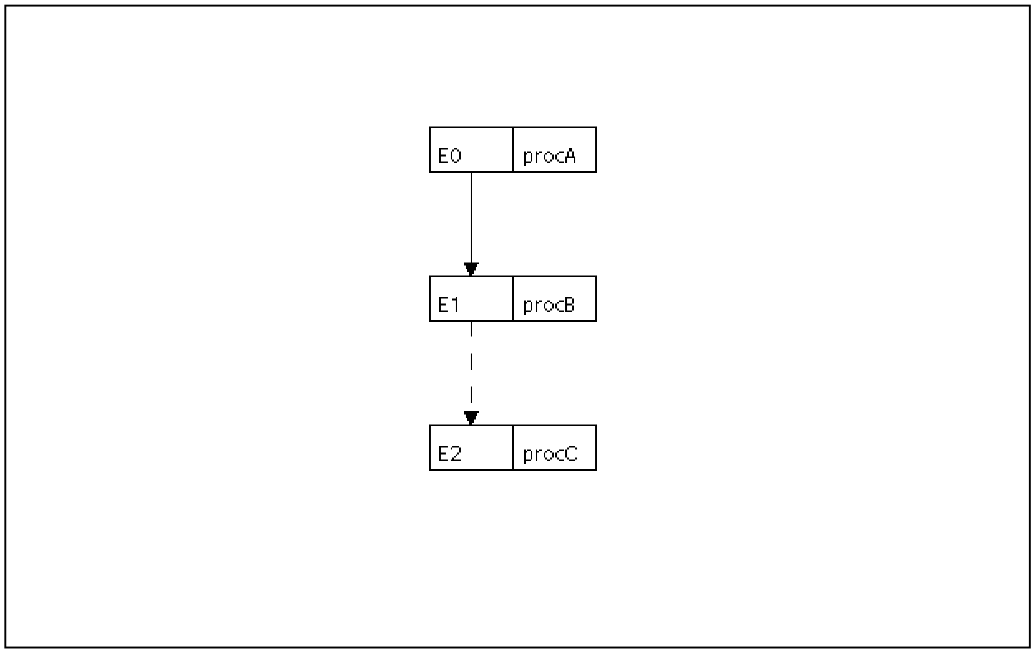


Figure 5.9: The graphical representation of LQN performance model

5.4 Test 4: Synchronous pattern with nested interaction

The time-line diagram in Figure 5.10 illustrates three processes, proc(A), proc(B) and proc(C) interacting with each other as follows:

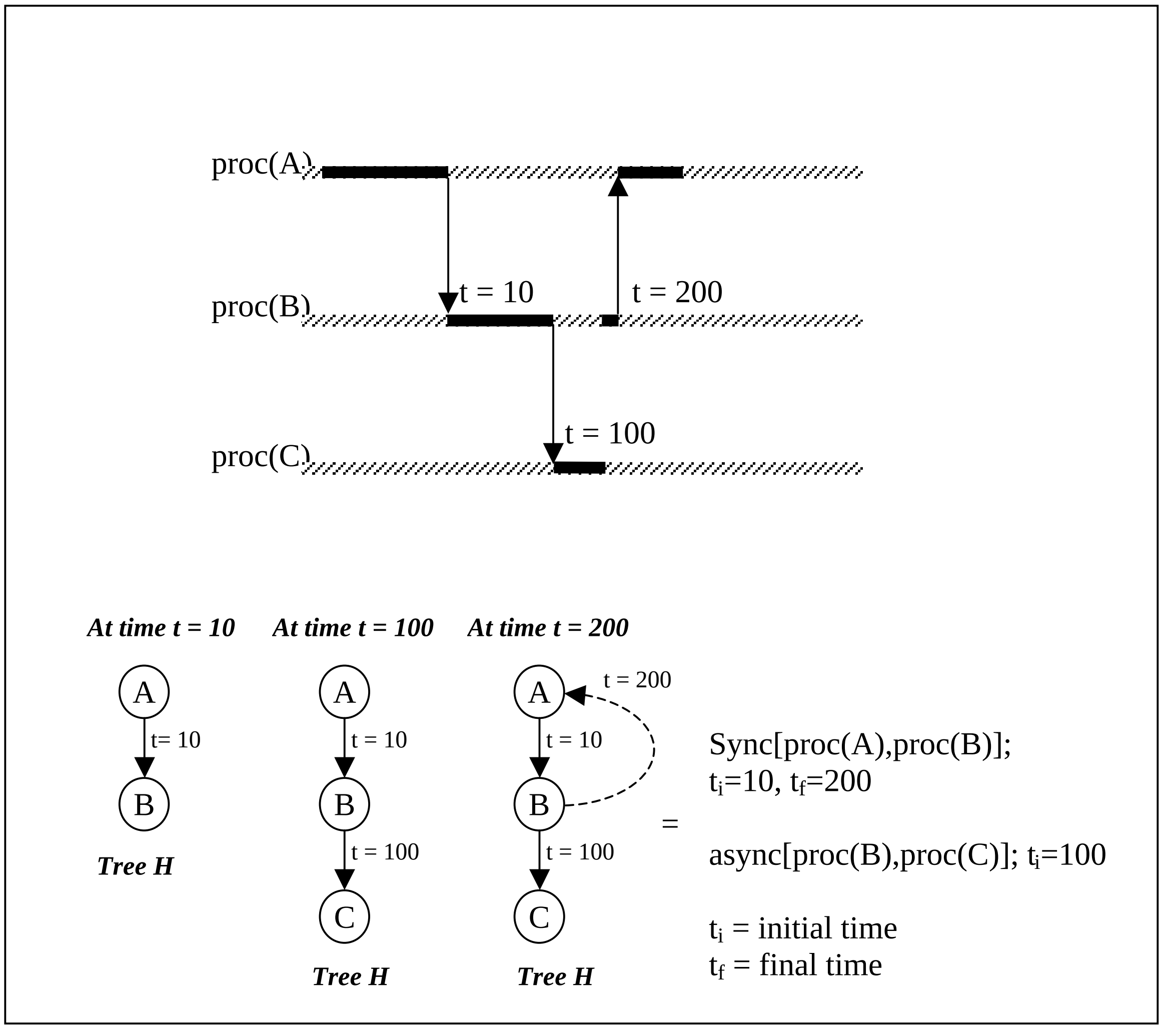


Figure 5.10: A nested asynchronous message

<u>At time t = 10</u>

Event Description:

proc(B) receives a message from proc(A).

SAME Action:

A corresponding tree, tree H is created where vertices A = R(H) and B = I C(A). The arc is labeled with time t = 10.

<u>At time t = 100</u>

Event Description:

Proc(C) receives message from proc(B).

SAME Action:

Vertex C is attached adjacent to vertex B, such that C = IC(B). The arc joining vertices B and C is labeled with time t = 100.

<u>At time t = 200</u>

Event Description:

Proc(A) receives a message from proc(B). Since proc(A) did not perform any activities between times t = 10 and t = 200, it is assumed blocked waiting for the reply.

SAME Action:

When proc(A) receives a message from proc(B), it is considered a reply, represented as a dashed arc connecting vertices B and A, to proc(A)'s request sent earlier at time = 10. A synchronous communication pattern between proc(A) and proc(B) is produced.

Since proc(B) did not wait for the reply from proc(C) of the request sent at time t = 100, under the case 2 in chapter 4, an asynchronous message from proc(B) to proc(C) at time t = 100 is produced.

The trace corresponding to the scenario above is given below. The result of SAME1 program is shown in Figure 5.11 and the graphical representation of the LQN network model, which is the output of SAME2 program, is shown in Figure 5.12.

```
 Trace4{
10 procA send %request1
10 procB receive %request1
100 procB send %request2
100 procC receive %request2
200 procB send %complete
200 procA receive %complete
}
```

```
S procA procB 10 200
A procB procC 100
```

Figure 5.11: The output of SAME1 program

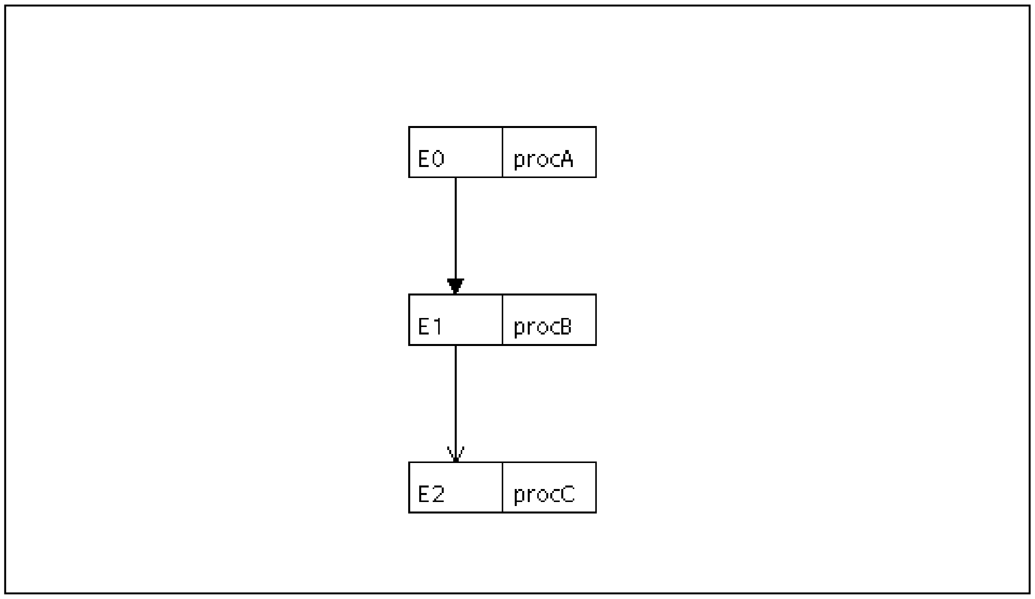


Figure 5.12: The graphical representation of LQN performance model

5.5 Test 5: Two step forwarding pattern

Timeline diagram in Figure 5.13 illustrates four processes, proc(A), proc(B), proc(C), and proc(D) interacting with each other as follows:

<u>At time t = 10</u>

Event Description:

proc(B) receives a message from proc(A).

SAME Action:

Correspondingly, a tree H is created with ver tices A = R(H) and B = IC(A) as illustrated in Figure 5.2.2.

<u>At time t = 100</u>

Event Description:

proc(C) receives a message from proc(B).

SAME Action:

Since there exists a tree with active vertex B, a vertex C, representing a new activation of process proc(C), is attached adjacent to it, such that C = IC(B).

<u>At time t = 150</u>

Event Description:

proc(D) receives a message from proc(C).

SAME Action:

Similar to above, vertex D is attached adjacent to vertex C such that D = IC(C).

<u>At time t = 200</u>

Event Description:

proc(B) receives a message from proc(D). Since proc(B) did not perform any activities between times t = 100 and t = 200, it is considered blocked, waiting for the reply of the message sent earlier at time t = 100 to proc(C).

SAME Action:

Under case 16, a forwarding pattern between processes proc(B), proc(C) and proc(D) is produced. Tree H is transformed such that vertices involved in the forwarding pattern, C and D are removed, except for vertex B. The reason why vertex B is not removed i s because it can still play a role in the detection of a communication pattern involving its parent, vertex A.

<u>At time t = 210</u>

Event Description:

proc(A) receives a message from proc(B). Since proc(A) did not perform any operations between time t = 10 an d t = 210, it is presumed blocked for a reply to the message it had sent earlier at time t = 10.

SAME Action:

In tree H, B = IC(A) and under case 14, a synchronous message is produced between processes proc(A) and proc(B) as illustrated in Figure 5.13. Tree H is no longer beneficial and thus is removed.

The trace corresponding to the scenario is given below. The result of SAME1 program is shown in Figure 5.14 and the graphical representation of the LQN network model, which is the output of SAME2 program, is shown in Figure 5.15.

```
Trace5 {
10 procA send %request1
10 procB receive %request1
100 procB send %request2
100 procC receive %request2
150 procC send %request3
150 procD receive %request3
200 procD send %request4
200 procB receive %request4
210 procB send %complete
210 procA receive %complete
}
```

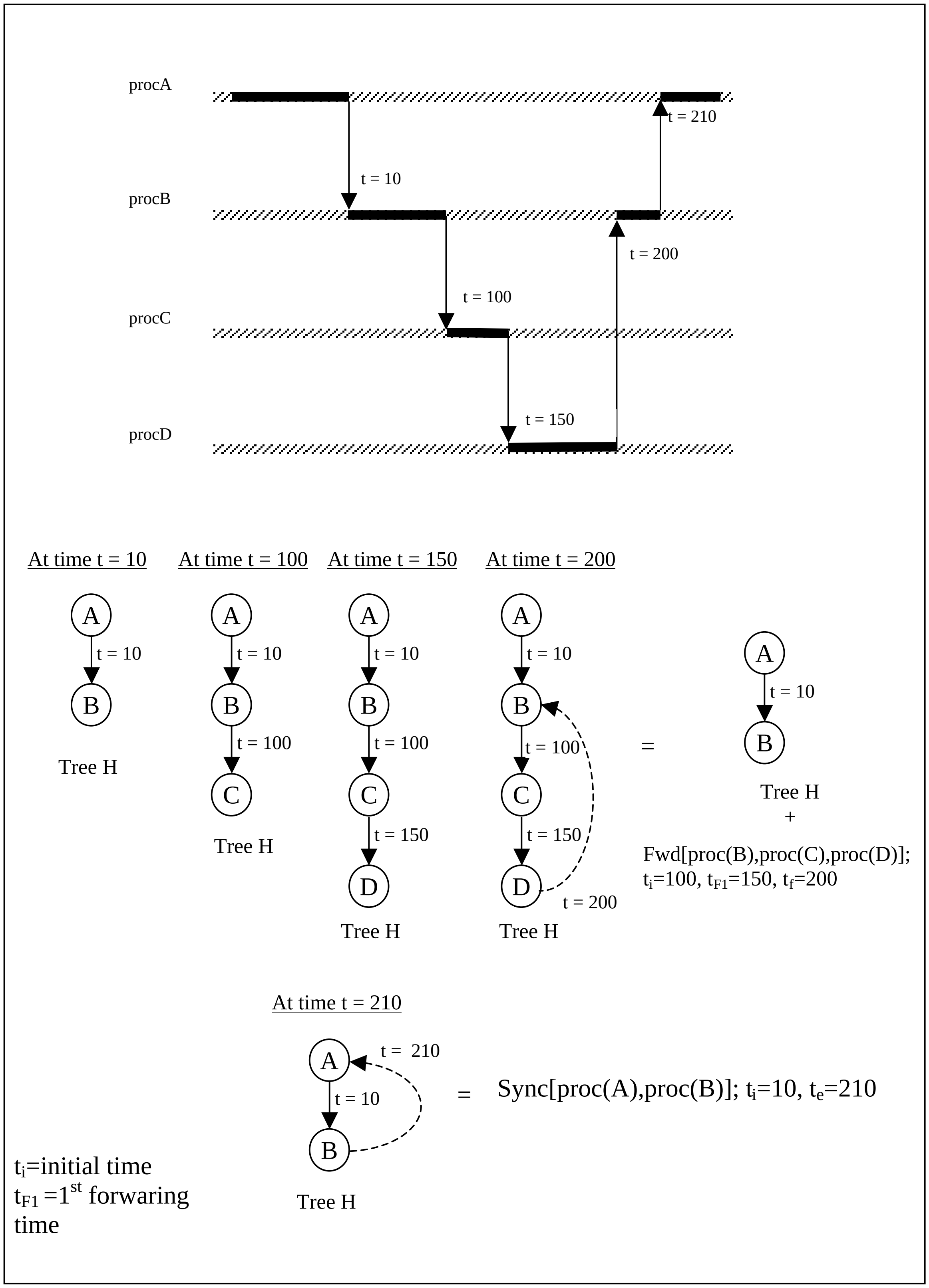


Figure 5.13: A nested forwarding communication pattern

```
F 3 procB procC procD 100 150 200
S procA procB 10 210
```

Figure 5.14: The output of SAME1 program

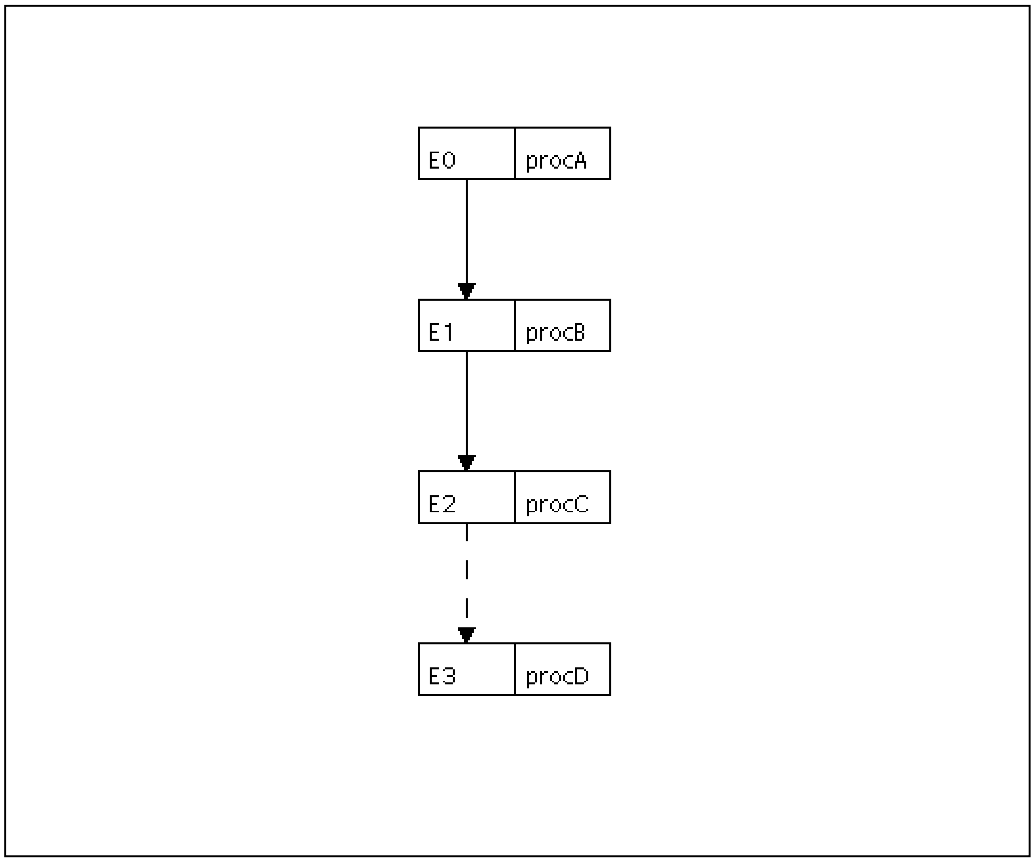


Figure 5.15: The graphical representation of LQN performance model

5.6 Test 6: Two concurrent threads

This section provides an example of a concurrent system and illustrates how SAME technique is applied to concurrent systems. SAME technique processes concurrent systems the same way it processes regular non -concurrent systems, the only difference being that in concurrent system, multiple trees are generated, each for a concurrent behavior. Figure 5.16 illustrates a concurrent system with 5 processes, proc(A), proc (B), proc(C), proc(D) and proc(E). proc(A) and proc(C) starts two simultaneous execution threads, interacting with different processes. SAME technique is applied in the following manner:

At time t = 10

Event Description:

proc(B) receives a message from proc(A).

SAME Action:

A new tree, tree H is generated with vertices A = R(H) and B = IC(A). The arc between them is labeled with time t = 10.

At time t = 20

Event Description:

proc(D) receives a message to proc(C).

SAME Action:

The only tree that e xists at this time is tree H and since there is no vertex C in tree H, a new tree, tree G is generated with C = R(G) and D = IC(C). The arc between the vertices is labeled with time t = 20.

At time t = 100

Event Description:

Proc(E) receives a message from proc(D).

SAME Action:

In tree G, there exists a vertex D and according to case 3 in chapter 4, an activation of proc(E), vertex E is added adjacent to vertex D. The arc between the vertices is labeled with time t = 100.

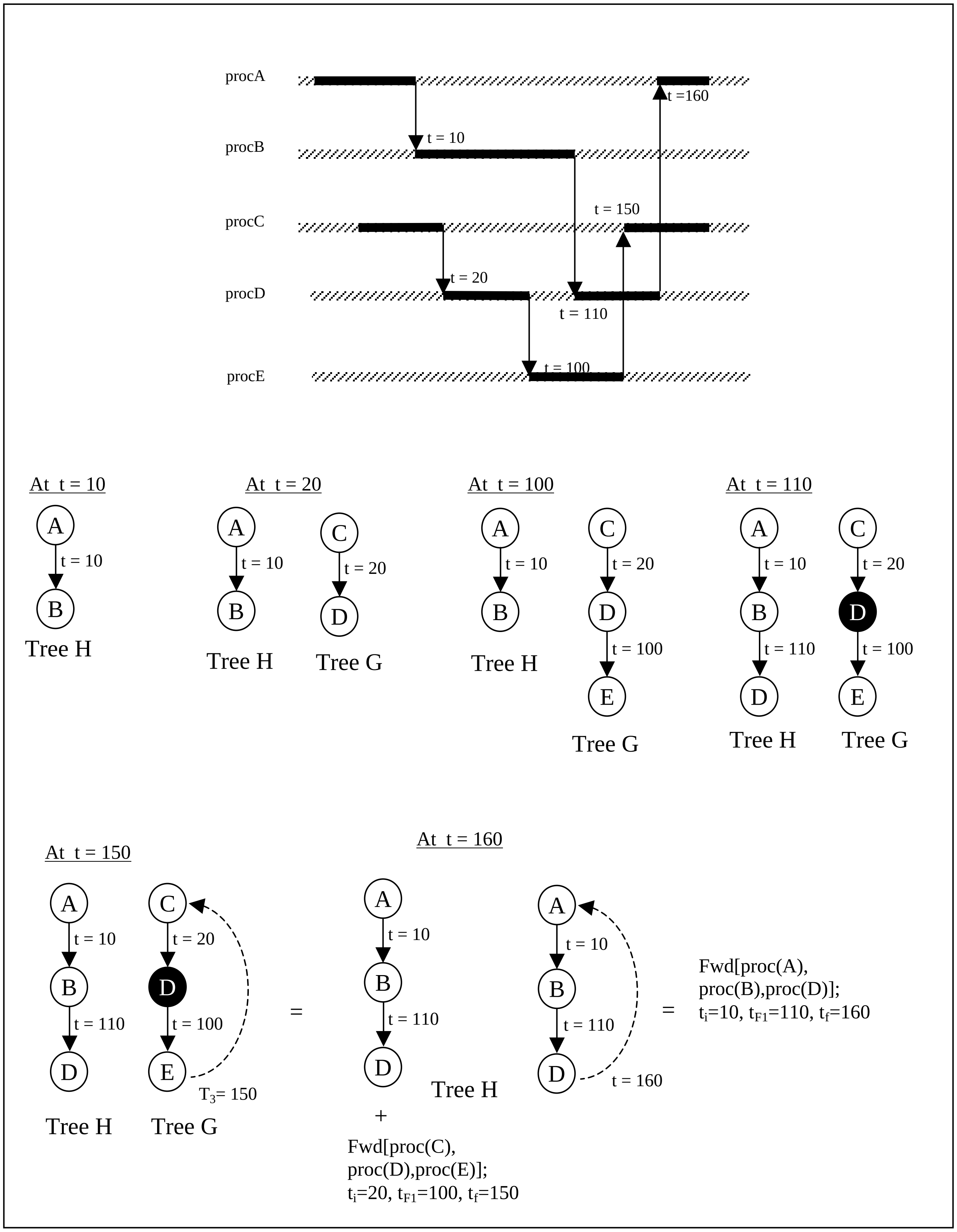


Figure 5.16: An example of a concurrent system

<u>At time t = 110</u>

Event Description:

proc(D) receives a message from proc(B).

SAME Action:

Vertex B is available in tree H where as vertex D is in tree G. According to case 9 in chapter 4, since vertex $D \neq R(G)$ nor $D \notin L\{G\}$, vertex D is made inert. The reason why it is made inert is because, proc(E) may reply to proc(C) completing a forwarding pattern and there should be a way to determine what processes forwarded the request. Keeping an inert vertex D in tree G serves that purpose. Vertex D also becomes a member of PA{proc(D)}. A new active vertex, vertex D is added adjacent to vertex B in tree H.

<u>At time = 150</u>

Event Description:

Proc(C) receives a message from proc(E).

SAME Action:

In tree G, C = R(G) and $E < D < C$. According to case 16 in chapter 4, it is a forwarding pattern. Here keeping an inert vertex D serves the purpose of determining what process has forwarded a message and when. Therefore, a forwarding pattern is produced between proc(C), proc(D) and proc(E), starting at time t = 20, forwarding time t = 100 and the replying time t = 150. Tree G is no longer required and is removed.

<u>At time t = 160</u>

Event Description:

Proc(A) receives a message from proc(D).

SAME Action:

In tr ee H, vertex A = R(H) and $D < B < A$.   According to case 16 in chapter 4, a forwarding pattern is produced between processes proc(A), proc(B) and proc(D), starting at time t = 10, forwarding time t = 110 and the replying time t = 160. Tree H is no longer required and thus is removed.

The trace corresponding to the scenario is given below. The result of SAME1 program is shown in Figure 5.17 and the graphical representation of the LQN network model, which is the output of SAME2 program, is shown in Figure 5.18.

```
Trace6{
10 procA send %request1
10 procB receive %request1
20 procC send %requestc1
20 procD receive %requestc1
100 procD send %requestc2
100 procE receive %requestc2
110 procB send %request2
110 procD receive %request2
150 procE send %completec1
150 procC receive %completec1
160 procD send %complete1
160 procA receive %complete1
}
```

```
F 3 procC procD procE 20 100 150
F 3 procA procB procD 10 110 160
```

Figure 5.17: The output of SAME1 program

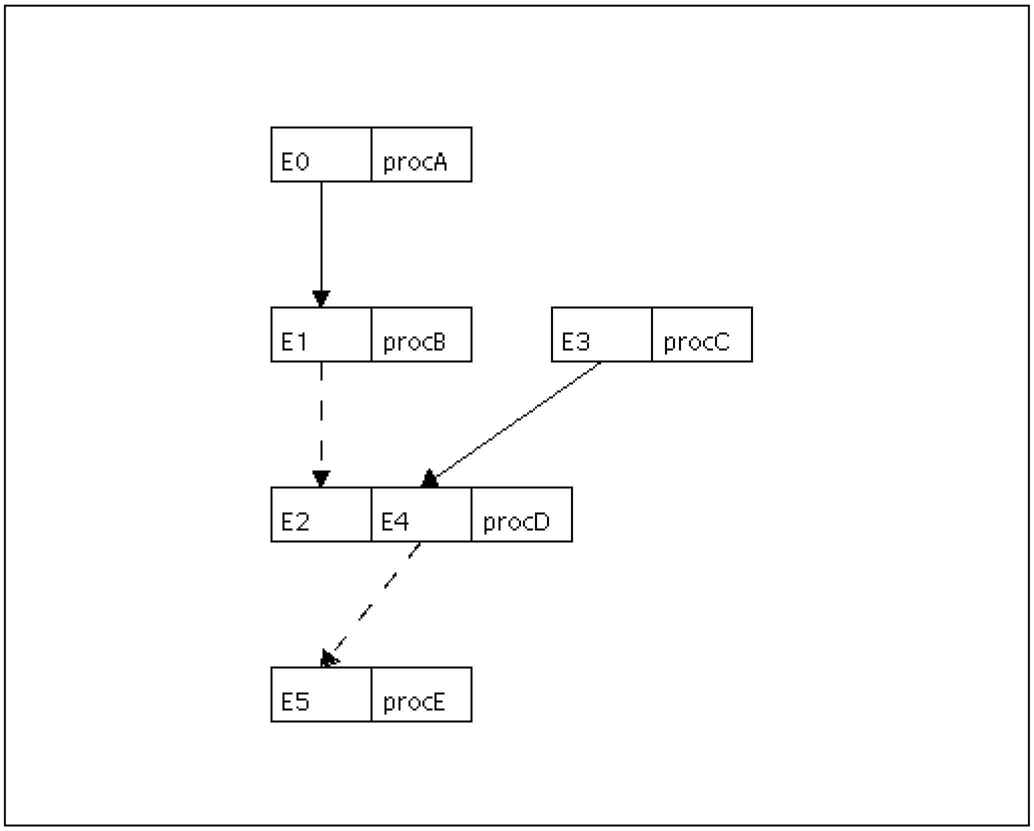


Figure 5.18: The graphical representation of LQN performance model

Chapter 6: Case Study- the ATM-GSM network model

This chapter presents a case study of an ATM-GSM network system. The system is designed using ObjecTime Developer Tool and is provided by ObjecTime Ltd. (now Rational Software Inc.). The ATM-GSM network is chosen for this case study as a typical distributed real-time application in which the driving scenarios are inherently concurrent, and the events initiating use cases come from multiple sources

This chapter first presents the overview of the ATM-GSM network system, after which SAME is applied on it to extract communication patterns and performance models.

6.1 Overview

The chosen ATM-GSM network system model was created by ObjecTime Ltd. (now Rational Software Inc.) as a demonstration of ObjecTime Developer Tool capabilities. It contains four distributed modules, which describe of two cellular phones, an ATM access device and an ATM-GSM network model. The Cellular phone and the ATM access device modules communicate with the ATM-GSM network module through proxies located with in the ATM-GSM network module itself.

The cellular phones are represented through Java-applet GUIs (Graphical User Interface) and have the functionality of basic cellular phones. They can register/un-register themselves with the ATM-GSM network through power on/off button, make calls to each other by dialing each others' distinct phone numbers and wait for incoming calls.

The ATM-GSM network module is the brain of the system. Figure 6.1-a shows an overview diagram of the ATM-GSM network module with top-level modules as designed in ObjecTime Developer Tool and Figure 6.1-b shows the internal structures of modules. The network model has a complicated design with 11 top-level modules or actors, and well over 100 contained actors. Due to the complexity and size of the design, the behavioral aspect of the actors will not be described fully. The 11 top -level actors represent the following:

1. One phone GUI (which is divided into two proxies, one for each cellular phone).
2. Two GSM mobile stations, one for each cellular phone.
3. Two GSM base stations, one for each cellular phone.
4. Two GSM services switching center, one for each cellular phone.
5. An ATM access device
6. An ATM access device proxy (which communicates with the ATM access device on a different node),
7. An ATM network module
8. A GSM registry module.

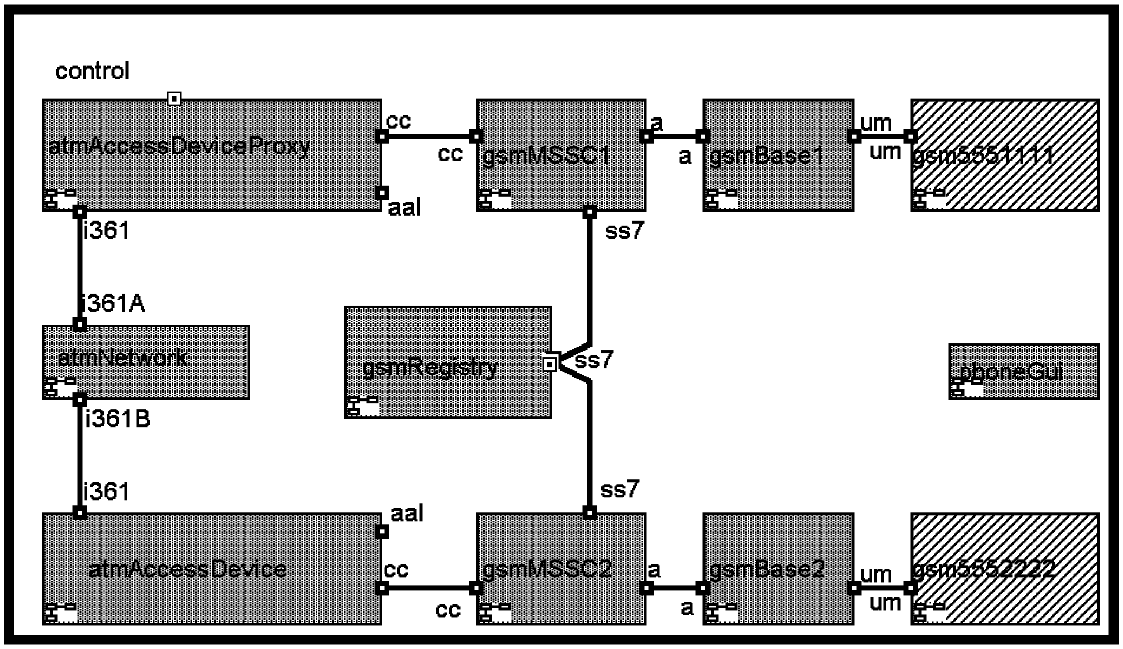


Figure 6.1: The top-level diagram of ATM-GSM network model

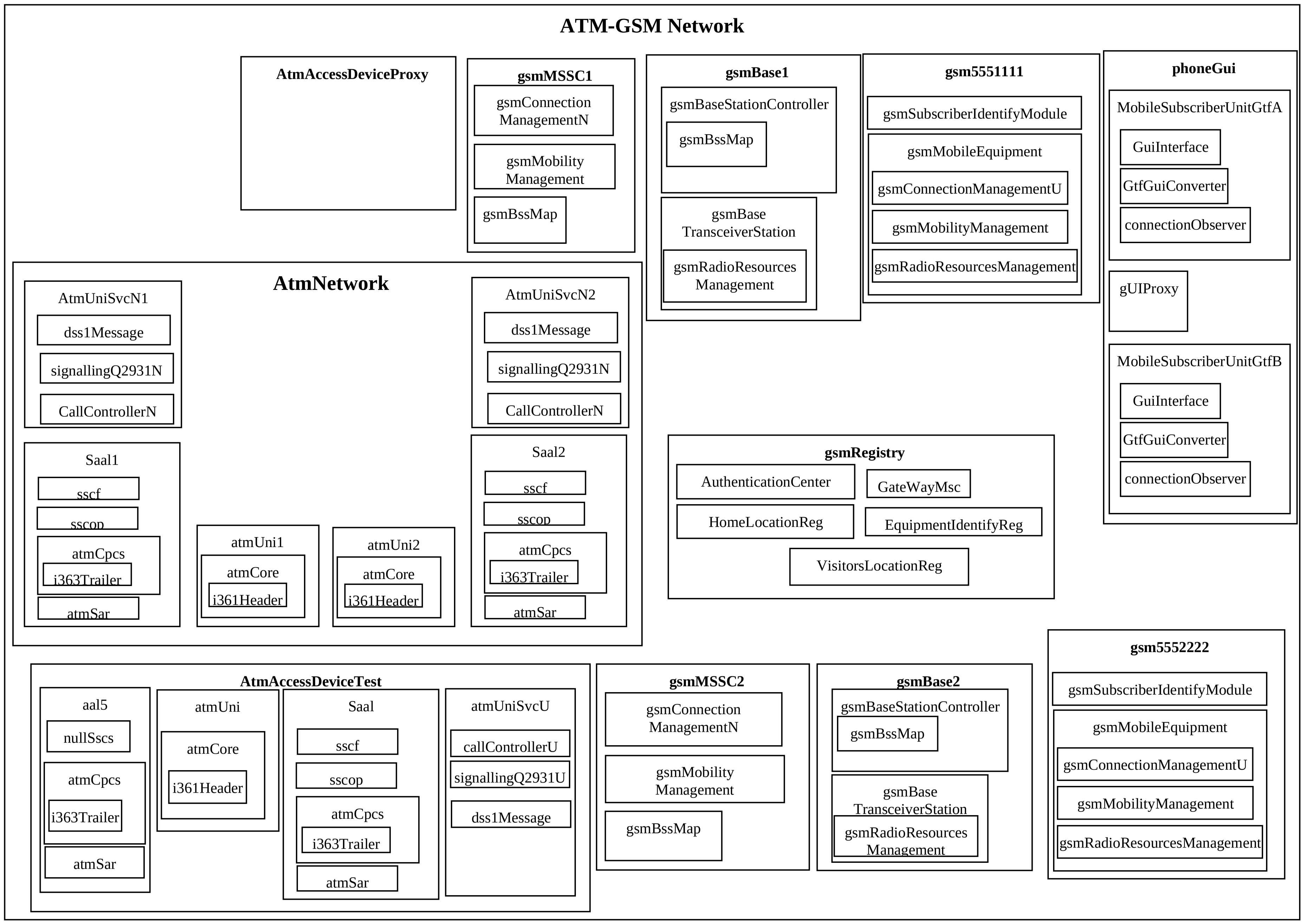


Figure 6.1-b: The internal structure of ATM-GSM network module

6.1.1 The Cellular Phones

The cellular phones are implemented by Java applets as graphical user interface, as shown in Figure 6.2. Each of the cellular phones has a private phone number. They are 555-1111 and 555 -2222. Cellular phones in this study are basic POTs (plain old telephones) and do not have extra features such as call waiting, caller-id and so on. Their basic functions include the following:

1. Power up
2. Dialing a number
3. Sending a call request
4. Accepting a call request
5. Clear display screen.
6. Power down

The cellular phones are connected to the "phoneGui" module in ATM-GSM network model via communication port 2711 .

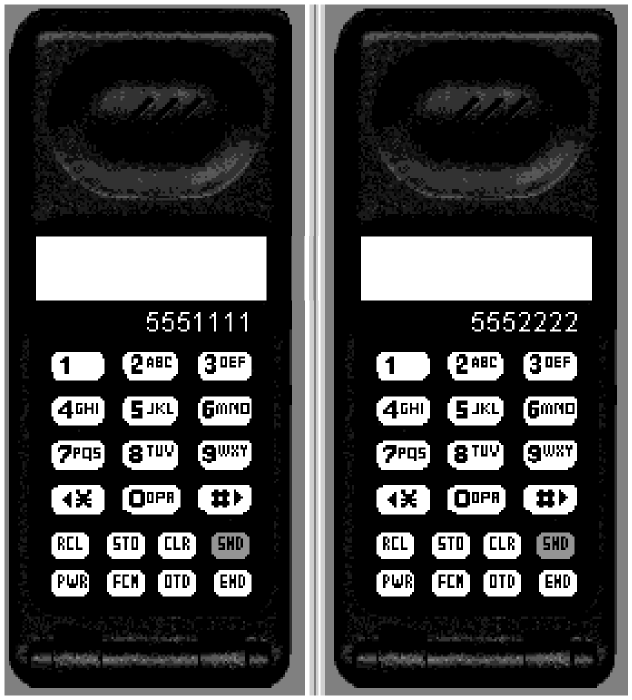


Figure 6.2 The cellular phones in ATM-GSM network system.

6.1.2 PhoneGui Module

The PhoneGui actor is connected to cellular phones implemented in Java applet GUI. PhoneGui acts as a proxy for comm unication between cellular phones and ATM -GSM network module. PhoneGui Module has three sub-modules as follows:

1. GUIProxy
2. MobileSubscriberUnitGtAA
    a. GUInterface
    b. GtfGuiConverter
    c. ConnectionObserver
3. MobileSubscriberUnitGtFB
    a. GUInterface
    b. GtfGuiConverter
    c. ConnectionObserver

GUInterface, GtfGuiConverter and Connection Observer are sub -modules of MobileSubscriberUnitGtfA and MobileSubscriberUnitGtfB. Figure 6.3 shows the internal structure of phoneGui actor with three contained actors.

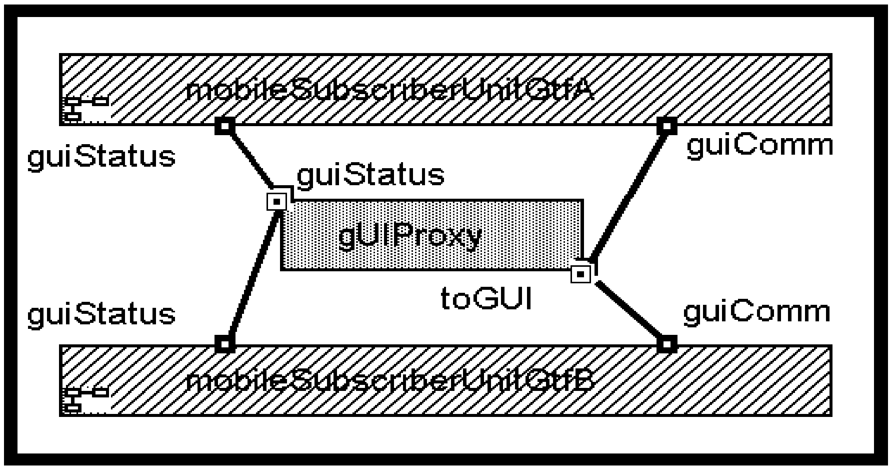


Figure 6.3: Internal Structure of phoneGui Actor

6.1.3 GSM5551111 & GSM5552222 Modules

Actors "GSM5551111" and "GSM5552222" represents GSM mobile stations, one for each cellular phone. They are the front -ends of the ATM -GSM network system. These modules are identical in structure and behavior. They are responsible for managing GSM connection, mobility, radio resources and identifying subscriber. Each contains two sub - modules as follows:

1. GSMSubscriberIdentifyModule
2. GSMMobileEquipment
    a. GSMConnectionManagementU
    b. GSMMobilityManagement
    c. GSMRadioResourcesManagement

Figure 6.4 shows the internal structure of GSM mobile stations (GSM5551111 and GSM5552222).

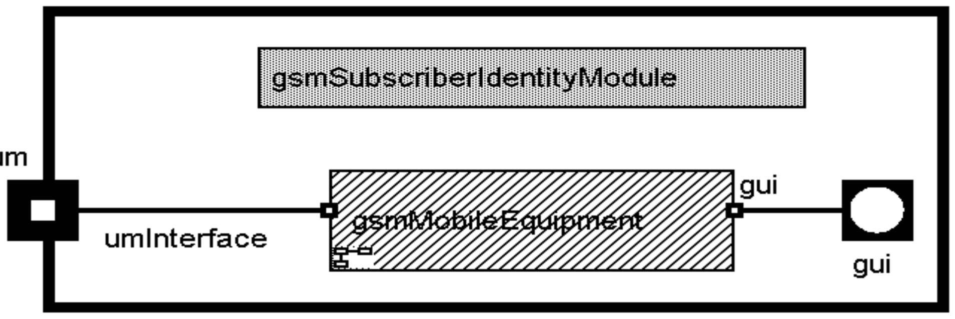


Figure 6.4: Internal Structure of GSM mobile Station module

6.1.4 GSMBase1 & GSMBase2 Modules

GSMBase1 and GSMBase2 represent th e GSM base stations, and are identical in behavior and structure. Their responsibilities are to act as GSM base station controllers and GSM transceiver stations. They have the following sub modules.

1. GSMBaseStationController

        a. GSMBssMap
2. GSMBaseTransceiverStation
        a. GSMRadioResources Management

Figure 6.5 shows the internal structure of GSMBase1 module.

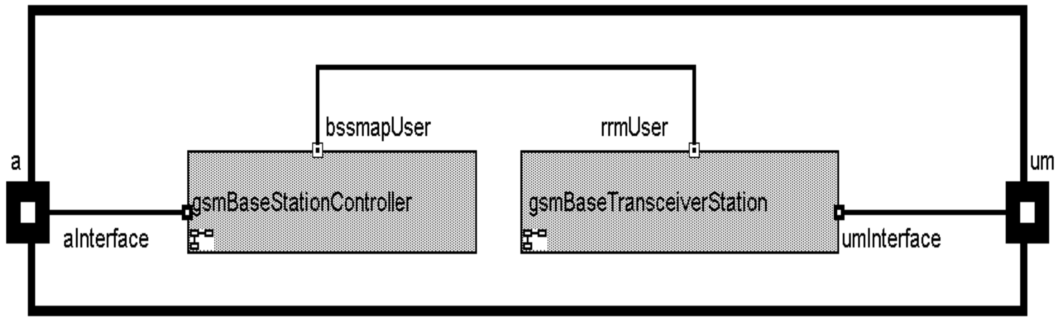


Figure 6.5: Internal Structure of GSM Base Station module

6.1.5 GSMMSSC1 and GSMMSSC2 modules

GSMMSSC1 and GSMMSSC2 modules are identical in behavior and structure. They represent GSM mobility services switching centers and have the following sub-modules.

1. GSMConnectionManagementN
2. GSMMobilityManagement
3. GSMBaseMap

Figure 6.6 shows the internal structure of the GSMMSSC1 which is also the same for GSMMSSC2.

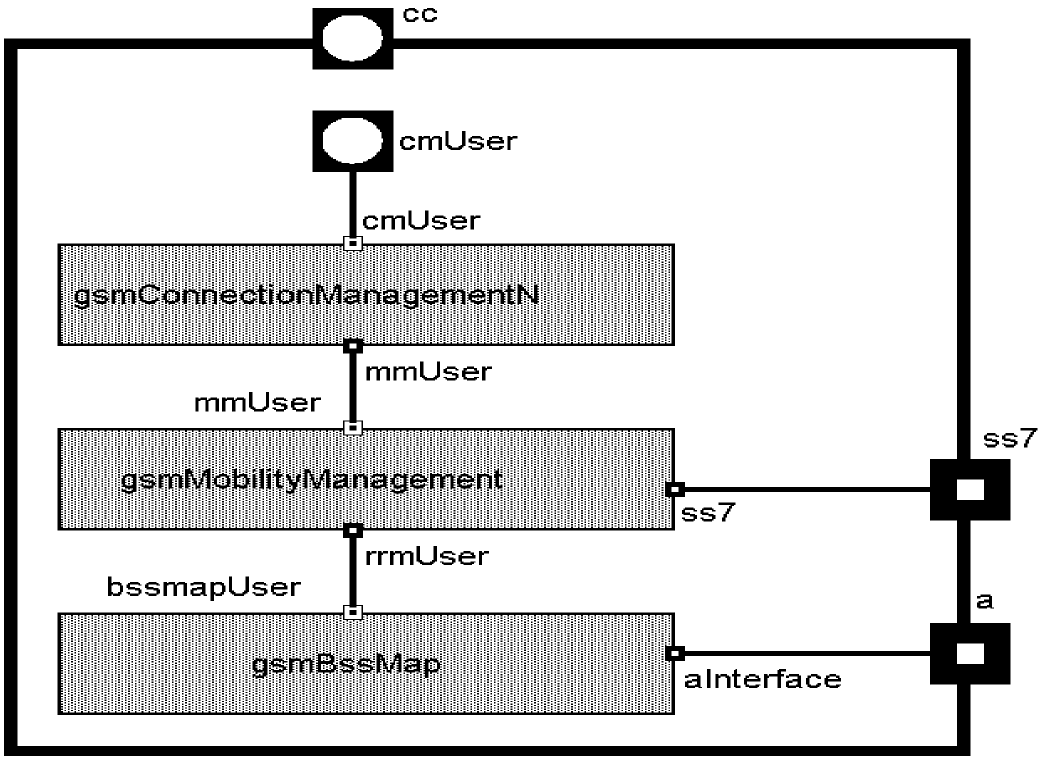


Figure 6.6: Internal Structure of GsmMSSC1 module

6.1.6 GSMRegistry Module

GSMRegistry module represents a registration center for the GSM cellular phones. It serves as a central station with responsibilities of authenticating, identifying and registering local and visitor cellular phones. It has the following sub-modules.

1. AuthenticationCenter
2. GateWayMsc
3. HomeLocationReg
4. VisitorsLocationReg

Figure 6.7 shows the internal structure of GSMRegistry module.

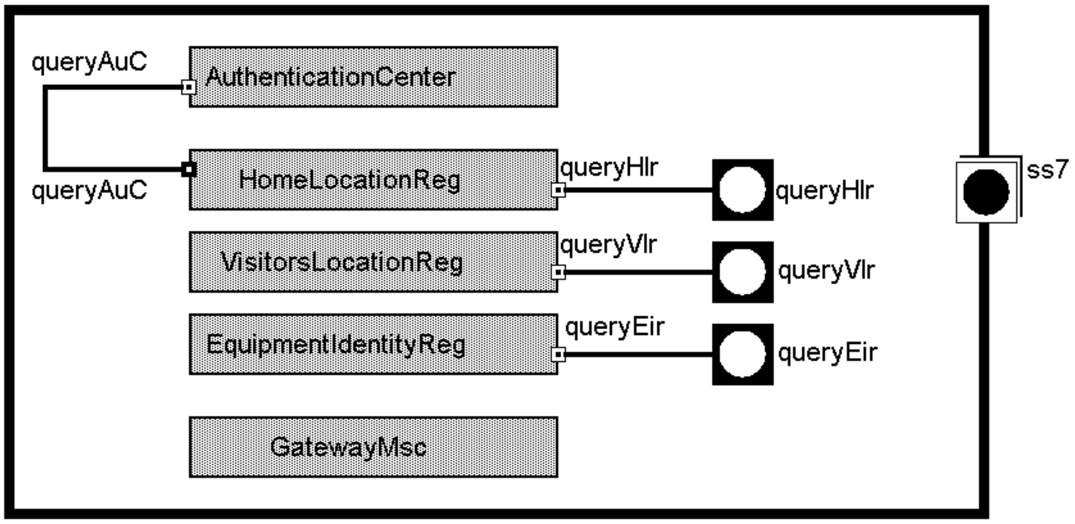


Figure 6.7: Internal Structure of gsmRegitries module

6.1.7 ATMAccessDeviceProxy Module

ATMAccessDeviceProxy Module is located on the GSM5551111 side of the model as shown in Figure 6.1. It serves as a proxy for ATMAccessDeive module located on a different node. Figure 6.8 shows the internal structure of ATMAccessDeviceProxy.

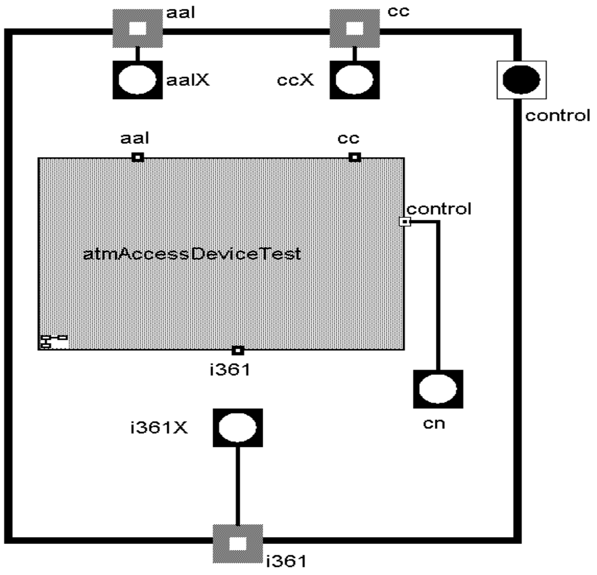


Figure 6.8: Internal Structure of atmAccessDeviceProxy module

6.1.8 ATMAccessDevice Modules

There are two AtmAccessDevice modules in this ATM-GSM model. One is located on a different node, which is accessed through AtmAccessDeviceProxy and the other in the same model on GSM5552222 side of the model. They are identical in structure and behavior and serves as a front end to ATM network. Following are the sub-modules located in AtmAccessDevice modules and Figure 6.9 shows the internal structure of AtmAccessDevice module.

1. aal5
    a. nullSscs
    b. atmCpcs
        i. i363Trailer
        ii. atmSar
2. atmUni
    a. atmCore
    b. i361Header
3. saal
    a. sscf
    b. sscop
    c. atmCpcs
        i. i363Trailer
    d. atmSar
4. atmUniSvcU
    a. callControllerU
    b. signallingQ2931U
    c. dss1Message

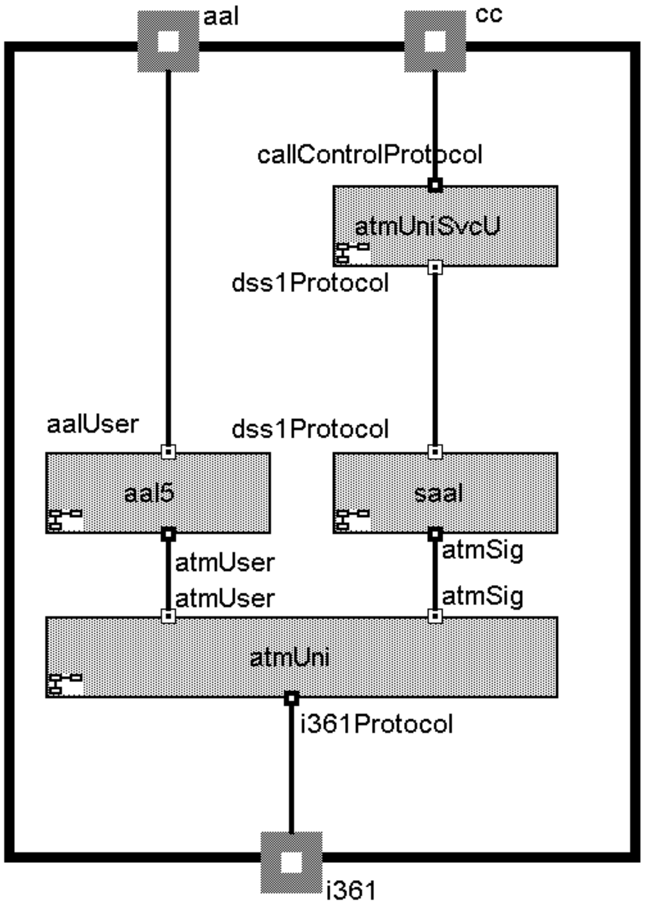


Figure 6.9: Internal Structure of atmAccessDevice module

6.1.9 AtmNetwork Module

AtmNetwork module acts as a central switching station for the ATM network. To represents modules or actors involved for each cellular phone, it is designed symmetrically. Figure 6.10 shows the internal s tructure of the AtmNetwork module. Following are sub-modules of AtmNetwork module.

1. AtmUniScvN1
    a. Dss1Message
    b. SignallingQ2931N
    c. CallControllerN
2. Saal1

        a. Sscf
        b. Sscop
        c. AtmCpcs
            i. I363Trailer
        d. AtmSar
3. atmUni1
    a. atmCore
        i. i361Header
4. AtmUniScvN2
    a. Dss1Message
    b. SignallingQ2931N
    c. CallControllerN
5. Saal2
    a. Sscf
    b. Sscop
    c. AtmCpcs
        i. I363Trailer
    d. AtmSar
6. atmUni2
    a. atmCore
        i. i361Header

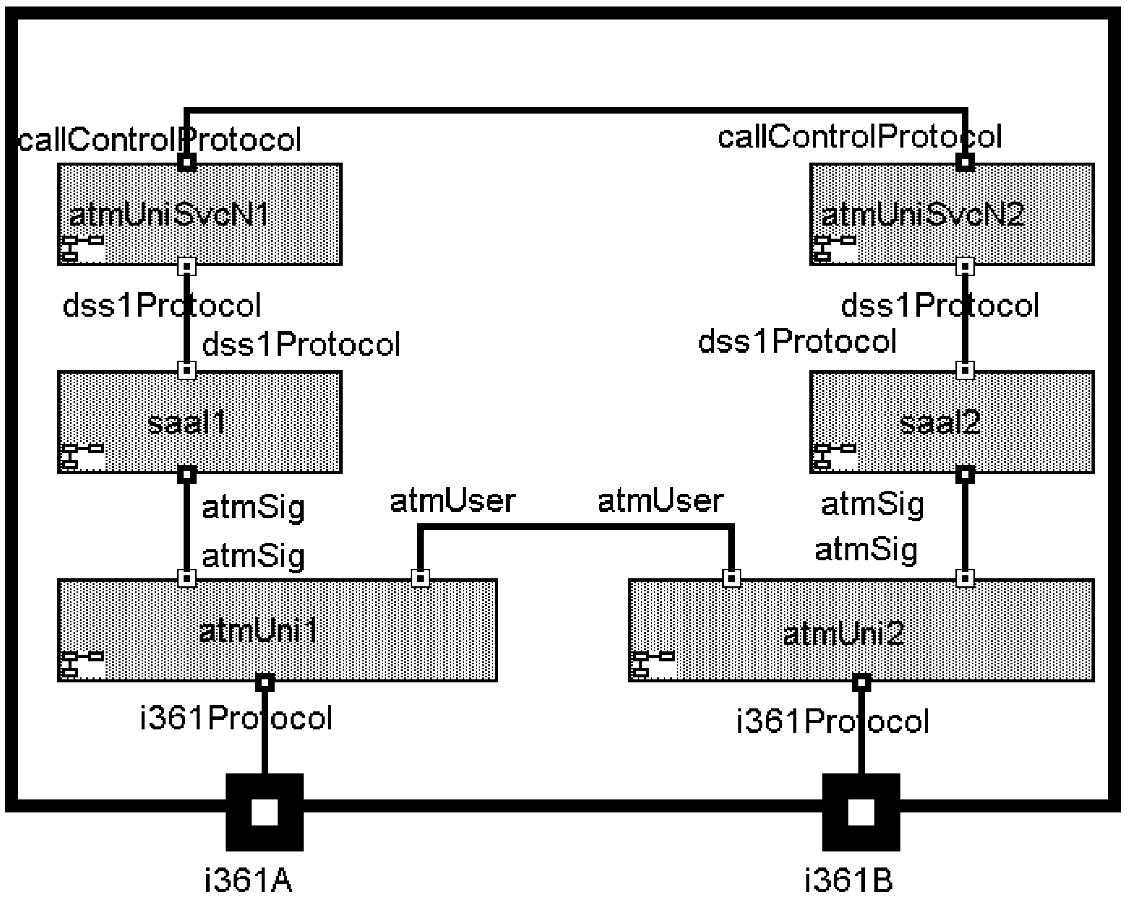


Figure 6.10: Internal Structure of atmNetwork module

6.2 The Scenarios

This section describes the extraction of communication patterns and performance models using SAME technique with ATM-GSM network model. The scenarios are derived from use cases based on functionality of cellular phones. In this study five scenarios are chosen as follows:

1. Phone 555-2222 turns on through power on/off button.
2. Phone 555-2222 makes a call to phone 555-1111
3. Phone 555-2222 accepts an incoming call from 555-1111
4. Phone 555-2222 terminates the call.
5. Phone 555-2222 turns off through power on/off button.

Each scenario is processed separately with SAME technique to extract communication patterns and performance models.

6.2.1 Switching Power to ON

In this scenario, cellular phone 555-2222 switches its power on. GtfGuiConverter informs this action to GSMMobileStation, which informs GSMBaseStation. GsmBaseStation further informs to GSMMobility ServiceSwitchingcenter. From there the signal goes to GSMRegistry, which registers the phone. Figure 6.11 illustrates a message sequence chart of the scenario. The trace of the scenario can be found in Appendix A1. SAME technique is applied to the trace and communication patterns are obtained, which are shown in Appendix A2. The communication patterns are processed with 2nd Pass technique to generate a LQN performance model. The LQN performance model in text form is in Appendix A3, and graphical version is displayed in Appendix A4.

6.2.2 Requesting a Phone Call

In this scenario, cellular phone 555-2222 makes a phone call to phone 555-1111. The scenario is up to the point where phone 555-1111 is informed of an incoming call but it has not yet picked up so the phone is ringing. This scenario contains over 53 actors and due to the complexity and size of the scenario and distributed modules involved, ObjecTime Developer tool is unable to create a message sequence chart. However an execution trace is included in Appendix B1. The trace is processed with the SAME technique and the communication patterns are extracted, as shown in Appendix B2. The communication patterns are processed with 2nd Pass technique and a LQN performance model is extracted. The model in textual form is in Appendix B3 and graphically displayed in Appendix B4.

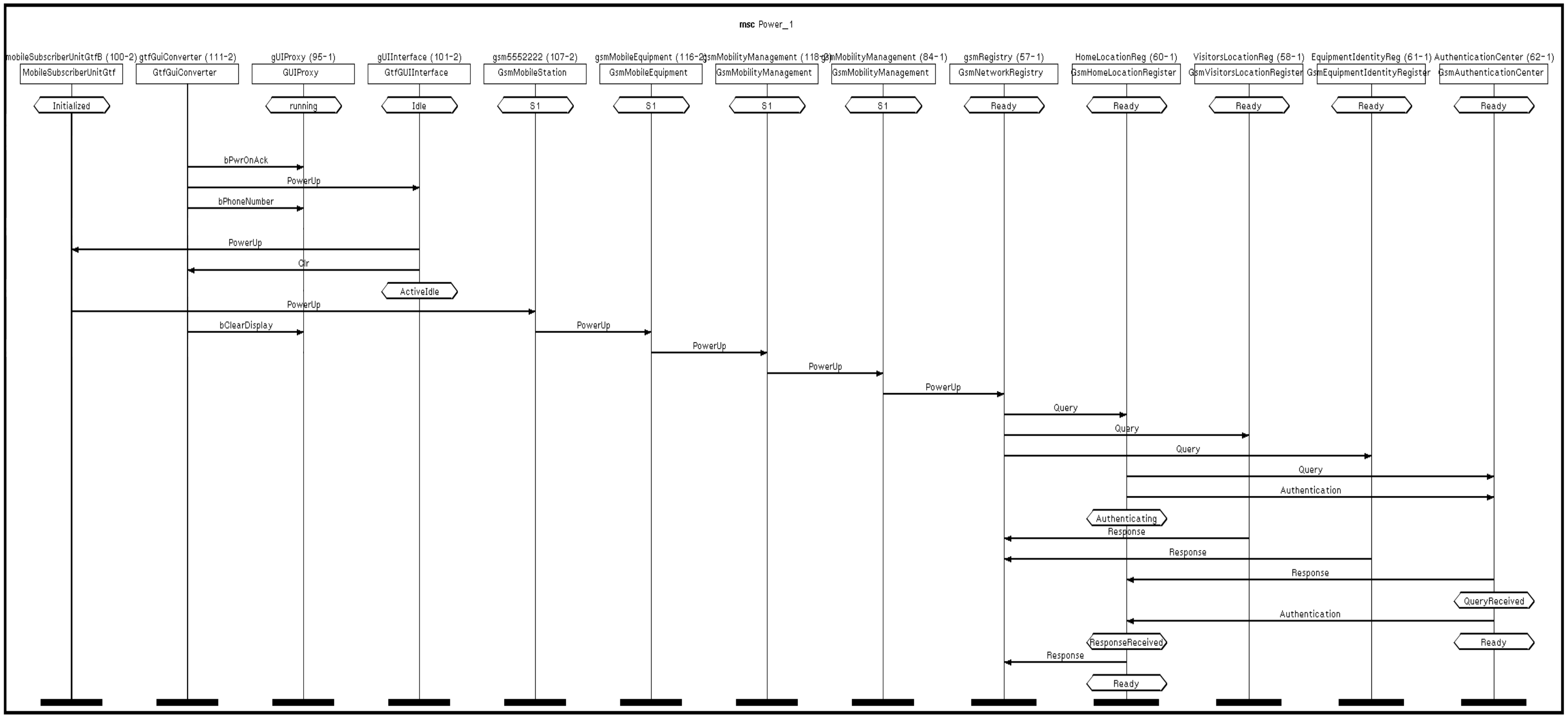


Figure 6.11: The message sequence chart of powerUp scenario

6.2.3 Accepting a Phone Call

In this scenario, cellular phone 555 -2222 accepts a phone call from phon e 555-1111 and a phone line is connected between them. Again due to the complexity and size of the scenario and distributed modules involved, ObjecTime Developer tool is unable to create a message sequence chart. However an execution trace is available in Appendix C1. The trace is processed with SAME technique and the communication patterns are extracted, as shown in Appendix C2. The communication patterns are processed with 2 nd Pass technique and a LQN performance model is extracted. The model in te xtual form is in Appendix C3 and graphically displayed in Appendix C4.

6.2.4 Termination of a Phone Call

In this scenario, cellular phone 555 -2222 terminates a phone call. An execution trace of the scenario is available in Appendix D1. The trace is pr ocessed with SAME technique and communication patterns are extracted, as shown in Appendix D2. The communication patterns are processed with 2 nd Pass technique and a LQN performance model is extracted. The model in textual form is in Appendix D3 and grap hically displayed in Appendix D4.

6.2.5 Switching Power Off

In this scenario, cellular phone 555 -2222 switches its power off. GtfGuiConverter informs this action to GSMMobileStation, which informs GSMBaseStation. GsmBaseStation further informs to GSMMo bilityServiceSwitchingcenter. From there the signal goes to GSMRegistry, which deregisters the phone. The trace of the scenario can be found in Appendix E1. SAME technique is applied to the trace and communication patterns are obtained, which are shown in Appendix E2. The communication patterns are processed with 2 nd Pass technique to generate a LQN performance model. The LQN performance model in text form is in Appendix E3 and graphically displayed in Appendix E4.

6.3 Conclusion

The SAME tools scale up to industrial-size traces. The trace analysis algorithm handled large traces with 600 events and large sub-trees up-to 250 nodes correctly.

Syntactically and semantically correct LQN models are produced. The models produced by the five scenarios were successfully parsed and executed by the LQN simulator ("parasrvn") and gave performance results. In scenario 3, a trivial addition had to be made to the model, to add a driver task.

Chapter 7: Conclusion

This chapter summarizes the work performed in this thesis and presents the main areas, which would require additional research and work to be done.

7.1 Contributions

There are several contributions of this research, which are as follows:

7.1.1. A complete algorithm for the detection and extraction of the communication patterns.

SAME technique extracts three different communication patterns, which are extremely important from the point of view of performance engineering and can be found in almost all message communication based systems. These patterns are as follows:

1. Synchronous communication pattern
2. Asynchronous communication pattern
3. Forwarding communication pattern.

The technique is based on tree transformations to recognize and extract communication patterns. Tree transformation is a dynamic w ay of extracting communication patterns; it processes events as they occur, hence it is fast and efficient.

7.1.2 An algorithm for development of LQN performance model from communication patterns.

The 2nd pass algorithm of the SAME technique generates LQN performance models.

7.1.3 The validation of the algorithm

The algorithms of the SAME technique are verified in chapter 5, through step-by-step process.

7.1.4 The development of general trace-based model building technique

There is some previous research done in the automated extraction of performance models from an execution trace. The closest study is done by Curtis Hrischuk, where performance models are created through a template matching technique, called TLC. It was based on a specific kind of trace, an angio-trace where each thread of execution was given a specific identification. This is no longer the case in this study. The SAME technique processes any general trace, as long as tasks have unique identifiers, and events recorded in trace are in chronological order or have timestamps.

Furthermore, the previous research was limited to extracting performance models from one scenario at a time. This is also no longer the case. SAME technique can process and extract communication patterns and performance models from a trace containing different scenarios.

7.1.5 The analysis of substantial telephone case study

The SAME technique is applied to a distributed ATM-GSM network model, a complicated design built in ObjecTime Developer tool. Syntactically and semantically correct LQN models are produced. The models produced by the five scenarios were successfully parsed and executed by the LQN simulator ("parasrvn") and gave performance results.

7.1.6 The implementation of the SAME algorithms

The SAME technique is implemented in Java programming language as two programs, SAME1 and SAME2. SAME1 program extracts communication patterns from a trace through tree transformation technique and SAME2 program extract LQN performance model from the trace. The SAME tools scale up to industrial-size traces. The trace analysis algorithm handled large traces with 600 events and large sub-trees up-to 250 nodes correctly.

7.2 Limitations

There are several main points to highlight in this section:

- SAME technique requires that the events should be in chronological order with timestamps. This requires the notion of a global clock or similar which may pose problems in the case of obtaining correct traces in heterogeneous distributed systems.
- In the case where message signals do not have unique identification and it is not certain which task receives which message, SAME assumes FIFO queue for messages.
- SAME technique does not tackle the notion of thread join and further research is required in this area.

References

[Buhr96] R.J.A Buhr and R.S. Casselman, "Use Case Maps for object-oriented systems". Prentice Hall, 1996.

[El-Sayed99] Hesham M. El-Sayed, "A Framework For Automated Performance Engineering of Distributed Real-Time Systems", PHD Thesis, Dept. of Systems and computer engineering, Carleton University, 1999.

[Fenton97] Fenton, N.E. and S.L. Pfleeger. "Software Metrics: A Rigorous and Practical Approach". PWS Publishing Company, 1997.

[Gentleman81] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays", SPIE Real- Time Signal Processing IV 298 1981.

[Hayes-Roth78] F. Hayes-Roth and D. Waterman. "Principles of pattern-directed inference systems". In D. Waterman and F. Hayes-Roth, editors, Pattern-Distributed Inference Systems, pages 577-601. Academic Press, 1978.

[Hrischuk 99a] C. Hrischuk, C.M. Woodside, J. Rolia and R. Iversen, "Trace-based load characterization for generaling software performance models", IEEE Transactions on Software Engineering, vol. 25, no. 1, January 1999.

[Hrischuk95] C. Hrischuk, J. Rolia, C.M. Woodside, "Automated generation of software performance model using an object-oriented prototype", International Workshop on Modeling and Simulation, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '95), pp. 399-409, Durham, NC, 1995.

[Hrischuk95a] Curtis Hrischukm “The automatic generation of software performance models using prototypes”. Master’s Thesis, Dept. of Systems and computer engineering, Carleton University, 1995.

[Jain91] R. Jain. “The Art of Computer Systems Performance Analysis”, John Wiley & Sons, 1991

[Liu73] C.L.Liu and J.W. Layland. “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment” J. ACM, 20, pp. 40-61.

[Menace98] D. Menasce and H. Gomaa, “On a language based method for software performance engineering of client/server systems”. Proceedings of the First International Workshop on Software and Performance WOSP 98, Santa Fe, New Mexico, October 12-16, 1998.

[Menace99] D. Menasce and H. Gomaa, “A method for design and performance modeling of client/server systems”. to appear in the IEEE Transactions on Software Engineering.

[Narain96] R. Narain, D. Rimel and P. Fingar, “Asynchronous message communication between distributed business objects”, http://www.trcinc.com/

[Neilson95] J.E. Neilson, C.M.Woodside, D.C. Petriu, S.Majumdar "Software Bottlenecking in Client-Server Systems and Rendezvous Networks," *IEEE Transactions on Software Engineering,* Vol. 21, No. 9, pp.776-782, Sept. 1995.

[Rational97] Rational Software Corporation, “A Rational Approach to Software Development Using Rational Rose 4.0”, “http://www.rational.com/products/whitepapers/293.jsp”

[Rolia95] J. R. Rolia and Kenneth Sevcik, “The method of layers”, IEEE Transactions on Software Engineering, Vol. 21, No. 8, pp. 689-700, 1995.

[schwarz94] R. Schwarz and F. Mattern. “Detecting casual relationshrips in distributed computations: in search of the Holy Grail. Distributed Computing”, 7(3):149-174, 1994.

[Scratchley99] W.C. Scratchley and C.M. Woodside, “Evaluating concurrency options in software specifications”, Submitted to MASCOTS 99.

[Smith90] C.U. Smith, “Performance Engineering of Software Systems”, Addison-Wesley Publishing Co., New York, NY, 1990.

[Strosnider96] Jay K. Strosnider, “Performance Engineering Real-Time/Multimedia Systems”, “http://www.ece.cmu.edu/afs/ece/usr/jks/web/home.html”

[Waheed95] A. Waheed, V. Mehfi, and D.T. Rover. “A model for instrumentation system management in concurrent systems.” Proceedings of the Twenty Eigth Hawaii International Conference on Systems Sciences, January 1995.

[Woodside89] C.M. Woodside, “Throughput calculation for basic stochastic rendezvous networks”, Performance Evaluation, Vol. 9, No. 2, pp. 143-160, 1989.

[Woodside95] C.M. Woodside and G. Ragunath, "General Bypass Architecture for High-Performance Distributed Algorithms", *Proc. 6th IFIP Conference on Performance of Computer Networks*, Istanbul, Oct. 23-26, 1995, in "Data Communications and their Performance", eds. S. Fdida and R.U. Onvural, Chapman and Hall, 1996, pp 51-65.

APPENDIX A1- PowerOn: sequence Trace

```
thread Power.1 {
760209 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%bPwrOnAck;
760209 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) spawn
Power.1.1;
760214 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%PowerUp;
760214 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) spawn
Power.1.2;
};
thread Power.1.1 {
760211 GUIProxy:phoneGui/gUIProxy(95-1) state running;
760211 GUIProxy:phoneGui/gUIProxy(95-1) receive %bPwrOnAck;
};
thread Power.1.2 {
760214 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state Idle;
760214 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) receive
%PowerUp;
760230 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send
%PowerUp;
760230 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
Power.1.2.1;
760234 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send %Clr;
760234 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
Power.1.2.2;
760239 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/ActiveIdle;
};
thread Power.1.2.1 {
760230 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) receive %PowerUp;
760251 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) send %PowerUp;
760253 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) state
Initialized;
760251 GsmMobileStation:gsm5552222(107-2) state S1;
760251 GsmMobileStation:gsm5552222(107-2) receive %PowerUp;
760259 GsmMobileStation:gsm5552222(107-2) send %PowerUp;
760260 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) state S1;
760260 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) receive %PowerUp;
760272 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) send %PowerUp;
760272 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
state S1;
760272 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
receive %PowerUp;
760276 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
send %PowerUp;
760276 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) state S1;
760276 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) receive %PowerUp;
760280 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) send %PowerUp;
760280 GsmNetworkRegistry:gsmRegistry(57-1) state Ready;
760280 GsmNetworkRegistry:gsmRegistry(57-1) receive %PowerUp;
760286 GsmNetworkRegistry:gsmRegistry(57-1) send %Query;
760286 GsmNetworkRegistry:gsmRegistry(57-1) spawn Power.1.2.1.1;
760287 GsmNetworkRegistry:gsmRegistry(57-1) send %Query;
760287 GsmNetworkRegistry:gsmRegistry(57-1) spawn Power.1.2.1.2;
760289 GsmNetworkRegistry:gsmRegistry(57-1) send %Query;
760289 GsmNetworkRegistry:gsmRegistry(57-1) spawn Power.1.2.1.3;
};
thread Power.1.2.1.1 {
760286 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) state Ready;
760286 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) receive %Query;
760293 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) send %Query;
760293 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) spawn Power.1.2.1.1.1;
760296 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) send %Authentication;
760296 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) spawn Power.1.2.1.1.2;
760297 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) state Authenticating;
};
```

```
thread Power.1.2.1.1.1 {
760293 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) state Ready;
760293 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) receive %Query;
760306 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) send %Response;
760307 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) state
QueryReceived;
760306 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) receive %Response;
760313 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) state ResponseReceived;
};
thread Power.1.2.1.1.2 {
760296 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) receive
%Authentication;
760309 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) send
%Authentication;
760310 GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) state Ready;
760309 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) receive %Authentication;
760315 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) send %Response;
760316 GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) state Ready;
760315 GsmNetworkRegistry:gsmRegistry(57-1) receive %Response;
760317 GsmNetworkRegistry:gsmRegistry(57-1) state Ready;
};
thread Power.1.2.1.2 {
760287 GsmVisitorsLocationRegister:gsmRegistry/VisitorsLocationReg(58-1) state Ready;
760287 GsmVisitorsLocationRegister:gsmRegistry/VisitorsLocationReg(58-1) receive %Query;
760299 GsmVisitorsLocationRegister:gsmRegistry/VisitorsLocationReg(58-1) send %Response;
760299 GsmNetworkRegistry:gsmRegistry(57-1) receive %Response;
760311 GsmNetworkRegistry:gsmRegistry(57-1) state Ready;
};
thread Power.1.2.1.3 {
760289 GsmEquipmentIdentityRegister:gsmRegistry/EquipmentIdentityReg(61-1) state Ready;
760289 GsmEquipmentIdentityRegister:gsmRegistry/EquipmentIdentityReg(61-1) receive
%Query;
760303 GsmEquipmentIdentityRegister:gsmRegistry/EquipmentIdentityReg(61-1) send
%Response;
760303 GsmNetworkRegistry:gsmRegistry(57-1) receive %Response;
760312 GsmNetworkRegistry:gsmRegistry(57-1) state Ready;
};
thread Power.1.2.2 {
760235 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) receive
%Clr;
};
```

APPENDIX A2- PowerOnsequence: Communication Patterns

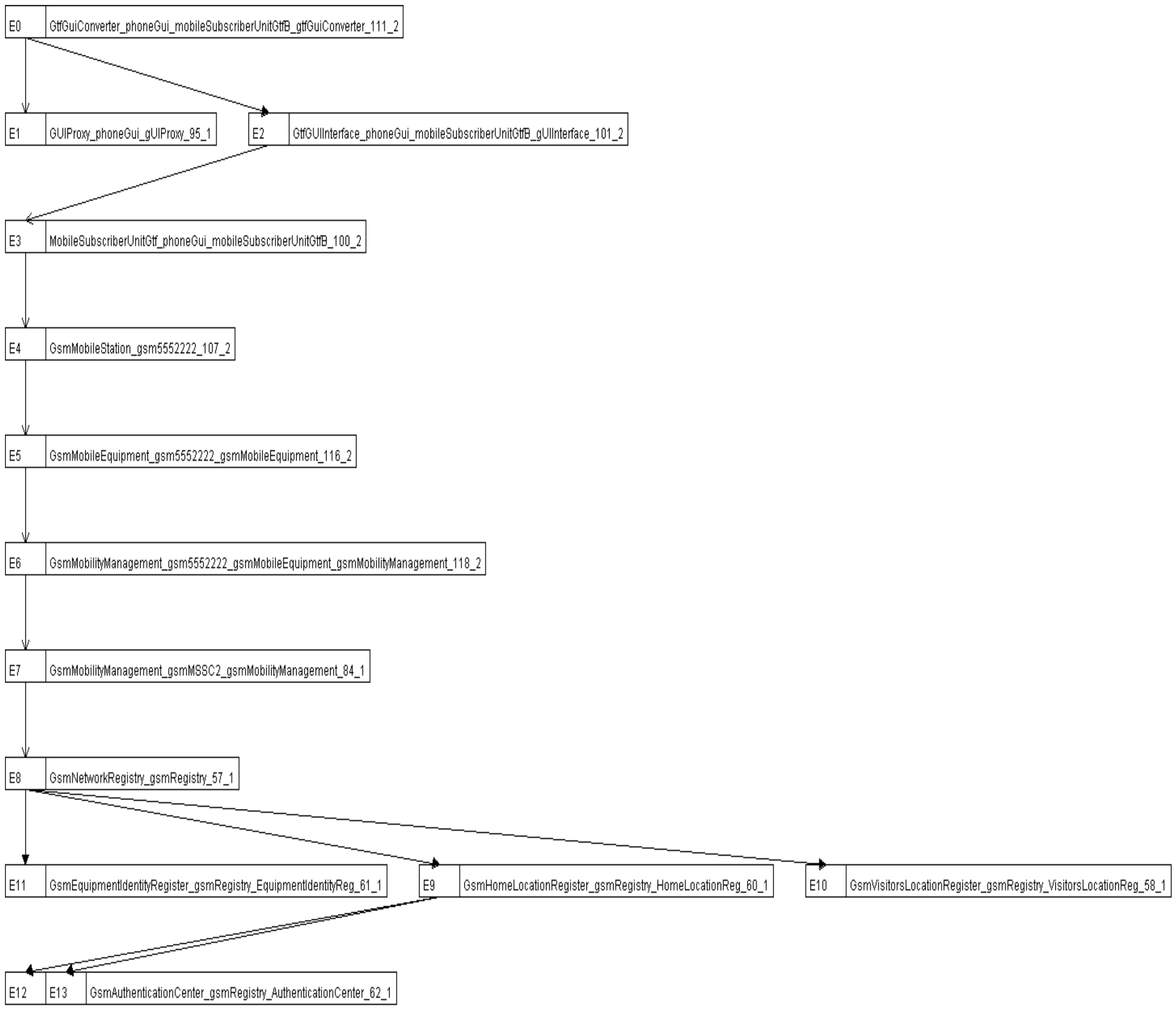
```
A
GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-
2) GUIProxy:phoneGui/gUIProxy(95-1) 760211
S GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1)
GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) 760293
760306
S GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1)
GsmAuthenticationCenter:gsmRegistry/AuthenticationCenter(62-1) 760296
760309
S GsmNetworkRegistry:gsmRegistry(57-1)
GsmHomeLocationRegister:gsmRegistry/HomeLocationReg(60-1) 760286 760315
S GsmNetworkRegistry:gsmRegistry(57-1)
GsmVisitorsLocationRegister:gsmRegistry/VisitorsLocationReg(58-1)
760287 760299
S GsmNetworkRegistry:gsmRegistry(57-1)
GsmEquipmentIdentityRegister:gsmRegistry/EquipmentIdentityReg(61-1)
760289 760303
S
GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-
2) GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-
2) 760214 760235
A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2)
MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) 760230
A Root
GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-
2) 760209
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2)
GsmMobileStation:gsm5552222(107-2) 760251
A GsmMobileStation:gsm5552222(107-2)
GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) 760260
A GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2)
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManageme
nt(118-2) 760272
A
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManageme
nt(118-2) GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1)
760276
A GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1)
GsmNetworkRegistry:gsmRegistry(57-1) 760280
```

APPENDIX A3- PowerOnsequence: LQN Performance Model

```
G
""
0.0
0
0
0.9
-1

P 1
p proc1 f
-1

T 13
t GUIProxy_phoneGui_gUIProxy_95_1 f E1  -1 proc1
t GsmAuthenticationCenter_gsmRegistry_AuthenticationCenter_62_1 f E12 E13  -1
proc1
t GsmEquipmentIdentityRegister_gsmRegistry_EquipmentIdentityReg_61_1 f E11 -1
proc1
t GsmHomeLocationRegister_gsmRegistry_HomeLocationReg_60_1 f E9 -1 proc1
t GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2 f E5  -1 proc1
t GsmMobileStation_gsm5552222_107_2 f E4  -1 proc1
t
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManageme
nt_118_2 f E6  -1 proc1
t GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1 f E7  -1 proc1
t GsmNetworkRegistry_gsmRegistry_57_1 f E8  -1 proc1
t GsmVisitorsLocationRegister_gsmRegistry_VisitorsLocationReg_58_1 f E10  -1 proc1
t GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2 f E2  -1
proc1
t GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2 r E0
-1 proc1
t MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2 f E3  -1 proc1
-1

E 14
s E1 1.0 -1
s E12 1.0 -1
s E13 1.0 -1
s E11 1.0 -1
s E9 1.0 -1
y E9 E12 1.0 -1
y E9 E13 1.0 -1
```

```
s E5 1.0 1.0 -1
z E5 E6 0.0 1.0 -1
s E4 1.0 1.0 -1
z E4 E5 0.0 1.0 -1
s E6 1.0 1.0 -1
z E6 E7 0.0 1.0 -1
s E7 1.0 1.0 -1
z E7 E8 0.0 1.0 -1
s E8 1.0 1.0 -1
y E8 E10 0.0 1.0 -1
y E8 E11 0.0 1.0 -1
y E8 E9 0.0 1.0 -1
s E10 1.0 -1
s E2 1.0 -1
z E2 E3 1.0 -1
s E0 1.0 1.0 -1
z E0 E1 1.0 0.0 -1
y E0 E2 0.0 1.0 -1
s E3 1.0 1.0 -1
z E3 E4 0.0 1.0 -1
-1
```

APPENDIX A4- PowerOnsequence: Graphical LQN Performance Model


E0
GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2
E1
GUIProxy_phoneGui_gUIProxy_95_1
E2
GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2
E3
MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2
E4
GsmMobileStation_gsm5552222_107_2
E5
GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2
E6
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManagement_118_2
E7
GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1
E8
GsmNetworkRegistry_gsmRegistry_57_1
E11
GsmEquipmentIdentityRegister_gsmRegistry_EquipmentIdentityReg_61_1
E9
GsmHomeLocationRegister_gsmRegistry_HomeLocationReg_60_1
E10
GsmVisitorsLocationRegister_gsmRegistry_VisitorsLocationReg_58_1
E12
E13
GsmAuthenticationCenter_gsmRegistry_AuthenticationCenter_62_1

APPENDIX B1- SendingRequest: Sequence Trace

```
thread SendingCallrequest.1.1 {
174863 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%Snd;
174864 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/CollectDigits;
174864 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) receive
%Snd;
174868 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send %Snd;
174868 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
SendingCallrequest.1.1.1;
174871 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send
%Display;
174871 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
SendingCallrequest.1.1.2;
174874 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/SettingUp;
};
thread SendingCallrequest.1.1.1 {
174869 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) receive %Snd;
174876 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) send %Snd;
174878 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) state
Initialized;
174876 GsmMobileStation:gsm5552222(107-2) state S1;
174876 GsmMobileStation:gsm5552222(107-2) receive %Snd;
174885 GsmMobileStation:gsm5552222(107-2) send %Snd;
174885 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) state S1;
174885 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) receive %Snd;
174899 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) send %CallSetup;
174900 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
state S1;
174900 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
receive %CallSetup;
174903 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
send %CallSetup;
174903 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) state S1;
174903 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) receive %CallSetup;
174909 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) send %CallSetup;
174909 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) state S1;
174909 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) receive %CallSetup;
174913 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) send %CallSetup;
174913 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) state Ready;
174913 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) receive
%CallSetup;
174917 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) send %SetupReq;
174917 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state Null0;
174917 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) receive
%SetupReq;
174920 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send %Setup;
174920 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) spawn
SendingCallrequest.1.1.1.1;
174922 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send %timeout
delay 20000;
174922 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) spawn
SendingCallrequest.1.1.1.2;
174923 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
CallInitiated1;
};
thread SendingCallrequest.1.1.1.1 {
174921 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
174921 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive %Setup;
174926 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %Dss1Pdu;
174926 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
174926 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Dss1Pdu;
```

```
174929 Sscop:atmAccessDevice/saal/sscop(78-1) send %Aal5Sdu;
174929 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
174929 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %Aal5Sdu;
174933 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) invoke %CpcsSdu;
174934 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
174934 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsSdu;
174938 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) reply %CpcsPdu;
174938 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsPdu;
174942 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %SarSdu;
174942 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
174942 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %SarSdu;
174946 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmUu1;
174946 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn SendingCallrequest.1.1.1.1.1;
174948 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmCellSdu;
174948 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn SendingCallrequest.1.1.1.1.2;
};
thread SendingCallrequest.1.1.1.1.1 {
174946 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
174946 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmUu1;
};
thread SendingCallrequest.1.1.1.1.2 {
174948 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
174953 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) invoke %AtmCellSdu;
174953 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
174953 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellSdu;
174957 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) reply %AtmCellPdu;
174957 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
174959 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellPdu;
174961 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
174959 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
174959 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
174963 AtmCore:atmNetwork/atmUni2/atmCore(43-1) invoke %AtmCellPdu;
174963 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
174963 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellPdu;
174967 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) reply %AtmCellSdu;
174967 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
174970 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellSdu;
174971 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
174971 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %AtmCellSdu;
174975 AtmSar:atmNetwork/saal2/atmSar(56-1) send %SarIdu;
174975 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
174975 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %SarIdu;
174978 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) invoke %CpcsPdu;
174978 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
174978 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsPdu;
174982 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) reply %CpcsSdu;
174982 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsSdu;
174984 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %Aal5Sdu;
174984 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
174984 Sscop:atmNetwork/saal2/sscop(53-1) receive %Aal5Sdu;
174987 Sscop:atmNetwork/saal2/sscop(53-1) send %Dss1Pdu;
174987 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
174987 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %Dss1Pdu;
174991 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %Setup;
174991 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state Null0;
174991 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive %Setup;
174995 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %SetupInd;
174997 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
CallInitiation1;
174995 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) state Ready;
174995 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) receive %SetupInd;
174999 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) send %CallSetup;
174999 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) spawn
SendingCallrequest.1.1.1.1.2.1;
175001 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) send %ProceedingReq;
175001 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) spawn
SendingCallrequest.1.1.1.1.2.2;
};
thread SendingCallrequest.1.1.1.1.2.1 {
174999 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) state Ready;
174999 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) receive %CallSetup;
```

```
175004 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) send %SetupReq;
175004 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) state Null0;
175004 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) receive %SetupReq;
175010 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) send %Setup;
175010 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) spawn
SendingCallrequest.1.1.1.1.2.1.1;
175012 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) send %timeout delay
20000;
175012 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) spawn
SendingCallrequest.1.1.1.1.2.1.2;
175015 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) state CallPresent6;
};
thread SendingCallrequest.1.1.1.1.2.1.1 {
175010 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) state Ready;
175010 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) receive %Setup;
175020 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) send %Dss1Pdu;
175020 Sscop:atmNetwork/saal1/sscop(47-1) state Ready;
175020 Sscop:atmNetwork/saal1/sscop(47-1) receive %Dss1Pdu;
175027 Sscop:atmNetwork/saal1/sscop(47-1) send %Aal5Sdu;
175027 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) state Ready;
175027 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) receive %Aal5Sdu;
175040 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) invoke %CpcsSdu;
175040 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) state Ready;
175040 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) receive %CpcsSdu;
175044 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) reply %CpcsPdu;
175044 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) receive %CpcsPdu;
175046 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) send %SarSdu;
175046 AtmSar:atmNetwork/saal1/atmSar(50-1) state Ready;
175046 AtmSar:atmNetwork/saal1/atmSar(50-1) receive %SarSdu;
175055 AtmSar:atmNetwork/saal1/atmSar(50-1) send %AtmUu1;
175055 AtmSar:atmNetwork/saal1/atmSar(50-1) spawn SendingCallrequest.1.1.1.1.2.1.1.1;
175057 AtmSar:atmNetwork/saal1/atmSar(50-1) send %AtmCellSdu;
175057 AtmSar:atmNetwork/saal1/atmSar(50-1) spawn SendingCallrequest.1.1.1.1.2.1.1.2;
};
thread SendingCallrequest.1.1.1.1.2.1.1.1 {
175055 AtmCore:atmNetwork/atmUni1/atmCore(40-1) state Ready;
175055 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmUu1;
};
thread SendingCallrequest.1.1.1.1.2.1.1.2 {
175057 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmCellSdu;
175076 AtmCore:atmNetwork/atmUni1/atmCore(40-1) invoke %AtmCellSdu;
175076 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) state Ready;
175076 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) receive %AtmCellSdu;
175080 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) reply %AtmCellPdu;
175080 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmCellPdu;
175082 AtmCore:atmNetwork/atmUni1/atmCore(40-1) send %AtmCellPdu;
175085 AtmCore:atmNetwork/atmUni1/atmCore(40-1) state Ready;
175083 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
175083 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) receive %AtmCellPdu;
175098 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %AtmCellPdu;
175222 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) receive %AtmCellPdu;
175228 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %AtmCellPdu;
175233 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
175229 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) state 2;
175229 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) receive %AtmCellPdu;
175238 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) invoke %AtmCellPdu;
175238 AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) state Ready;
175238 AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) receive
%AtmCellPdu;
175245 AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) reply
%AtmCellSdu;
175245 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) receive %AtmCellSdu;
176238 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %CallSetup;
176243 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
176239 GsmMobileServicesSwitchingCenter:gsmMSSC1(20-1) state S1;
176239 GsmMobileServicesSwitchingCenter:gsmMSSC1(20-1) receive %CallSetup;
176247 GsmMobileServicesSwitchingCenter:gsmMSSC1(20-1) send %CallSetup;
176247 (22-1) receive %CallSetup;
176285 (22-1) send %CallSetup;
176289 (22-1) state 2;
```

```
176285 GsmMobilityManagement:gsm5551111/gsmMobileEquipment/gsmMobilityManagement(114-2)
state S1;
176285 GsmMobilityManagement:gsm5551111/gsmMobileEquipment/gsmMobilityManagement(114-2)
receive %CallSetup;
176293 GsmMobilityManagement:gsm5551111/gsmMobileEquipment/gsmMobilityManagement(114-2)
send %CallSetup;
176293 GsmMobileEquipment:gsm5551111/gsmMobileEquipment(112-2) state S1;
176293 GsmMobileEquipment:gsm5551111/gsmMobileEquipment(112-2) receive %CallSetup;
176302 GsmMobileEquipment:gsm5551111/gsmMobileEquipment(112-2) send %Alerting;
176302 GsmMobileStation:gsm5551111(104-2) state S1;
176302 GsmMobileStation:gsm5551111(104-2) receive %Alerting;
176312 GsmMobileStation:gsm5551111(104-2) send %Alerting;
176312 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfA(96-2) state Initialized;
176312 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfA(96-2) receive %Alerting;
176394 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfA(96-2) send %Alerting;
176395 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfA/gUIInterface(97-2) state
PowerIdle/ActiveIdle;
176395 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfA/gUIInterface(97-2) receive
%Alerting;
176404 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfA/gUIInterface(97-2) send
%Display;
176404 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfA/gtfGuiConverter(110-2) state
top;
176404 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfA/gtfGuiConverter(110-2) receive
%Display;
};
thread SendingCallrequest.1.1.1.1.2.2 {
175001 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive
%ProceedingReq;
175007 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %CallProceeding;
175008 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
CanBeCleared/OutgoingCallProceeding3;
175007 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %CallProceeding;
175017 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %Dss1Pdu;
175018 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
175017 Sscop:atmNetwork/saal2/sscop(53-1) receive %Dss1Pdu;
175024 Sscop:atmNetwork/saal2/sscop(53-1) send %Aal5Sdu;
175025 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
175024 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %Aal5Sdu;
175031 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) invoke %CpcsSdu;
175031 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsSdu;
175034 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
175034 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) reply %CpcsPdu;
175034 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsPdu;
175036 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %SarSdu;
175038 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
175036 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %SarSdu;
175049 AtmSar:atmNetwork/saal2/atmSar(56-1) send %AtmUu1;
175049 AtmSar:atmNetwork/saal2/atmSar(56-1) spawn SendingCallrequest.1.1.1.1.2.2.1;
175051 AtmSar:atmNetwork/saal2/atmSar(56-1) send %AtmCellSdu;
175051 AtmSar:atmNetwork/saal2/atmSar(56-1) spawn SendingCallrequest.1.1.1.1.2.2.2;
175053 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
};
thread SendingCallrequest.1.1.1.1.2.2.1 {
175049 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmUu1;
175061 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
};
thread SendingCallrequest.1.1.1.1.2.2.2 {
175051 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
175064 AtmCore:atmNetwork/atmUni2/atmCore(43-1) invoke %AtmCellSdu;
175064 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellSdu;
175067 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
175067 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) reply %AtmCellPdu;
175067 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
175069 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellPdu;
175072 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
175070 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
175089 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) invoke %AtmCellPdu;
175089 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellPdu;
175092 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
175092 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) reply %AtmCellSdu;
```

```
175092 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
175094 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellSdu;
175096 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
175094 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %AtmCellSdu;
175164 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %SarIdu;
175169 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
175164 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %SarIdu;
175175 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) invoke %CpcsPdu;
175175 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsPdu;
175227 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
175227 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) reply %CpcsSdu;
175227 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsSdu;
175232 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %Aal5Sdu;
175235 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
175232 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Aal5Sdu;
175239 Sscop:atmAccessDevice/saal/sscop(78-1) send %Dss1Pdu;
175241 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
175239 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive %Dss1Pdu;
175246 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %CallProceeding;
175250 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
175246 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) receive
%CallProceeding;
175257 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
CanBeCleared/OutgoingCallProceeding3;
};
thread SendingCallrequest.1.1.1.2 {
};
thread SendingCallrequest.1.1.2 {
174871 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) receive
%Display;
};
```

APPENDIX B2- SendingRequest: Communication Patterns

S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 174934 174938
A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 174946
S AtmCore:atmNetwork/atmUni2/atmCore(43-1)
I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 174963 174967
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1)
I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 174978 174982
S AtmCpcs:atmNetwork/saal1/atmCpcs(48-1)
I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) 175040 175044
A AtmSar:atmNetwork/saal1/atmSar(50-1) AtmCore:atmNetwork/atmUni1/atmCore(40-1) 175055
S AtmCore:atmNetwork/atmUni1/atmCore(40-1)
I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) 175076 175080
A AtmCore:atmNetwork/atmUni1/atmCore(40-1)
AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) 175083
S AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1)
AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) 175238 175245
A AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1)
AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) 175229
S SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1)
CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) 174995 175001
A CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1)
CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) 174999
S Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1)
SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) 174991 175007
S Sscop:atmNetwork/saal2/sscop(53-1)
Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) 174987 175017
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) Sscop:atmNetwork/saal2/sscop(53-1) 174984 175024
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1)
I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 175031 175034
S AtmSar:atmNetwork/saal2/atmSar(56-1) AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) 174975 175036
S AtmCore:atmNetwork/atmUni2/atmCore(43-1) AtmSar:atmNetwork/saal2/atmSar(56-1) 174971 175049
A AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
AtmCore:atmNetwork/atmUni2/atmCore(43-1) 174959
S AtmCore:atmNetwork/atmUni2/atmCore(43-1)
I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 175064 175067

A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 174948
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) 175089 175092
A AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
AtmSar:atmAccessDevice/saal/atmSar(81-1) 174942
A Sscop:atmAccessDevice/saal/sscop(78-1)
AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) 174929
S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 175175 175227
A Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1)
Sscop:atmAccessDevice/saal/sscop(78-1) 174926
A SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1)
Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) 174921
A CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1)
SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) 174917
S GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2)
GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) 174864
174871
A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2)
MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) 174869
A AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1)
GsmMobileServicesSwitchingCenter:gsmMSSC1(20-1) 176239
A GsmMobileServicesSwitchingCenter:gsmMSSC1(20-1)
GsmMobilityManagement:gsm5551111/gsmMobileEquipment/gsmMobilityManagement
(114-2) 176285
A
GsmMobilityManagement:gsm5551111/gsmMobileEquipment/gsmMobilityManagement
(114-2) GsmMobileEquipment:gsm5551111/gsmMobileEquipment(112-2) 176293
A GsmMobileEquipment:gsm5551111/gsmMobileEquipment(112-2)
GsmMobileStation:gsm5551111(104-2) 176302
A GsmMobileStation:gsm5551111(104-2)
MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfA(96-2) 176312
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfA(96-2)
GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfA/gUIInterface(97-2) 176395
A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfA/gUIInterface(97-2)
GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfA/gtfGuiConverter(110-2) 176404
A CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1)
SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) 175004
A SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1)
Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) 175010
A Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1)
Sscop:atmNetwork/saal1/sscop(47-1) 175020
A Sscop:atmNetwork/saal1/sscop(47-1) AtmCpcs:atmNetwork/saal1/atmCpcs(48-1)
175027

A AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) AtmSar:atmNetwork/saal1/atmSar(50-1) 175046
A AtmSar:atmNetwork/saal1/atmSar(50-1) AtmCore:atmNetwork/atmUni1/atmCore(40-1) 175057
A AtmSar:atmNetwork/saal2/atmSar(56-1) AtmCore:atmNetwork/atmUni2/atmCore(43-1) 175051
A AtmCore:atmNetwork/atmUni2/atmCore(43-1) AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 175070
A AtmCore:atmAccessDevice/atmUni/atmCore(69-1) AtmSar:atmAccessDevice/saal/atmSar(81-1) 175094
A AtmSar:atmAccessDevice/saal/atmSar(81-1) AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) 175164
A AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) Sscop:atmAccessDevice/saal/sscop(78-1) 175232
A Sscop:atmAccessDevice/saal/sscop(78-1) Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) 175239
A Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) 175246
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) GsmMobileStation:gsm5552222(107-2) 174876
A GsmMobileStation:gsm5552222(107-2) GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) 174885
A GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2) 174900
A GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2) GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) 174903
A GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) 174909
A GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) 174913

APPENDIX B3- SendingRequest: LQN Performance Model

```
G
""
0.0
0
0
0.9
-1

P 0
p proc1 f
-1

T 0
t AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) f E40  -1 proc1
t AtmCore:atmAccessDevice/atmUni/atmCore(69-1) f E15 E16 E38  -1 proc1
t AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) f E47  -
1 proc1
t AtmCore:atmNetwork/atmUni1/atmCore(40-1) f E35 E36  -1 proc1
t AtmCore:atmNetwork/atmUni2/atmCore(43-1) f E17 E34  -1 proc1
t AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) f E12 E43  -1 proc1
t AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) f E30  -1 proc1
t AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) f E20  -1 proc1
t AtmSar:atmAccessDevice/saal/atmSar(81-1) f E14 E42  -1 proc1
t AtmSar:atmNetwork/saal1/atmSar(50-1) f E33  -1 proc1
t AtmSar:atmNetwork/saal2/atmSar(56-1) f E19  -1 proc1
t AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) f E45  -1 proc1
t CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) f E26  -1 proc1
t CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) f E25  -1 proc1
t CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) f E8  -1 proc1
t Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) f E10 E48  -1 proc1
t Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) f E28  -1 proc1
t Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) f E23  -1 proc1
t GsmMobileEquipment:gsm5551111/gsmMobileEquipment(112-2) f E52  -1 proc1
t GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) f E4  -1 proc1
t GsmMobileServicesSwitchingCenter:gsmMSSC1(20-1) f E50  -1 proc1
t GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) f E7  -1 proc1
t GsmMobileStation:gsm5551111(104-2) f E53  -1 proc1
t GsmMobileStation:gsm5552222(107-2) f E3  -1 proc1
t
GsmMobilityManagement:gsm5551111/gsmMobileEquipment/gsmMobilityManagement
(114-2) f E51  -1 proc1
```

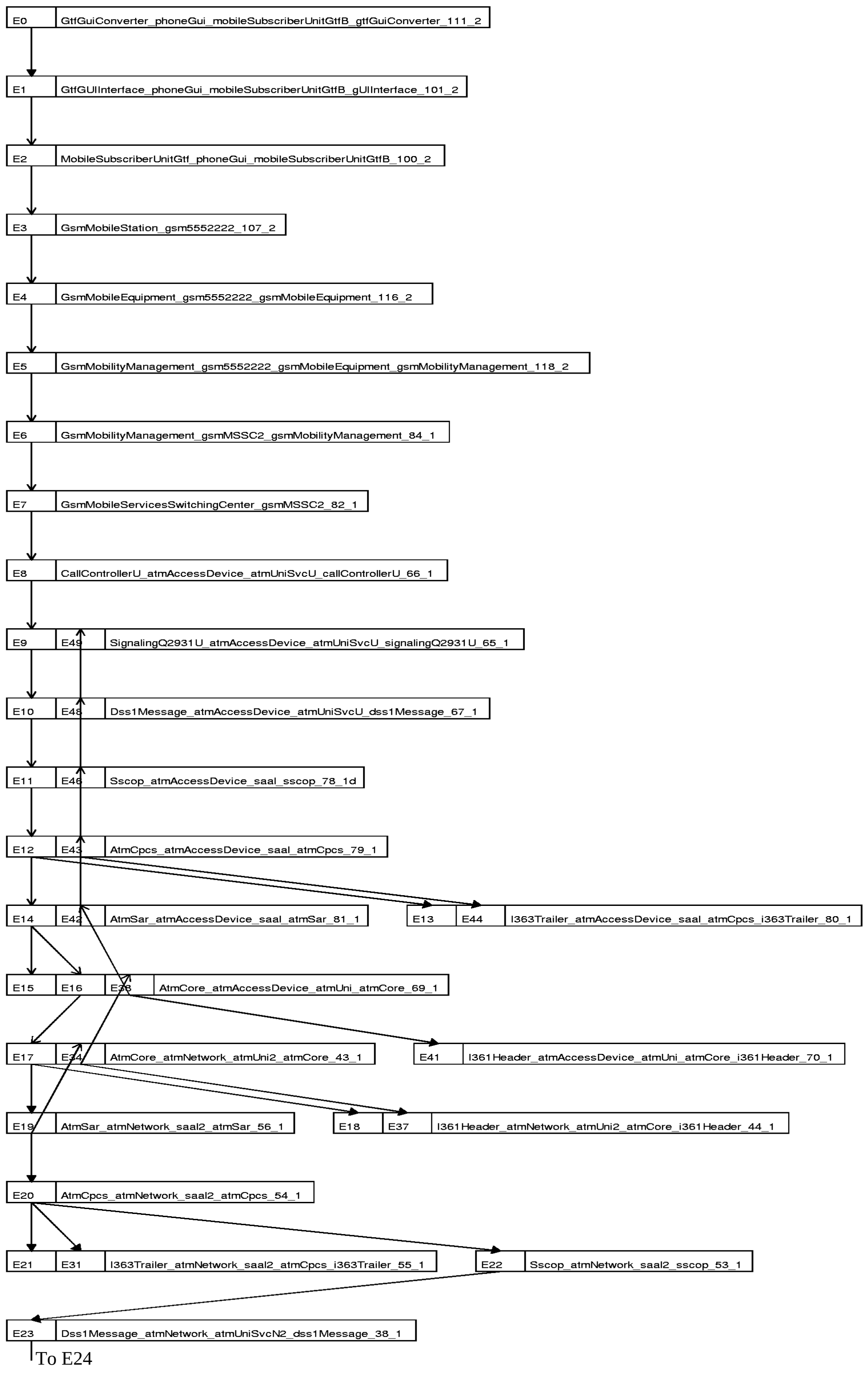
```
t
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) f E5  -1 proc1
t GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) f E6  -1 proc1
t GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfA/gUIInterface(97-2) f E55  -1
proc1
t GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) f E1  -1
proc1
t GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfA/gtfGuiConverter(110-2) f E56  -
1 proc1
t GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) r E0  -1
proc1
t I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) f E41  -1 proc1
t I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) f E39  -1 proc1
t I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) f E18 E37  -1 proc1
t I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) f E13 E44  -1 proc1
t I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) f E32  -1 proc1
t I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) f E21 E31  -1 proc1
t MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfA(96-2) f E54  -1 proc1
t MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) f E2  -1 proc1
t SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) f E27  -1 proc1
t SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) f E24  -1 proc1
t SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) f E9 E49  -1
proc1
t Sscop:atmAccessDevice/saal/sscop(78-1) f E11 E46  -1 proc1
t Sscop:atmNetwork/saal1/sscop(47-1) f E29  -1 proc1
t Sscop:atmNetwork/saal2/sscop(53-1) f E22  -1 proc1
-1

E 0
s E0 1.0 -1
y E0 E1 1.0 -1
s E1 1.0 -1
z E1 E2 1.0 -1
s E10 1.0 1.0 -1
z E10 E11 0.0 1.0 -1
s E11 1.0 1.0 -1
z E11 E12 0.0 1.0 -1
s E12 1.0 1.0 -1
y E12 E13 0.0 1.0 -1
z E12 E14 0.0 1.0 -1
s E13 1.0 -1
s E14 1.0 1.0 -1
z E14 E15 0.0 1.0 -1
z E14 E16 0.0 1.0 -1
s E15 1.0 -1
```

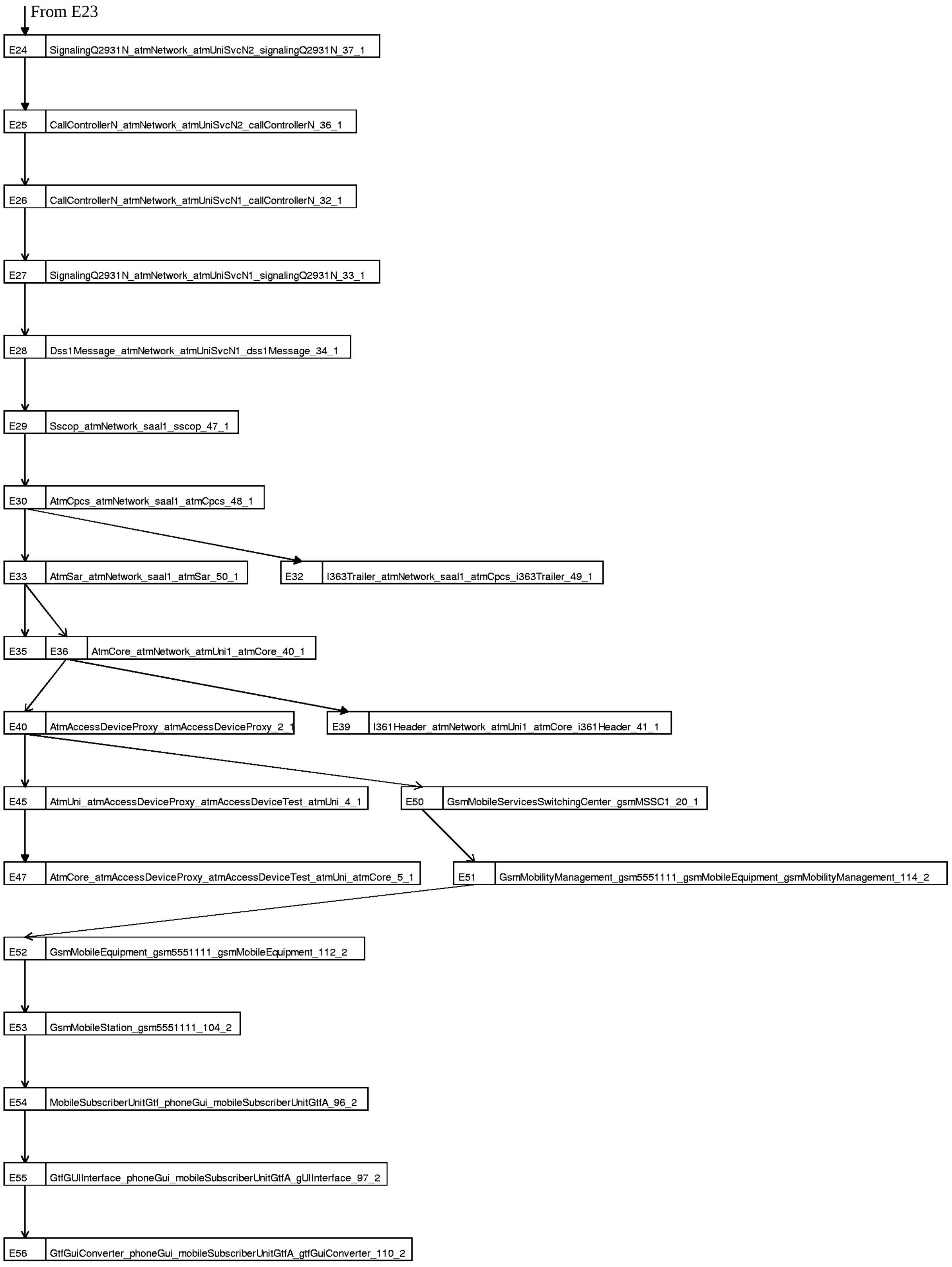
```
s E16 1.0 1.0 -1
z E16 E17 0.0 1.0 -1
s E17 1.0 1.0 -1
y E17 E18 0.0 1.0 -1
y E17 E19 0.0 1.0 -1
s E18 1.0 -1
s E19 1.0 1.0 -1
y E19 E20 1.0 0.0 -1
z E19 E34 0.0 1.0 -1
s E2 1.0 1.0 -1
z E2 E3 0.0 1.0 -1
s E20 1.0 -1
y E20 E21 1.0 -1
y E20 E22 1.0 -1
y E20 E31 1.0 -1
s E21 1.0 -1
s E22 1.0 -1
y E22 E23 1.0 -1
s E23 1.0 -1
y E23 E24 1.0 -1
s E24 1.0 -1
y E24 E25 1.0 -1
s E25 1.0 -1
z E25 E26 1.0 -1
s E26 1.0 1.0 -1
z E26 E27 0.0 1.0 -1
s E27 1.0 1.0 -1
z E27 E28 0.0 1.0 -1
s E28 1.0 1.0 -1
z E28 E29 0.0 1.0 -1
s E29 1.0 1.0 -1
z E29 E30 0.0 1.0 -1
s E3 1.0 1.0 -1
z E3 E4 0.0 1.0 -1
s E30 1.0 1.0 -1
y E30 E32 0.0 1.0 -1
z E30 E33 0.0 1.0 -1
s E31 1.0 -1
s E32 1.0 -1
s E33 1.0 1.0 -1
z E33 E35 0.0 1.0 -1
z E33 E36 0.0 1.0 -1
s E34 1.0 1.0 -1
y E34 E37 0.0 1.0 -1
z E34 E38 0.0 1.0 -1
s E35 1.0 -1
```

```
s E36 1.0 1.0 -1
y E36 E39 0.0 1.0 -1
z E36 E40 0.0 1.0 -1
s E37 1.0 -1
s E38 1.0 1.0 -1
y E38 E41 0.0 1.0 -1
z E38 E42 0.0 1.0 -1
s E39 1.0 -1
s E4 1.0 1.0 -1
z E4 E5 0.0 1.0 -1
s E40 1.0 1.0 -1
z E40 E45 0.0 1.0 -1
z E40 E50 0.0 1.0 -1
s E41 1.0 -1
s E42 1.0 1.0 -1
z E42 E43 0.0 1.0 -1
s E43 1.0 1.0 -1
y E43 E44 0.0 1.0 -1
z E43 E46 0.0 1.0 -1
s E44 1.0 -1
s E45 1.0 1.0 -1
y E45 E47 0.0 1.0 -1
s E46 1.0 1.0 -1
z E46 E48 0.0 1.0 -1
s E47 1.0 -1
s E48 1.0 1.0 -1
z E48 E49 0.0 1.0 -1
s E49 1.0 -1
s E5 1.0 1.0 -1
z E5 E6 0.0 1.0 -1
s E50 1.0 1.0 -1
z E50 E51 0.0 1.0 -1
s E51 1.0 1.0 -1
z E51 E52 0.0 1.0 -1
s E52 1.0 1.0 -1
z E52 E53 0.0 1.0 -1
s E53 1.0 1.0 -1
z E53 E54 0.0 1.0 -1
s E54 1.0 1.0 -1
z E54 E55 0.0 1.0 -1
s E55 1.0 1.0 -1
z E55 E56 0.0 1.0 -1
s E56 1.0 -1
s E6 1.0 1.0 -1
z E6 E7 0.0 1.0 -1
s E7 1.0 1.0 -1
```

```
z E7 E8 0.0 1.0 -1
s E8 1.0 1.0 -1
z E8 E9 0.0 1.0 -1
s E9 1.0 1.0 -1
z E9 E10 0.0 1.0 -1
-1
```

APPENDIX B4- SendingRequest: Graphical LQN Performance Model


E0
GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2
E1
GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2
E2
MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2
E3
GsmMobileStation_gsm5552222_107_2
E4
GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2
E5
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManagement_118_2
E6
GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1
E7
GsmMobileServicesSwitchingCenter_gsmMSSC2_82_1
E8
CallControllerU_atmAccessDevice_atmUniSvcU_callControllerU_66_1
E9
E49
SignalingQ2931U_atmAccessDevice_atmUniSvcU_signalingQ2931U_65_1
E10
E48
Dss1Message_atmAccessDevice_atmUniSvcU_dss1Message_67_1
E11
E46
Sscop_atmAccessDevice_saal_sscop_78_1d
E12
E43
AtmCpcs_atmAccessDevice_saal_atmCpcs_79_1
E14
E42
AtmSar_atmAccessDevice_saal_atmSar_81_1
E13
E44
I363Trailer_atmAccessDevice_saal_atmCpcs_i363Trailer_80_1
E15
E16
E38
AtmCore_atmAccessDevice_atmUni_atmCore_69_1
E17
E34
AtmCore_atmNetwork_atmUni2_atmCore_43_1
E41
I361Header_atmAccessDevice_atmUni_atmCore_i361Header_70_1
E19
AtmSar_atmNetwork_saal2_atmSar_56_1
E18
E37
I361Header_atmNetwork_atmUni2_atmCore_i361Header_44_1
E20
AtmCpcs_atmNetwork_saal2_atmCpcs_54_1
E21
E31
I363Trailer_atmNetwork_saal2_atmCpcs_i363Trailer_55_1
E22
Sscop_atmNetwork_saal2_sscop_53_1
E23
Dss1Message_atmNetwork_atmUniSvcN2_dss1Message_38_1
To E24


From E23
E24 SignalingQ2931N_atmNetwork_atmUniSvcN2_signalingQ2931N_37_1
E25 CallControllerN_atmNetwork_atmUniSvcN2_callControllerN_36_1
E26 CallControllerN_atmNetwork_atmUniSvcN1_callControllerN_32_1
E27 SignalingQ2931N_atmNetwork_atmUniSvcN1_signalingQ2931N_33_1
E28 Dss1Message_atmNetwork_atmUniSvcN1_dss1Message_34_1
E29 Sscop_atmNetwork_saal1_sscop_47_1
E30 AtmCpcs_atmNetwork_saal1_atmCpcs_48_1
E33 AtmSar_atmNetwork_saal1_atmSar_50_1
E32 I363Trailer_atmNetwork_saal1_atmCpcs_i363Trailer_49_1
E35
E36 AtmCore_atmNetwork_atmUni1_atmCore_40_1
E40 AtmAccessDeviceProxy_atmAccessDeviceProxy_2_1
E39 I361Header_atmNetwork_atmUni1_atmCore_i361Header_41_1
E45 AtmUni_atmAccessDeviceProxy_atmAccessDeviceTest_atmUni_4_1
E50 GsmMobileServicesSwitchingCenter_gsmMSSC1_20_1
E47 AtmCore_atmAccessDeviceProxy_atmAccessDeviceTest_atmUni_atmCore_5_1
E51 GsmMobilityManagement_gsm5551111_gsmMobileEquipment_gsmMobilityManagement_114_2
E52 GsmMobileEquipment_gsm5551111_gsmMobileEquipment_112_2
E53 GsmMobileStation_gsm5551111_104_2
E54 MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfA_96_2
E55 GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfA_gUIInterface_97_2
E56 GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfA_gtfGuiConverter_110_2

APPENDIX C1- AcceptingRequest: Sequence Trace

```
thread acceptingcall.1.1 {
2582501 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%Snd;
2582501 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/ActiveIdle;
2582501 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) receive
%Snd;
2582506 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send
%Connect;
2582506 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
acceptingcall.1.1.1;
2582509 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send
%Display;
2582509 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
acceptingcall.1.1.2;
};
thread acceptingcall.1.1.1 {
2582507 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) receive
%Connect;
2582515 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) send %Connect;
2582517 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) state
Initialized;
2582515 GsmMobileStation:gsm5552222(107-2) state S1;
2582515 GsmMobileStation:gsm5552222(107-2) receive %Connect;
2582526 GsmMobileStation:gsm5552222(107-2) send %Connect;
2582526 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) state S1;
2582526 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) receive %Connect;
2582537 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) send %Connect;
2582537 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
state S1;
2582537 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
receive %Connect;
2582545 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
send %Connect;
2582545 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) state S1;
2582545 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) receive %Connect;
2582550 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) send %Connect;
2582550 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) state S1;
2582550 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) receive %Connect;
2582553 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) send %CallSetupResp;
2582553 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) state Ready;
2582553 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) receive
%CallSetupResp;
2582557 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) send %SetupResp;
2582557 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
CanBeCleared/IncomingCallProceeding9;
2582557 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) receive
%SetupResp;
2582561 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send %Connect;
2582562 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
CanBeCleared/ConnectReq8;
2582561 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
2582561 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive %Connect;
2582564 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %Dss1Pdu;
2582564 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
2582564 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Dss1Pdu;
2582568 Sscop:atmAccessDevice/saal/sscop(78-1) send %Aal5Sdu;
2582568 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
2582568 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %Aal5Sdu;
2582572 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %CpcsSdu;
2582572 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
2582572 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsSdu;
2582577 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) send %CpcsPdu;
2582577 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsPdu;
2582581 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %SarSdu;
2582581 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
```

```
2582581 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %SarSdu;
2582584 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmUu1;
2582584 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn acceptingcall.1.1.1.1;
2582586 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmCellSdu;
2582586 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn acceptingcall.1.1.1.2;
};
thread acceptingcall.1.1.1.1 {
2582584 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
2582584 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmUu1;
};
thread acceptingcall.1.1.1.2 {
2582586 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
2582591 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellSdu;
2582591 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
2582591 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellSdu;
2582595 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) send %AtmCellPdu;
2582595 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
2582597 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellPdu;
2582599 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
2582597 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
2582597 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
2582602 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellPdu;
2582602 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
2582602 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellPdu;
2582607 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) send %AtmCellSdu;
2582607 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
2582609 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellSdu;
2582609 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
2582609 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %AtmCellSdu;
2582613 AtmSar:atmNetwork/saal2/atmSar(56-1) send %SarIdu;
2582613 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
2582613 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %SarIdu;
2582617 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %CpcsPdu;
2582617 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
2582617 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsPdu;
2582620 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) send %CpcsSdu;
2582620 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsSdu;
2582622 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %Aal5Sdu;
2582622 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
2582622 Sscop:atmNetwork/saal2/sscop(53-1) receive %Aal5Sdu;
2582626 Sscop:atmNetwork/saal2/sscop(53-1) send %Dss1Pdu;
2582626 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
2582626 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %Dss1Pdu;
2582629 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %Connect;
2582629 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
CanBeCleared/IncomingCallProceeding9;
2582629 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive %Connect;
2582632 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %SetupConf;
2582634 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
CanBeCleared/ConnectReq8;
2582633 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) state Ready;
2582633 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) receive %SetupConf;
2582636 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) send %CallSetupResp;
2582636 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) spawn
acceptingcall.1.1.1.2.1;
2582638 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) send
%SetupCompleteReq;
2582638 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) spawn
acceptingcall.1.1.1.2.2;
};
thread acceptingcall.1.1.1.2.1 {
2582636 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) state Ready;
2582636 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) receive
%CallSetupResp;
2582641 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) send %SetupResp;
2582641 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) state
CanBeCleared/OutgoingCallProceeding3;
2582641 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) receive %SetupResp;
2582647 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) send %Connect;
2582650 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) state
CanBeCleared/Active10;
```

```
2582647 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) state Ready;
2582647 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) receive %Connect;
2582655 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) send %Dss1Pdu;
2582655 Sscop:atmNetwork/saal1/sscop(47-1) state Ready;
2582655 Sscop:atmNetwork/saal1/sscop(47-1) receive %Dss1Pdu;
2582662 Sscop:atmNetwork/saal1/sscop(47-1) send %Aal5Sdu;
2582662 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) state Ready;
2582662 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) receive %Aal5Sdu;
2582674 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) send %CpcsSdu;
2582675 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) state Ready;
2582675 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) receive %CpcsSdu;
2582678 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) send %CpcsPdu;
2582678 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) receive %CpcsPdu;
2582680 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) send %SarSdu;
2582680 AtmSar:atmNetwork/saal1/atmSar(50-1) state Ready;
2582680 AtmSar:atmNetwork/saal1/atmSar(50-1) receive %SarSdu;
2582689 AtmSar:atmNetwork/saal1/atmSar(50-1) send %AtmUu1;
2582689 AtmSar:atmNetwork/saal1/atmSar(50-1) spawn acceptingcall.1.1.1.2.1.1;
2582690 AtmSar:atmNetwork/saal1/atmSar(50-1) send %AtmCellSdu;
2582690 AtmSar:atmNetwork/saal1/atmSar(50-1) spawn acceptingcall.1.1.1.2.1.2;
};
thread acceptingcall.1.1.1.2.1.1 {
2582689 AtmCore:atmNetwork/atmUni1/atmCore(40-1) state Ready;
2582689 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmUu1;
};
thread acceptingcall.1.1.1.2.1.2 {
2582690 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmCellSdu;
2582709 AtmCore:atmNetwork/atmUni1/atmCore(40-1) send %AtmCellSdu;
2582709 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) state Ready;
2582709 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) receive %AtmCellSdu;
2582712 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) send %AtmCellPdu;
2582712 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmCellPdu;
2582714 AtmCore:atmNetwork/atmUni1/atmCore(40-1) send %AtmCellPdu;
2582716 AtmCore:atmNetwork/atmUni1/atmCore(40-1) state Ready;
2582714 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
2582714 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) receive %AtmCellPdu;
2582728 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %AtmCellPdu;
2625499 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) receive %AtmCellPdu;
2625507 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %AtmCellPdu;
2625532 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
};
thread acceptingcall.1.1.1.2.2 {
2582638 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive
%SetupCompleteReq;
2582644 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %ConnectAck;
2582645 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
CanBeCleared/Active10;
2582644 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %ConnectAck;
2582652 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %Dss1Pdu;
2582653 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
2582652 Sscop:atmNetwork/saal2/sscop(53-1) receive %Dss1Pdu;
2582659 Sscop:atmNetwork/saal2/sscop(53-1) send %Aal5Sdu;
2582660 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
2582659 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %Aal5Sdu;
2582666 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %CpcsSdu;
2582666 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsSdu;
2582669 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
2582669 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) send %CpcsPdu;
2582669 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsPdu;
2582671 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %SarSdu;
2582672 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
2582671 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %SarSdu;
2582684 AtmSar:atmNetwork/saal2/atmSar(56-1) send %AtmUu1;
2582684 AtmSar:atmNetwork/saal2/atmSar(56-1) spawn acceptingcall.1.1.1.2.2.1;
2582685 AtmSar:atmNetwork/saal2/atmSar(56-1) send %AtmCellSdu;
2582685 AtmSar:atmNetwork/saal2/atmSar(56-1) spawn acceptingcall.1.1.1.2.2.2;
2582686 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
};
thread acceptingcall.1.1.1.2.2.1 {
2582684 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmUu1;
2582695 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
```

```
};
thread acceptingcall.1.1.1.2.2.2 {
2582685 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
2582697 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellSdu;
2582697 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellSdu;
2582701 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
2582701 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) send %AtmCellPdu;
2582701 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
2582703 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellPdu;
2582704 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
2582703 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
2582718 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellPdu;
2582718 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellPdu;
2582721 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
2582722 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) send %AtmCellSdu;
2582722 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
2582724 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellSdu;
2582725 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
2582724 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %AtmCellSdu;
2582732 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %SarIdu;
2582734 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
2582732 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %SarIdu;
2582820 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %CpcsPdu;
2582820 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsPdu;
2582829 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
2582829 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) send %CpcsSdu;
2582829 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsSdu;
2582832 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %Aal5Sdu;
2582835 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
2582832 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Aal5Sdu;
2582884 Sscop:atmAccessDevice/saal/sscop(78-1) send %Dss1Pdu;
2582886 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
2582884 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive %Dss1Pdu;
2582892 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %ConnectAck;
2582894 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
2582892 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) receive
%ConnectAck;
2582899 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send
%SetupCompleteInd;
2582902 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
CanBeCleared/Active10;
2582899 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) receive
%SetupCompleteInd;
2582904 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) state Ready;
};
thread acceptingcall.1.1.2 {
2582509 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) receive
%Display;
};
```

APPENDIX C2- AcceptingRequest: Communication Patterns

S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 2582572 2582577
A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 2582584
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) 2582591 2582595
S AtmCore:atmNetwork/atmUni2/atmCore(43-1)
I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 2582602 2582607
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1)
I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 2582617 2582620
S AtmCpcs:atmNetwork/saal1/atmCpcs(48-1)
I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) 2582675 2582678
A AtmSar:atmNetwork/saal1/atmSar(50-1) AtmCore:atmNetwork/atmUni1/atmCore(40-1) 2582689
S AtmCore:atmNetwork/atmUni1/atmCore(40-1)
I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) 2582709 2582712
A AtmCore:atmNetwork/atmUni1/atmCore(40-1)
AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) 2582714
S SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1)
CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) 2582633 2582638
A CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1)
CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) 2582636
S Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1)
SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) 2582629 2582644
S Sscop:atmNetwork/saal2/sscop(53-1)
Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) 2582626 2582652
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) Sscop:atmNetwork/saal2/sscop(53-1) 2582622 2582659
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1)
I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 2582666 2582669
S AtmSar:atmNetwork/saal2/atmSar(56-1) AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) 2582613 2582671
S AtmCore:atmNetwork/atmUni2/atmCore(43-1) AtmSar:atmNetwork/saal2/atmSar(56-1) 2582609 2582684
A AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
AtmCore:atmNetwork/atmUni2/atmCore(43-1) 2582597
S AtmCore:atmNetwork/atmUni2/atmCore(43-1)
I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 2582697 2582701
A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 2582586
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) 2582718 2582722

```
A AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
AtmSar:atmAccessDevice/saal/atmSar(81-1) 2582581
A Sscop:atmAccessDevice/saal/sscop(78-1)
AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) 2582568
S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 2582820 2582829
A Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1)
Sscop:atmAccessDevice/saal/sscop(78-1) 2582564
A SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1)
Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) 2582561
A CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1)
SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) 2582557
A GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1)
CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) 2582553
S GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2)
GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) 2582501
2582509
A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2)
MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) 2582507
A CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1)
SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) 2582641
A SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1)
Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) 2582647
A Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1)
Sscop:atmNetwork/saal1/sscop(47-1) 2582655
A Sscop:atmNetwork/saal1/sscop(47-1) AtmCpcs:atmNetwork/saal1/atmCpcs(48-1)
2582662
A AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) AtmSar:atmNetwork/saal1/atmSar(50-1)
2582680
A AtmSar:atmNetwork/saal1/atmSar(50-1) AtmCore:atmNetwork/atmUni1/atmCore(40-
1) 2582690
A AtmSar:atmNetwork/saal2/atmSar(56-1) AtmCore:atmNetwork/atmUni2/atmCore(43-
1) 2582685
A AtmCore:atmNetwork/atmUni2/atmCore(43-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 2582703
A AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
AtmSar:atmAccessDevice/saal/atmSar(81-1) 2582724
A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) 2582732
A AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
Sscop:atmAccessDevice/saal/sscop(78-1) 2582832
A Sscop:atmAccessDevice/saal/sscop(78-1)
Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) 2582884
A Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1)
SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) 2582892
```

A SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1)
CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) 2582899
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2)
GsmMobileStation:gsm5552222(107-2) 2582515
A GsmMobileStation:gsm5552222(107-2)
GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) 2582526
A GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2)
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) 2582537
A
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) 2582545
A GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1)
GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) 2582550

APPENDIX C3- AcceptingRequest: LQN Performance Model

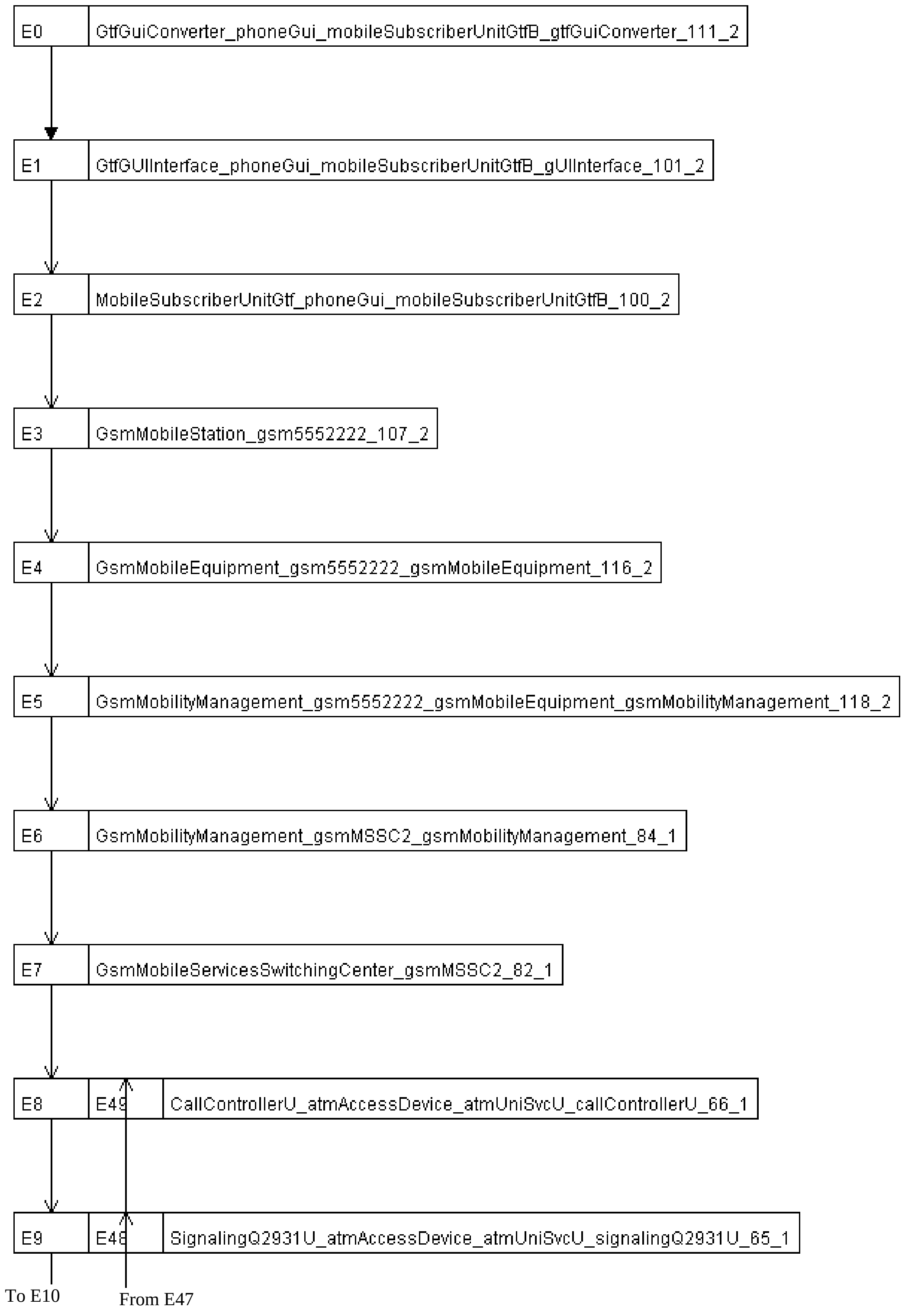
```
G
""
0.0
0
0
0.9
-1

P 0
p proc1 f
-1

T 0
t AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) f E41  -1 proc1
t AtmCore:atmAccessDevice/atmUni/atmCore(69-1) f E15 E16 E39  -1 proc1
t AtmCore:atmNetwork/atmUni1/atmCore(40-1) f E36 E37  -1 proc1
t AtmCore:atmNetwork/atmUni2/atmCore(43-1) f E18 E35  -1 proc1
t AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) f E12 E44  -1 proc1
t AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) f E31  -1 proc1
t AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) f E21  -1 proc1
t AtmSar:atmAccessDevice/saal/atmSar(81-1) f E14 E43  -1 proc1
t AtmSar:atmNetwork/saal1/atmSar(50-1) f E34  -1 proc1
t AtmSar:atmNetwork/saal2/atmSar(56-1) f E20  -1 proc1
t CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) f E27  -1 proc1
t CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) f E26  -1 proc1
t CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) f E8 E49  -1
proc1
t Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) f E10 E47  -1 proc1
t Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) f E29  -1 proc1
t Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) f E24  -1 proc1
t GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) f E4  -1 proc1
t GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) f E7  -1 proc1
t GsmMobileStation:gsm5552222(107-2) f E3  -1 proc1
t
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) f E5  -1 proc1
t GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) f E6  -1 proc1
t GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) f E1  -1
proc1
t GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) r E0  -1
proc1
t I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) f E17 E42  -1 proc1
t I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) f E40  -1 proc1
```

```
t I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) f E19 E38  -1 proc1
t I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) f E13 E45  -1 proc1
t I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) f E33  -1 proc1
t I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) f E22 E32  -1 proc1
t MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) f E2  -1 proc1
t SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) f E28  -1 proc1
t SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) f E25  -1 proc1
t SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) f E9 E48  -1
proc1
t Sscop:atmAccessDevice/saal/sscop(78-1) f E11 E46  -1 proc1
t Sscop:atmNetwork/saal1/sscop(47-1) f E30  -1 proc1
t Sscop:atmNetwork/saal2/sscop(53-1) f E23  -1 proc1
-1

E 0
s E0 1.0 -1
y E0 E1 1.0 -1
s E1 1.0 -1
z E1 E2 1.0 -1
s E10 1.0 1.0 -1
z E10 E11 0.0 1.0 -1
s E11 1.0 1.0 -1
z E11 E12 0.0 1.0 -1
s E12 1.0 1.0 -1
y E12 E13 0.0 1.0 -1
z E12 E14 0.0 1.0 -1
s E13 1.0 -1
s E14 1.0 1.0 -1
z E14 E15 0.0 1.0 -1
z E14 E16 0.0 1.0 -1
s E15 1.0 -1
s E16 1.0 1.0 -1
y E16 E17 0.0 1.0 -1
z E16 E18 0.0 1.0 -1
s E17 1.0 -1
s E18 1.0 1.0 -1
y E18 E19 0.0 1.0 -1
y E18 E20 0.0 1.0 -1
s E19 1.0 -1
s E2 1.0 1.0 -1
z E2 E3 0.0 1.0 -1
s E20 1.0 1.0 -1
y E20 E21 1.0 0.0 -1
z E20 E35 0.0 1.0 -1
s E21 1.0 -1
y E21 E22 1.0 -1
```

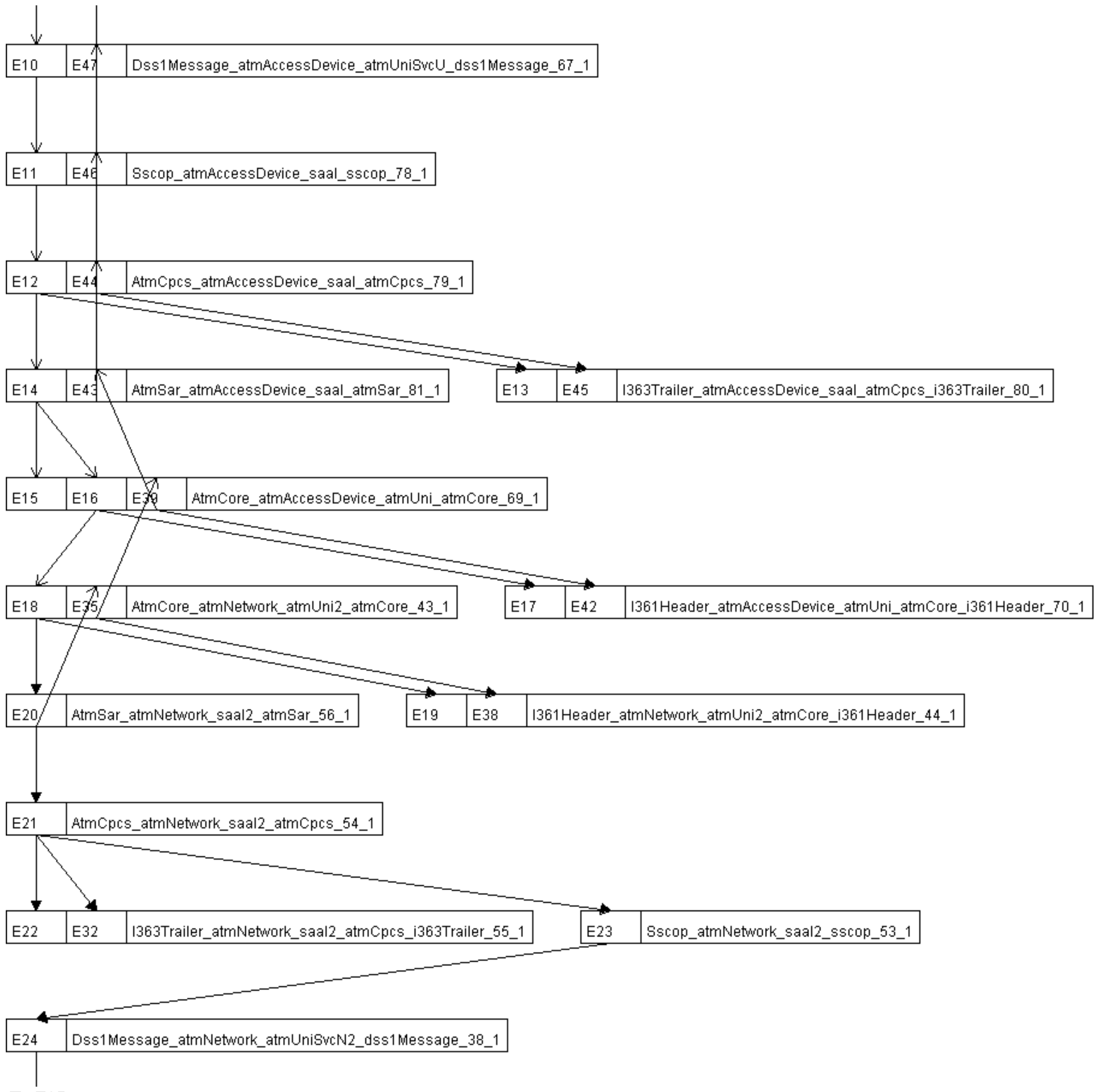
```
y E21 E23 1.0 -1
y E21 E32 1.0 -1
s E22 1.0 -1
s E23 1.0 -1
y E23 E24 1.0 -1
s E24 1.0 -1
y E24 E25 1.0 -1
s E25 1.0 -1
y E25 E26 1.0 -1
s E26 1.0 -1
z E26 E27 1.0 -1
s E27 1.0 1.0 -1
z E27 E28 0.0 1.0 -1
s E28 1.0 1.0 -1
z E28 E29 0.0 1.0 -1
s E29 1.0 1.0 -1
z E29 E30 0.0 1.0 -1
s E3 1.0 1.0 -1
z E3 E4 0.0 1.0 -1
s E30 1.0 1.0 -1
z E30 E31 0.0 1.0 -1
s E31 1.0 1.0 -1
y E31 E33 0.0 1.0 -1
z E31 E34 0.0 1.0 -1
s E32 1.0 -1
s E33 1.0 -1
s E34 1.0 1.0 -1
z E34 E36 0.0 1.0 -1
z E34 E37 0.0 1.0 -1
s E35 1.0 1.0 -1
y E35 E38 0.0 1.0 -1
z E35 E39 0.0 1.0 -1
s E36 1.0 -1
s E37 1.0 1.0 -1
y E37 E40 0.0 1.0 -1
z E37 E41 0.0 1.0 -1
s E38 1.0 -1
s E39 1.0 1.0 -1
y E39 E42 0.0 1.0 -1
z E39 E43 0.0 1.0 -1
s E4 1.0 1.0 -1
z E4 E5 0.0 1.0 -1
s E40 1.0 -1
s E41 1.0 -1
s E42 1.0 -1
s E43 1.0 1.0 -1
```

```
z E43 E44 0.0 1.0 -1
s E44 1.0 1.0 -1
y E44 E45 0.0 1.0 -1
z E44 E46 0.0 1.0 -1
s E45 1.0 -1
s E46 1.0 1.0 -1
z E46 E47 0.0 1.0 -1
s E47 1.0 1.0 -1
z E47 E48 0.0 1.0 -1
s E48 1.0 1.0 -1
z E48 E49 0.0 1.0 -1
s E49 1.0 -1
s E5 1.0 1.0 -1
z E5 E6 0.0 1.0 -1
s E6 1.0 1.0 -1
z E6 E7 0.0 1.0 -1
s E7 1.0 1.0 -1
z E7 E8 0.0 1.0 -1
s E8 1.0 1.0 -1
z E8 E9 0.0 1.0 -1
s E9 1.0 1.0 -1
z E9 E10 0.0 1.0 -1
-1
```

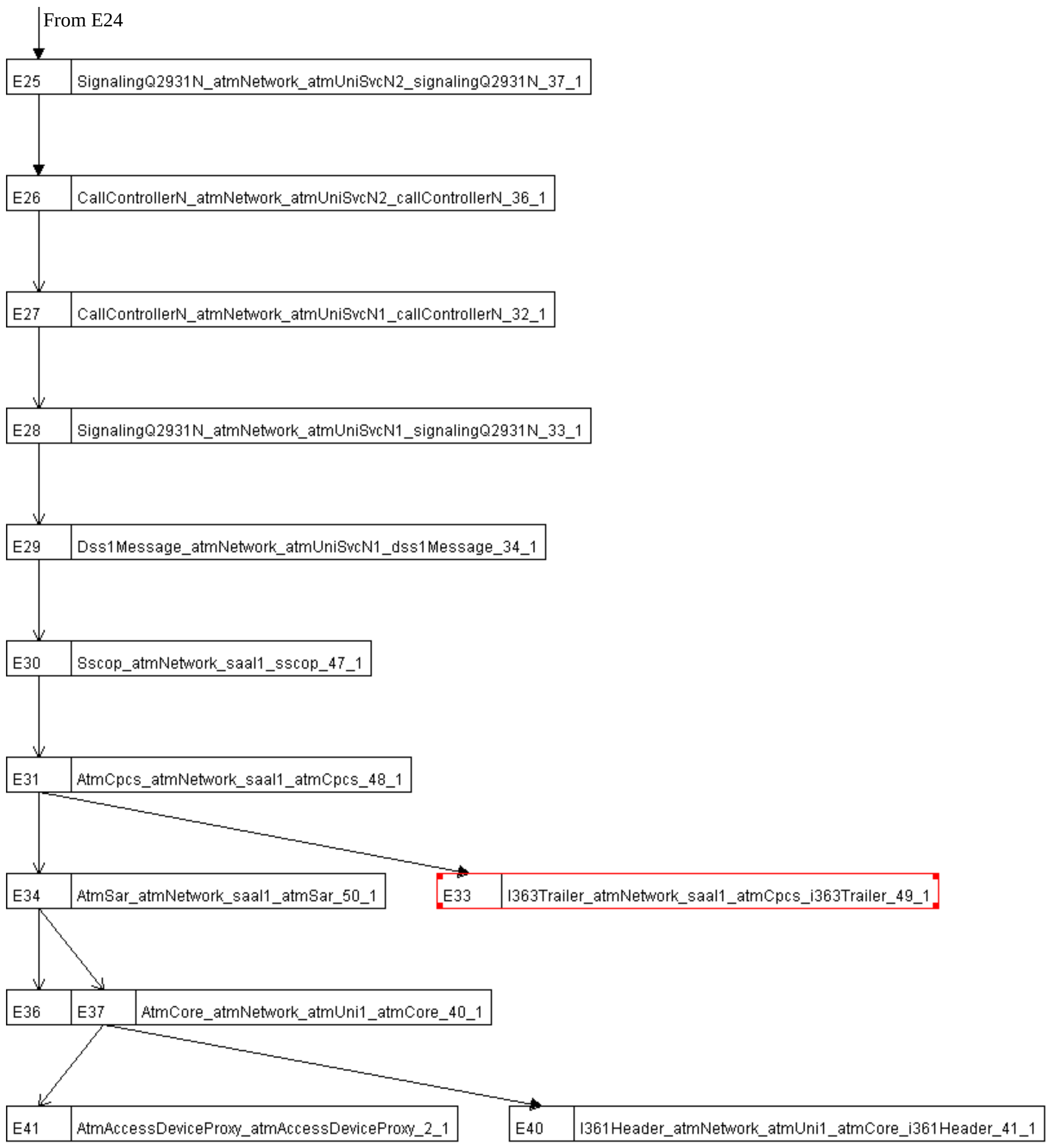
APPENDIX C4- AcceptingRequest: Graphical LQN Performance Model


E0
GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2
E1
GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2
E2
MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2
E3
GsmMobileStation_gsm5552222_107_2
E4
GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2
E5
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManagement_118_2
E6
GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1
E7
GsmMobileServicesSwitchingCenter_gsmMSSC2_82_1
E8
E49
CallControllerU_atmAccessDevice_atmUniSvcU_callControllerU_66_1
E9
E48
SignalingQ2931U_atmAccessDevice_atmUniSvcU_signalingQ2931U_65_1
To E10
From E47

From E9   to E48


E10
E47
Dss1Message_atmAccessDevice_atmUniSvcU_dss1Message_67_1
E11
E46
Sscop_atmAccessDevice_saal_sscop_78_1
E12
E44
AtmCpcs_atmAccessDevice_saal_atmCpcs_79_1
E14
E43
AtmSar_atmAccessDevice_saal_atmSar_81_1
E13
E45
I363Trailer_atmAccessDevice_saal_atmCpcs_i363Trailer_80_1
E15
E16
E39
AtmCore_atmAccessDevice_atmUni_atmCore_69_1
E18
E35
AtmCore_atmNetwork_atmUni2_atmCore_43_1
E17
E42
I361Header_atmAccessDevice_atmUni_atmCore_i361Header_70_1
E20
AtmSar_atmNetwork_saal2_atmSar_56_1
E19
E38
I361Header_atmNetwork_atmUni2_atmCore_i361Header_44_1
E21
AtmCpcs_atmNetwork_saal2_atmCpcs_54_1
E22
E32
I363Trailer_atmNetwork_saal2_atmCpcs_i363Trailer_55_1
E23
Sscop_atmNetwork_saal2_sscop_53_1
E24
Dss1Message_atmNetwork_atmUniSvcN2_dss1Message_38_1

To E25


From E24
E25
SignalingQ2931N_atmNetwork_atmUniSvcN2_signalingQ2931N_37_1
E26
CallControllerN_atmNetwork_atmUniSvcN2_callControllerN_36_1
E27
CallControllerN_atmNetwork_atmUniSvcN1_callControllerN_32_1
E28
SignalingQ2931N_atmNetwork_atmUniSvcN1_signalingQ2931N_33_1
E29
Dss1Message_atmNetwork_atmUniSvcN1_dss1Message_34_1
E30
Sscop_atmNetwork_saal1_sscop_47_1
E31
AtmCpcs_atmNetwork_saal1_atmCpcs_48_1
E34
AtmSar_atmNetwork_saal1_atmSar_50_1
E33
I363Trailer_atmNetwork_saal1_atmCpcs_i363Trailer_49_1
E36
E37
AtmCore_atmNetwork_atmUni1_atmCore_40_1
E41
AtmAccessDeviceProxy_atmAccessDeviceProxy_2_1
E40
I361Header_atmNetwork_atmUni1_atmCore_i361Header_41_1

APPENDIX D1- CallTermination: Sequence Trace

```
thread endingacall.1 {
4249758 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%End;
4249759 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/ActiveIdle;
4249759 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) receive
%End;
4249765 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send %Clr;
4249765 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
endingacall.1.1;
4249768 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send %End;
4249768 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
endingacall.1.2;
};
thread endingacall.1.1 {
4249766 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) receive
%Clr;
4249774 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%bClearDisplay;
4249778 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) state
top;
4249775 GUIProxy:phoneGui/gUIProxy(95-1) state running;
4249775 GUIProxy:phoneGui/gUIProxy(95-1) receive %bClearDisplay;
};
thread endingacall.1.2 {
4249769 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) receive %End;
4249781 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) send %End;
4249783 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) state
Initialized;
4249781 GsmMobileStation:gsm5552222(107-2) state S1;
4249781 GsmMobileStation:gsm5552222(107-2) receive %End;
4249794 GsmMobileStation:gsm5552222(107-2) send %End;
4249795 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) state S1;
4249795 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) receive %End;
4249798 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) send %CallDisconnect;
4249798 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
state S1;
4249798 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
receive %CallDisconnect;
4249802 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
send %CallDisconnect;
4249802 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) state S1;
4249802 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) receive
%CallDisconnect;
4249806 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) send %CallDisconnect;
4249806 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) state S1;
4249806 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) receive %CallDisconnect;
4249809 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) send %CallDisconnect;
4249811 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) state Ready;
4249811 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) receive
%CallDisconnect;
4249814 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) send
%DisconnectReq;
4249815 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
CanBeCleared/Active10;
4249815 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) receive
%DisconnectReq;
4249818 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send
%Disconnect;
4249820 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state
DisconnectRequest11;
4249818 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
4249818 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive %Disconnect;
4249822 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %Dss1Pdu;
4249822 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
```

```
4249822 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Dss1Pdu;
4249827 Sscop:atmAccessDevice/saal/sscop(78-1) send %Aal5Sdu;
4249827 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
4249827 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %Aal5Sdu;
4249831 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %CpcsSdu;
4249831 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
4249831 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsSdu;
4249836 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) send %CpcsPdu;
4249836 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsPdu;
4249839 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %SarSdu;
4249839 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
4249839 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %SarSdu;
4249843 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmUu1;
4249843 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn endingacall.1.2.1;
4249844 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmCellSdu;
4249844 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn endingacall.1.2.2;
};
thread endingacall.1.2.1 {
4249843 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
4249843 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmUu1;
};
thread endingacall.1.2.2 {
4249844 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
4249850 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellSdu;
4249850 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
4249850 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellSdu;
4249853 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) send %AtmCellPdu;
4249853 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
4249856 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellPdu;
4249858 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
4249856 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
4249856 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
4249861 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellPdu;
4249861 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
4249861 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellPdu;
4249864 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) send %AtmCellSdu;
4249864 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
4249867 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellSdu;
4249867 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
4249867 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %AtmCellSdu;
4249870 AtmSar:atmNetwork/saal2/atmSar(56-1) send %SarIdu;
4249871 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
4249871 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %SarIdu;
4249874 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %CpcsPdu;
4249874 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
4249874 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsPdu;
4249878 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) send %CpcsSdu;
4249878 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsSdu;
4249880 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %Aal5Sdu;
4249880 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
4249880 Sscop:atmNetwork/saal2/sscop(53-1) receive %Aal5Sdu;
4249884 Sscop:atmNetwork/saal2/sscop(53-1) send %Dss1Pdu;
4249884 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
4249884 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %Dss1Pdu;
4249887 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %Disconnect;
4249887 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
CanBeCleared/Active10;
4249887 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive %Disconnect;
4249891 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %DisconnectInd;
4249892 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state
DisconnectRequest11;
4249891 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) state Ready;
4249891 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) receive
%DisconnectInd;
4249895 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) send
%CallDisconnect;
4249895 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) spawn
endingacall.1.2.2.1;
4249896 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) send %ReleaseReq;
4249896 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) spawn
endingacall.1.2.2.2;
```

```
};
thread endingacall.1.2.2.1 {
4249895 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) state Ready;
4249895 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) receive
%CallDisconnect;
4249901 CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) send %DisconnectReq;
4249901 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) state
CanBeCleared/Active10;
4249901 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) receive
%DisconnectReq;
4249910 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) send %Disconnect;
4249911 SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) state
DisconnectIndication12;
4249910 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) state Ready;
4249910 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) receive %Disconnect;
4249917 Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) send %Dss1Pdu;
4249917 Sscop:atmNetwork/saal1/sscop(47-1) state Ready;
4249917 Sscop:atmNetwork/saal1/sscop(47-1) receive %Dss1Pdu;
4249924 Sscop:atmNetwork/saal1/sscop(47-1) send %Aal5Sdu;
4249924 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) state Ready;
4249924 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) receive %Aal5Sdu;
4249936 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) send %CpcsSdu;
4249936 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) state Ready;
4249936 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) receive %CpcsSdu;
4249940 I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) send %CpcsPdu;
4249940 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) receive %CpcsPdu;
4249943 AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) send %SarSdu;
4249943 AtmSar:atmNetwork/saal1/atmSar(50-1) state Ready;
4249943 AtmSar:atmNetwork/saal1/atmSar(50-1) receive %SarSdu;
4249953 AtmSar:atmNetwork/saal1/atmSar(50-1) send %AtmUu1;
4249953 AtmSar:atmNetwork/saal1/atmSar(50-1) spawn endingacall.1.2.2.1.1;
4249954 AtmSar:atmNetwork/saal1/atmSar(50-1) send %AtmCellSdu;
4249954 AtmSar:atmNetwork/saal1/atmSar(50-1) spawn endingacall.1.2.2.1.2;
};
thread endingacall.1.2.2.1.1 {
4249953 AtmCore:atmNetwork/atmUni1/atmCore(40-1) state Ready;
4249953 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmUu1;
};
thread endingacall.1.2.2.1.2 {
4249954 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmCellSdu;
4249972 AtmCore:atmNetwork/atmUni1/atmCore(40-1) send %AtmCellSdu;
4249973 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) state Ready;
4249973 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) receive %AtmCellSdu;
4249979 I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) send %AtmCellPdu;
4249979 AtmCore:atmNetwork/atmUni1/atmCore(40-1) receive %AtmCellPdu;
4249981 AtmCore:atmNetwork/atmUni1/atmCore(40-1) send %AtmCellPdu;
4249983 AtmCore:atmNetwork/atmUni1/atmCore(40-1) state Ready;
4249981 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
4249981 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) receive %AtmCellPdu;
4249995 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %AtmCellPdu;
4253253 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) receive %AtmCellPdu;
4253258 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) send %AtmCellPdu;
4253261 AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) state S1;
4253258 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) state 2;
4253258 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) receive %AtmCellPdu;
4253264 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) send %AtmCellPdu;
4253265 AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) state Ready;
4253265 AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) receive
%AtmCellPdu;
4253274 AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) send
%AtmCellSdu;
4253274 AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) receive %AtmCellSdu;
};
thread endingacall.1.2.2.2 {
4249896 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive %ReleaseReq;
4249905 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %Release;
4249905 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) spawn
endingacall.1.2.2.2.1;
4249906 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %timeout delay
20000;
```

```
4249906 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) spawn
endingacall.1.2.2.2.2;
4249908 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state ReleaseReq19;
};
thread endingacall.1.2.2.2.1 {
4249905 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %Release;
4249913 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %Dss1Pdu;
4249915 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
4249913 Sscop:atmNetwork/saal2/sscop(53-1) receive %Dss1Pdu;
4249920 Sscop:atmNetwork/saal2/sscop(53-1) send %Aal5Sdu;
4249921 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
4249920 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %Aal5Sdu;
4249927 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %CpcsSdu;
4249927 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsSdu;
4249931 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
4249931 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) send %CpcsPdu;
4249931 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsPdu;
4249933 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %SarSdu;
4249934 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
4249933 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %SarSdu;
4249948 AtmSar:atmNetwork/saal2/atmSar(56-1) send %AtmUu1;
4249948 AtmSar:atmNetwork/saal2/atmSar(56-1) spawn endingacall.1.2.2.2.1.1;
4249949 AtmSar:atmNetwork/saal2/atmSar(56-1) send %AtmCellSdu;
4249949 AtmSar:atmNetwork/saal2/atmSar(56-1) spawn endingacall.1.2.2.2.1.2;
4249950 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
};
thread endingacall.1.2.2.2.1.1 {
4249948 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmUu1;
4249958 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
};
thread endingacall.1.2.2.2.1.2 {
4249949 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
4249960 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellSdu;
4249960 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellSdu;
4249964 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
4249964 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) send %AtmCellPdu;
4249964 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
4249966 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellPdu;
4249968 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
4249967 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
4249986 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellPdu;
4249986 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellPdu;
4249989 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
4249989 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) send %AtmCellSdu;
4249989 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
4249992 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellSdu;
4249993 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
4249992 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %AtmCellSdu;
4250116 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %SarIdu;
4250118 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
4250116 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %SarIdu;
4250142 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %CpcsPdu;
4250142 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsPdu;
4250221 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
4250221 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) send %CpcsSdu;
4250221 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsSdu;
4250224 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %Aal5Sdu;
4250227 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
4250224 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Aal5Sdu;
4250232 Sscop:atmAccessDevice/saal/sscop(78-1) send %Dss1Pdu;
4250235 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
4250232 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive %Dss1Pdu;
4250240 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %Release;
4250242 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
4250240 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) receive
%Release;
4250281 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send
%ReleaseInd;
4250281 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) spawn
endingacall.1.2.2.2.1.2.1;
```

```
4250283 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) send
%ReleaseComplete;
4250283 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) spawn
endingacall.1.2.2.2.1.2.2;
4250286 SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) state Null0;
};
thread endingacall.1.2.2.2.1.2.1 {
4250281 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) receive
%ReleaseInd;
4250291 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) send
%CallDisconnect;
4250295 CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) state Ready;
4250291 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) receive %CallDisconnect;
4250307 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) send %CallDisconnect;
4250310 GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) state S1;
4250307 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) receive
%CallDisconnect;
4250358 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) send %CallDisconnect;
4250369 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) state S1;
4250360 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
receive %CallDisconnect;
4250393 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
send %CallDisconnect;
4250458 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
state S1;
4250393 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) receive %CallDisconnect;
4250565 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) send %End;
4250568 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) state S1;
4250565 GsmMobileStation:gsm5552222(107-2) receive %End;
4250737 GsmMobileStation:gsm5552222(107-2) send %End;
4250738 GsmMobileStation:gsm5552222(107-2) state S1;
4250737 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) receive %End;
4250757 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) send %End;
4250759 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) state
Initialized;
4250757 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) receive
%End;
4250775 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send %Clr;
4250778 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/ActiveIdle;
4250775 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) receive
%Clr;
4250794 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%bClearDisplay;
4250798 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) state
top;
4250795 GUIProxy:phoneGui/gUIProxy(95-1) receive %bClearDisplay;
};
thread endingacall.1.2.2.2.1.2.2 {
4250283 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) receive
%ReleaseComplete;
4250300 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) send %Dss1Pdu;
4250303 Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) state Ready;
4250300 Sscop:atmAccessDevice/saal/sscop(78-1) receive %Dss1Pdu;
4250314 Sscop:atmAccessDevice/saal/sscop(78-1) send %Aal5Sdu;
4250353 Sscop:atmAccessDevice/saal/sscop(78-1) state Ready;
4250315 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %Aal5Sdu;
4250373 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %CpcsSdu;
4250374 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) receive %CpcsSdu;
4250381 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) state Ready;
4250381 I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) send %CpcsPdu;
4250381 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) receive %CpcsPdu;
4250385 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) send %SarSdu;
4250388 AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) state Ready;
4250385 AtmSar:atmAccessDevice/saal/atmSar(81-1) receive %SarSdu;
4250537 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmUu1;
4250537 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn endingacall.1.2.2.2.1.2.2.1;
4250540 AtmSar:atmAccessDevice/saal/atmSar(81-1) send %AtmCellSdu;
4250540 AtmSar:atmAccessDevice/saal/atmSar(81-1) spawn endingacall.1.2.2.2.1.2.2.2;
4250543 AtmSar:atmAccessDevice/saal/atmSar(81-1) state Ready;
};
```

```
thread endingacall.1.2.2.2.1.2.2.1 {
4250537 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmUu1;
4250632 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
};
thread endingacall.1.2.2.2.1.2.2.2 {
4250540 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellSdu;
4250704 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellSdu;
4250704 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) receive %AtmCellSdu;
4250731 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) state Ready;
4250731 I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) send %AtmCellPdu;
4250731 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) receive %AtmCellPdu;
4250733 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) send %AtmCellPdu;
4250735 AtmCore:atmAccessDevice/atmUni/atmCore(69-1) state Ready;
4250733 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellPdu;
4250746 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellPdu;
4250746 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) receive %AtmCellPdu;
4250749 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) state Ready;
4250749 I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) send %AtmCellSdu;
4250749 AtmCore:atmNetwork/atmUni2/atmCore(43-1) receive %AtmCellSdu;
4250751 AtmCore:atmNetwork/atmUni2/atmCore(43-1) send %AtmCellSdu;
4250753 AtmCore:atmNetwork/atmUni2/atmCore(43-1) state Ready;
4250752 AtmSar:atmNetwork/saal2/atmSar(56-1) receive %AtmCellSdu;
4250772 AtmSar:atmNetwork/saal2/atmSar(56-1) send %SarIdu;
4250773 AtmSar:atmNetwork/saal2/atmSar(56-1) state Ready;
4250772 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %SarIdu;
4250785 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %CpcsPdu;
4250785 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) receive %CpcsPdu;
4250789 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) state Ready;
4250789 I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) send %CpcsSdu;
4250789 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) receive %CpcsSdu;
4250791 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) send %Aal5Sdu;
4250792 AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) state Ready;
4250791 Sscop:atmNetwork/saal2/sscop(53-1) receive %Aal5Sdu;
4250812 Sscop:atmNetwork/saal2/sscop(53-1) send %Dss1Pdu;
4250813 Sscop:atmNetwork/saal2/sscop(53-1) state Ready;
4250812 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) receive %Dss1Pdu;
4250828 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) send %ReleaseComplete;
4250829 Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) state Ready;
4250828 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) receive
%ReleaseComplete;
4250836 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) send %ReleaseConf;
4250837 SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) state Null0;
4250836 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) receive
%ReleaseConf;
4250843 CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) state Ready;
};
thread endingacall.1.2.2.2.2 {
};
```

APPENDIX D2- CallTermination: Communication Patterns

S GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) 4249759 4249766
S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 4249831 4249836
A AtmSar:atmAccessDevice/saal/atmSar(81-1) AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 4249843
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1) I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) 4249850 4249853
S AtmCore:atmNetwork/atmUni2/atmCore(43-1) I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 4249861 4249864
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 4249874 4249878
S AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) 4249936 4249940
A AtmSar:atmNetwork/saal1/atmSar(50-1) AtmCore:atmNetwork/atmUni1/atmCore(40-1) 4249953
S AtmCore:atmNetwork/atmUni1/atmCore(40-1) I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) 4249973 4249979
A AtmCore:atmNetwork/atmUni1/atmCore(40-1) AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) 4249981
S AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) 4253265 4253274
S SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) 4249891 4249896
A CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) 4249895
S Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) 4249887 4249905
S Sscop:atmNetwork/saal2/sscop(53-1) Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) 4249884 4249913
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) Sscop:atmNetwork/saal2/sscop(53-1) 4249880 4249920
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 4249927 4249931
S AtmSar:atmNetwork/saal2/atmSar(56-1) AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) 4249871 4249933
S AtmCore:atmNetwork/atmUni2/atmCore(43-1) AtmSar:atmNetwork/saal2/atmSar(56-1) 4249867 4249948
A AtmCore:atmAccessDevice/atmUni/atmCore(69-1) AtmCore:atmNetwork/atmUni2/atmCore(43-1) 4249856

S AtmCore:atmNetwork/atmUni2/atmCore(43-1)
I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 4249960 4249964
A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 4249844
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) 4249986 4249989
A AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
AtmSar:atmAccessDevice/saal/atmSar(81-1) 4249839
A Sscop:atmAccessDevice/saal/sscop(78-1)
AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) 4249827
S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 4250142 4250221
A Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1)
Sscop:atmAccessDevice/saal/sscop(78-1) 4249822
A SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1)
Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) 4249818
A CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1)
SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) 4249815
A GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1)
CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) 4249811
A GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1)
GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) 4249806
A
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) 4249802
A GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2)
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) 4249798
A GsmMobileStation:gsm5552222(107-2)
GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) 4249795
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2)
GsmMobileStation:gsm5552222(107-2) 4249781
A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2)
MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) 4249769
A GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2)
GUIProxy:phoneGui/gUIProxy(95-1) 4249775
S Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1)
SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) 4250240
4250283
A SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1)
CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) 4250281
S Sscop:atmAccessDevice/saal/sscop(78-1)
Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) 4250232 4250300
S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
Sscop:atmAccessDevice/saal/sscop(78-1) 4250224 4250315

```
S AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1)
I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) 4250374 4250381
S AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) 4250116 4250385
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
AtmSar:atmAccessDevice/saal/atmSar(81-1) 4249992 4250537
A AtmCore:atmNetwork/atmUni2/atmCore(43-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 4249967
S AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) 4250704 4250731
A Root AtmCore:atmNetwork/atmUni2/atmCore(43-1) 4249949
S AtmCore:atmNetwork/atmUni2/atmCore(43-1)
I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) 4250746 4250749
S AtmCpcs:atmNetwork/saal2/atmCpcs(54-1)
I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) 4250785 4250789
A AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1)
AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) 4253258
A CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1)
SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) 4249901
A SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1)
Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) 4249910
A Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1)
Sscop:atmNetwork/saal1/sscop(47-1) 4249917
A Sscop:atmNetwork/saal1/sscop(47-1) AtmCpcs:atmNetwork/saal1/atmCpcs(48-1)
4249924
A AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) AtmSar:atmNetwork/saal1/atmSar(50-1)
4249943
A AtmSar:atmNetwork/saal1/atmSar(50-1) AtmCore:atmNetwork/atmUni1/atmCore(40-
1) 4249954
A CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1)
GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) 4250291
A GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1)
GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) 4250307
A GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1)
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) 4250360
A
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) 4250393
A GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2)
GsmMobileStation:gsm5552222(107-2) 4250565
A GsmMobileStation:gsm5552222(107-2)
MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) 4250737
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2)
GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) 4250757
```

A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2)
GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) 4250775
A GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2)
GUIProxy:phoneGui/gUIProxy(95-1) 4250795
A AtmSar:atmAccessDevice/saal/atmSar(81-1)
AtmCore:atmAccessDevice/atmUni/atmCore(69-1) 4250540
A AtmCore:atmAccessDevice/atmUni/atmCore(69-1)
AtmCore:atmNetwork/atmUni2/atmCore(43-1) 4250733
A AtmCore:atmNetwork/atmUni2/atmCore(43-1) AtmSar:atmNetwork/saal2/atmSar(56-
1) 4250752
A AtmSar:atmNetwork/saal2/atmSar(56-1) AtmCpcs:atmNetwork/saal2/atmCpcs(54-1)
4250772
A AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) Sscop:atmNetwork/saal2/sscop(53-1)
4250791
A Sscop:atmNetwork/saal2/sscop(53-1)
Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) 4250812
A Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1)
SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) 4250828
A SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1)
CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) 4250836

APPENDIX D3- CallTermination: LQN Performance Model

```
G
""
0.0
0
0
0.9
-1

P 0
p proc1 f
-1

T 0
t AtmAccessDeviceProxy:atmAccessDeviceProxy(2-1) f E43  -1 proc1
t AtmCore:atmAccessDevice/atmUni/atmCore(69-1) f E16 E17 E41 E57  -1 proc1
t AtmCore:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni/atmCore(5-1) f E74  -
1 proc1
t AtmCore:atmNetwork/atmUni1/atmCore(40-1) f E38 E39  -1 proc1
t AtmCore:atmNetwork/atmUni2/atmCore(43-1) f E19 E37 E60  -1 proc1
t AtmCpcs:atmAccessDevice/saal/atmCpcs(79-1) f E13 E46  -1 proc1
t AtmCpcs:atmNetwork/saal1/atmCpcs(48-1) f E32  -1 proc1
t AtmCpcs:atmNetwork/saal2/atmCpcs(54-1) f E22 E65  -1 proc1
t AtmSar:atmAccessDevice/saal/atmSar(81-1) f E15 E45  -1 proc1
t AtmSar:atmNetwork/saal1/atmSar(50-1) f E35  -1 proc1
t AtmSar:atmNetwork/saal2/atmSar(56-1) f E21 E63  -1 proc1
t AtmUni:atmAccessDeviceProxy/atmAccessDeviceTest/atmUni(4-1) f E73  -1 proc1
t CallControllerN:atmNetwork/atmUniSvcN1/callControllerN(32-1) f E28  -1 proc1
t CallControllerN:atmNetwork/atmUniSvcN2/callControllerN(36-1) f E27 E72  -1 proc1
t CallControllerU:atmAccessDevice/atmUniSvcU/callControllerU(66-1) f E9 E51  -1
proc1
t Dss1Message:atmAccessDevice/atmUniSvcU/dss1Message(67-1) f E11 E49  -1 proc1
t Dss1Message:atmNetwork/atmUniSvcN1/dss1Message(34-1) f E30  -1 proc1
t Dss1Message:atmNetwork/atmUniSvcN2/dss1Message(38-1) f E25 E70  -1 proc1
t GUIProxy:phoneGui/gUIProxy(95-1) f E3 E69  -1 proc1
t GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) f E5 E56  -1 proc1
t GsmMobileServicesSwitchingCenter:gsmMSSC2(82-1) f E8 E52  -1 proc1
t GsmMobileStation:gsm5552222(107-2) f E4 E58  -1 proc1
t
GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement
(118-2) f E6 E54  -1 proc1
t GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) f E7 E53  -1
proc1
```

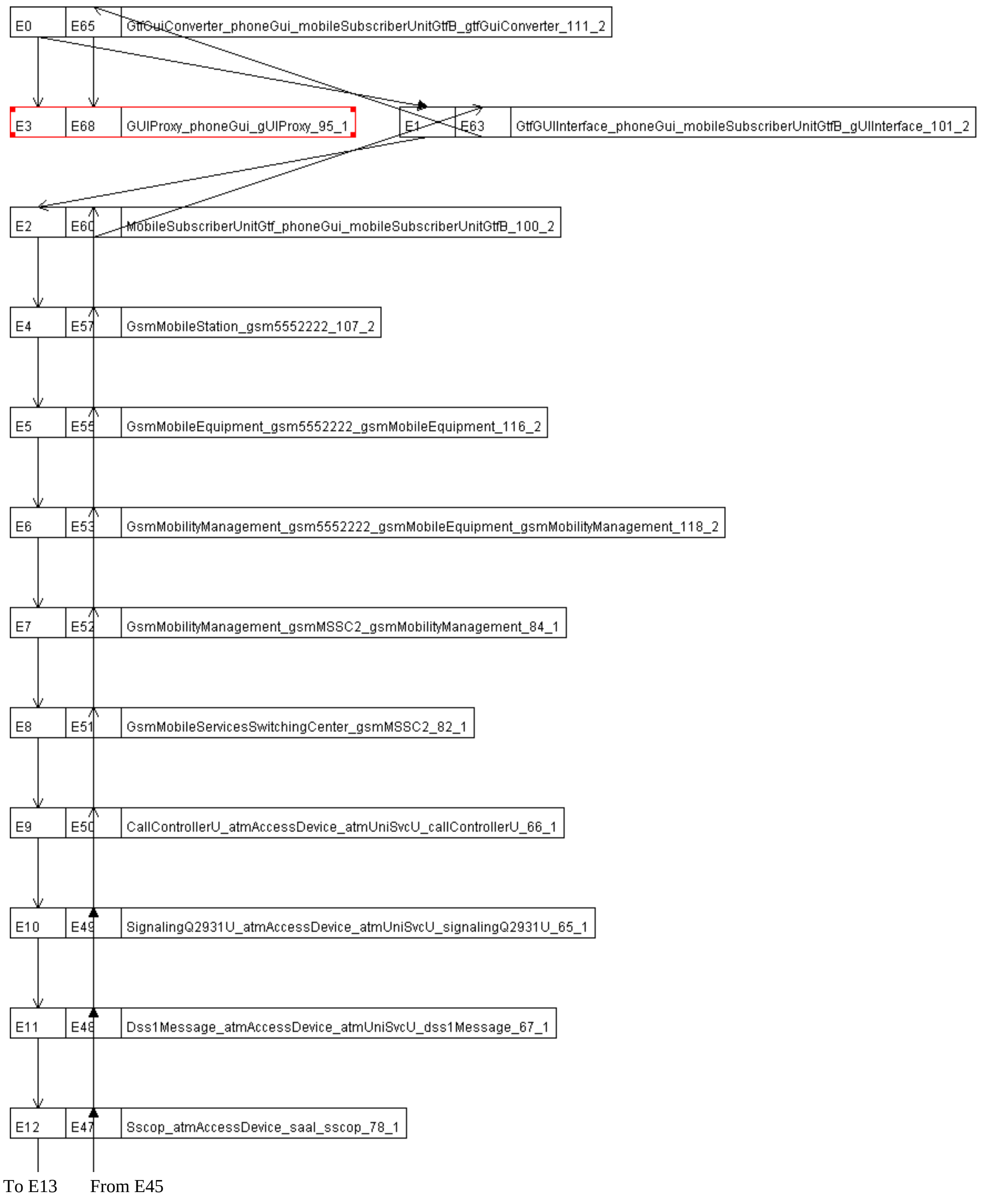
```
t GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) f E1 E64  -
1 proc1
t GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) r E0
E66  -1 proc1
t I361Header:atmAccessDevice/atmUni/atmCore/i361Header(70-1) f E18 E44 E59  -1
proc1
t I361Header:atmNetwork/atmUni1/atmCore/i361Header(41-1) f E42  -1 proc1
t I361Header:atmNetwork/atmUni2/atmCore/i361Header(44-1) f E20 E40 E62  -1 proc1
t I363Trailer:atmAccessDevice/saal/atmCpcs/i363Trailer(80-1) f E14 E47 E55  -1 proc1
t I363Trailer:atmNetwork/saal1/atmCpcs/i363Trailer(49-1) f E34  -1 proc1
t I363Trailer:atmNetwork/saal2/atmCpcs/i363Trailer(55-1) f E23 E33 E67  -1 proc1
t MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) f E2 E61  -1
proc1
t Root r E36  -1 proc1
t SignalingQ2931N:atmNetwork/atmUniSvcN1/signalingQ2931N(33-1) f E29  -1 proc1
t SignalingQ2931N:atmNetwork/atmUniSvcN2/signalingQ2931N(37-1) f E26 E71  -1
proc1
t SignalingQ2931U:atmAccessDevice/atmUniSvcU/signalingQ2931U(65-1) f E10 E50  -
1 proc1
t Sscop:atmAccessDevice/saal/sscop(78-1) f E12 E48  -1 proc1
t Sscop:atmNetwork/saal1/sscop(47-1) f E31  -1 proc1
t Sscop:atmNetwork/saal2/sscop(53-1) f E24 E68  -1 proc1
-1

E 0
s E0 1.0 1.0 -1
y E0 E1 1.0 0.0 -1
z E0 E3 0.0 1.0 -1
s E1 1.0 1.0 -1
z E1 E2 0.0 1.0 -1
s E10 1.0 1.0 -1
z E10 E11 0.0 1.0 -1
s E11 1.0 1.0 -1
z E11 E12 0.0 1.0 -1
s E12 1.0 1.0 -1
z E12 E13 0.0 1.0 -1
s E13 1.0 1.0 -1
y E13 E14 0.0 1.0 -1
z E13 E15 0.0 1.0 -1
s E14 1.0 -1
s E15 1.0 1.0 -1
z E15 E16 0.0 1.0 -1
z E15 E17 0.0 1.0 -1
s E16 1.0 -1
s E17 1.0 1.0 -1
y E17 E18 0.0 1.0 -1
```

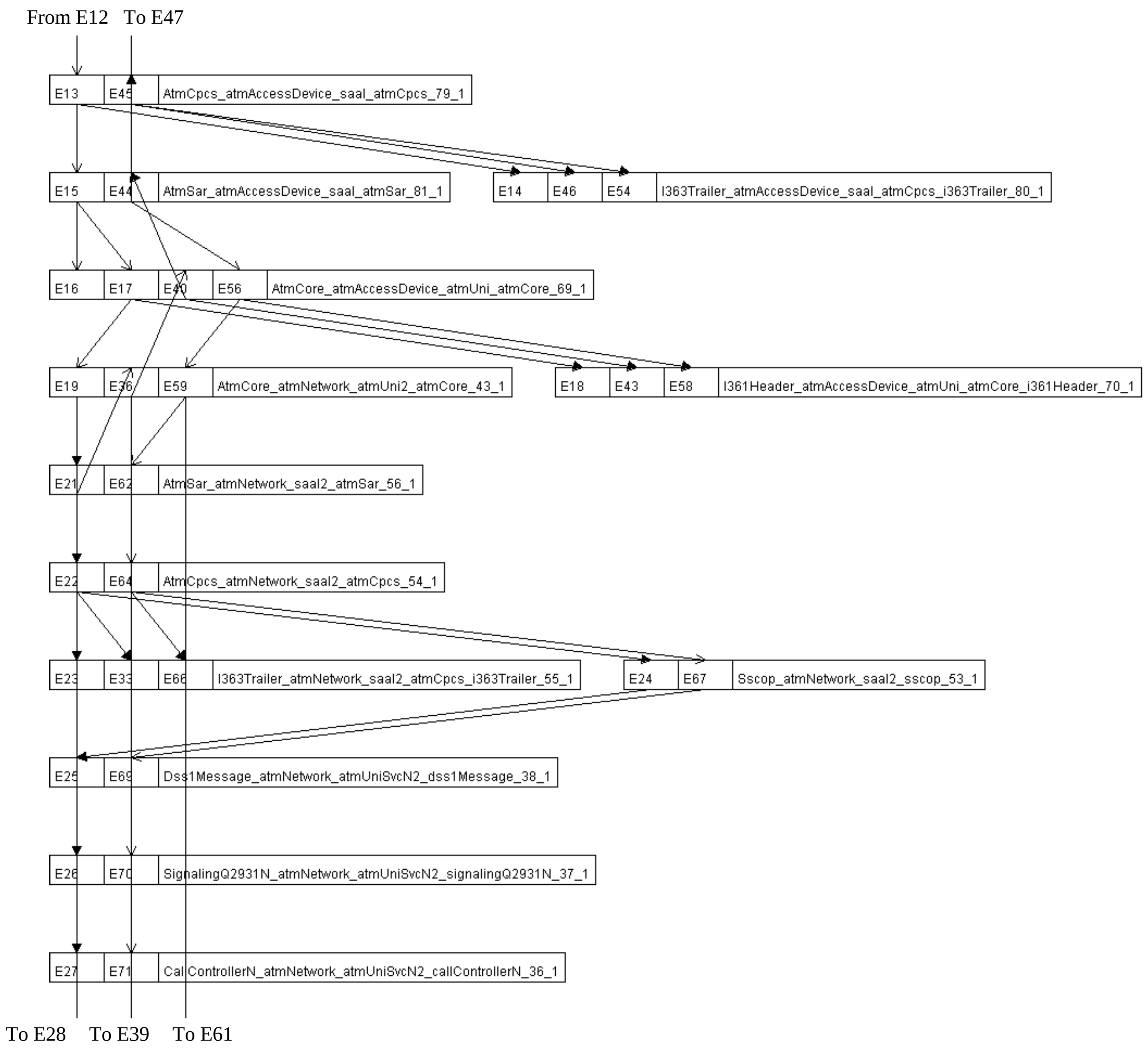
```
z E17 E19 0.0 1.0 -1
s E18 1.0 -1
s E19 1.0 1.0 -1
y E19 E20 0.0 1.0 -1
y E19 E21 0.0 1.0 -1
s E2 1.0 1.0 -1
z E2 E4 0.0 1.0 -1
s E20 1.0 -1
s E21 1.0 -1
y E21 E22 1.0 -1
s E22 1.0 -1
y E22 E23 1.0 -1
y E22 E24 1.0 -1
y E22 E33 1.0 -1
s E23 1.0 -1
s E24 1.0 -1
y E24 E25 1.0 -1
s E25 1.0 -1
y E25 E26 1.0 -1
s E26 1.0 -1
y E26 E27 1.0 -1
s E27 1.0 -1
z E27 E28 1.0 -1
s E28 1.0 1.0 -1
z E28 E29 0.0 1.0 -1
s E29 1.0 1.0 -1
z E29 E30 0.0 1.0 -1
s E3 1.0 -1
s E30 1.0 1.0 -1
z E30 E31 0.0 1.0 -1
s E31 1.0 1.0 -1
z E31 E32 0.0 1.0 -1
s E32 1.0 1.0 -1
y E32 E34 0.0 1.0 -1
z E32 E35 0.0 1.0 -1
s E33 1.0 -1
s E34 1.0 -1
s E35 1.0 1.0 -1
z E35 E38 0.0 1.0 -1
z E35 E39 0.0 1.0 -1
s E36 1.0 -1
z E36 E37 1.0 -1
s E37 1.0 1.0 -1
y E37 E40 0.0 1.0 -1
z E37 E41 0.0 1.0 -1
s E38 1.0 -1
```

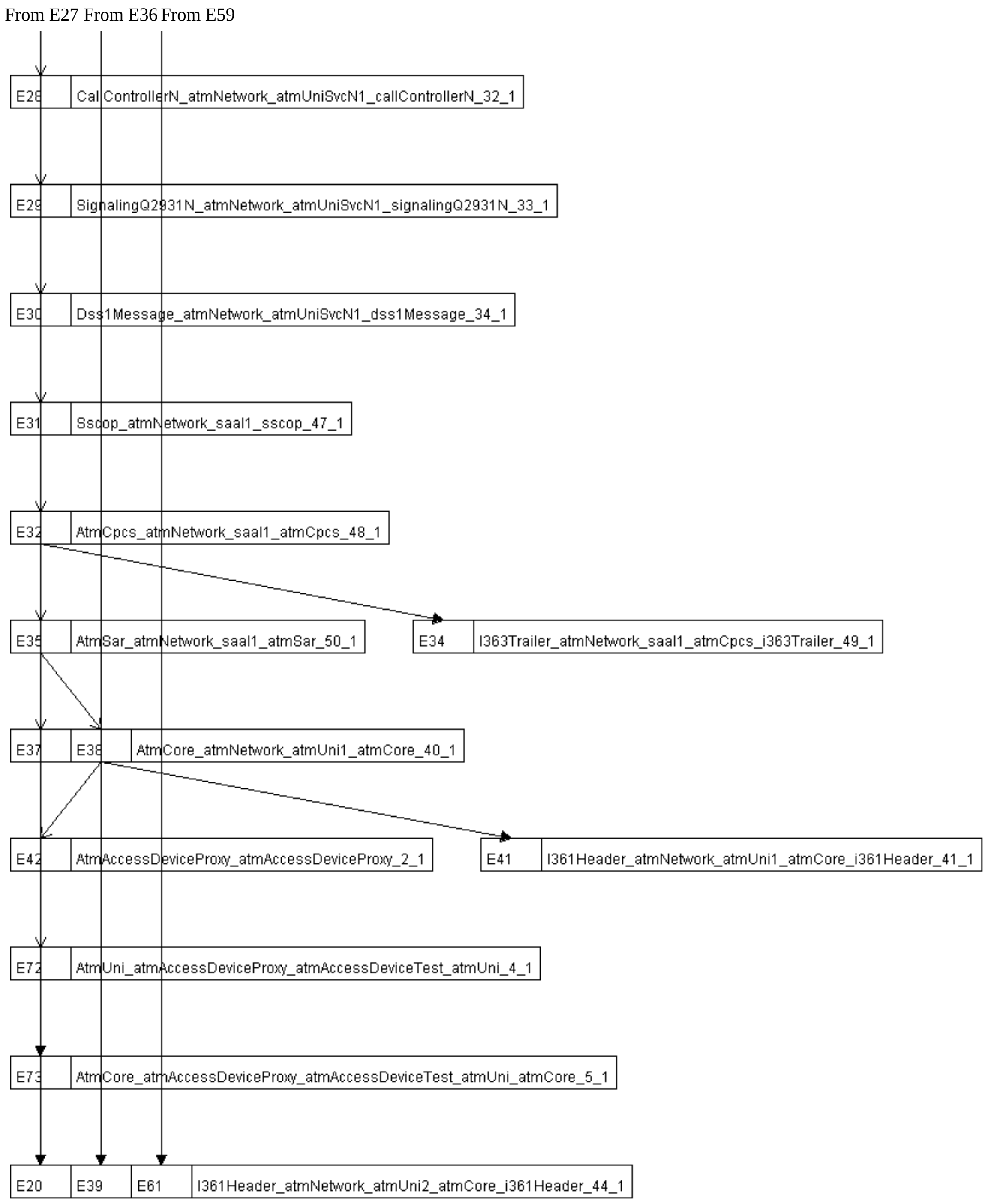
```
s E39 1.0 1.0 -1
y E39 E42 0.0 1.0 -1
z E39 E43 0.0 1.0 -1
s E4 1.0 1.0 -1
z E4 E5 0.0 1.0 -1
s E40 1.0 -1
s E41 1.0 1.0 -1
y E41 E44 0.0 1.0 -1
y E41 E45 0.0 1.0 -1
s E42 1.0 -1
s E43 1.0 1.0 -1
z E43 E73 0.0 1.0 -1
s E44 1.0 -1
s E45 1.0 1.0 -1
y E45 E46 1.0 0.0 -1
z E45 E57 0.0 1.0 -1
s E46 1.0 -1
y E46 E47 1.0 -1
y E46 E48 1.0 -1
y E46 E55 1.0 -1
s E47 1.0 -1
s E48 1.0 -1
y E48 E49 1.0 -1
s E49 1.0 -1
y E49 E50 1.0 -1
s E5 1.0 1.0 -1
z E5 E6 0.0 1.0 -1
s E50 1.0 -1
z E50 E51 1.0 -1
s E51 1.0 1.0 -1
z E51 E52 0.0 1.0 -1
s E52 1.0 1.0 -1
z E52 E53 0.0 1.0 -1
s E53 1.0 1.0 -1
z E53 E54 0.0 1.0 -1
s E54 1.0 1.0 -1
z E54 E56 0.0 1.0 -1
s E55 1.0 -1
s E56 1.0 1.0 -1
z E56 E58 0.0 1.0 -1
s E57 1.0 1.0 -1
y E57 E59 0.0 1.0 -1
z E57 E60 0.0 1.0 -1
s E58 1.0 1.0 -1
z E58 E61 0.0 1.0 -1
s E59 1.0 -1
```

```
s E6 1.0 1.0 -1
z E6 E7 0.0 1.0 -1
s E60 1.0 1.0 -1
y E60 E62 0.0 1.0 -1
z E60 E63 0.0 1.0 -1
s E61 1.0 1.0 -1
z E61 E64 0.0 1.0 -1
s E62 1.0 -1
s E63 1.0 1.0 -1
z E63 E65 0.0 1.0 -1
s E64 1.0 1.0 -1
z E64 E66 0.0 1.0 -1
s E65 1.0 1.0 -1
y E65 E67 0.0 1.0 -1
z E65 E68 0.0 1.0 -1
s E66 1.0 1.0 -1
z E66 E69 0.0 1.0 -1
s E67 1.0 -1
s E68 1.0 1.0 -1
z E68 E70 0.0 1.0 -1
s E69 1.0 -1
s E7 1.0 1.0 -1
z E7 E8 0.0 1.0 -1
s E70 1.0 1.0 -1
z E70 E71 0.0 1.0 -1
s E71 1.0 1.0 -1
z E71 E72 0.0 1.0 -1
s E72 1.0 -1
s E73 1.0 1.0 -1
y E73 E74 0.0 1.0 -1
s E74 1.0 -1
s E8 1.0 1.0 -1
z E8 E9 0.0 1.0 -1
s E9 1.0 1.0 -1
z E9 E10 0.0 1.0 -1
-1
```

Appendix D4- CallTermination: Graphical LQN Performance Model


E0
E65
GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2
E3
E68
GUIProxy_phoneGui_gUIProxy_95_1
E1
E63
GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2
E2
E60
MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2
E4
E57
GsmMobileStation_gsm5552222_107_2
E5
E55
GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2
E6
E53
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManagement_118_2
E7
E52
GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1
E8
E51
GsmMobileServicesSwitchingCenter_gsmMSSC2_82_1
E9
E50
CallControllerU_atmAccessDevice_atmUniSvcU_callControllerU_66_1
E10
E49
SignalingQ2931U_atmAccessDevice_atmUniSvcU_signalingQ2931U_65_1
E11
E48
Dss1Message_atmAccessDevice_atmUniSvcU_dss1Message_67_1
E12
E47
Sscop_atmAccessDevice_saal_sscop_78_1
To E13
From E45


From E12
To E47
E13
E45
AtmCpcs_atmAccessDevice_saal_atmCpcs_79_1
E14
E46
E54
I363Trailer_atmAccessDevice_saal_atmCpcs_i363Trailer_80_1
E15
E44
AtmSar_atmAccessDevice_saal_atmSar_81_1
E16
E17
E40
E56
AtmCore_atmAccessDevice_atmUni_atmCore_69_1
E19
E36
E59
AtmCore_atmNetwork_atmUni2_atmCore_43_1
E18
E43
E58
I361Header_atmAccessDevice_atmUni_atmCore_i361Header_70_1
E21
E62
AtmSar_atmNetwork_saal2_atmSar_56_1
E22
E64
AtmCpcs_atmNetwork_saal2_atmCpcs_54_1
E23
E33
E66
I363Trailer_atmNetwork_saal2_atmCpcs_i363Trailer_55_1
E24
E67
Sscop_atmNetwork_saal2_sscop_53_1
E25
E69
Dss1Message_atmNetwork_atmUniSvcN2_dss1Message_38_1
E26
E70
SignalingQ2931N_atmNetwork_atmUniSvcN2_signalingQ2931N_37_1
E27
E71
CallControllerN_atmNetwork_atmUniSvcN2_callControllerN_36_1
To E28
To E39
To E61


From E27
From E36
From E59
E28
CallControllerN_atmNetwork_atmUniSvcN1_callControllerN_32_1
E29
SignalingQ2931N_atmNetwork_atmUniSvcN1_signalingQ2931N_33_1
E30
Dss1Message_atmNetwork_atmUniSvcN1_dss1Message_34_1
E31
Sscop_atmNetwork_saal1_sscop_47_1
E32
AtmCpcs_atmNetwork_saal1_atmCpcs_48_1
E35
AtmSar_atmNetwork_saal1_atmSar_50_1
E34
I363Trailer_atmNetwork_saal1_atmCpcs_i363Trailer_49_1
E37
E38
AtmCore_atmNetwork_atmUni1_atmCore_40_1
E42
AtmAccessDeviceProxy_atmAccessDeviceProxy_2_1
E41
I361Header_atmNetwork_atmUni1_atmCore_i361Header_41_1
E72
AtmUni_atmAccessDeviceProxy_atmAccessDeviceTest_atmUni_4_1
E73
AtmCore_atmAccessDeviceProxy_atmAccessDeviceTest_atmUni_atmCore_5_1
E20
E39
E61
I361Header_atmNetwork_atmUni2_atmCore_i361Header_44_1

Appendix E1- PowerDown: Sequence Trace

```
thread PowerOff.1 {
1320599 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%bPwrOffAck;
1320599 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) spawn
PowerOff.1.1;
1320604 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%PowerDown;
1320604 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) spawn
PowerOff.1.2;
};
thread PowerOff.1.1 {
1320601 GUIProxy:phoneGui/gUIProxy(95-1) state running;
1320601 GUIProxy:phoneGui/gUIProxy(95-1) receive %bPwrOffAck;
};
thread PowerOff.1.2 {
1320604 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state
PowerIdle/ActiveIdle;
1320604 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) receive
%PowerDown;
1320614 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send
%PowerDown;
1320614 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
PowerOff.1.2.1;
1320617 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) send %Clr;
1320617 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) spawn
PowerOff.1.2.2;
1320621 GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) state Idle;
};
thread PowerOff.1.2.1 {
1320614 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) receive
%PowerDown;
1320624 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) send %PowerDown;
1320626 MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) state
Initialized;
1320624 GsmMobileStation:gsm5552222(107-2) state S1;
1320624 GsmMobileStation:gsm5552222(107-2) receive %PowerDown;
1320633 GsmMobileStation:gsm5552222(107-2) send %PowerDown;
1320638 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) state S1;
1320638 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) receive %PowerDown;
1320647 GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) send %PowerDown;
1320647 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
state S1;
1320647 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
receive %PowerDown;
1320651 GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement(118-2)
send %PowerDown;
1320652 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) state S1;
1320652 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) receive %PowerDown;
1320656 GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) send %PowerDown;
1320656 GsmNetworkRegistry:gsmRegistry(57-1) state Ready;
1320656 GsmNetworkRegistry:gsmRegistry(57-1) receive %PowerDown;
};
thread PowerOff.1.2.2 {
1320618 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) receive
%Clr;
1320628 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) send
%bClearDisplay;
1320630 GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) state
top;
1320628 GUIProxy:phoneGui/gUIProxy(95-1) receive %bClearDisplay;
};
```

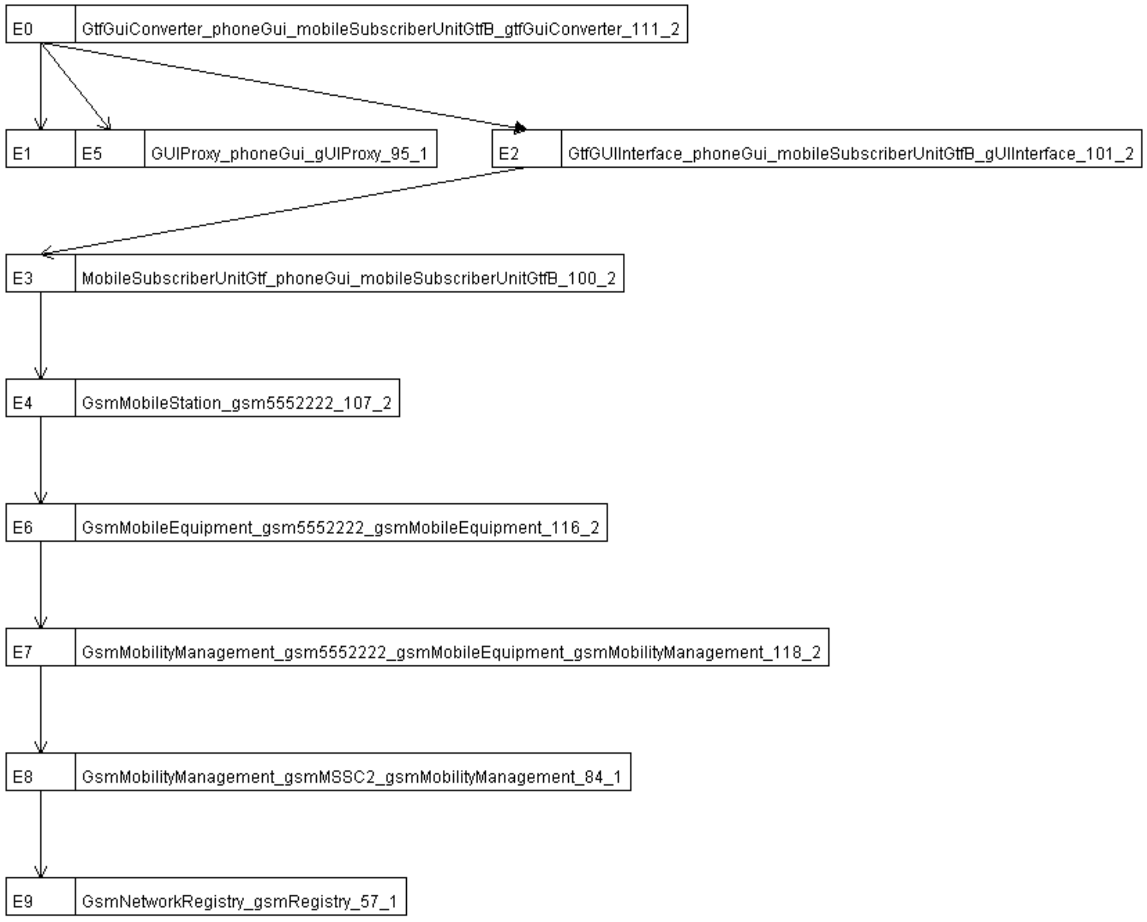
Appendix E2- PowerDown: Communication Patterns

A GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) GUIProxy:phoneGui/gUIProxy(95-1) 1320601
S GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) 1320604 1320618
A GtfGUIInterface:phoneGui/mobileSubscriberUnitGtfB/gUIInterface(101-2) MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) 1320614
A MobileSubscriberUnitGtf:phoneGui/mobileSubscriberUnitGtfB(100-2) GsmMobileStation:gsm5552222(107-2) 1320624
A GsmMobileStation:gsm5552222(107-2) GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) 1320638
A GsmMobileEquipment:gsm5552222/gsmMobileEquipment(116-2) GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement (118-2) 1320647
A GsmMobilityManagement:gsm5552222/gsmMobileEquipment/gsmMobilityManagement (118-2) GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) 1320652
A GsmMobilityManagement:gsmMSSC2/gsmMobilityManagement(84-1) GsmNetworkRegistry:gsmRegistry(57-1) 1320656
A GtfGuiConverter:phoneGui/mobileSubscriberUnitGtfB/gtfGuiConverter(111-2) GUIProxy:phoneGui/gUIProxy(95-1) 1320628

Appendix E3- PowerDown: LQN Performance Model

```
G
"" 9.0E-4 1  1 0.9 -1

P 1
p proc1 f
-1

T 9
t GUIProxy_phoneGui_gUIProxy_95_1 f E1 E5  -1 proc1
t GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2 f E6  -1 proc1
t GsmMobileStation_gsm5552222_107_2 f E4  -1 proc1
t
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManageme
nt_118_2 f E7  -1 proc1
t GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1 f E8  -1 proc1
t GsmNetworkRegistry_gsmRegistry_57_1 f E9  -1 proc1
t GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2 f E2  -1
proc1
t GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2 r E0
-1 proc1
t MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2 f E3  -1 proc1
-1

E 10
s E1 1.0 -1
s E5 1.0 -1
s E6 1.0 1.0 -1
z E6 E7 0.0 1.0 -1
s E4 1.0 1.0 -1
z E4 E6 0.0 1.0 -1
s E7 1.0 1.0 -1
z E7 E8 0.0 1.0 -1
s E8 1.0 1.0 -1
z E8 E9 0.0 1.0 -1
s E9 1.0 -1
s E2 1.0 -1
z E2 E3 1.0 -1
s E0 1.0 1.0 -1
z E0 E1 1.0 0.0 -1
y E0 E2 0.0 1.0 -1
z E0 E5 0.0 1.0 -1
s E3 1.0 1.0 -1
z E3 E4 0.0 1.0 -1
-1
```

Appendix E4- PowerDown: LQN Performance Graphical Model


E0
GtfGuiConverter_phoneGui_mobileSubscriberUnitGtfB_gtfGuiConverter_111_2
E1
E5
GUIProxy_phoneGui_gUIProxy_95_1
E2
GtfGUIInterface_phoneGui_mobileSubscriberUnitGtfB_gUIInterface_101_2
E3
MobileSubscriberUnitGtf_phoneGui_mobileSubscriberUnitGtfB_100_2
E4
GsmMobileStation_gsm5552222_107_2
E6
GsmMobileEquipment_gsm5552222_gsmMobileEquipment_116_2
E7
GsmMobilityManagement_gsm5552222_gsmMobileEquipment_gsmMobilityManagement_118_2
E8
GsmMobilityManagement_gsmMSSC2_gsmMobilityManagement_84_1
E9
GsmNetworkRegistry_gsmRegistry_57_1