ERICSSON TELECOM AB

Simulation of an ATM based PNNI switch system

Jorge Abarca Pereda
Royal Institute of Technology, Sweden

Master's Thesis in
Computer Science
September 1997

Supervisor: Kent Konradsson
Section: Network Access Signalling
Department: ATM broadband applications and product development

Examiner: Enn Tyugu
Supervisor: Vladimir Vlasov
Department of Teleinformatics
Royal Institute of Technology, Sweden

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	**Jorge Abarca Pereda**		
Dokansv/Godkänd - Doc respons/Approved **Kent Konradsson ETX/XBN**	Kontr/Checked	Datum/Date **1997-11-12** Rev	Tillhör/Referens-File/Reference

Master Thesis: Simulation of an ATM based PNNI switch network system

Abstract

This master thesis presents a simulation of an ATM network system based on PNNI (Private Network-to-Network Interface or Private Network Node Interface) protocol. PNNI is designed for use between private ATM switches and between groups of private ATM switches. PNNI is specified by ATM forum and is the chosen protocol for networking and resource management with respect to routing and signalling.

PNNI includes two types of protocols, a routing and a signalling protocol. The routing protocol is dynamic, which means that a node or a switch will announce or update information regarding own identity, resources on outgoing links and in node, and address reachability to all neighbouring nodes. The topology information is distributed between switches and clusters of switches. The signalling protocol is defined for messaging flow and is used to establish point-to-point and point-to-multipoint connections across the ATM network.

The simulation program is implemented in Erlang. The simulation technique is called Event Driven Simulation. This thesis focuses on the routing protocol. Dijkstra routing algorithm was used to build up the routing table. The simulation techique was used to set the functionality of the PNNI ATM based switch system. We could vary parameters of a link and look at the behaviour of the system to collect the statistic needed.

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	Jorge Abarca Pereda		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		1997-11-12	

Table of contents

1 Introduction

This document is the report of a master thesis work for Master of Science in Computer Science at the Department of Teleinformatics at the Royal Institute of Technology, Stockholm. The work was done at Ericsson Telecom AB, department of ATM broadband applications and product development, at the section of Network Access Signalling (referred to as XBN).

1.1 Overview

PNNI adopts source routing technique for its path selection. In source routing, the originating (or source) node selects the path to the destination node, and other nodes on the path obey the source's routing instruction. This means that the consequence of an inefficient routing decision will affect a connection for as long as that connection remains open. Thus the shortest-path-first (or Dijkstra's) algorithm is widely used for calculation of the best paths from the originating node to every other node in the network viewed from the originating node. Several studies have shown that the shortest-path-algorithm demands a lot of processor capacity, and it takes time to find the best paths, especially for multi-level hierarchical PNNI networks with many nodes.

1.2 The scope of Master Thesis work

The objective of this thesis was to simulate a PNNI ATM based network system. It was necessary to model and implement such network in a way that could give the characteristics and functionality of PNNI. The execution of the simulation program should give a view of how the system behaves under the presence of a certain number of parameters. The network should be dimensioned i.e. the parameters of nodes and links should be set in concordance to a real system. The value of each parameter within the dimensioned network could have repercussions on network traffic management.

The system developed in this project should be able to simulate the behaviour of an ATM switch network under the presence of thousands of calls incoming to the network through its nodes, each representing an autonomous switch with a specific arrival random interval intensity, which is modelled to be of Poisson type. A switch takes time to manage a call request in order to send it to the apropiate neighbour node. This time delay is insignificant if the network link has simulated distance in terms of time required for a call to pass through the link.

An important feature of PNNI is the ability to calculate the routes in a dynamic form, that is, continuously announce and update information about the topology (i.e. nodes, links, resources, reachability) with its neighbours. This form of interchange of information between the nodes makes necessary for a node to have a topology database over all the system. In the real system a minor change in the topology of the network updates immediately the topology databases of all nodes in the system . In this thesis a global database was maintained which contained the system topology, and a minor change in it could be viewed by all nodes.

An ATM connection implies a traffic contract which essentially specifies the capacity, behaviour and Quality of Service (QoS), for that connection. The traffic contract is maintained for the lifetime of the connection. To this purpose the most interesting QoS for the objective of the thesis is service category Variable Bit Rate (VBR). Under VBR, when after the path connection is done, differents amounts of data are sent at different times. It's necessary to investigate a least-cost route that can be accepted by the Generic Connection Admission Control (GCAC), which will check for the requested path, from information stored in the networks topology, if the links specified by the requested route have enough resources available to support the proposed connection. The least-cost route which takes into account the most interested parameters will be investigated trough this work.

The remainder of the report is structured as follows:

Chapter 2. Terminology.

Chapter 3. Presents the theory of PNNI and Source Routing.

Chapter 4. Presents GCAC as the admission control during the connection setup phase, to determine if a connection request can be accepted without violating QoS and throughput requirements.

Chapter 5. Presents the ATM topology state parameters.

Chapter 6. Presents how the Routing Algorithm is implemented and what routing policy it follows.

Chapter 7. Presents the theory behind Event driven simulation.

Chapter 8. Presents how event driven simulation is implemented.

Chapter 9. Presents concrete rsults based on simulations of 2 types: for telephony service and video service.

2 Terminology

2.1 Abbreviations

Abbreviation	Explanation
ABR	Available Bit Rate
ATM	Asynchronous Transfer Mode
ACR	Available Cell Rate
AW	Administrative Weight
CAC	Connection Admission Control
CBR	Constant Bit Rate
CDV	Cell Delay Variation
CLR	Cell Loss Ratio
CRM	Cell Rate Margin
CTD	Cell Transfer Delay
DTL	Designated Transit List
GCAC	Generic Connection Admission Control
IG	Information Group
MCR	Maximum Cell Rate
maxCTD	Maximum Cell Transfer Delay
MBS	Maximum Burst Size
OSPF	Open Shortest Path First
PCR	Peak Cell Rate
PGL	Peer Group Leader
PTSE	PNNI Topology State Element
PTSP	PNNI Topology State Packet
PNNI	Private Network-to-Network Interface
QoS	Quality of Service
SCR	Sustainable Cell Rate
UBR	Unspecified Bit Rate
UNI	User to Network Interface
VBR	Variable Bit Rate
VBRnrt	Variable Bit Rate non real time
VBRrt	Variable Bit Rate real time

Abbreviation	Explanation
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier

2.2 Definitions

Term	Explanation
Address prefix	A string of 0 or more bits up to a maximum of 152 bits that is the lead portion of one or more ATM addresses.
Aggregation token	A number assigned to an outside link by the border nodes at the end of the outside link. The same number is associated with all uplinks and induced uplinks associated with the outside link. In the parent and all higher-level peer groups, all uplinks with the same aggregation token are aggregated.
ATM address	A twenty-byte address used to identify an ATM connection termination point.
Border node	A logical node that is in a specified peer group, and has at least one link that crosses the peer group boundary.
Complex node representation	A collection of nodal state parameters that provide detailed state information associated with a logical node.
Cost	The word cost is used in this document to mean all resources and quality of a link or a node.
Crankback	A mechanism for partially releasing a connection setup in progress which has encountered a failure. This mechanism allows PNNI to perform alternate routing
Default cost	The cost given by the default node representation.
Default node representation	A single value of each nodal state parameter giving the presumed value between any entry to or exit from the logical node and the nucleus.
Designated transit list	A list of nodes and optionally link IDs that completely specify a path across a single PNNI group.
Dijkstra's algorithm	An algorithm that is sometimes used to calculate routes given a link and nodal state topology database.
End system	A system on which connection termination points are located.

Term	Explanation
Exception	A connectivity advertisement in a PNNI complex node representation that represents something other than the default node representation.
Exterior link	A link which crosses the boundary of the PNNI routing domain. The PNNI protocol does not run over an exterior link.
Exterior reachable address	An address that can be reached through a PNNI routing domain, but is not located in that PNNI routing domain.
External node	A node located in another peer group.
Generic connection admission control	Algorithm to decide if a node is likely to accept the call or not.
Hierarchically complete source node	A stack of DTLs representing a route across a PNNI routing domain such that a DTL is included for each hierarchical level between and including the current level and the lowest visible level in which the source and destination are reachable.
Horizontal link	A link between two nodes in the same peer group (uni - directional).
Inport	A port where a given route enters a node.
Internal reachable address	An address that can be reached through a PNNI routine domain and which is located in that PNNI routing domain.
Logical group node	A logical node that represents a lower level peer group as a single point for purposes of operation at one level of the PNNI routing hierarchy.
Logical node	An abstract representation of a peer group or of a switch system as a single point.
Neighbour node	A node that is connected to a particular node via a link
Nodal state parameter	Information that captures an aspect or property of a node.
Node	Synonymous with logical node.
Nucleus	The interior reference point of a logical node in the PNNI complex node representation.
Outport	A port where a given route leaves a node.
Peer group	A set of logical nodes which are grouped for purposes of creating a routing hierarchy. PTSEs are exchanged among all members of the group.
PNNI topology state element	A collection of PNNI information that is flooded among all logical nodes between a peer group.

Term	Explanation
PNNI topology state packet	A type of PNNI routing packet that is used for flooding PTSEs among logical nodes within a peer group.
Port identifier	The identifier assigned by a logical node to represent the point of attachment of a link to a that node.
Scope	A range describing what levels information shall be distributed to.
Spoke	A connection in a complex node representation between the nucleus and a port representing links to other nodes
Switching systems	A set of or more systems that act together and appear as a single switch for the purposes of PNNI routing
Topology aggregation	The database that describes the entire PNNI routing domain as seen by a node.
Topology database	The database that describes the entire PNNI routing domain as seen by a node.
Uplink	Represents the connectivity from a border node to a node at a higher hierarchical level (unidirectional).

3 PNNI routing description

The PNNI protocol is defined for use between private ATM switches and between private ATM networks. The abbreviation PNNI stands for either Private Network Node Interface or Private Network-to-Network Interface, reflecting these two possible usage [1]. PNNI routing applies to private network of ATM switching systems.

PNNI includes a routing protocol for distribution of topology information between switches and groups of switches. A hierarchy mechanism supports scaling of the protocol. Topology and routing are based on the fact that all nodes have a copy of the network map, which is regularly updated.

3.1 PNNI overview

The functions of the PNNI routing protocol include:

- Discovery of neighbours and link status.
- Synchronization of topology databases.
- Flooding of PNNI Topology State Elements (PTSEs).
- Election of Peer Group Leaders (PGLs).
- Summarization of topology state and address reachability information.
- Construction of the routing hierarchy.

PNNI establishment of calls between origin-destination pairs in the network, applies to a network of lowest-level nodes (fig. 1 illustrates such a network). This is referred as signalling in PNNI. Data passes between lowest-level nodes and reaches an ATM end user at an end system. End systems are points of origin and termination of connection. But when the source routes are computed to be able to handle calls, PNNI routing applies to a network of hierarchical structure, where the nodes are both lowest-level nodes and higher-level nodes.

3.2 PNNI hierarchy

In this work the use of logical node (see definition) and node in PNNI are meant to be treated equal with the same meaning.

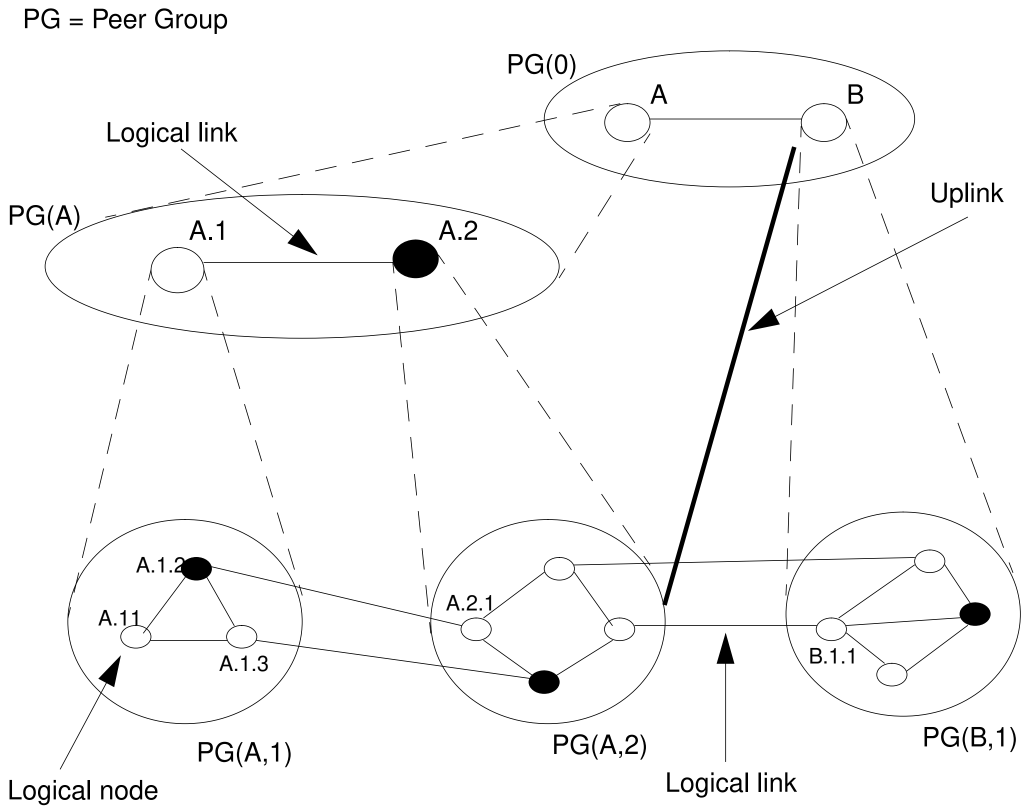


Figure 1: *A real network is represented by PG(A,1), PGH(A,2) and PG(B,1), the aggregation to a higher level (dashed lines) and the higher lines (thin lines).*

3.2.1 Lowest Hierarchical level

If the PNNI protocol only supported the flat network representation, then each lowest-level node has to maintain the entire topology of the network, including information for every physical link in the network and reachability information for every node in the network. This could create enormous overhead for larger networks. The PNNI routing hierarchy is designed to reduce this overhead while providing for efficient routing.

The PNNI hierarchy begins at lowest level, where the lowest-level nodes are organized into peer groups. A logical node in the context of the lowest hierarchy level is a lowest-level node. A peer group (PG) is a collection of logical nodes, each of which exchanges information with other members of the group, such that all members maintain an identical view of the group and ancestor peer groups (which are represented as logical group nodes). This means that nodes outside the peer group only see a single node, not the entire group. Logical nodes are uniquely identified by logical node IDs. A peer group is identified by its peer group identifier. A border node has at least a link that crosses the peer group boundary.

Logical nodes are connected by logical links. Between lowest level nodes, a logical link is either a physical link or a VPC between two lowest-level nodes. Link between lowest-level nodes within the same peer group are not aggregated. Logical links inside a peer group are inside (horizontal) links whereas links that connect two peer groups are outside links.

PNNI Hello protocol support among other things the exchange of peer group IDs so that neighbouring nodes can determine whether they belong to a same peer group or to a different peer groups. Each node exchanges Hellos packets with its immediate neighbour and thereby determine its identity and link status. This information includes the ATM address of node, node identity and peer group membership of the nodes immediate neighbours and in case of outside links, information about the status of its links(uplink information) to the neighbours.

Each node bounds its state information in PNNI Topology State Elemnts (PTSE), which are reliably flooded throughout the peer group and child peer groups. PTSEs contain among other things, topology state parameters (see 5.2). Topology state parameters includes link state parameters, which describes the resources of nodes.

Topology state information is distributed through the network as information groups (IGs) encapsulated in PTSEs. PTSEs are the smallest connection of PNNI routing information that is flooded as a unit among all logical nodes within a peer group. A node's topology database consist of a collection of all PTSEs received, which represent that node's present view of the PNNI routing domain. In particular the topology database provides all the information required to compute a route from the given node to any adress reachable within or through that routing domain.

Each node generates a PTSE that describes its own identity and capabilities, information used to elect, the peer group leader, as well as information used in establishing the PNNI hierarchy. This is referred as nodal information.

Reachability information consists of addresses and adress prefixes which describe the destinations to which calls may be routed. This information is advertised in PTSEs by nodes in the PNNI routing domain.

Flooding is the reliable hop-by-hop propagation of PTSEs throughout a peer group and child peer groups. It ensures that each node in a peer group maintains an identical topology database. Flooding is the advertising mechanism in PNNI.

3.2.2 Higher Hierarchical Levels

A peer group is represented at the next hierarchical level by a single node called a logical group node. The functions needed to perform this role are executed by a node, called the peer group leader (PGL), that is a member of a peer group being represented. The peer group leader in a peer group is responsible for aggregating the peer group information which will be advertised by the logical group node to the other peer group nodes in the higher level. The peer group leader is also responsible for flooding information from the other higher levels nodes to the nodes in the peer group. (Apart from its specific role in aggregation and distribution of information for maintaining the PNNI hierarchy, the PGL doesn't have any specific role in the peer group).

A logical group node is an abstraction of a peer group for the purpose of representing that peer group in the next PNNI routing hierarchy level. For example in fig. 1, logical group node A represents peer group A at the next higher level peer group that consist of A.1 and A.2 which by the same way represents peer group A at the next higher level peer group that consist of A and B. (The functions of a logical group node include aggregating and summarizing information about its child peer group and flooding that information into its own peer group). A logical group node passes information flooded within its peer group to the PGL of its child peer group for flooding.

Links that goes from a lower level node to one on a higher level are called uplinks (see fig 1). If there are several uplinks to the same higher level node from one peer group, those links maybe represented as one horizontal link between the higher level nodes.

On every higher level information is summarized as much as possible, and feasible. Uplinks are bundled together using aggregation tokens and all links with the same token are represented by the same aggregated horizontal link on a higher level. PNNI routing hierarchy completion is achieved by creating ever higher levels of peer groups until the entire network is encompassed in a single highest level peer group.

Adress summarization reduces the amount of addressing information which needs to be distributed in a PNNI network an thus contributing to scaling in large networks. It consists of using a single reachable adress prefix to represent a collection of end system and node adress that begin with the given prefix.

Reachability information, i.e. reachable adress prefixes, are fed throughout the PNNI routing hierarchy so that all nodes can reach the end systems with addresses summarized by these prefixes (see fig 1).

Links are advertised uni-directional (outgoing direction as viewed by a node) in PNNI, but since a connection are always setup bidirectional and both directions uses the same route, links will be referred to as bidirectional. The associated total link cost for that bidirectional link takes both forward and backward directions differents set of networks resources into consideration when computing the link cost.

3.3 Reference Model

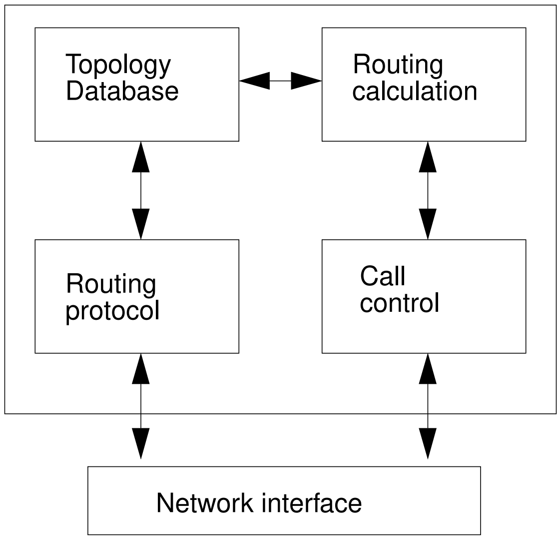


Figure 2: *The relation between entities between entities in PNNI*

Topology information is distributed in the network by the Routing Protocol, and in every node the information is stored in the node's Topology Database. The routing algorithm uses the topology information in the database to build routes to different nodes. When the Call Control receives a call setup request, the Routing Calculation can deliver routes on which to forward a call.

3.4 Designated Transit Lists

In processing a call, PNNI Call Control may request a route from PNNI Routing Algorithm, PNNI specifies routes using Designated Transit Lists (DTLs). A DTL is a complete path across a peer group, consisting of a sequence of nodes IDs and optionally ports IDs. It is provided by the source node, i.e. the originator of a hierarchically complete source route, or an entry border node to a peer group. A hierarchically complete source route represents a route across a PNNI routing domain that includes each hierarchical routing level between the current level and the lowest visible level in which the source and destination are reachable. This is expressed as a sequence of DTLs ordered from lowest to highest peer group level and organized as a stack (LIFO structure). The DTL at the top of the stack is the DTL that corresponds to the lowest level peer group.

3.5 Source Routing

There are two basic routing techniques that have been used in networking: source routing and hop-by-hop routing. In source routing the originating (or source) switching system selects the path to the destination, and other systems on the path obey the source's routing instructions. In hop-by-hop routing, each system independently selects the next hop for that path, which results in a progress toward the destination provided that the decisions made at each hop are sufficiently consistent (e.g. that no loops occurs).

In principle other routing technique can be applied to connection-oriented networks, such as ATM. Hop-by-hop routing has some disadvantages in this type of network. The first is the creation of routing loops. There are several possible causes of routing loops: inconsistency on routing decisions when switches uses different routing algorithms, inconsistency in distributed routing databases among the switches, which is typically due to changes in topology information that have not yet fully propagated. Another disadvantage of hop-by-hop routing is that it replicates the cost of the path selection in each system.

In source routing the source is responsible for selecting the path to the destination. It does this based on its local knowledge of the network topology. Since only one distributed database per source node is involved, loops are not possible. Furthermore, since only the source selects the path, the algorithm used not need to be the same in every ATM switch system, and the cost of executing it only occurs once.

To avoid the difficulties of using hop-by-hop routing stated above with connection oriented service and to gain the advantages of source routing, PNNI uses source routing for all connection setup requests. This implies that the first node in a peer group selects the entire path across that peer group and to all its ancestors for the purpose of reaching the destination. The path is encoded as a set of Designated Transit Lists (DTLs) which are explicitly included in the connection setup request. The DTLs specify every node used in transit across the peer group and may also optionally specify the logical links to be used between these nodes. If a node along the path is unable to follow the DTL for a specific connection setup request due to lack of resources, lack of connectivity or that a node is lost, then the node must refuse that request and cranckback the request to the node that created the DTL.

4 Resource management

4.1 Generic Connection Admission Control

As part of providing quality of service (QoS) and throughput guarantees, a switching system performs connection admission control (CAC) during the connection setup phase to determine if a connection request can be accepted without violating existing connections's QoS and throughput requirements.

To enable routing to produce paths that are likely to be accepted, it is necessary for switching systems to expose detailed and up-to-date resource information, however, may result in unacceptable high routing traffic. Furthermore since CAC is not subject to standardization in PNNI [1], it is not feasible to perform the actual CAC to determine if the switching systems along the path will admit a new connection. Generic Connection Admission Control (GCAC) solves this problem by allowing switching systems to advertise resource information that is generic i.e. independent of the actual CAC used in the switching systems, and compact, but yet rich enough to support most CAC.

For every incoming connection request, a path is chosen from the pre calculated routing table. The routing table is computed based on information stored in the node's topology database about traffic characteristics. Bandwidth and QoS requirements of the connection to be established are then compared to the available resources of the route chosen from the table. The path is chosen to satisfy the connection's end-to-end bandwidth and QoS requirements as well as to meet network efficiency criteria. During connection setup, each switching system along the chosen path performs CAC to ensure that the connection can be accommodated without jeopardizing QoS and bandwidth guarantees to the existing connections.

A GCAC is needed in the path selection process to determine if a link or node is likely to have enough resources available to support the proposed connection. In essence, a GCAC predicts the outcomeof the actual CAC performed at a switching system.

5 ATM Service Architecture

5.1 Topology State Parameters

Topology state parameter is a generic form that refers to either a link state parameter or a nodal state parameter. Topology state parameters fall into two categories: topology metric and topology attribute. A topology metric is a topology state parameter that requires the values of the state parameters of all links and nodes along a given path to be combined in a additive manner to determine whether the path is acceptable and/or desirable for carrying a given connection. A topology attribute is a state topology state parameter that is considered individually to determine whether a given link or node is acceptable and/or desirable for carrying a given connection.

5.2 ATM Service Category Parameters and Attributes

In the Private Network-Network Interface Specification (version 1.0) [1], the following classification is made for rt-VBR service category regarding resources available on a link and in a node:

Topology Metrics:

- Cell Delay Variation (CDV)
- Maximum Cell Transfer Delay (maxCTD)
- Administrative Weight (AW)

Topology Attributes:

- Cell Loss Ratio (CLR)
- Available Cell Rate (ACR)
- Maximum Cell Rate (MCR)
- Cell Rate Margin (CRM)
- Variance Factor (VF)

In the PNNI Routing Specification, topology state parameters are classified from the network resources point of view. Every node in the network announces from its distributed topology database, the resources CDV, maxCTD, AW, CLR, ACR, MCR, CRM and VF on outgoing links and for the node itself dynamically.

The Administrative Weight is a value set by the network operator. It is used to indicate the relative desirability of using a link or node for whatever reason significant to the network operator. A higher value describes a link or node which is less desirable to be used than a link with lower value.

In this work there is not notice taken about CRM, MCR,CLR,maxCTD and VF. Instead the Least-cost route method specified below is used (see 6.1).

6 Specification of the least-cost route generation

6.1 Dijkstra Algorithm

Dijkstra algorithm for finding the shortest path from a source node to a destination node in a graph is the most used and implemented algorithm in a network. In this work it's defined first what is the best route from a source node to a destination node in comparison with other routes that have the same source node and the same destination node.

The routing algorithm must be able to adapt a routing policy that can change when a network operator wants to. The routing policy in turn reflects the network operator's criterias on resource utilization in the network. Statistics on what kind of application most hosts use, hosts requirements of delay, bandwidth and cell loss, will be used by the network operator to control which topology state parameter to be optimized and the priority between parameters in the routing policy.

Today's routing algorithms, such as Dijkstra, commonly used and implemented in conjuction with protocols like the Open Shortest Path First (OSPF), are capable of routing based on one constraint. In other words, given a connectivity graph and a single metric associated with each link in the graph, Dijkstra algorithm can quickly find the path between two points that minimizes the sum of links metrics in the path. In this work it is defined a routing policy in order to give priorities to each metric in a link, Dijkstra optimizes the metric of a link which has highest priority, and then goes down in the set of priorities if two or more routes have the same values (see least-cost route). The metrics of a link represent the link resources in terms of parameters specified in the PNNI protocol specification. This form of work of the Least-cost route generation is determinant for the stability of the network, it minds that if a link has reduced its resources, is necessary to try to avoid the incoming traffic through this link so giving more priority to a link with more resources.

In order to apply Dijkstra Algorithm we meet two kind of problems: one that considers building the routing table and maintaining them and one that considers finding an acceptable route when there comes a request for a connection i.e path selection.

So Dijkstra Algorithm described above implies two kind of problems; first a link must be described in terms of multiple parameters, and second whether a link can participate in a path depends on the requirements of the connection being routed. An approach to both problems may be solved if we can find a good relation of those parameters that distinguish and became important for a link, this relation can be managed by the operator giving the necessary priority to a differents parameters on the link. As a example suppose that a path should be found for which delay is minimized, then is entirely possible that such a path could be found but with unacceptable low bandwidth or delay variation. Similarly if routed only by bandwidth or delay variation a path may be found with unacceptable delay.

A possible solution is to run the Dijkstra algorithm one time for every different types of parameters in the link. Then every parameter would be optimized separately. Then we get a set of overlapping routes where every route is optimized on only one parameter. Then one must choose in the set of overlapping routes to find the convenient one. This is not efficient because it will take long time to compute a shortest path route.

Another solution is to optimize one topology state parameter, one that is considered the most significant, under the condition that the others satisfies the requirements on transmission performance. Then it‘s not possible to build a routing table, because routing tables implies to consider the topology of all network and not only of one host each time.

The Dijkstra algorithm used in this work is a routing algorithm that takes every topology parameter into account and give a priority to each metrics. That is the operator has the possibility to give differents priorities to the metrics according to a routing policy. When the priorities are defined then Dijkstra runs and find the best routes in accord to the routing policy.

6.2 Least-cost route

A least-cost route is the route with the highest resource margin and which requires minimum allocation of resources compared to other routes, where the resources of the network are nodes and links, with their associated available relative capacity an QoS. As resources are allocated along the least-cost route, it may as a result turn into a non least-cost route, another route will then become least-cost. To rebuild routing tables, the routing policy must be done first to be able to set the priorities needed and then compute the least-cost route.

There are 4 factors that this work takes in consideration in order to define the best route among many others: ACR, CDV, AW, and the number of links a route has. ACR is a selective parameter, the ACR cost of a route is determined by the minimal ACR value of a link within the route and the biggest ACR value of all routes will be chosen as best. For CDV is valid that the minimal CDV value is considered best and the CDV value of a route will be the sum of CDV values of the links a route consists of. The same assertion as for CDV is valid for AW. The minimal number of links between 2 or more routes is considered best. The algorithm works in the following way; if two routes have the same cost in terms of ACR, then the algorithm looks at CDV parameter and the route which has the minimal value is chosen before the others, if still two o more routes have the same value in terms of ACR and CDV, the algorithm looks at AW cost of the routes in question and choose that one which has the minimal value. If still two or more routes have the same value, the algorithm looks at the number of links or hops the routes have and it chooses the route. The probability for the latter case is very little because the values of ACR, CDV, and AW are big and the amount of resources taken from them in each case are very small (In case of telephony each call request is 64Kb of 155 Mb ACR a link has) so after a small number of calls is very little the probability that 2 paths have still the same amount of ACR, and this probability reduces more in each hop down on the set of priorities.

By this way of calculate routes we try to avoid links that has minimum resources, and a call will take a route which has the maximum of the least resources over the network.As example suppose that link a->b has 100 Mb b->c has 1Mb, the total cost of the route a->b->c is 1Mb, and suppose there is other link a->x with 100 Mb and link x->c has 2Mb with the total cost of route a->x->c as 2Mb, in this case Dijkstra choose route a->x->c as the best one.

After an amount of time the resources of the network have been reduced, and as result of the utilization of the network, some links have been used more than others, and is essential to balance the utilization of the resources over the network for the incoming calls, so that the links which loose more resources have reduced theirs ability in being the best route.

PNNI presents two approaches for the solution of this problem, the first one is by setting a Timer after which Dijkstra run again updating the routes, and the other solution is a procedure which detects when the resources of a link had reduced its value under a given minimum limit, in this case to run Dijkstra again is a necessary step. We examine these two approaches closer.

After a given time set by the Timer, had gone, the program executes Dijkstra again with the resources left, setting the best routes to those ones that have more resources. The cost of a route is determined by the ACR parameter or how much bandwidth there is left on a given link, the CDV or the rate for a call to setup a connection, and the AW parameter - the value the operator set to a link. Choosing the highest of the least ACR parameter over a route, and then the least CDV value over a route that results of adding the correspondents CDV values in each link across the route, and then choosing the highest AW value in a route that results of adding all AW values of links over the route and if necessary, looking at the least number of hoops for each route, the algorithm can determine the best route from a source node to a destination node.

The second approach uses the following strategy: if the current value of the parameter in question have been reduced to half of the last value it had (before the last executing of Dijkstra) and if this last value minus the current value is over the 3% of the initial value (the value this parameter had at beginning of the program) then Dijkstra executes again:

IF (Par < 50% of LastPar) AND (LastPar - Par > 3% of InitPar) THEN Dijkstra

So the last approach says that after the value of the given parameter reduces under a minimum limit, it doesn't important to have the following changes in consideration.

6.2.1 Denominations

- Cost is the bidirectional link cost function with complexity. Multiple topology state parameters for one link in the PNNI network forms the link cost.
- CTD is the topology metric Maximum Cell Transfer delay in the PNNI Routing specification.

- CDV is the topology metric peak-to-peak Cell Delay Variation in the PNNI Routing specification.

- ACR is the topology attribute Available Cell Rate in the PNNI Routing specification i.e. absolute Available Cell Rate.

- MCR is the topology attribute Maximum Cell Rate in the PNNI Routing specification.

- CLR is the topology attribute Cell Loss Ratio in the PNNI Routing specification.

- AW is the topology attribute Administrative Weight in the PNNI Routing specification.

- ACR/MCR is the relative Available Cell Rate.

- Hop is the number of hops between nodes in a network.

6.3 PNNI routing algorithm

The Dijkstra algorithm was implemented in ERLANG that is a declarative programming language developed at Ericsson.

We start to consider the cost of a link as the result of the routing policy we pointed to above. The start point is a file with the topology necessary for the simulation. In this file the topology of the network is specified, with links and resources in terms of topology state parameters, the arrival interval intensity to each node that is modelled as a Poisson process (in calls per sec), and the time that it takes for a call to reach the neighbour node (in ns). There are also timers for each node which are needed to compute once again the Dijkstra algorithm, when time out (see 6.4).

Initializing

(1) The program reads the initial file and keeps the topology of the network in a Erlang Term Storage (ETS) table that is a datastructure already implemented in Erlang which permit access in constant time, no restriction on data, searches/ updates between other features. A empty queue is created, this queue will keep all the routes that the Dijkstra algorithm find step by step from a source node to all nodes, in order of priority, i.e the best route at the front. A empty routing table is also created, which is a ETS table too. At this part we know the source node, as input in case we need to update a specified route or being any one (taken by order) in case we need to compute or update all routes in the network.

Main loop

(2) Assume that the source node is *i*. Find all neighbours nodes to *i* and build paths to these neighbours i.e. the old path + the new link generated. Given all state parameters, the resulting cost is computed, by the least-cost priority

procedure. Now put all these paths into the queue by order i.e. least cost path at the front.

(3) If Queue is empty. We computed all routes from node *i* to rest of nodes in the network. Then go to (4). Otherwise go to (5).

(4) If we need to compute all routes from node *j* to all nodes, supposed that *i /= j* and it was not computed yet, go to (2). Otherwise go to (6).

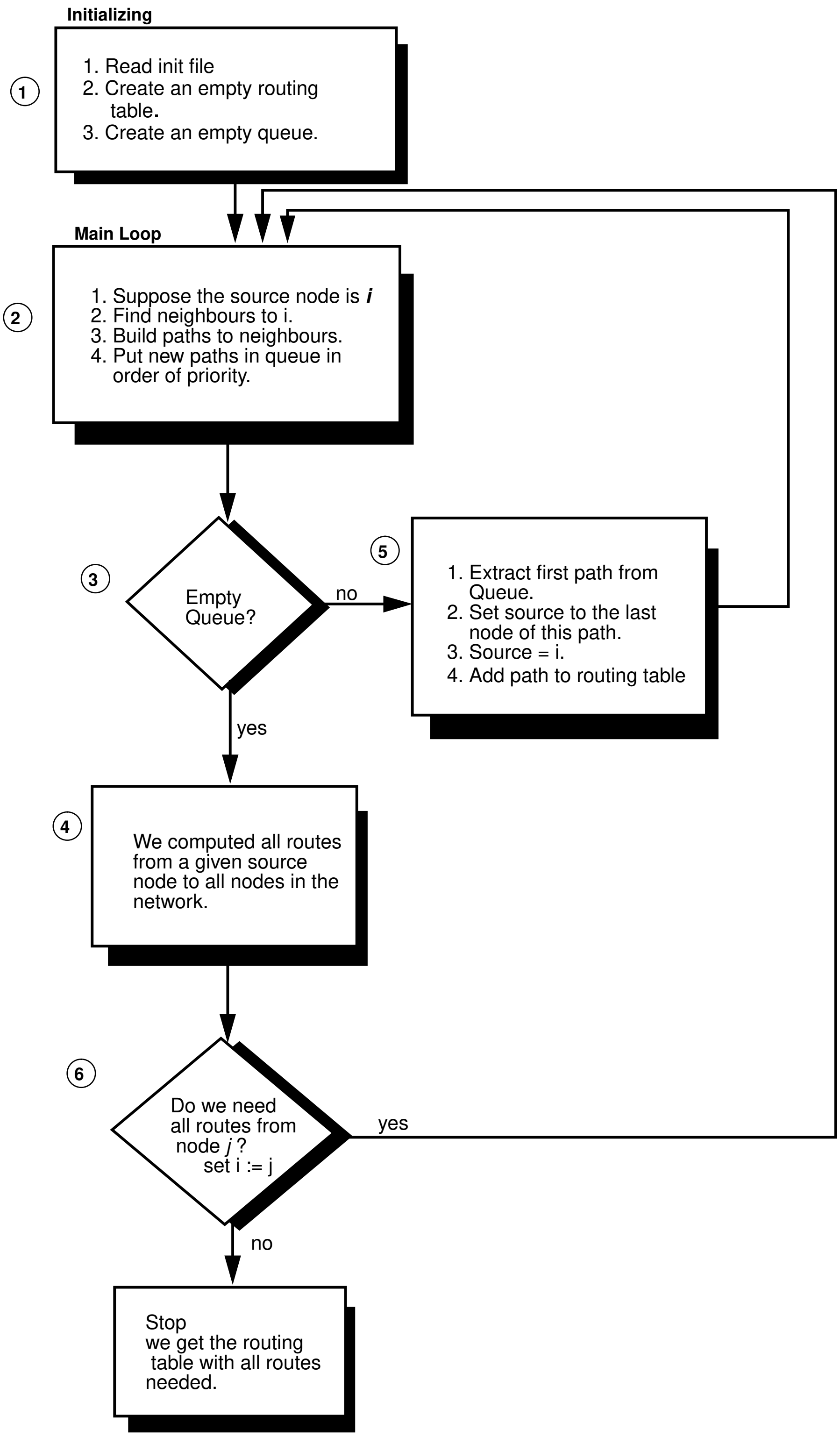
(5) Extract the first path from the queue, find neighbours to the last node in the path. Add this path to routing table. Continue at (2).

(6) The algorithm finishes. We get all routes from a given node or from all nodes to the rest of nodes in the network. Hence the routing table is filled with the needed routes.

The use of a declarative programming language like Erlang that provides a high level of abstraction, makes the algorithm easy to implement with a good performance. The use of ETS (Erlang Term Storage) simplifies the construction of datastructures where to store, update, read, write and maintain data.

The flow chart of the algorithm is showed below. An example of the routing table is showed in the Appendix.

6.4 Flow chart of Dijkstra algorithm



7 Event driven simulation

In order to simulate a system, it is needed to study the system in question and to set an apropiate model of it. There are a variety of methods and mechanisms to classify a system in order to create a model of it in an apropiate way. This chapter will describe why we have choosen an event driven simulation and how it works.

7.1 Basic definitions

A system is defined to be a collection of entities, e.g., people or machines, that act and interact together toward the accomplishment of some logical end. In practice what is meant by the system depends on the objectives of a particular study. The collection of entities that compose a system can be a subset of a other system. We define the state of a system to be a collection of variables necessary to describe a system at a particular time, relative to the objectives of a study. In a network system is reasonable to distinguish as state variables: the number of nodes, the number of links, the number of calls that come in to the system, and the time of arrival of each call to the system. We categorize systems to be of two types, discrete and continuous. A *discrete* system is one for which the state variables change instantaneously at separated points of time. A *continuous* system is one for which the state variables change continuously with respect to time. So a network system is obviously a discrete system, the incoming calls variates at separated point of time.

At separated points in time arrives a number of calls to the system. These points in time are the ones at which an *Event* occurs, where an event is defined as an instantaneous occurrence that may change the state of the system. But an event might be used to schedule an other event or the end of a simulation run at a particular time or to schedule a decision about the system's operation at a particular time and might not actually result in a change in the state of the system. This is why (we said that) an event might change the state of the system.

7.2 Time-advance mechanisms

Because of the dynamic nature of discrete-event simulation models, we must keep track of the current value of simulated time as the simulation proceeds, and we need also a mechanism to advance simulated time from one value to another. We call the variable in a simulation model that gives the current value of simulated time the *simulation clock.* The unit of time for the simulation clock is never stated explicitly for a given model, it's assumed to be in the same units as the input parameters. That is as we set a dimension for the network each incoming call have a given time unit when it occurs, how long time it will takes in the network system, etc. Also, generally there is not a relationship between simulated time and the time needed to run a simulation on the computer.

An approach generally used in simulations programs, is the next-event time-advance approach. The simulation clock is initialized to zero, and the times of occurrence of future events are determined. The simulation clock is advanced to the time of occurrence of the most imminent (first) of these future events, at which point the state of the system is updated to account for the fact that an event has occurred, and our knowledge of the times of occurrence of future events is also updated. Then the simulation clock is advanced to the time of the most imminent event, the state of the system is updated, and future event times are determined, etc. This process of advancing the simulation clock from one event time to another is continued until eventually some pre-specified stopping condition is satisfied. Since all state changes occur only at event times for a discrete-event simulation model, periods of inactivity are skipped over by jumping the clock from event time to event time.

7.3 Components and Organization of a Discrete-Event Simulation Model

There are a number of components and a logical organization for these components that are common to a typical discrete-event simulation which uses the next-event time-advance approach:

System state: The collection of state variables necessary to describe the system at a particular time.

Simulation clock: A variable giving the current value of simulated time.

Event list: A list ordered according to event time stamps i.e. when each type of event will occur.

Statistical Counters: Variables used to store statistical information about system performance.

Initialization routine: A subprogram to initialize the simulation model at time zero.

Timing routine: A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur.

Event routine: A subprogram that updates the system state when a event of particular type occurs (there is one event routine for each event type).

Library routines: A set of subprograms used to generate random observations from probability distributions that were determined as part of the simulation model.

Report generator: A subprogram that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends.

Main program: A subprogram that invokes the timing routine to determine the next event and then transfers control to the corresponding event routine to update the system state appropriately. The main program may also check for termination and invoke the report generator when the simulation is over.

The logical relationships (flow of control) among these components is shown in the following flow chart (8.5). This corresponds to my particular model.

8 The simulation model of PNNI

8.1 The model of the system

The PNNI ATM based switch system is modelled as a system with two basic entities; Node, which is an abstraction of a switch or collection of them with the same function and link which is the abstraction of the physical link used to connect two nodes. The state of the system is defined by the collection of state variables which describes the system at a particular time. The state variables are the parameters of the entities or the resources a link and a node have to offer and an incoming call with it's traffic contract. As calls arrives to the system each of them with an specific traffic contract, the state of the system is determined by the set of entities and the resources capacity of them, which varies undeterministically under the time an amount of calls uses the system. The system implemented maintain a fixed amount of entities just because we are not interested in the behaviour of the system when it varies the number of nodes or links. It is easy to see that the state of the system changes instantaneously at separated points of time, that is the system is Discrete. So we are interested in modelling the system as it evolves over time by a representation in which the resources change instantaneously at separated points in time.

8.2 Requirements

The followings requirements are set to the simulation program:

- A PNNI network with (2-20) nodes in one peer group can be simulated in the simulation system. Each node could run its own source routing algorithm with its own timer.
- There should be an interface by which the different source routing algorithm could be dinamically adopted for each node.
- Implement least one source algorithm. It should be possible to measure its execution time per node.The optimizings parameters (CDV, maxCDV, and AW at least) can be changed dynamically.
- The topology parameters for each node and link could be changed dynamically. The parameters should always be accessible.
- Some incoming and outgoing distribution for the behaviour of each node.
- Some duration distribution for each call.
- Measures of rejected calls in percent.
- Summary of statistics.

8.2.1 Details

A setup of an end-to-end connection (i.e. from the originating node to the destination node) may fail due to the real topology has been changed since the last route calculation. The reason that caused such fails can be:

- The calculation of the routing table used an old topology. In this case we have choosed a 'wrong' route before the setup.
- Since a setup along a route is done step by step, it may be interfered by other setups. E.g. a setup for a->b->c->d->e may be interfered by the following setup: x->y->c->d->z. In this case, the route is correct before the setup, but it fails during the setup process (the topology changes during the setup, i.e. when the first setup is connecting a->b, the second setup is connecting c->d).
- It is mainly important to run the routing algorithm after a while when a certain condition is reached, with the resources that the network has left. This should be done with the purpose of updating the routing table with the routes which have more resources as the first to pick up. The condition reached is specified bellow.
- The simulation program should consider either the procedure described at 6.1 or set a timer which is specified in seconds after which the routing algorithm should be run again, with the purposes given at 6.1.

8.3 The structure of the program

At the network system, 2 entities (the most importants for the subject of this work) define the state of the system at any point in time: The link and the incoming calls. A CALL is defined as a datastructure with the followings elements:

- ID. The calls identity number or the call number.
- Parameter. How much bandwidth the call require.
- Duration. How long time the call will take. This time is generated stochastically, by a Poisson process.
- From. Where the node has arrived. This number is generated randomly uniform.
- TemporaryFrom. Where the call is on the way to the destination node. This is determined by the route, taken from the routing table.
- To. The destination node, and is also randomly uniform generated.

Due to simulation purposes, it is necessary to define a datastructure called ITEM which is unique for a call, the task of this Item will be to carry information that is not inherent to a call but to the simulation itself. So A ITEM will be defined as follows:

- Time. The time a call has arrived to the system.
- Event. What type of event a call has. See below.
- CALL. the call itself.
- Route. The current route extracted from the routing table. A call will follow this route.
- NrNode. The source node a call has arrived to.

Our model is Dynamic because our simulation model represents a system that evolves over time. It's stochastic because our model contains some probabilistics components, and Discrete. Due to the requirements the followings events can occur:

- Arrival. A call arrives to the system according to a Poisson process. Each node has different arrival intensity (rate).
- Request. The incoming call has arrived to a node and request for connection, that is a route is set for that purpose, at a first moment the best route in terms of resources will be received from the routing table.
- Setup. A call is in a node on the way from the source node to the destination, and require a connection to the next node. A check will be done at the link that binds both nodes if there is enough resources to accept the call. If the check fails the call will do a cranckback returning to the node it started from, leaving all resources it took from them and choosing the next route from the list of remainings routes.Cranckback is done at once. This procedure is done step-by-step under simulation time.
- Release. A call reached to its destination node. After this call have used a short time in the destination node specified by the element *duration* in datastructure *Call,* it will give back the resources taken on the route and disappear leaving the statistics counters updated.
- Timer. After the simulation program runs a time specified by the datatype *timer,* is necessary to run again the routing algorithm with the remained resources on the network. The point here is to update the routing table with new routes that has more resources than the old ones.

The events described above lead us to the following event diagram depicted in Fig 3:

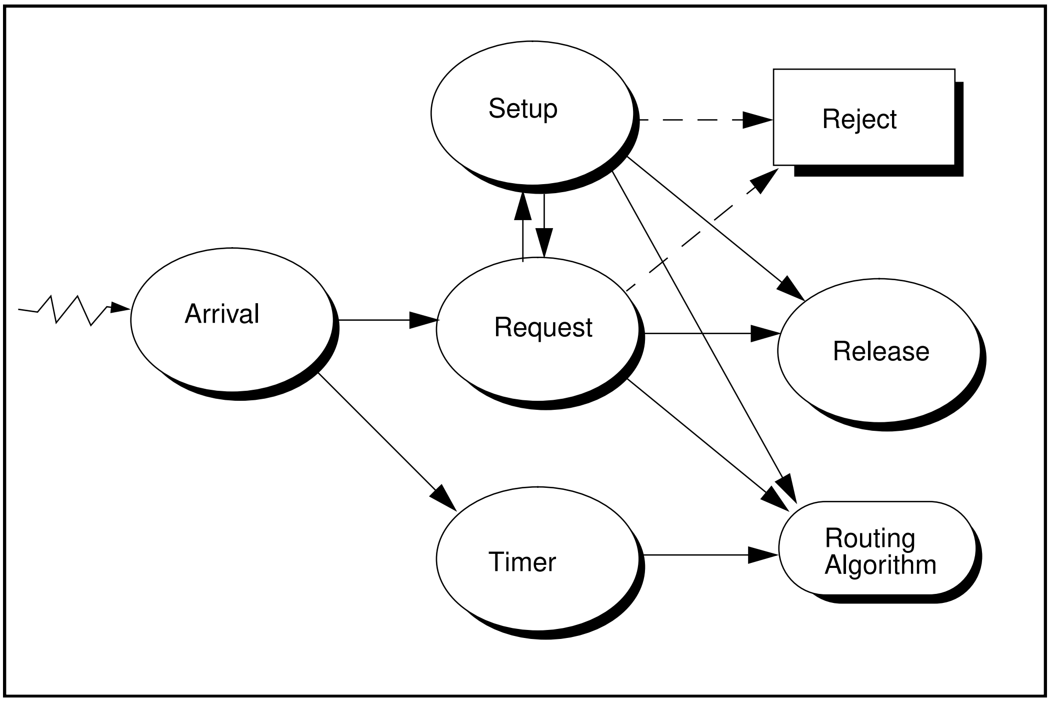


Figure 3: *Event diagram.*

When a call arrives to a source node it has the type stamp Arrival event, with the time when it arrived, and it's scheduled to the event list as that one. When the next event to occur is a call with stamp Arrival (the first from the event list), it will do a first request to the arrived node and it will be scheduled with a new stamp: Request event, and put in the event list with its new event stamp and a new updated time, due to the time it will take to reach the next node if it is not the destination node nor is rejected. So when the new event to occur is a call with the stamp event Request, it will be scheduled with a new event stamp Setup if the next node is not the destination node or Release if it's the case, then updating all elements in the call datastructure. If the new event to occur is a call with a stamp Setup event then it is scheduled as Setup event again if it has not reached yet the destination node or Request event if it has been rejected by a link due to lack of resources, or Release event if it has reached the destination node.

The timer event is an empty datastructure, it is in the event list, and will occur after a determined time different for all nodes. This datastructure is created at beginning of the execution. When the next event to occur is the Timer event then the routing algorithm is run again with the remainings resources and so updating the routing table. At the end, a new timer event is scheduled and added to at the event list with the same time as the one treated.

8.4 The simulation program

Initializing:

The simulation program begins at time 0 with the main program invoking the initialization routine, were the simulation clock is set to zero, the system state and the statistical counters are initialized, and the event list is initialized.

Main loop:

After control has been returned to the main program, it invokes the timing routine to determine which type of event is the most imminent. If an event of type *i* is the next to occur, the simulation clock is advanced to the time that event type *i* will occur, and control is returned to the main program. Then the main program invokes event routine *i*, where the followings activities can occur:

1. The system state is updated to account for the fact that the event of type *i* has occurred.

2. Information about system performance is collected by updating the statistical counters

3. The times of occurrence of future events are generated and this information is added to the event list.

4. Dijkstra Algorithm is executed if it's needed.

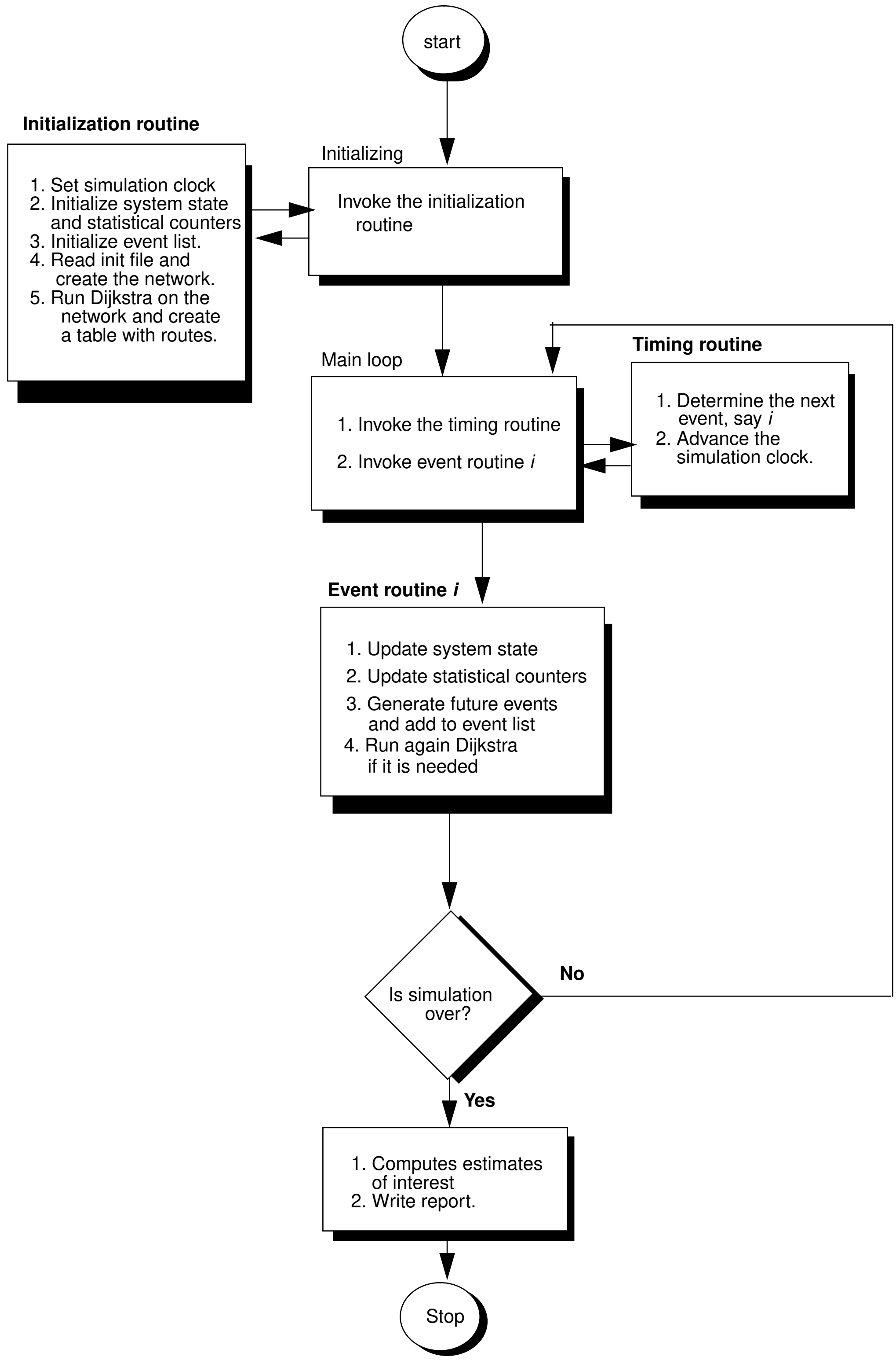
It is necessary to generate random observations from probability distributions in order to determine these future event times. There is a interval time for the incomings calls to a determined node that is modelled by a Poisson process. This is the random variate for each call related to the arrival node. Before all this process is continued, a check is made to determine if the simulation should be now terminated. In the case that the timer is not set up, a mechanism to run the routing algorithm after a certain condition is fulled, is executed, namely the verification if a link has reduced its resources under a given level, otherwise the timer will determine when it will occur.

If it is time to terminate the simulation, the report generator is invoked from the main program to compute estimates (from the statistical counters) of the desired measures of performance and to produce a report. If it is not time for termination, control is passed back to the main program and the cycle: main program - timing routine - main program - event routine - termination check, is repeated until the stopping condition is eventually satisfied, namely when the simulation time set in the beginning of the simulation has expired.

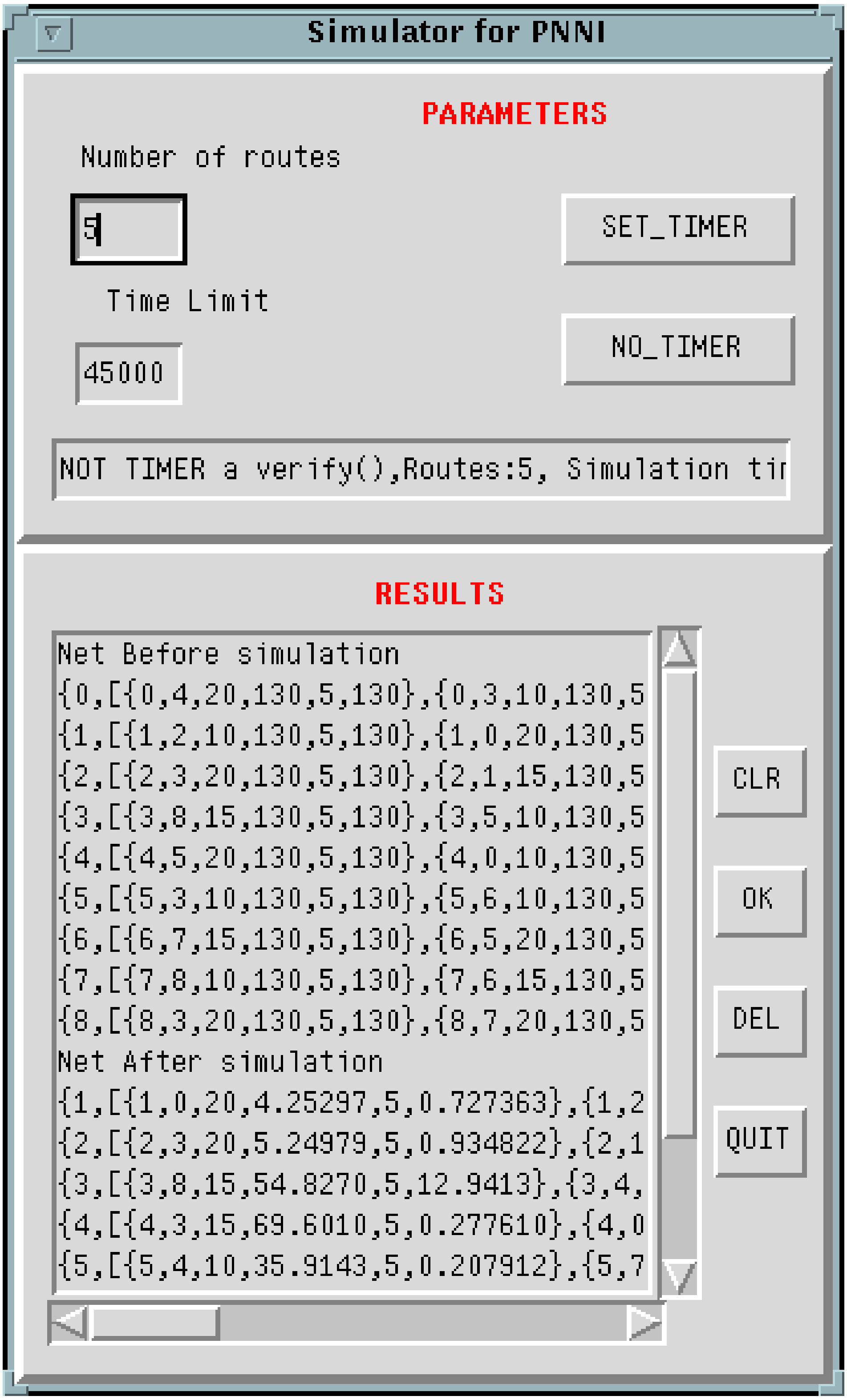
The flow chart of the algorithm is showed below.

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	Jorge Abarca Pereda		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		1997-11-12	

8.5 Flow chart of the simulation algorithm



8.6 Graphic user interface



The user Interface can be interpreted as follows:

In the upper window, Number of Routes gives the number of paths from each node to all nodes (except to it self), according to the routing policy. Time Limit is the parameter which indicates the simulation time (in seconds), the program will execute. It shoud be chosen if the simulation program will execute with Timer or with the procedure Verify() (See 6.1 and 8.3).

In the below window, we have a canvas Results which shows the Network state described in terms of Erlang table ETS. And the same network at end of simulation. The right buttons are to clear (CLR) the chosen upper parameters, to run (OK) the program with the parameters specified, and quit (QUIT) the simulation program.

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	**Jorge Abarca Pereda**		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		**1997-11-12**	

9 Simulations

In order to get a good sense about the performance of the network, the simulator was executed for two cases, due to the difference in QoS parameters (the most important of which is Bandwith): when the users request service for video applications and when the network is used for telephony services. The input parameters for both services were specified by Ericsson.

In the first case the following indata is used: a request uses a capacity of 3Mbytes, for a mean time of 90 minutes, the arrival intensity to each node is 0.0023 arrivals per second, i.e. 1 call every 435 seconds, during 250 hours (simulation time), the timer is 30000 sec (see 6.1), and the number of routes calculated are 8.

In order to get a histogram that will show the average percent of blockings during the simulation time, the 900000 sec simulation time was divided into 300 intervals of 3000 sec each, and the number of replications is 10. The partition could be other but what is needed is an indicator in a histogram of the average number of blockings. So the following diagram shows the average number of blockings for 300 intervals of 3000 sec each (see fig 4).

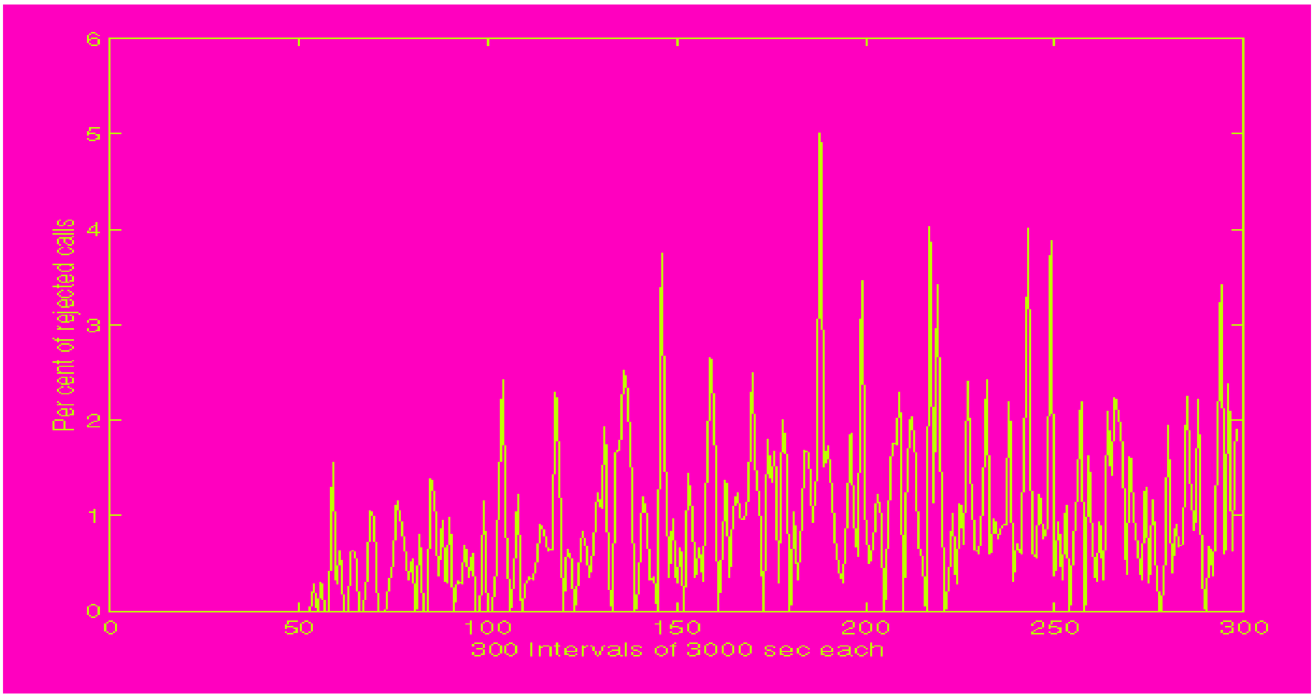


Figure 4: *Average percent of rejected calls for video service request.*

And a histogram that shows the corresponding average number of rejected calls is as follows:

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	**Jorge Abarca Pereda**		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		**1997-11-12**	

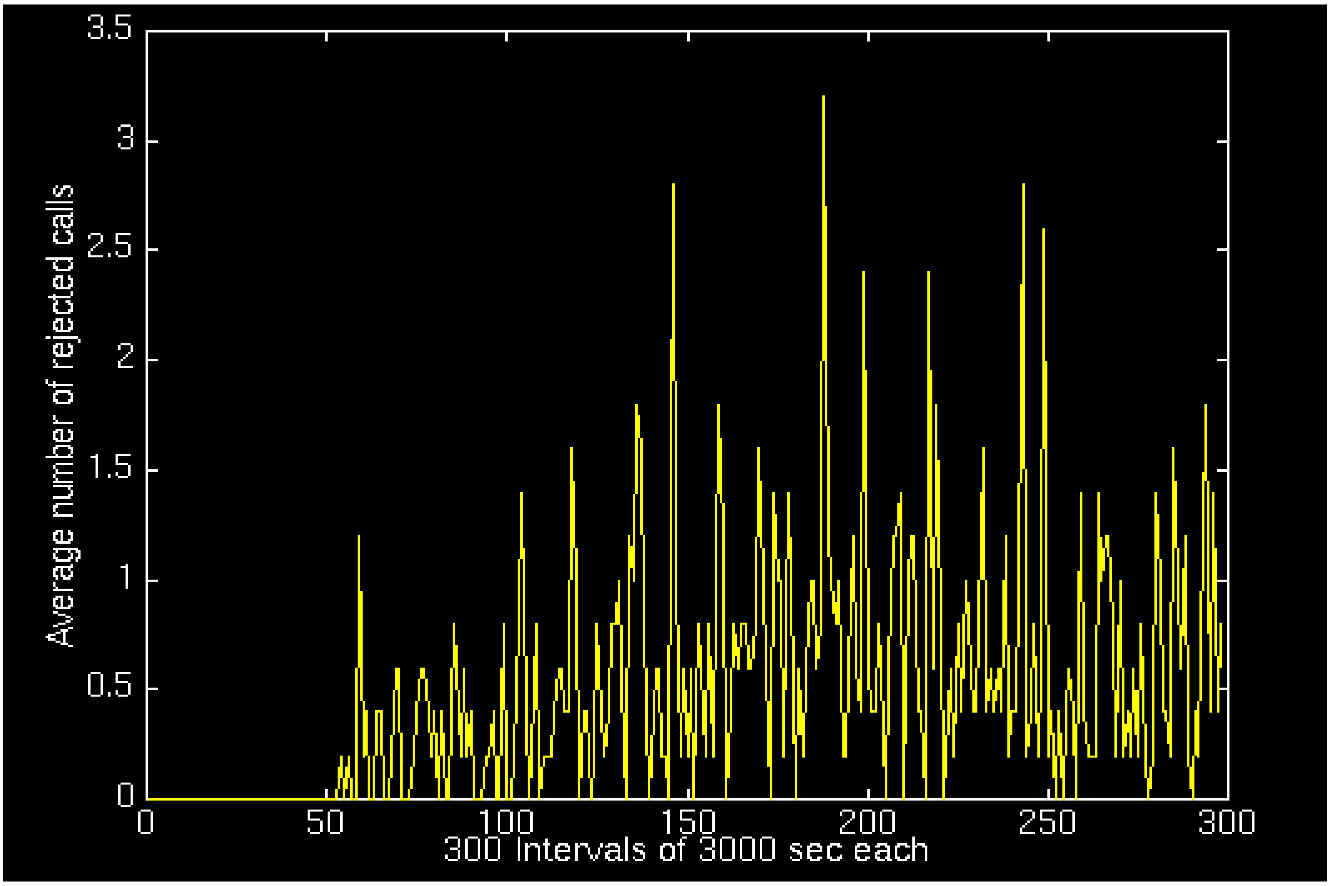


Figure 5: *Averaged number of rejected calls for video service request.*

This histogram shows that after a period of time of 160 intervals (approx. 4800 sec simulation time), the number of blockings for each interval time of 30 sec, variates in between 0 and 3.2 blockings approx., and the correspondent 0% - 5% approx. of rejected calls.

In case of telephony services the following input parameters are used: a request uses a capacity of 64 Kilobytes, for a mean time of 3 minutes, the arrival intensity to each node is 6 calls per second, during 2.5 hours (simulation time), the timer is 3 sec, and the number of routes calculated are 8.

In order to get a histogram that shows the average number of blockings during the simulation time, the 9000 sec simulation time was divided into 300 intervals of 30 sec each, and the number of replications is 10.The partition could be other but what is needed is a indicator in a histogram of the average number of blockings. So the following diagram shows the average number of blockings for 300 intervals of 30 sec each (fig 6).

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	**Jorge Abarca Pereda**		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		**1997-11-12**	

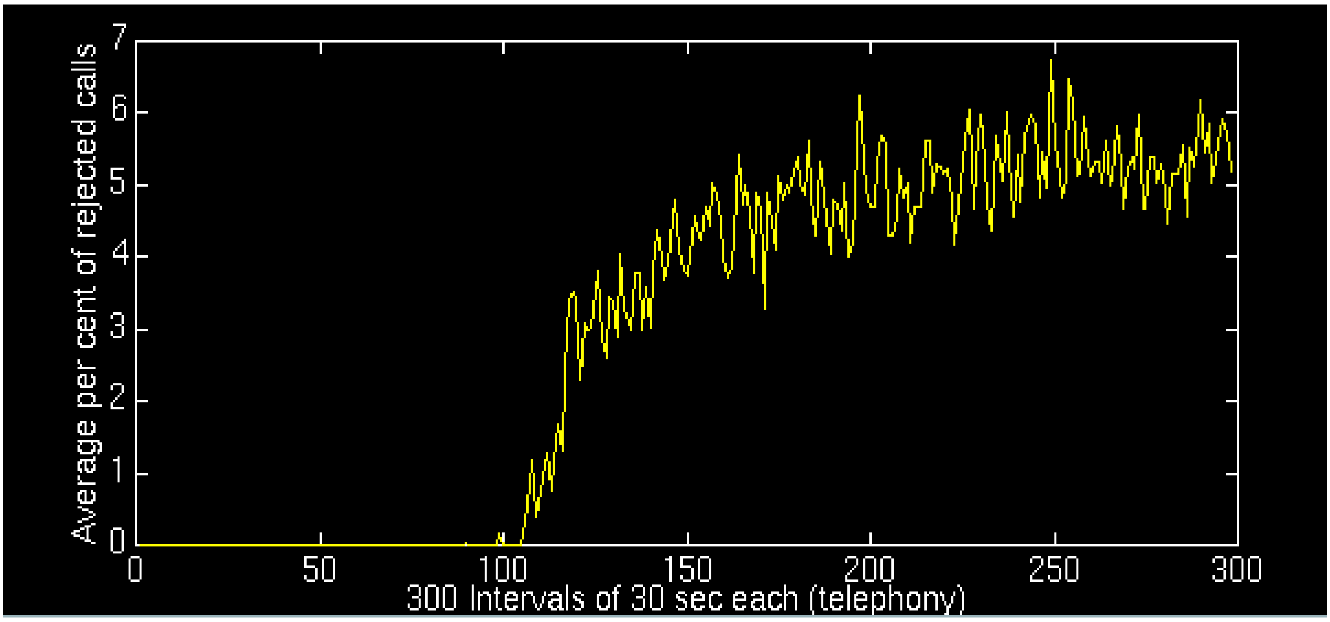


Figure 6: *Histogram for the average percent of blockings (telephony service)*

And a histogram that shows the corresponding average number of rejected calls is as follows:

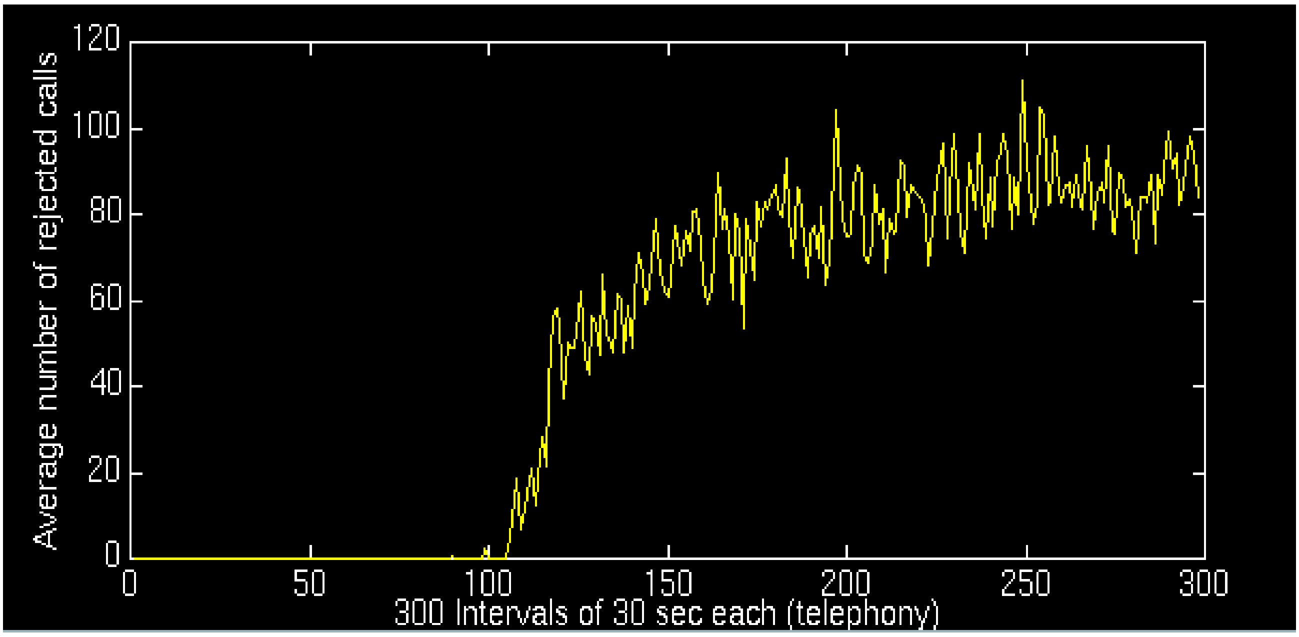


Figure 7: *Histogram for the average number of blockings (telephony service)*

This histogram shows that after a period of time of 170 intervals (approx. 5100 sec simulation time), the number of blockings for each interval time of 30 sec, variates in between 65 and 110 blockings approx., and the correspondent 4.2% - 6.8% rejected calls.

9.1 Considerations and results

In the case of request for video service it was done 10 replications, and the average results is as follows:

Number of routes: 8 Simulation time: 900000 sec (250 hours)

9 nodes: NodeNr 0, NodeNr 1, NodeNr 2, NodeNr 3, NodeNr 4, NodeNr 5, NodeNr 6, NodeNr 7, NodeNr 8.

Total average arrived requests: 18697.2 calls Rejected: 151.4

Number of times, Dijkstra was executed: 59.6. Timer: 30000 sec.

Number of arrived/rejected requests per node:

Nodes	Calls arrived (mean number)	Calls rejected (mean number)
0	2081.20	4.20
1	2075.80	4.80
2	2063.80	0.40
3	2050.80	0.20
4	2103.00	0.40
5	2076.60	85.80
6	2072.60	49.60
7	2030.80	1.20
8	2043.10	8.60

In the case of request for telephony service it was done 10 replications, and the average results is as follows:

Number of routes: 8 Simulation time: 9000 sec (2.5 hours)

9 nodes: NodeNr 0, NodeNr 1, NodeNr 2, NodeNr 3, NodeNr 4, NodeNr 5, NodeNr 6, NodeNr 7, NodeNr 8.

Total average arrived requests: 486601.7 calls Rejected: 14411.8

Number of times, Dijkstra was executed: 72.6. Timer: 3 sec.

Number of arrived/rejected requests per node:

Nodes	Calls arrived (mean number)	Calls rejected (mean number)
0	54076.3	41.25
1	53963.8	2520.75
2	54107.3	2493.00
3	53923.5	334.50
4	53956.5	170.50
5	54021.8	2718.50
6	53924.5	5643.25
7	54071.3	490.00
8	55323.5	765.45

9.2 Output data analys and conclusions.

9.2.1 Output data analys

The simulation system is of type non-terminating, because there is no natural event to specify the length of a run. So it's used the Statistical Analys for Steady-State Parameters [4].

Let Y1,Y2,... be an output stochastic process from a single run of the non terminating simulation. Suppose that P(Yi =< y) = Fi(y) -> F(y) = P(Y=< y) as i -> ∞ , where Y is the steady -state random variable of interest with a distribution function F. Let the steady-state mean v = E(Y), then one difficulty in estimating v is that the distribution function of Yi (for i = 1,2,3...) is different from F, since it will generally not be possible to choose *I* (the set of initial conditions) to be representative of "steady-state behaviour". This causes an estimator of v based on the observations Y1,Y2,...,Ym not be "representative". This problem is called "the problem of the initial transient" or "the start-up problem".

The steady-state mean v = E(Y), is also generally defined by

v = lim E(Yi), i-> ∞

Thus, the transient means converge to the steady-state mean. The most serious consequence of the problem of the initial transient is probably that $E([\bar{Y}(m)] \neq \upsilon)$ for any *m*. The technique most often suggested for dealing with this problem is called *warming up the model* or *initial-data deletion.* The idea is to delete some number of observations from the beginning of a run and to use only the remaining observations to estimate *v*. So given the observations Y1,Y2,...,Ym, it is often suggested to use:

$$\bar{Y}(m,l) = \frac{\sum_{i=l+1}^{m} Y_i}{m-l}$$

($1 \le l \le m-1$) rather than $\bar{Y}(m)$ as an estimator of v . m = MAX (300 in case of video service and telephony service).

The question now is how to choose the *warm-up period* (or deletion amount) l . We would like to pick l (and m) such that $E([\bar{Y}(m)] \approx \upsilon)$. If l and m are chosen too small, then $E[\bar{Y}(m)]$ may be significantly different from v . On the other hand if l is chosen larger than necessary, then $\bar{Y}(m,l)$ will probably have an unnecessarily large variance. The simplest and most general technique for determining l is a graphical procedure due to Welch (1981, 1983) [4]. Its specific goal is to determine a time index l such that $E([\bar{Y}(m)] \approx \upsilon)$ for $i > l$, where l is the warm-up period. [This is equivalent to determining when the transient mean curve $E[Y_i]$ (for i = 1,2,...) "flattens out" at level v ; (see fig 9 or 10). In general it is very difficult to determine l from a single replication due to the inherent variability of the process Y_1, Y_2 ,,, (see fig 8) below. As a result Welch's procedure is based on making *n* independent replications of the simulation and employing the following four steps:

- It was done 10 replications, each one consisting of MAX observations.
- Let Yji be the i-th observation from the j-th replication; and let

$$\bar{Y}_i = \frac{\sum_{j=1}^{n} Y_{ij}}{n}$$

- for i = 1,2,...,m. The averaged process $\bar{Y}_1, \bar{Y}_2$,... has means $E(\bar{Y}_i) = E(Y_i)$ and variances $Var(\bar{Y}_i) = Var((Y_i)/n)$. Thus, the averaged process has the same transient mean curve as the original, but its plot has only (1/n)th the variance.
- To smooth out the high-frecuency oscillations in $\bar{Y}_1, \bar{Y}_2$,...(but leave the low-frequency oscillations or long-run trend of interest), we further define

the moving average $\overline{Y}_i(w)$ (where *w* is the *window* and is a positive integer such that $w \le \lfloor m/2 \rfloor$ as follows:

$$\overline{Y}_i(w) = \begin{cases} \dfrac{\sum_{s=-w}^{w} \overline{Y}_{i+s}}{2w+1} & if\, i = w+1, \ldots, m-w \\ \dfrac{\sum_{s=-(i-1)}^{i-1} \overline{Y}_{i+s}}{2i-1} & if\, i = 1, \ldots, w \end{cases}$$

- Thus if *i* is not too close to the beginning of the replications, then $\overline{Y}_i(w)$ is just the simple average of 2*w* + 1 observations of the averaged process centered at observation *i* (see fig. 8). It is called a moving average since *i* moves through time.

- Plot $\overline{Y}_i(w)$ for i=1,2,...,*m-w* and choose *l* to be that value of *i* beyond which $\overline{Y}_1(w)$, $\overline{Y}_2(w)$,...appears to have converged.

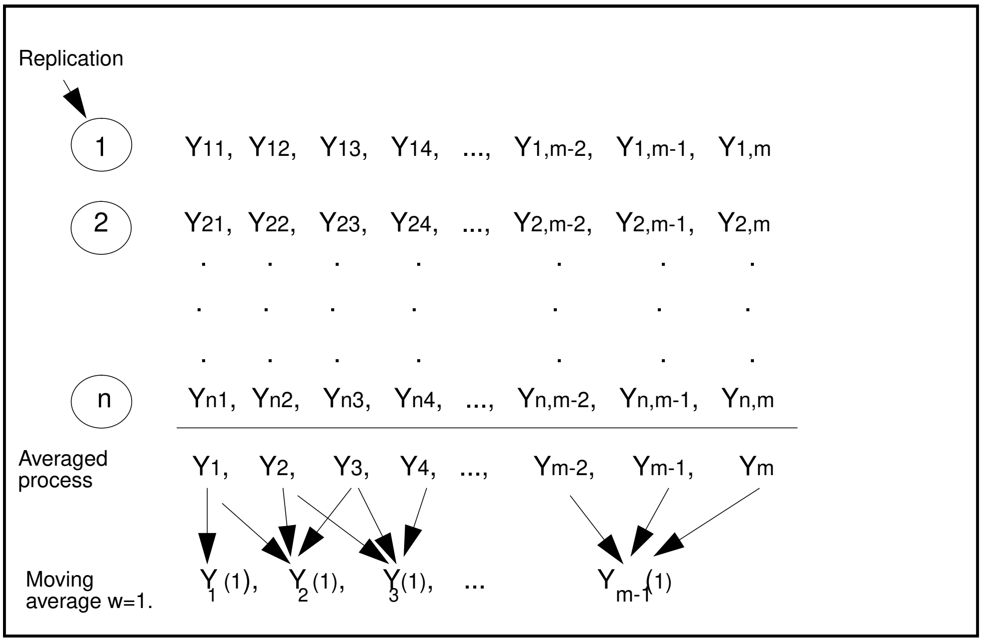


Figure 8: *Averaged process and moving average with w=1 based on n replications of length m.*

The plot $\overline{Y}_i(w)$ for i = 1,2,..., 300 - *w*; where $\overline{Y}_i$ are the averaged percent of rejected calls which occurs in an interval of time from a single run, for both cases are showed in the following histograms:

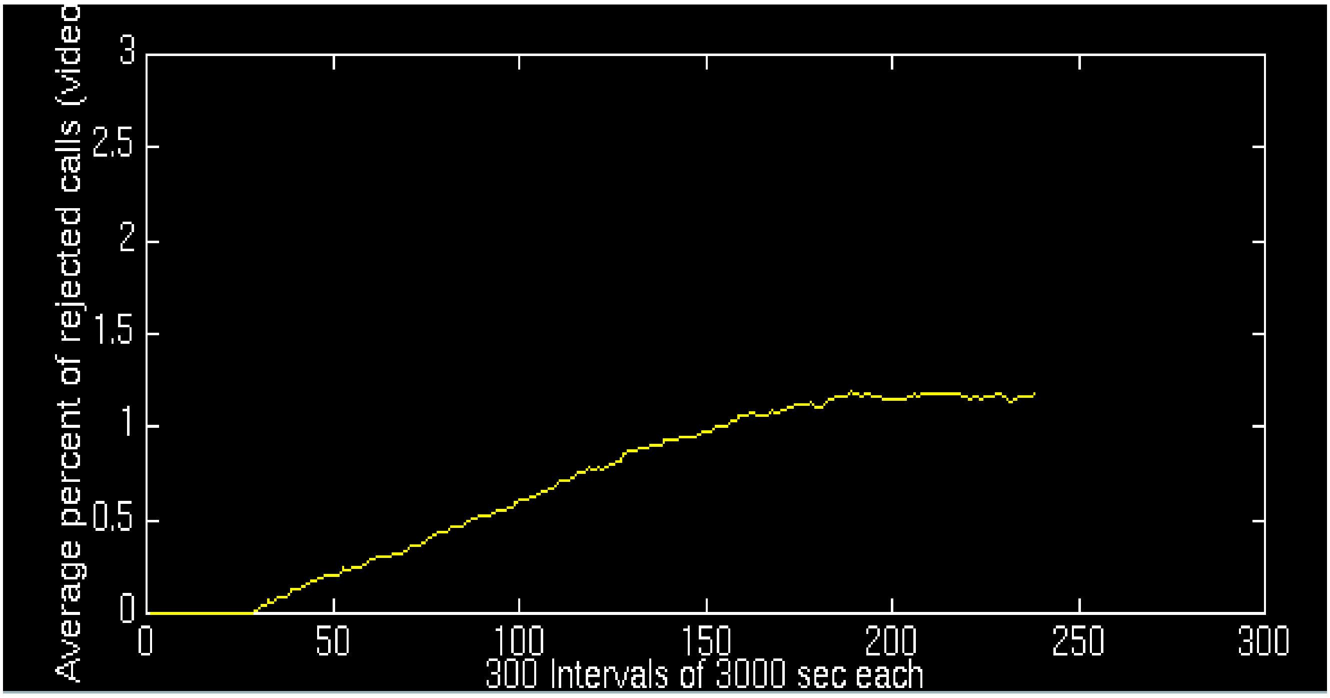


Figure 9: *Averaged percent of rejected calls for video request service. w= 60,l=160*

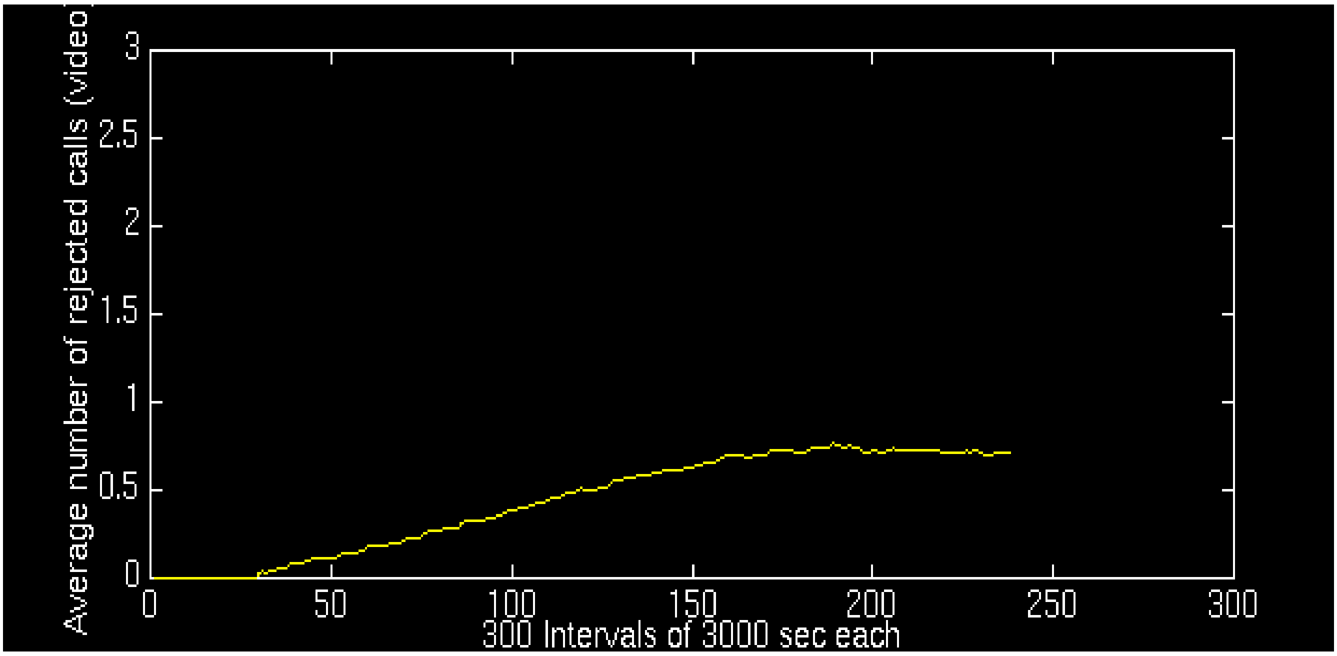


Figure 10: *Averaged number of rejected calls for video request service. w=60,l=160.*

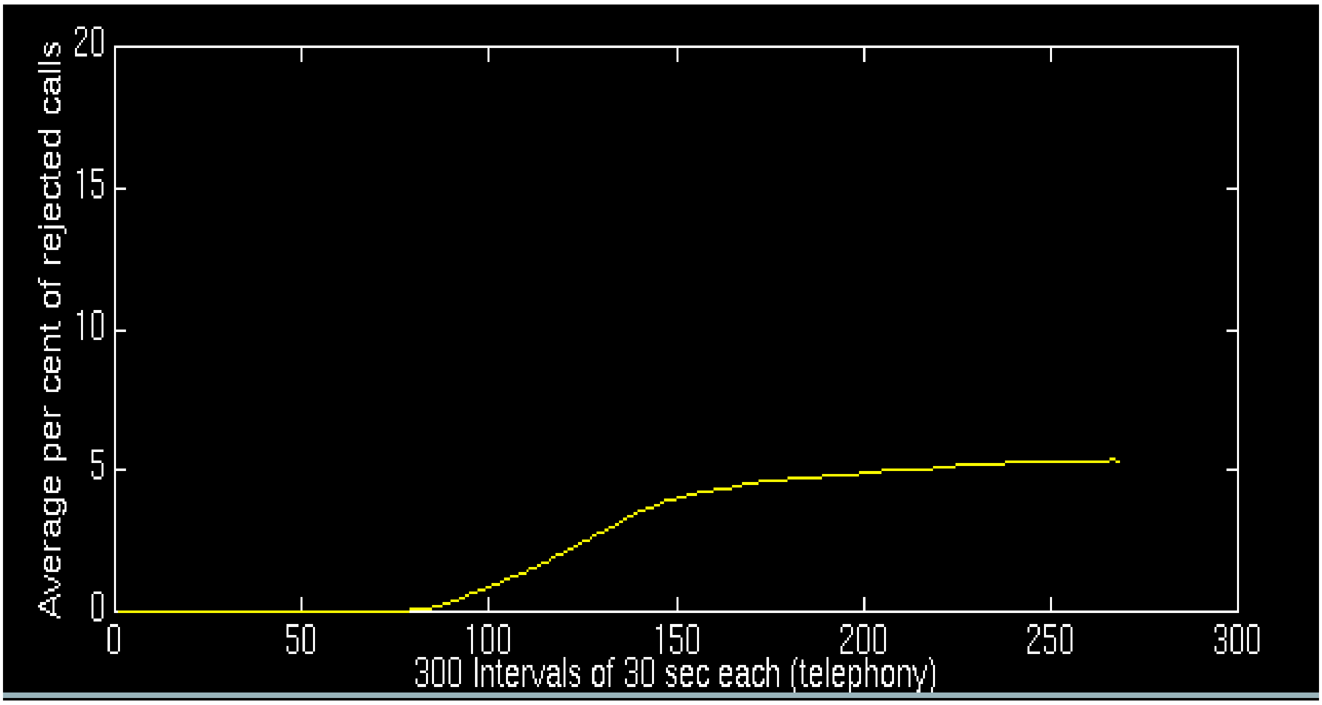


Figure 11: *Moving averages for 300 sec throughputs, telephony service: w = 30 l = 170*

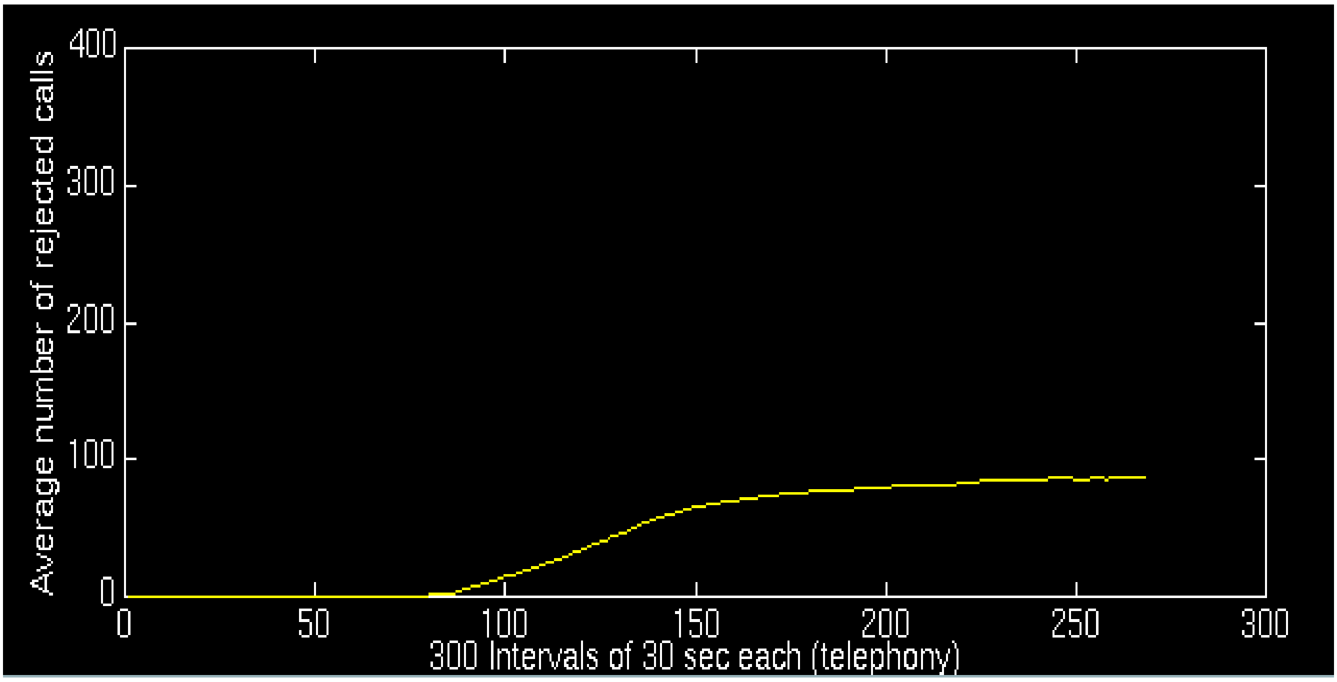


Figure 12: *Moving averages for 300 sec. throughputs, telephony service: w=30 l = 170*

For video service the value of *'l'* is that value of *'i'* beyond which $\overline{Y}_i(w)$, i=1,2,3,...,300-w; appears to be converged.'l' is calculated: *l*=160 intervals that corresponds to 48000 sec simulation time. And the averaged process $\overline{Y}_i$ converge around to 1.16% blockings per interval, i.e. per a period of 3000 sec. (see figs 9 and 10).

For telephony service the value of 'l' is that value of 'i' beyond which $\overline{Y}_i(w)$, i=1,2,3,...,300-w; appears to be converged.'l' is calculated: *l*=170 intervals that corresponds to 5100 sec simulation time. And the averaged process $\overline{Y}_i$ converge around to 5.11% blockings per interval, i.e. per a period of 30 sec. (see figs 10 and 11).

9.2.2 Estimation of means and variances

Suppose that X1,X2, ,Xn are IID (independent identically distributed) random variables (observations) with finite population mean μ and finite population variance ν^2 and our primary objective is to estimate μ ; the estimation of ν^2 is of secondary interest. Then the *sample mean*

$$\overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n}$$

is an unbiased (point) estimator of μ ; that is, $E([\overline{X}(n)] = \mu$. (Intuitively, $\overline{X}(n)$ an unbiased estimator of μ means that if we perform a very large number of independent experiments each resulting in an $\overline{X}(n)$, the average of the $\overline{X}(n)$'s will be μ) Similarly, the *sample variance*

$$S^2(n) = \frac{\sum_{i=1}^{n} [X_i - \overline{X}(n)]}{n-1}$$

is an unbiased estimator of ν^2, since $E[S\ (n)] = \nu^2$. The difficulty with using $\overline{X}(n)$ as an estimator of μ without any additional information is that we have no way of assessing how close $\overline{X}(n)$ is to μ Because $\overline{X}(n)$ is a random variable with variance $Var[\overline{X}(n)]$, on one experiment $\overline{X}(n)$ may be close to μ while on another $\overline{X}(n)$ may differ from μ by a large amount. The usual way to assess the precision of $\overline{X}(n)$ as an estimator of μ is to construct a confidence interval for μ .

9.2.3 Confidence intervals

Let Xi, μ , ν^2 be as above. We begin with a statement of the most important result in probability theory, the classical central limit theorem. Let Z*n* be the random variable $[\bar{X}(n)-\mu]/(\sqrt{\nu^2/n})$ and let F*n*(z) be the distribution function of Zn for a sample size of n; that is F*n(z)* = P(Z*n*<z). The theorem says, in effect that if *n* is "sufficiently large", the random variable Z*n* will be approximately distributed as a standard normal random variable, regardless of the distribution of the Xi's. It can also be shown for large *n* that the sample mean $\bar{X}(n)$ is approximately distributed as a normal random variable with mean μ and variance ν^2/n (note that μ and ν^2/n are the mean and variance of $\bar{X}(n)$.

The difficulty with using the above results in practice is that the variance ν^2 is generally unknown. However, since the sample variance $S\ (n)$ converges to ν^2 as *n* gets large, it can be shown that the Theorem remains true if we replace ν^2 by $S\ (n)$ in the expression of Zn. With this change the theorem says that If n is sufficiently large the random variable $t_n = [\bar{X}(n)-\mu]/(\sqrt{\nu^2/n})$ is approximately distributed as a standard normal random variable. Then it follows that

$$P(-z_{1-\alpha/2} \le [\bar{X}(n)-\mu]/(\sqrt{\nu^2/n}) \le z_{1-\alpha/2})$$

$$P(\bar{X}(n) - z_{1-\alpha/2}\sqrt{\nu^2/n} \le \mu \le z_{1-\alpha/2}\sqrt{\nu^2/n} + \bar{X}(n)) \quad = (1-\alpha)$$

The symbol = (in this case) means "approximately equal" and $z_{1-\alpha/2}$ (for 0 < a < 1) is the upper $1-\alpha/2$ critical point for a standard random variable. Therefore, if *n* is sufficiently large, an approximate 100 $(1-\alpha)$ percent confidence interval for μ is given by

$$\bar{X}(n) \pm z_{1-\alpha/2}\sqrt{\nu^2/n}$$

So if the Xi are random variables. the random variable *tn*= $[\bar{X}(n)-\mu]/(\sqrt{\nu^2/n})$ has a *t* distribution with *n-1* degrees of freedom [see 4 (p.288)] and an *exact* (for any n > 2) 100 $(1-\alpha)$ percent confidence interval for μ is given by

$$\bar{X}(n) \pm t_{n-1,\,1-\alpha/2}\sqrt{S^2(n)/n}$$

Where $t_{n-1,\,1-\alpha/2}$ is the upper $1-\alpha/2$ critical point for the *t* distribution with n-1 degrees of freedom. These criticals points are given in the Appendix of [4].

9.2.4 Replication/Deletion Approach for Means

The replication/deletion approach allows to obtain a point estimate and confidence interval for the steady-state mean v = E(Y) of the process Y1,Y2,Y3,...

So Let $\overline{Y}_{ij}$ be as defined before (see 9.2.1) and let Xj be given by:

$$X_j = \frac{\sum_{j=l+1}^{n} Y_{ji}}{m-l}$$

for j= 1,2,...,*n*. Where *n* - is the number of replications and *m* - number of stochastic processes.

Then Xj's are independent random variables with $E(Xj) \approx \upsilon$; X(n) is an approximately unbiased point estimator for 'v' and an 90 per cent confidence interval for 'v' is given by:

$$X(n) \pm t_{(n-1),(1-\alpha/2)} \sqrt{\frac{S(n)^2}{n}}$$

where X(n) and S(n) are computed above (see 9.2.2).

Then a point estimate and 90 per cent confidence interval for the steady-state mean of the process that simulate the percent of blockings are given by:

For video service request:

v = X(10) = 1.26469 and $S\ (10)$:1.1895 and t(9,0.9)=1.83

$$X(10) \pm t_{(9),(0.95)} \sqrt{\frac{1.1895}{10}} = 1.2647 \pm 0.631$$

And for telephony service request

A point estimate and 90 per cent confidence interval for the steady-state mean of the process that simulate the percent of blockings are given by:

v = X(10) = 5.61072 , $S\ (10)$: 0.1613 and t(9,0.9)=1.83

$$X(10) \pm t_{(9),(0.95)} \sqrt{\frac{0,1613}{10}} = 5.6103 \pm (0,2310)$$

ameters can also be adapted to a specified requirement.

Defining new requirements on the system or on a part of it, the simulation program could be extended and developed. So, the frontiers of a simulation of an ATM switch system remains open.

Looking at results, it seems better to execute PNNI without Timer, in case when the calls come in with high bandwidth of request (case of video). Timer is better for requests which not require to much bandwidth (case of telephony).

The mean time execution of the simulation program is about 3 min. in case of video service and 74 min. in case of telephony. In case of video the program uses 2Mbytes memory, in case of telephony it uses about 20 Mbytes of memory. Allocation of data in the memory and management of it is leaved to Erlang run-time machine, which executes Grabage Collection.

The results of the simulation program under certain conditions give a good understanding of how fine an ATM switch system manages a high transmission of requested calls.

11 Acknowledgements

I would like to thank the following people for their assistance:

Kent Konradsson, for his support and pacience in answering to my questions.

Jack Song, for supervising my work and pulling up with all my questions.

Tomas Ryden, Tomas Phil, Mikael Frank and Peter Karlsson at the department in Ericsson, for being a great support, and for fruitful discussions.

12 References

[1] PNNI Specification, The ATM Forum Technical Committee, Version 1.0, March 1996

[2] Traffic Management Specification, The ATM Forum Technical Committee, Version 4.0, March 1996

[3] Data and Computer Communications, William Stallings, Fourth edition, 1994

[4] Simulation Modelling & Analys, Averill M. Law, W. David Kelton , Second edition, 1991

[5] Köteori och Tillförlitlighetsteori, Ulf Körner, 1994

[6] Concurrent Programming in Erlang, Joe Amstrong, Robert Virding, Mike Williams, 1993

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	**Jorge Abarca Pereda**		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		**1997-11-12**	

Appendix A

The tested network

Here are all data of the tested network, the routing table presented and simulations results. The horizontals links are bi-directionals.

A.1 The 9 nodes Network

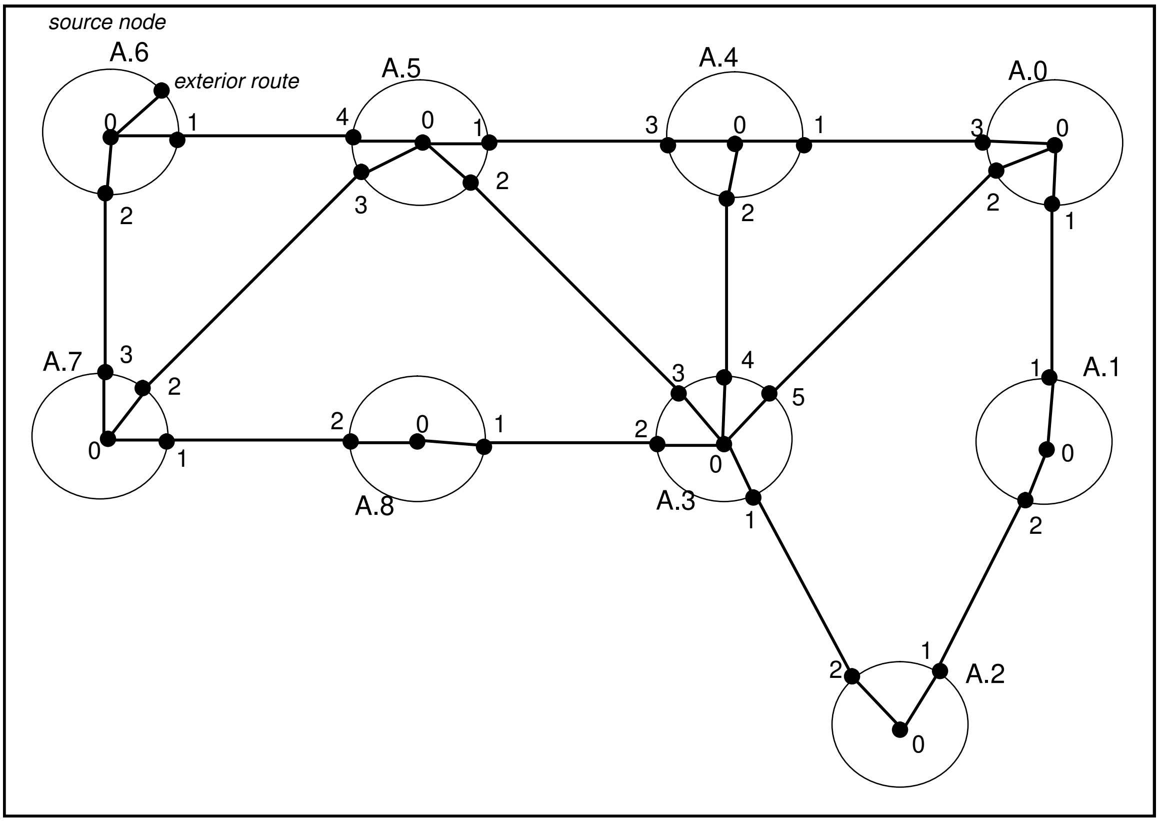


Figure 13: *The network used for test. Including a exterior route and horizontal links.*

When creating the routing table, following priority it does:

(1) ACR - Absolute Available Cell Rate.

(2) CDV - Cell Delay variation.

(3) AW - Administrative weight.

(4) Number of hops

A.2 Network’s dimension

Links	ACR	CDV	AW	Nr Hops	Link Cost
Inside A.0					
A.0:1 - A.1:1	130 Mb	250 μsec	5000	0	130
A.0:2 - A.3:5	130 Mb	250 μsec	5000	0	130
A.0:3 - A.4:1	130 Mb	250 μsec	5000	0	130
Inside A.1					
A.1:1 - A.0:1	130 Mb	250 μsec	5000	0	130
A.1:2 - A.2:1	130 Mb	250 μsec	5000	0	130
A.1:3 - A.3:6	130 Mb	250 μsec	5000	0	130
Inside A.2					
A.2:1 - A.1:2	130 Mb	250 μsec	5000	0	130
A.2:2 - A.3:1	130 Mb	250 μsec	5000	0	130
Inside A.3					
A.3:1 - A.2:2	130 Mb	250 μsec	5000	0	130
A.3:2 - A.8:1	130 Mb	250 μsec	5000	0	130
A.3:3 - A.5:2	130 Mb	250 μsec	5000	0	130
A.3:4 - A.4:2	130 Mb	250 μsec	5000	0	130
A.3:5 - A.0:2	130 Mb	250 μsec	5000	0	130
A.3:6 - A.1:3	130 Mb	250 μsec	5000	0	130
Inside A.4					
A.4:1 - A.0:3	130 Mb	250 μsec	5000	0	130
A.4:2 - A.3:4	130 Mb	250 μsec	5000	0	130
A.4:3 - A.5:1	130 Mb	250 μsec	5000	0	130
Inside A.5					
A.5:1 - A.4:3	130 Mb	250 μsec	5000	0	130
A.5:2 - A.3:3	130 Mb	250 μsec	5000	0	130
A.5:3 - A.7:2	130 Mb	250 μsec	5000	0	130
A.5:4 - A.6:1	130 Mb	250 μsec	5000	0	130
Inside A.6					
A.6:1 - A.5:4	130 Mb	250 μsec	5000	0	130
A.6:2 - A.7:3	130 Mb	250 μsec	5000	0	130

Links	ACR	CDV	AW	Nr Hops	Link Cost
Inside A.7					
A.7:1 - A.8:2	130 Mb	250 µsec	5000	0	130
A.7:2 - A.5:3	130 Mb	250 µsec	5000	0	130
A.7:3 - A.6:2	130 Mb	250 µsec	5000	0	130
Inside A.8					
A.8:1 - A.3:2	130 Mb	250 µsec	5000	0	130
A.8:2 - A.7:0	130 Mb	250 µsec	5000	0	130

Table 1: *Topology state parameters and link cost*

Additional data in cases of Video and Telephony services.

VIDEO	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8
Arrival intensity	23x 10-3	23x 10-3	23x 10-3	23x 10-3	23x 10-3	23x 10-3	23x 10-3	23x 10-3	23x 10-3
Mean Bandwidth	3 Mb	3 Mb	3 Mb	3 Mb	3 Mb	3 Mb	3 Mb	3 Mb	3 Mb
Mean Duration	5400 sec	5400 sec	5400 sec	5400 sec	5400 sec	5400 sec	5400 sec	5400 sec	5400 sec
Timer	3000 0 sec	3000 0 sec	3000 0 sec	3000 0 sec	3000 0 sec	3000 0 sec	3000 0 sec	3000 0 sec	3000 0 sec

Table 2: *For each call there is a mean bandwidth connection request, mean duration and an arrival Poisson distribution to every node (calls per sec), plus a Timer.*

TELE-PHONY	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8
Arrival intensity	6 c/ sec	6 c/ sec	6 c/ sec	6 c/ sec	6 c/ sec	6 c/ sec	6 c/ sec	6 c/ sec	6 c/ sec
Mean Bandwidth	64 Kb	64 Kb	64 Kb	64 Kb	64 Kb	64 Kb	64 Kb	64 Kb	64 Kb
Mean Duration	180 s	180 s	180 s	180 s	180 s	180 s	180 s	180 s	180 s
Timer	3 sec	3 sec	3 sec	3 sec	3 sec	3 sec	3 sec	3 sec	3 sec

Table 3: *.For each call there is a mean bandwidth connection request, mean duration and an arrival Poisson distribution to every node (calls per sec), plus a Timer.*

A.3 The routing table

Node	Node	Route
A.0	A.1	A.0 A.1 A.0 A.3 A.2 A.1
	A.2	A.0 A.3 A.2 A.0 A.1 A.2
	A.3	A.0 A.3 A.0 A.4 A.3
	A.4	A.0 A.4 A.0 A.3 A.4
	A.5	A.0 A.3 A.5 A.0 A.4 A.5
	A.6	A.0 A.3 A.5 A.6 A.0 A.4 A.5 A.6
	A.7	A.0 A.3 A.5 A.7 A.0 A.4 A.5 A.7
	A.8	A.0 A.3 A.8 A.0 A.4 A.3 A.8
A.1	A.0	A.1 A.0 A.1 A.2 A.3 A.0
	A.2	A.1 A.2 A.1 A.0 A.3 A.2
	A.3	A.1 A.0 A.3 A.1 A.2 A.3
	A.4	A.1 A.0 A.4 A.1 A.0 A.3 A.4
	A.5	A.1 A.0 A.3 A.5 A.1 A.2 A.3 A.5
	A.6	A.1 A.0 A.3 A.5 A.6 A.1 A.2 A.3 A.5 A.6
	A.7	A.1 A.0 A.3 A.5 A.7 A.1 A.2 A.3 A.8 A.7

Uppgjord/Prepared	Faktaansvarig - Subject responsible	Dokumentnr/Documentnr	
Jorge Abarca Pereda	Jorge Abarca Pereda		
Dokansv/Godkänd - Doc respons/Approved	Kontr/Checked	Datum/Date Rev	Tillhör/Referens-File/Reference
Kent Konradsson ETX/XBN		1997-11-12	

Node	Node	Route
	A.8	A.1 A.0 A.3 A.8 A.1 A.2 A.3 A.8
A.2	A.0	A.2 A.1 A.0 A.2 A.3 A.0
	A.1	A.2 A.1 A.2 A.3 A.0 A.1
	A.3	A.2 A.3 A.2 A.1 A.0 A.3
	A.4	A.2 A.3 A.4 A.2 A.3 A.0 A.4
	A.5	A.2 A.3 A.5 A.2 A.3 A.4 A.5
	A.6	A.2 A.3 A.5 A.6 A.2 A.3 A.5 A.7 A.6
	A.7	A.2 A.3 A.5 A.7 A.2 A.3 A.8 A.7
	A.8	A.2 A.3 A.8 A.2 A.3 A.5 A.7 A.8
A.3	A.0	A.3 A.0 A.3 A.4 A.0
	A.1	A.3 A.2 A.1 A.3 A.0 A.1
	A.2	A.3 A.2 A.3 A.0 A.1 A.2
	A.4	A.3 A.4 A.3 A.5 A.4
	A.5	A.3 A.5 A.3 A.4 A.5
	A.6	A.3 A.5 A.6 A.3 A.5 A.7 A.6
	A.7	A.3 A.8 A.7 A.3 A.5 A.7
	A.8	A.3 A.8 A.3 A.5 A.7 A.8

Node	Node	Route
A.4	A.0	A.4 A.0 A.4 A.3 A.0
	A.1	A.4 A.0 A.1 A.4 A.3 A.2 A.1
	A.2	A.4 A.3 A.2 A.4 A.0 A.1 A.2
	A.3	A.4 A.3 A.4 A.5 A.3
	A.5	A.4 A.5 A.4 A.3 A.5
	A.6	A.4 A.5 A.6 A.4 A.5 A.7 A.6
	A.7	A.4 A.5 A.7 A.4 A.5 A.6 A.7
	A.8	A.4 A.3 A.8 A.4 A.5 A.3 A.8
A.5	A.0	A.5 A.3 A.0 A.5 A.4 A.0
	A.1	A.5 A.3 A.0 A.1 A.5 A.3 A.2 A.1
	A.2	A.5 A.3 A.2 A.5 A.4 A.3 A.2
	A.3	A.5 A.3 A.5 A.4 A.3
	A.4	A.5 A.4 A.5,A.3 A.4
	A.6	A.5 A.6 A.5 A.7 A.6
	A.7	A.5 A.7 A.5 A.6 A.7
	A.8	A.5 A.7 A.8 A.5 A.3 A.8
A.6	A.0	A.6 A.5 A.4 A.0 A.6 A.5 A.3 A.0

Node	Node	Route
	A.1	A.6 A.5 A.3 A.2 A.1 A.6 A.5 A.4 A.0 A.1
	A.2	A.6 A.5 A.3 A.2 A.6 A.7 A.8 A.3 A.2
	A.3	A.6 A.5 A.3 A.6 A.7 A.8 A.3
	A.4	A.6 A.5 A.4 A.6 A.7 A.5 A.4
	A.5	A.6 A.5 A.6 A.7 A.5
	A.7	A.6 A.7 A.6 A.5 A.7
	A.8	A.6 A.7 A.8 A.6 A.5 A.7 A.8
A.7	A.0	A.7 A.5 A.4 A.0 A.7 A.8 A.3 A.0
	A.1	A.7 A.5 A.4 A.0 A.1 A.7 A.8 A.3 A.0 A.1
	A.2	A. 7 A.5 A.3 A.2 A.7 A.8 A.3 A.2
	A.3	A.7 A.8 A.3 A.7 A.5 A.3
	A.4	A.7 A.5 A.4 A.7 A.6 A.5 A.4
	A.5	A.7 A.5 A.7 A.6 A.5
	A.6	A.7 A.6 A.7 A.5 A.6
	A.8	A.7 A.8 A.7 A.5 A.3 A.8
A.8	A.0	A.8 A.3 A.0 A.8 A.3 A.4 A.0
	A.1	A.8 A.3 A.0 A.1 A.8 A.3 A.2 A.1

Node	Node	Route
	A.2	A.8 A.3 A.2 A.8 A.7 A.5 A.3 A.2
	A.3	A.8 A.3 A.8 A.7 A.5 A.3
	A.4	A.8 A.3 A.4 A.8 A.3 A.5 A.4
	A.5	A.8 A.3 A.5 A.8 A.7 A.5
	A.6	A.8 A.7 A.6 A.8 A.3 A.5 A.6
	A.7	A.8 A.7 A.8 A.3 A.5 A.7

Table 4: *Its presented 2 routes calculated by Dijkstra.*