ROUTING MANAGEMENT APPLICATION BASED ON MOBILE AGENTS ON THE INTERNET2

Angélica Reyes, Ernesto Sánchez, Antonio Barba [1]
Dept. of Applied Mathematics and Telematics.
Escuela Técnica Superior de Ingenieros de Telecomunicación de Barcelona
Technical University of Catalonia (UPC).
c/ Jordi Girona 1-3, Module C-3, Campus Nord, 08034 Barcelona, Spain.
angelica@mat.upc.es; ernesto@mat.upc.es; telabm@mat.upc.es

ABSTRACT

In this work, the design and development of a routing management application based on mobile agents in an Internet 2 (I2) environment is proposed. The objective of the application is to provide a routing based on the quality of service defined on a Service Level Agreement (SLA). For this purpose, the application obtains in each moment the best route according to a traffic and congestion control. As a consequence of the distributed operation scenario and the requirement of real time functionality, the application uses CORBA and intelligent mobile agent technologies.

1. INTRODUCTION

Nowadays, the Internet only offers one Quality of Service (QoS) level based on delivery of "best effort". Currently some applications do not work properly with this network model, because of varying delays and losses due to congestion.

Routing is one mechanism to alleviate congestion. The problem is that the current Internet routing protocols do not consider QoS explicitly, for example Open Shortest Path First (OSPF) and Routing Information Protocol (RIP), etc. are based on traffic parameters, number of hops, level of congestion, links, costs, etc. This paper presents a routing management application from another point of view, where the principal element is a QoS according to a predefined SLA (Service Level Agreement). The user terminal has a client where the QoS parametrs are specified. This information is negotiated with the server application. The server application, with the information of the intelligent agents, defines the best routing for the connection. In this case, the quality of network service depends basically on two issues: the sufficiency of network resources and the capability of a traffic-handling mechanism to efficiently utilize the available resources.

Routing is a mechanism implemented in the networks nodes to collect, maintain and distribute information about paths to different destination in the network. The node may treat the packets differently based on the field type of service (TOS) of the Internet protocol version 4 (Ipv4) packet header. The basic philosophy of Differentiated Services is to use the TOS octet in a way that enables service differentiation throughout the network, without keeping track of all flows at every node.

To allow different levels of service in the network it is necessary that the nodes have all the mechanisms to control QoS. Our application uses the potential of Differentiated Services to work with requirements of different applications, different QoSs, and to allow several tariffing levels in the Internet service use.

In this paper, the management Corba application chooses paths in every I2 domain that follow the packets. This choice is related to the contract between customer and service provider that specifies the forwarding service. This Service Level Agreement (SLA) is stored on an integrated Oracle database. Furthermore, an intelligent mobile agent application (Grasshopper) provides the control information to support this routing decision.

The article is structured as follows, the following chapter describes the basic operations scenario for the network management applications. Chapter three describes the routing Corba-based application. Chapter four talks about the traffic and congestion agent application. Finally, in the fifth chapter, different mathematical expressions are provided for the validation of the proposed scenario.

2. NETWORK SCENARIO

2.1 Internet2

Internet2 aim to develop common standards and support services for new classes of applications and to encourage the development of advanced real-time multimedia applications. Also, it will develop the

[1] This work has the support of CICYT
TIC1998-0495-C02-01

network infrastructure and service differentiation needed to support them. Future applications may require additional types of services. It is important that the Internet2 design be flexible enough to accommodate both currently anticipated requirements as well as new requirements, as they become known.

When multiple levels of service are available some form of resource control or cost accounting must be implemented with feedback to the end user to ensure that the appropriate level of service is requested. Since the best charging model for Internet2 is not obvious, Internet 2 will be used initially to develop and test methods of cost allocation. The management application proposed in this paper envisages the possibility of tariffing approaches to resolve this problem. The defined client application would provide different QoS level and management requests for at least five parameters: transmission speed, bounded delay and delay variance, throughput, schedule and loss rate.

2.2 Differentiated Services

The Differentiated Services architecture allows supporting different quality of service levels in a network. The ability to provide these services depends on management tools used in router monitoring. The Corba application proposed in this paper is intended to facilitate policy management to the nodes. Considering differentiated services, the treatment of flows should be similar in every node and over every hop to provide reasonable network service. From this perspective, different Per-Hop Behaviour (PHB) classes are proposed. This section defines four PHB groups.

1. Class selector Per-Hop Behaviour (CS PHB): CS PHB should give packets a probability of timely forwarding that is not lower than that given to packets marked with a lower class selector PHB, under reasonable operating conditions and traffic loads. Particularly CS PHB can conflict during congestion.

2. Expedited forwarding Per-Hop Behaviour (EF-PHB): the objective of EF-PHB is to provide tools to build a low loss, low latency, low jitter, assured bandwidth. EF-PHB means a strict bit-rate control in the boundary node and as quick forwarding in the core routers as possible.

3. Assured forwarding Per-Hop Behaviour (AF-PHB): AF-PHB can have a number of PHB classes (N), each with a number of drop precedence levels (M). It requires an active queue-management algorithm.

4. Dynamic RT/NRT Per-Hop Behaviour (DRT-PHB): The DRT-PHB group defines a system with two PHB classes and six PHBs in each of the classes. PHB classes offer two distinctly different delay characteristics: RT class is for flows needing real-time service, and NRT is for flows without strictly delay requirements. Six importance levels offer wide dynamics for various traffic-control and pricing schemes.[8]

The lowest importance level of each PHB class forms a best-effort service that has essentially the same characteristics all the time.

2.3 Multiprotocol Label Switching (MPLS)

In the Corba management application, Multiprotocol Label Switching (MPLS) is used in every domain to make the routing easier. MPLS makes use of labels for routing and forwarding packets, and potentially allows a common approach to traffic engineering, QoS routing, and other aspects of operation.

MPLS adds a 32-bit label, which has specific information of routing in every IP packet. The label allows address packets to the routers by predetermined paths. In the edge router, a label with information to alert the router about the next hop, is added to every packet,. When the router sends a packet to the next hop, this packet carries the label. Consequently, the network does not need to examine the packet header, only to read the MPLS label, to add it in a routing table and specify the next hop. As the packet crosses the network, if necessary, the packet is labelled again to indicate a better path.

In a Differentiated Services Network, the MPLS label should refer to a PHB class. Differentiated Services use various PHBs to operate packets with different traffic. To know which PHB is to be applied, it is necessary to indicate the PHB in the IP header of every packet. In this way, the same path and the same buffer transmit packets of the same PHB class. [8]

3. ROUTING MANAGEMENT APPLICATION

3.1 Introduction

As it was seen before, it is clear now that I2 architecture will need a network and service management platform in order to support new services evolution and implementation. This management platform should, among others, permanently monitor the network resources allocated to an application with a given QoS, perform actions over these resources in order to dinamically adapt them to the instantaneous traffic characteristics allowing supporting the QoS parameters initially agreed in the SLA. Furthermore, to implement security mechanisms like user authentication and establish a fair way to tariffing the service provided.

In order to reach some of the main objectives stated before, the management application proposed in this work could be divided in two parts: a CORBA –

based routing service management application, database directory, and a intelligent agents – based network resources management application.

The first part or server application will use the latest CORBA version, to take advantage of interoperability between different manufacturer platform whether hardware and software, and facilities for clients and servers in various programming languages [3]. It is composed by a Oracle database in which the Service – Level Agreement (SLA) between user and network will be saved, as well as routing and network resources availability information gathered by agents. Every time a user tries to establish a connection through the Network, he or she must to agree the required QoS parameters with the management application, using a SLA and through the edge router at which the user is connected. The management application consults its own database to know how much available the network resources are and then establish either the QoS parameters of user application and the route the user packets must follow in the network. This SLA is saved in the Oracle database, the routing information is communicated to the edge router and the connection is established.

Once the user connection has started, the management application creates static agents inside each node belonging to the followed path. These agents will permanently monitor the QoS parameters all along the connection, and mobile agents will be sent from the management platform, through every node in the connection, taking the information gathered by static agents in return to management application. If it detects some loss in the connection performance, it will use management mobile agents to take control over those affected network nodes adapting routing algorithms and queue strategies.

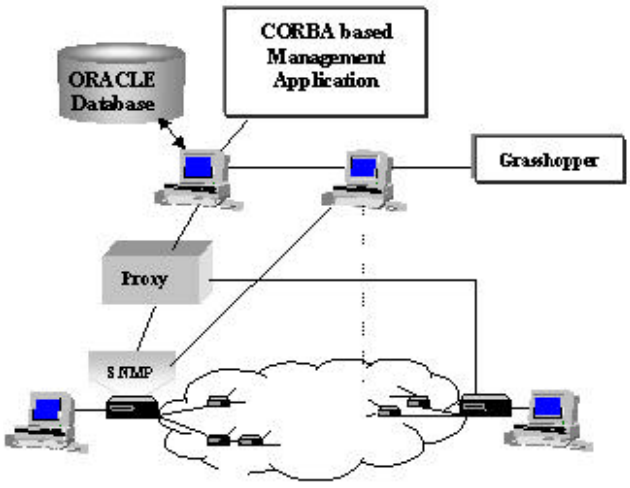

Figure 1. Scheme of routing management application

Furthermore, the Corba management application uses the Lightweight Directory Access Protocol (LDAP) to provide to each router with the service disciplines stored in the directories. The directory (Directory Enabled Network, DEN), implemented by Oracle, extends the information model to describe how the network elements and services behave. A directory service is a physically distributed, logically centralized repository of infrequently changing data that is used to manage the network. The directory service that is used by the application provides a means for locating and identifying users and available resources in a distributed system. Directory services also provide the foundation for adding, modifying, removing, renaming, and managing system components without disrupting the services provided by other systems components. DEN's use of LDAPv3 enables DEN schema information to be shared between any directory and any client that uses LDAPv3.

In order to explain with more detail the proposed application environment, the next section enumerates the different basic processes that work with the management application. Figure 1 shows graphically the routing management application environment that is proposed in this paper.

3.2 Task Process of the routing management application

The basic sequence of processes taken into account between the Corba management server and the client of the user terminal is as follows:

1. The terminal is connected with the edge router. The terminal does an identification process.

2. The edge router consults information from the directory (DEN) of the management application using LDAP protocol.

3. The application uses SNMPv2 routers and these routers can not support CORBA, for this reason, it is necessary to use a proxy SNMP/CORBA. There are some proxies between SNMP and CORBA, for example AdventNet of SNMPv2, which allow interactions between CORBA manager and SNMP agents.

4. The edge router consults to the management application server, the best point-to-point routing path that the packets must to follow in the network domain. The edge router sends to the application the router source address, the router target address and the customer reference.

5. The server makes an identification process of the edge router and the identification and authentication of the terminal.

6. The management Corba system consults the SLA that the customer agreed with the service provider in the Oracle directory. For this operation the application uses the customer reference.

7. The Corba management application needs the next information: QoS, Differentiated Service priority policies, SLA and routing dynamics tables to obtain the best path. This information is encapsulated in the General Inter ORB Protocol (GIOP) request like a component of the object reference from CORBA.

8. The customer exports the object reference as a GIOP parameter request in the server object. This is possible by the Portable Object Adapter (POA). The ORB is saved on the Oracle directory database.

9. The management application obtains a routing reference sequence of the core routers. The application gives this sequence to the edge router.

10. During the transmission, the routing is based on MPLS protocol, the management system sends to the routers updated routing tables based on QoS criteria.

11. Also, during the transmission, mobile agents transport parameters (packet-loss ratio, bandwidth, delay average, etc.) and some delivery features (times of birth, dead of request, reliability, and queue management) to the routers in order to provide differentiated services policies and traffic management.

Basically, mobile agents during the connection perform three functions. First, as depicted in figure 2, they circulate between all the routers involved in each connection gathering the monitored information from static agents, and taking this data to the management application in order to allow it to make the appropriate decisions and measures, considering the received data from agents and the connection QoS parameters specified in SLA, like change priority in queues, moving edges for each service in queues, modifying routing strategies, etc. The second mobile agents' function is now clear: they must to take the control by means of the Grasshopper application and deliver the resulting information to the CORBA application. Lastly, this application sends commands to the appropriate devices, i. e. routers, in the connection. Thus, there is a complete, immediate control and monitoring over the QoS of the service the Net is giving. The third function of mobile agents is to maintain updated the routing data in the Oracle Database in order to allow the management application to generate route tables for new connections. In order to perform this task, mobile agents explore the entire Internet2 domain where they are hosted, taking information about state of links and routers and deliver it to the management application.

4. GRASSHOPPER

Grasshopper v2.0 is an OMG's CORBA based Distributed Agent Environment, completely developed in Java by GMD FOKUS and IKV++. In this case, it is used for the development of a congestion and traffic management application based on intelligent mobile agents. Also, Grasshopper is MASIF[9] and FIPA[10] compliant. These features guarantee the integration with other middleware technologies, like Java and CORBA, and with third party agent platforms MASIF compliant as well. Therefore, this platform provides an integrated environment for both client/server and mobile agent technologies [11, 12].

4.1. Grasshopper Components

Grasshopper components are, essentially, Agents, Agencies, Places and Regions. These components will be defined in the following sections, as well as the importance they have in the proposed application, the role they play in Network routing and the reasons why an intelligent agent based technology has been chosen as an essential part of this application.

4.1.1. Agents

An agent is a piece of software that runs in a certain environment, and it has some degree of autonomy. Also, a mobile agent has the ability of migrate, i. e. it can change his physical location in the network, in order to perform his tasks locally, even in the middle of a task execution.

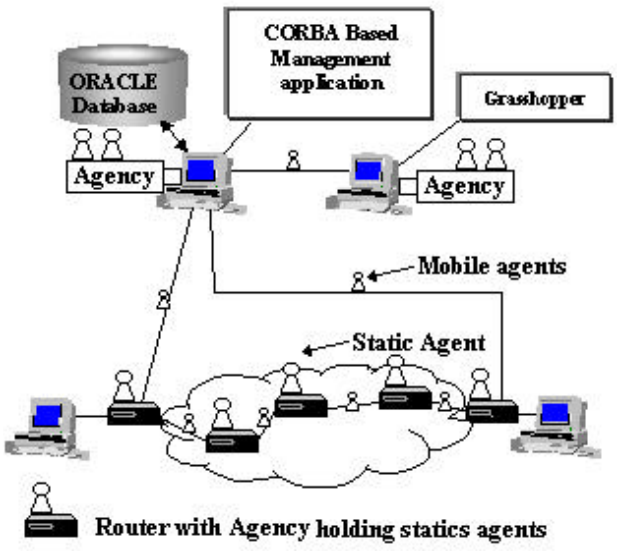

Figure 2. Management Application and Agents.

There are two types of a Grasshopper agent: Mobile and Static Agents. The later does not have the ability of migrate between different network locations. Instead, it is associated with a specific location and performs specialized tasks.

In this context, either static and mobile agents perform meaningful monitoring tasks in both routers and their service connections.

Static agents contain permanently updated information about all the variables associated with routers: queues occupation, current throughput of

each link in the router, etc. Also, they maintain statistical data from every connection as well, like delays, packet loss rate, delay jitter, current service rate, etc. in order to keep track on QoS parameters of each connection. Optionally, they could perform more sophisticated management functions, including control features, like security aspects, multicast and broadcast services management, alarms management, etc. distributing either network and service management tasks, just including a few code lines.

4.1.2. Agencies

Agencies, or *Agent Systems*, are the very runtime environments for both mobile and stationary agents.

A Grasshopper Agency comprises two parts: Core Agency and one or more Places. In the Core Agency reside the minimal functionalities in order to support agents' execution, providing the following services:

Communication services, responsible of all kind of interaction between all Grasshopper distributed components. All interaction can be performed through sockets connections, Java RMI or CORBA IIOP.

Registry services, which allow to find any entity in the environment, contributing with its management tasks.

Management services, that allows to human user perform monitoring and control tasks over places and agents.

Security services, that comprises both internal and external security mechanisms. The former, based on JDK security mechanism, offers protection to agencies resources against not authorized access. The later assure secure remote interactions between distributed components. Due to it, the confidentiality, data integrity and mutual authentication between both communication components can be achieved.

Persistence services, that enables the storage of data and code inside agents, places and agencies on a persistent medium for information recovery purposes.

As was noted before, this scheme requires one agency on each Network device involved in either transmission and control functions. Agencies in routers will provide all functionalities needed by agents in order to perform their monitoring tasks. Agencies and their services provide mechanisms to allow mobile agents to move between network nodes, offer a communication path between mobile and static agents in order to change monitoring data as well as commands from the management application, authenticating each agent arriving the node, assuring that any not authorized agent could damage the data integrity in static agents, without involving security mechanisms from the protocol stack of the Network. The Network does not need to be worry about this security aspects: agencies do among each other.

Thereafter, this agencies system will allow a smooth evolution toward a more complete, better supervised and, therefore, more efficient management systems and networks. Agencies, through their places (defined in the following section), could provide more sophisticated services to agents, allowing them to perform specialized management tasks and contributing in an efficient use of networks resources, improving QoS offered to final users. For example, since agents and agencies are monitoring traffic *in situ*, i. e. on each network node, they could collaborate in the calculation of service costs, allowing the network to offer its services in a fair way.

4.1.3. Places

Places reside in agencies and comprise several resources sets that enhance the core agency functionalities. Specific and specialized functions and services are defined and implemented in places.

Places are Grasshopper entities very useful in the application proposed in this article, because the core agency can not be modified by the application developer/manager: it offers basic functionalities and services to agents. Instead, special functions or services, those that could be useful to implement and all future enhancements that should be included in the application, must to be implemented as places inside agencies. This special functions could be related with cost calculating, alarms management, etc.

4.1.4. Regions

Regions are constituted by registries, in which both agencies and their places are associated with by means of their registration. Regions allow perform management of distributed components in this environment. Also, when an agent migrate to an agency, it is automatically registered in the appropriate region registry. Since an agent is able to move between agencies of different regions, every time an agent changes its location, the region registries are updated. Thus, entities are able to locate agents, agencies or places inside a region, and connections between agents and/or agencies are performed in a effective way.

In this context, each Internet2 Domain could be associated with a Grasshopper Region, enabling the network administrator to distribute network and services management among all the network components. For example, it would be possible to monitor and control locally internal routers and their queues, as well as the traffic circulating among them, by specialized agents enabled to make decisions on routing algorithms considering the actual traffic behavior.

Furthermore, it would be possible to split very large Internet2 Domains in several Grasshopper Regions in

order to facilitate and enhance its distributed management.

This way, it is possible to establish work spaces for those agents that explore de domain, i. e. the region, in order to take network resources occupation information and update the management application routing database.

4.2. Why Agents?

At a first sight, it seems that introducing agents in a network will introduce some drawbacks as well [13]: as agents migrate between network nodes, they inevitably will consume network resources, like bandwidth, processor time, etc. Also, introducing intelligent, autonomous code segments (agents) some doubts about security aspects will be outlined. Furthermore, it will be difficult to simulate network performance due to undeterministic behavior of agents.

However, these drawbacks are not as important as they look like if compared with advantages that are introduced with agents' technology: In spite of resource consumption by agents, the number of those running among the network nodes limits this consumption. It has been found that a small population of routing agents improves QoS in certain network environment [14] without significantly overloading network resources. On the other hand, besides security mechanisms inherent to Internet2 protocol layer architecture, Grasshopper introduces its own security mechanisms, based on *X.509* certificates and Secure Socket Layer Protocol, and thus, security aspects depends on Grasshopper based application implementation and should not be a weak point in the Net.

Besides, in general, agent based network and service management applications will offer more flexible and adaptive software solutions, allowing creating and supplying services in a more automatic fashion, distributing and decentralizing management tasks by migration of control agents to those network nodes where they are needed.

On the other hand, Grasshopper features makes it an ideal platform for network management applications implementation, since it is MASIF and FIPA compliant, and therefore is able to completely integrate with CORBA and/or Java based applications, or with other agents platforms; Grasshopper interfaces and protocols, like CORBA IIOP, MAF IIOP, RMI, SSL etc.

It could be argued that there is no need to employ agent based technology to perform network nodes supervision, QoS monitoring and routing information gathering, because there are protocols that could do this work as well, like SNMP. However, it can be shown that using mobile agents in these tasks will improve the management application performance [15].

5. TRAFFIC AND CONGESTION CONTROL

In this section, the described agent scenario is modeled and mathematically evaluated. The purpose of this evaluation is to obtain design parameters for the maximum delay allowed and/or the size of the mobile agents.

The Grasshopper application is based on the use of predictive agents distributed in the network nodes. These nodes support agent algorithms with diverse threshold conditions according to the traffic levels of the network. The agents are integrated in an intelligent network that tries to predict traffic and to take measures in order to anticipate a congestion situation in the network. There are static agents allocated in the nodes and other elements of the network, they are autonomous and communicate among each other to adapt the traffic of the network. Other agents are mobile agents and they are sent by the management system through the connection paths and nodes collecting information and co-operating with the static agents in order to obtain the global control in the network.

The static agents work in the switches to establish an appropriate quality of service testing differently each type of traffic. Four classes of traffic have been defined, classified according to the priority level. The agents will adopt different management policies according to quality of service requirements defined for each traffic class and priority level. Furthermore, the results of the operations that perform the mobile agent of the managed node are periodically transferred to the agent of the manager node (Grasshopper application) by means of intelligent agents.

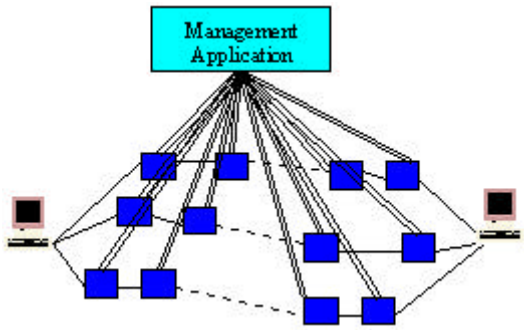

Figure 3. Network scenario with routing alternatives

Our framework incorporates the following features of the real system to capture the basis of the mobile agent architecture.

- Network management center as on-off source, according to the existence or not of a mobile agents.

- Routers considered as buffers with a processor on-off for the agents.
- Delays considered for the transmission and propagation of the mobile agents through the links.

The management center waits the return of all the mobile agents in each periodic cycle for the processing of the global status of the network and to obtain the policies for routing according to the quality of service.

In the algorithm, an agent feedback scheme is adopted. That is, each router sends a "stop" agent to the source when its queue length crosses a high threshold, and it sends a "start" agent when the queue length drops below a low threshold. The source stops sending traffic whenever at least one receiver has sent a "stop" signal. It sends at peak rate otherwise.

In the figure 4 can be seen a buffer with variable thresholds in function of the charge of messages in the input of the node.

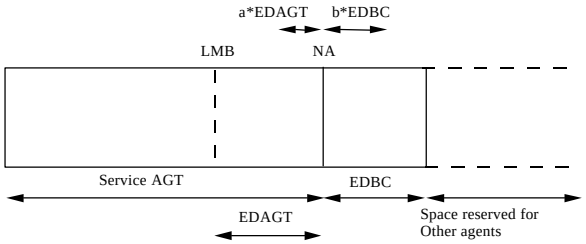

Fig. 4. Scheme of the node buffer of the core network with the threshold levels according to a congestion situation.

- Delay of medium service according the mean length of the buffer AGT (LMB) and the RATE.

- T_{sum}: Increase in b*(nº slots of available packets /RATE).

- T_{res}: decrease in a*[(NA-LMB)/RATE].

- Trend: trend = d(EDAGT)/dt.

being:

- Mean length of the buffer AGT (LMB).

- Available space in the common buffer (EDBC) with (a, b < 1).

- RATE of the port AGT of output (RATE).

- High threshold of the buffer AGT (NA).

The algorithm [17] in the zero-delay case, shuts off whenever one of the routers queues tends to overflow. That is:

- It is a lossless system, and achieves maximum throughput of agents (or cells) for a given buffer size.
- Buffer occupancy evolves periodically with alternation of one queue staying full and the other queue staying below a full buffer until they exchange position.

A formal description of the algorithm is as follows:

$$Trend_i = \{ l(t) - m_i(t), \text{ if } 0 < q_i(t) < NA_i, \text{ i = 1,2. (1)}$$

$$l(t) = \begin{cases} 0 \\ \\ m = 1 \end{cases} \quad \text{if} \quad (q_1(t) = NA_1 \text{ and } m_1(t) = 0) \quad \text{or}$$

$(q_2(t) = NA_2 \text{ and } m_2(t) = 0)$. Equal to 1 otherwise. (2)

In this case, $q_i(t)$ is the queue length at time *t* for queue *i*; $Trend_i$ is the queue growth rate at time *t*. $m_i(t)$ is the instantaneous throughput of the on-off router. $l(t)$ is the cell rate with which the source sends traffic at time *t*, Na_i is the buffer size for receiver queue *i*.

Note that, although the $m_i(t)$'s appear in the condition (2), the information comes from the feedback on the queue size, and no additional information is assumed.

The algorithm in the zero-delay case forces $l(t) = m_i(t)$ when $q_i(t)$ stays full. Notice that is says exactly "management center listen the slowest router" because, while queue 1 stays full and queue 2 is below a full buffer, which means that during this period router 1 has a lower capability that router 2, the source listens to router 1.

Furthermore, during the period in which $q_1(t) = NA_1$, the growth rate of queue 2 is $m_1(t) - m_2(t)$ where $m_1(t)$ and $m_2(t)$ are independent on-off Markov processes. Thus the two queues can be decoupled except at the boundary points, and the stationary distribution of each queue can be studied as a fluid-flow analysis.

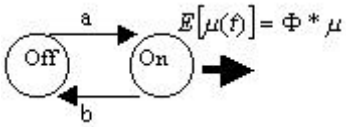

Figure 5. Notation for a router on-off model.

Probability of being *on*: $\Phi = \dfrac{a}{a+b}$

Instantaneous throughput capability: $m(t)$

Channel capability, constant: m

Average throughput capability: $E[m(t)] = \Phi * m$

In case of algorithm with zero delay propagation delay, routers with the same m and Φ. Condition $m_1 = m_2 = 1$ and $\Phi_1 = \Phi_2 := \Phi$.

For $i, j \in \{1,2\}$, we have:

$$\Pr(q_i \le x) = \frac{x + c_i}{NA_1 + NA_2 + c_1 + c_2}, \text{ for } 0 \le x \le NA_i \quad (3)$$

$$\Pr(q_i = 0) = \frac{c_i}{NA_1 + NA_2 + c_1 + c_2} \quad (4)$$

$$\Pr(q_i = NA_i) = \frac{NA_j + c_j}{NA_1 + NA_2 + c_1 + c_2} \quad (5)$$

where

$$c_i = (1-\Phi)\left[\frac{\Phi}{b_i} + \frac{(1-\Phi)}{b_j}\right] \text{ with } i \neq j \quad (6)$$

The throughput of the routers is the same and is given by

$$d_i = \Phi - \frac{\Phi(1-\Phi)(c_1 + c_2)}{NA_1 + NA_2 + c_1 + c_2} \quad (7)$$

the average buffer occupancy is given by

$$E\{q_i\} = \frac{1}{2} NA_i \left(\frac{NA_i + 2(NA_j + c_j)}{NA_1 + NA_2 + c_1 + c_2} \right) \quad (8)$$

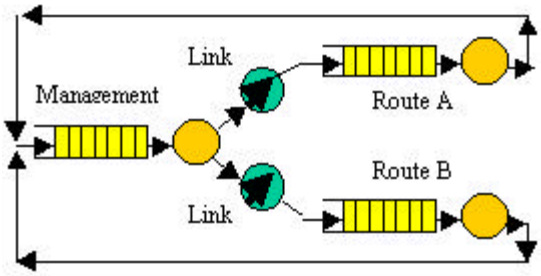

Figure 6. Mathematical model of the proposed scenario.

The effect of the propagation delay has not been considered in this case, however would be taken into account in the control capability of the source policy. Furthermore, the delay makes the system non-Markovian, and the fluid-flow analysis cannot be applied directly.

The obtained expressions have been defined in two routing paths with an only node for each route. However the results are useful for a general system with k-arcs (Routers) and different routing options. In [16] we can obtain more details about the construction of this kind of transitive graphs without loops.

6. CONCLUSIONS

The described architecture provides an improvement in the performance as a consequence of the use of mobile intelligent agents in a distributed network management environment. The CORBA and Grasshopper applications are multiplatform and provide a fast management for large networks. Both are secure environments, flexible to changes and allow a good resource management. The MASIF and FIPA standards provide also an open and interoperable system with other domains.

Furthermore, the LDAP/DEN architecture allows to handle user data, network elements, management policies, service disciplines, etc. Finally, the use of a QoS criteria allow to the network management CORBA application provide the best routing decision according to the SLA of each subscriber.

7. REFERENCES

[1] Real Time CORBA Joint Revised Submission. March 1999

[2] CORBA Messaging Joint Revised Submission. May 18,1998

[3] CORBA Version 3, http://www.omg.org/news/pr98/component.html

[4] Internet2, http://www.ietf.org

[5] I2 QoS, http://www.internet2.edu/qos/

[6] M. Carlson, W.Weiss, S.Blake, Z.Wang, D.Black, E.Davies. *"An Architecture for Differentiaded Services."*, RFC *2475.* Dec. 1998

[7] Internet Engineering Task Force, *"A conceptual Modelfor Diffserv Routers."*, March 2000.

[8] Kahevi Kilkki. *"Differentiated Services for the Internet."*, Macmillan Technology Series. 1999

[9] The OMG MASIF Standard, http://www.fokus.gmd.de/research/cc/ecco/masif/index.html

[10] FIPA, Foundation for Intelligent Physical Agents, http://drogo.cselt.stet.itlfiDa/sDec/fiya98/fiya98.htm

[11] IKV++ Grasshopper, a Platform for Mobile Software Agents, http://www.ikv.de/yroducts/grasshopper/

[12]IKV++ Grasshopper User's Guide, http://www.grasshoPDer.de/download/uguide.pdf

[13]N. Minar, K. Hultman Kramer and P. Maes, *"Cooperating Mobile Agents for Dynamic Network Routing"*, Proceedings of the 1st Hungarian National Conference on Agent Based Computation, 1999.

[14] T. White and B. Pagurek, *"Artificial Life, Adaptive Behavior, Agents Application Oriented Routing with Biologically-Jnspired Agents"*, In Proceedings of the Genetic and Evolutionary Computation Conference (GECCQ-99), July 1999.

[15] A. Barba, *"Gestión de Red*", Edicions UPC, 1999.

[16] S. P. Mansilla and O. Serra, *"Construction of k-arc transitive digraphs*.", Discrete Mathematics. Ed. Elsevier.

[17] A. Barba, *"Performance of Routing Management based on agents*.", UPC Internal Report. 2000.