Designing the Simulative Evaluation of an Architecture for Supporting QoS on a Large Scale

H. Tarasiuk, J. Mongay Batalla, R. Janowski, W. Burakowski
Institute of Telecommunications, Warsaw University of Technology, Poland
{halina; jordim; robert; wojtek}@tele.pw.edu.pl

G. Stea, C. Cicconetti
Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy
{g.stea; c.cicconetti}@iet.unipi.it

G. García-de Blas, F. J. Ramón Salguero
Telefónica I+D, Madrid, Spain
{ggdb; fjrs}@tid.es

J. Sá Silva
Lab. of Communications and Telematics CISUC-DEI, University of Coimbra, Portugal
sasilva@dei.uc.pt

ABSTRACT

The EuQoS system is a complete QoS system, scalable to large dimensions and addressing QoS at all relevant layers, which has been developed within the framework of the IST-EuQoS project. Its design has been aided by a considerable amount of modeling and simulation work, aimed at testing the various QoS mechanisms devised and their interaction. This paper describes the modeling and simulation work done in the framework of the project. We describe the three simulation models which have been developed, based on the different timescales at which the QoS mechanisms have effect. Furthermore, as a sample case of performance evaluation, involving different simulators, we describe the performance evaluation of the EuQoS signaling subsystem.

Categories and Subject Descriptors

C.4 [**Computer systems organization**] Performance of systems – design studies, performance attributes.

General Terms

Algorithms, Performance, Design

Keywords

Simulation, EuQoS, QoS, Performance Evaluation

1. INTRODUCTION

The greatest challenge for the future Internet is to support the provisioning of Quality of Service (QoS) on a wide scale. In order to achieve such an ambitious goal, the cooperation of several building blocks (e.g., routing algorithms, resource management schemes, admission control algorithms, traffic analysis techniques, signaling protocols) is required. While the above topics have been subject of extensive research in the last fifteen years, as testified by the abundance of related literature, such a scientific effort has not been translated so far into wide-scale QoS deployment.

A first considerable effort to fill this gap has been carried out in the framework of the IST-EuQoS project, [1], a joint effort of European universities, research centers and major telecom operators with the goal of defining and prototyping an architectural network model, called the *EuQoS system*. The latter (a first version of which is described in [3]), is in fact a complete architecture designed for supporting QoS on a large scale. The EuQoS system is able to satisfy QoS requirements i) on an end-to-end basis; ii) in a heterogeneous network scenario; iii) at all relevant layers, ranging from the application to the network control and data planes, and at all relevant timescales, ranging from network planning to packet forwarding; iv) scalable to large dimensions, in terms of number of calls and of involved Autonomous Systems, and v) relying on standard protocols as much as possible, possibly enhanced with new functionalities. The EuQoS system addresses both the core IP network and the currently most popular access technologies, such as UMTS (Universal Mobile Telecommunications System), WiFi, xDSL (x Digital Subscriber Line) and LAN/Ethernet. It specifies a set of end-to-end *Classes of Services* (CoSs), which are then used to support a broad range of applications, such as Voice over IP (VoIP), Video on Demand (VoD), Tele Engineering, Distance Learning, and Telemedicine, besides obviously standard TCP-based applications. A prototype of the EuQoS system has been deployed into a pan-European testbed, consisting of twelve fully functional testbeds located in various European countries, connected through national research networks (NRENs) and through GEANT.

The design of EuQoS has been supported by an extensive performance evaluation activity, aimed at both comparing alternative design choices and verifying the compatibility and performance of all the proposed solutions. Such evaluation, mainly carried out through simulation, has entailed a considerable modeling effort. More specifically, the EuQoS system has been modeled at three different levels of abstractions, reflecting the three different timescales at which QoS building blocks are active, i.e. the *resource provisioning*, *flow lifetime*, and *packet transmission* timescales. Furthermore, as the EuQoS system is designed for large-scale functionality (in term of number of involved networks/domains and calls), a specific attention has been devoted to the twofold problem of scale: on one hand, the simulators have been designed to be able to execute "large" simulations in reasonable time. On the other hand, the simulators have been used to assess the scalability of the critical components of the EuQoS architecture, testing scenarios with scales much larger than those achievable in the EuQoS prototype.

All the above activities have been made more challenging by the fact that the simulation and modeling effort was carried out in parallel with the system design, and by the general lack of a comprehensive reference prior work. In fact, although the simulation and modeling literature is rich with models for single QoS components, networks, and so on, this is – to the best of our knowledge – the first time that all the pieces are brought together in a coherent framework.

This paper describes the simulation models and the simulators used within the EuQoS project. Furthermore, it reports some sample results obtained with those simulators, related to performance evaluation of signaling, and a summary of the lessons, learned through simulation, which have steered the design process.

QoSim'08, March 3, 2008, Marseille, France.

The rest of the paper is organized as follows: Section 2 provides the necessary background on the EuQoS system. The general simulation models are presented in Section 3, and results of performance evaluation of the signaling system are shown in Section 4. Finally, we report conclusions in Section 5.

2. THE EUQOS SYSTEM

In this Section we outline the aspects of the EuQoS architecture, namely the QoS model, the architectural building blocks, the QoS interdomain routing and traffic engineering, the signaling framework, the resource reservation models and the QoS control mechanisms, which are essential to understand the contribution of this paper. We underline that there are many fundamental aspects of the EuQoS system which are not mentioned here due to lack of space (e.g., the application-to-network interface, the multicast middleware, the monitoring and measurement sub-system, and others). The interested reader is referred to [1], [3], [28] for more details.

2.1 The QoS Model

Since QoS must be provided on an end-to-end basis, a consistent view of CoSs and related QoS requirements is needed in the whole network. For this purpose, *end-to-end CoSs* are implemented in EuQoS following IETF recommendations [9]. The submitted traffics are marked accordingly (by the applications and/or the ingress node). These CoSs (e.g. *real-time telephony*, *multimedia streaming*, etc.) specify the target QoS requirements on end-to-end delay, jitter, and packet loss rate [10], [11] (henceforth collectively referred to as the *QoS parameters*). A best-effort, *standard* class is also defined for non-QoS traffic. Furthermore, a *signaling* CoS is defined. Each Autonomous System (AS) is free to offer transport services for a subset of the above end-to-end CoSs. In order to achieve a consistent forwarding treatment *across* AS boundaries, neighboring ASs negotiate per-CoS *peering Service Level Specifications* (*p-SLSs*). The latter are bi-lateral agreements, specifying the amount and shape of traffic that the downstream AS (*provider*) is willing to accept from the upstream AS (*customer*), and the QoS that the provider AS is committed to provide to conformant traffic. The traffic submitted from the customer has to conform to a *Traffic Specification* (TSPEC) [25], while the QoS guarantees from the provider describe the target value of the QoS parameters to traverse the AS, from the ingress to the exit points. Traffic of a given CoS is not allowed to flow at an inter-AS boundary unless a related p-SLS exists. p-SLSs come with some kind of financial settlement between the customer and the provider, whose modeling is not in the scope of the EuQoS project.

2.2 The EuQoS Architecture

As the primary goal of EuQoS is managing heterogeneity, the network functionalities in each AS are partitioned between a *Technology Independent* (TI) and a *Technology Dependent* (TD) network layers, shown in Figure 1. At the TI level, *Resource Managers* (RMs) possess the AS-wide knowledge of the available and reserved network resources and of the routing topology. RMs are in charge of enforcing Call Admission Control (CAC) and resource accounting. When the RM decides that the status of the resources needs to be changed, mainly as a result of administrative decisions (e.g. periodic network optimization cycles), it communicates with one or more *Resource Allocators* (RAs), lying at the border between the TI and the TD layers. The latter actually enforce the RM decisions by performing TD CAC and configuring each specific underlying network device.

2.3 Resource Provisioning Timescale: Routing and Traffic Engineering

As far as interdomain routing is concerned, EuQoS defines the EQ-BGP protocol [3], which extends BGP-4. EQ-BGP includes an optional path attribute, named QoS_NLRI that conveys information about the QoS parameters of a path, a configurable QoS-aware decision process for selecting the best end-to-end path also based on the QoS parameter values, and separate, per-CoS routing tables. A Traffic Engineering and Resource Optimization (TERO) module, located in the RM, interacts with EQ-BGP routers for controlling and configuring the interdomain routing. TERO configures queues and policers at interdomain links so as to provision the necessary resources to allow traffic to flow across neighboring ASs. TERO routing and provisioning decisions are taken either as a reaction to the variation of the network topology (e.g., negotiation of a new p-SLS with a neighboring AS) or periodically, for maintenance and optimization. Furthermore, TERO configures the EQ-BGP decision process so as a) to account for QoS parameters, and b) to select EQ-BGP updates that balance resource utilization. Finally, it configures EQ-BGP for updating outgoing update messages before they are advertised to another AS or inside an AS. In fact, TERO provides EQ-BGP routers with the *nominal* QoS parameter related to the transit along *intra-AS* paths and at *inter-AS* links. The latter correspond to those achievable around the *Maximum Admissible Load*, i.e. the point at which CAC would start rejecting calls.

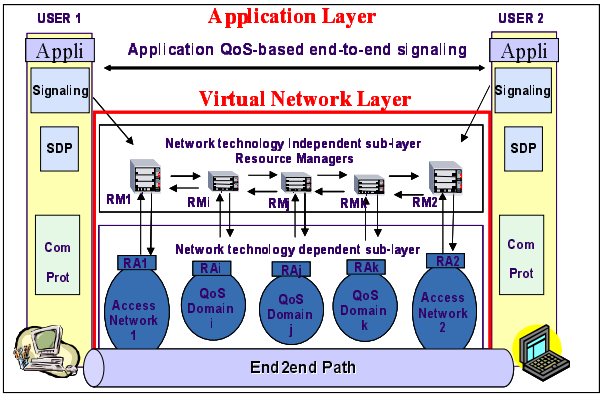

Figure 1 – High-level architecture of the EuQoS system

2.4 Flow Lifetime Timescale: Signaling

Three different types of flow lifetime signaling are required in EuQoS for handling calls: *inter-application*, *horizontal* and *vertical* signaling. The former is originated by the calling application as a first step, so as to agree on the QoS requirements with the remote endpoint(s). This is achieved through extensions of the Session Initiation Protocol (SIP) [26] and Session Description Protocol (SDP) [27] called EQ-SIP and EQ-SDP, respectively [5], which include mechanisms for the negotiation of QoS parameters. Horizontal signaling is again originated by the calling application, but it involves the exchange of messages through all the RMs along the end-to-end path (hence the name *horizontal*) to transport CAC requests, which are checked in each RM, up to the destination. The Next Step In Signaling (NSIS) protocol [6] has been extended (and called EQ-NSIS,[7]) in order to serve the purpose. EQ-NSIS ensures that the signaling messages traverse the same path as the data (i.e., through the same ASs, and border routers within each AS), *and* each RMs receives the message from the

ingress router and can therefore perform CAC before the signaling proceeds further.
Vertical signaling, instead, takes place between a RM and a RA, and it is not necessarily connected to horizontal signaling. It is performed via EQ-COPS, an extended version of the Common Open Policy Server protocol [8]. Each AS maintains its own Policy Repository (PR) and its own Policy Decision Point (PDP), the latter being usually co-located with the RM. The PR stores AS-specific policies, which define the high-level behavior of the RMs, i.e. which QoS requests should be satisfied and under what circumstances. The policies are communicated to the Policy Enforcement Points (PEPs), which are instead located in the RAs, where they are translated into specific network device configurations.
Two different resource reservation models have been implemented in EuQoS in order to solve this issue: resources are either bound to the end-to-end path *dynamically*, through RMs and RAs (*loose* model), or they are *statically* provisioned along it (*hard* model). The two models coexist in the EuQoS system. In the hard model, the processing of horizontal and vertical signaling at call setup time is reduced, since transit RMs/RAs along an end-to-end path need not control the admission of new calls and reserve resources accordingly.

2.5 Packet Timescale: QoS Control Mechanisms

Within the EuQoS architecture, a key role is played by the QoS control mechanisms enforced both in the core and in the access networks. The latter guarantee the correct forwarding behavior for the packets of each end-to-end CoS, by committing resources (bandwidth and buffer) to each of them. The QoS control mechanisms (i.e., packet schedulers, buffer management algorithms, policers) and CoSs are specific to each technology: for instance, in IP networks we implement DiffServ PHBs (Per Hop Behaviors) and the CoSs specified in [9]. Moreover, in some access technologies and in inter-domain links, end-to-end CoSs can be aggregated into broader, network CoSs. As a consequence, mapping criteria for end-to-end CoSs into network CoSs have to be defined and implemented in the RAs. Thanks to that the RA can enforce of the appropriate QoS control mechanisms in each specific network technology.

3. SIMULATION MODEL

Having presented the most relevant features of the EuQoS system, we now describe how we modeled it for performance evaluation purposes. The key observation to modeling such a complex system was that the above described QoS mechanisms work at three different timescales, namely the *resource provisioning timescale* (i.e., days or weeks), the *flow lifetime timescale* (i.e., seconds to hours), and the *packet transmission timescale* (i.e., milliseconds and below). As a consequence, different simulation models and simulators were defined, with different objectives. More specifically, a *packet transmission level (PTL)* simulator has been devised for assessing the effectiveness of the proposed QoS control mechanisms within a single network and their mutual compatibility in an end-to-end scenario. Furthermore, a *call invocation level (CIL)* simulator has been devised for assessing the performance of signaling for call handling. Finally, a *provisioning level (PL)* simulator has been devised to evaluate the behavior of routing, e.g. the time to converge after a route change or loss.

3.1 Packet Transmission Level Simulator

The purpose of the PTL simulator is to assess end-to-end packet-related performance metrics as defined in [20] and [21]. The PTL simulation model is composed of two *access networks* connected through a *core network*, as depicted in Figure 2 [2]. The access networks taken into consideration are WiFi, Ethernet and xDSL, the most widespread at the time of writing. The core network is a generic IP DiffServ-based network. Access networks are connected to the core network by means of a single link between border nodes. In the general case, a set of flows is established between nodes in the different access networks, representing *foreground* traffic. Furthermore, *background* traffic is setup between pairs of nodes (e.g., between edge nodes in the core network). The latter is meant to simulate the behavior of (possibly many) aggregated traffic sources sharing resources (e.g., link bandwidth) with foreground traffic. Decoupling traffic into foreground and background traffic makes simulations faster and more scalable, without losing in accuracy. Background traffic is in fact generated according to well-established network-specific models taken from the literature. As for foreground traffic, accurate models of some popular applications have been devised, including VoIP, videoconference, and video streaming.
In the PTL simulations, the number of established flows (both foreground and background) is kept fixed, and flow setup/teardown, with the associated signaling and resource reservation mechanisms, is not simulated (those are in fact covered by CIL simulation). This allows us to infer steady-state forwarding behavior for the various traffic classes, so as to compute the admission control limits in various scenarios and to assess the effectiveness of the QoS control mechanisms. The PTL simulator has been implemented under the ns2 simulation framework [14]. Specifically, modules have been added to ns2 for access networks which were not available (such as xDSL), for generating traffic and taking performance measures and for all the QoS control mechanisms devised within EuQoS. It is publicly available as a standalone package at [17].

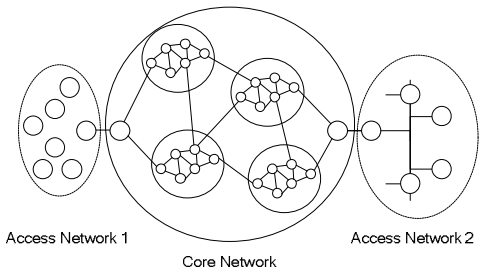

Figure 2 - PTL simulation model

3.2 Call Invocation Level Simulators

The Call Invocation Level (CIL) simulators have been developed to evaluate call signaling and call setup delay. In EuQoS the call setup process touches many components and protocols, all of which were modeled. As mentioned above, we distinguish inter-application, horizontal and vertical signaling. All signaling messages are transported using the Signaling CoS. For evaluating impact of all signaling system elements on setup delay, taking into account worst-case signaling traffic on each system level, we decided to develop three simulation models and consequently three simulators. The first one, SIM-EUQOS-SIP [19], evaluates the impact of the inter-application signaling, mainly focusing on the server processing time of signaling messages; the second one, SIM-EUQOS-TI/TD, evaluates the processing of the call setup messages at the RMs and RAs, and the last one, SIM-EUQOS-SIG [18], evaluates the performance of the Signaling CoS defined at the Network layer. Their simulation models are different, although not entirely disjoint. For instance, for the first two we do not need to simulate the whole state machine of the signaling protocols

(since the focus is on the processing times at the servers, as will be clarified in Section 4). The state machine of EQ-NSIS and EQ-SIP are instead required in the third simulator, in which packets actually flow in the network.

3.3 Provisioning Level Simulator

The Provisioning Level Simulator was devised for evaluating the performance of the EQ-BGP protocol [3], in terms of convergence speed after a route advertisement or withdrawal, and traffic overhead due to the exchange of EQ-BGP messages. The simulation model - besides all the mechanisms of BGP-4 - incorporates all the QoS-specific protocol enhancements. Specifically, QoS_NLRI attributes containing the information about type of e2e CoS, a QoS assembling function which calculates the QoS values for outgoing updates based on those of the received updates, a QoS-aware decision algorithm for comparing alternative paths for the same destination, and support for separate per-CoS routing tables. The evaluation was carried out in the four representative network topologies: a ring, a full mesh, a bi-clique, and an Internet-like topology with 29 ASs. The obtained results confirmed that the EQ-BGP protocol produces stable routing, with a similar convergence speed and message overhead as BGP-4. More details on these performance evaluation studies are presented in [3] and [4].

3.4 Interaction with the EuQoS System

The simulation activity has been carried out in close contact with the design and the development of the EuQoS system. Thus, information has been exchanged between the simulation and architecture work packages, in both directions. On one side, this has helped to validate the simulation models and to construct simulation scenarios based on realistic data obtained from commercial products as network devices. For instance, simulations related to the inter-application signaling (described in Section 4.3) include processing times which were actually measured in the EuQoS testbed.

On the other side, the simulation activity has identified potential performance problems, e.g. prototype implementation of EQ-SIP server. Based on simulation studies, we identified scalability problems, which led to reconsidering some aspects of the software architecture after the first phase of the project. For instance, based on the simulation of the Application layer, we identified the software modules causing the highest processing time, which have been revised and optimized. Based on Signaling CoS simulation studies and separate EQ-NSIS protocol evaluation, we recommended to use UDP (instead of TCP) for carrying both EQ-NSIS and EQ-SIP messages. Moreover, to provision Signaling CoS for the worst case traffic assumption considering the system without packet losses. The system with packet losses requires more bandwidth to keep setup delay requirements since the lost packets require re-transmissions [12].

This input was also exploited by the development work package to design the final EuQoS system and to prepare effective trial procedures in the pan-European EuQoS testbed.

4. PERFORMANCE EVALUATION OF THE EUQOS SIGNALING SUBSYSTEM

In this section, we describe in more detail how we modeled, simulated and evaluated the EuQoS signaling subsystem.

The EuQoS signaling subsystem is compliant with the ITU-T recommendations for IP QoS networks [22] and resource and admission control functions [23]. The typical multi-domain call setup scenario in EuQoS is shown in Figure 3: a caller client application submits its call setup request, together with its QoS requirements (basically the set of media types and the set of codecs required for each media) to an application server (A-SSN, Application Signaling and Service Negotiation), using EQ-SIP. After authenticating the caller, the application server locates and contacts its peer in the destination domain, which in turn contacts the callee. As soon as the two endpoints agree on the QoS settings for the call, the resource reservation process is triggered, which involves horizontal and possibly vertical signaling, as well as processing in the involved RMs and RAs. Once the resources have been reserved, the call setup continues until the caller application is given the final confirmation.

The three potential bottlenecks for the above mentioned process are i) the resource reservation process, ii) the transport of signaling messages through the network, and iii) the application server performance. As the rate of calls is increased, in fact, all the three subsystems may become congested. However, a call setup does have delay constraints: users would not be willing to wait (say) 30s for a call setup. As for what setup delay is tolerable, lacking references specifically related to large-scale multiservice IP networks, we used those available for PSTN networks. The ITU requirements referring to international PSTN calls (which appear to be more representative for a multi-domain scenario) [15], report that the acceptable setup delay should be not greater than 11s for 95% of calls under normal load conditions. The same requirement has been specified for Internet telephony in [24]. Building on this requirement, we decompose the signaling system into its three possible bottleneck, we set a target value for the contribution to the setup delay of each bottleneck, and we assemble the simulation results accordingly. More specifically, we fix at t_1=3.5s, t_2=3s, and t_3=4.5s the threshold for the 95th percentile of the delay introduced by the resource reservation, the transport of signaling messages, and the application signaling respectively. These target values were obtained as an input from the EuQoS testbed and software development process. Note that assuming that the 95th percentile of the setup delay introduced by each component stays below the target is a sufficient condition for the 95th percentile of the total setup delay to be within 11 seconds.

In the remainder of this section, we report the simulative modeling and evaluation for each single part of the signaling subsystem. The goal of the evaluation is to estimate the call setup time as a function of the rate of call arrivals, so as to provide guidelines for dimensioning the signaling subsystem for a given expected load. Throughout this section, we refer to the setup of a successful call in the *loose* resource reservation model, in which transit domains may be involved in processing signaling messages. As already anticipated, the setup delay in this case is expected to be larger than for a call routed using the *hard* model.

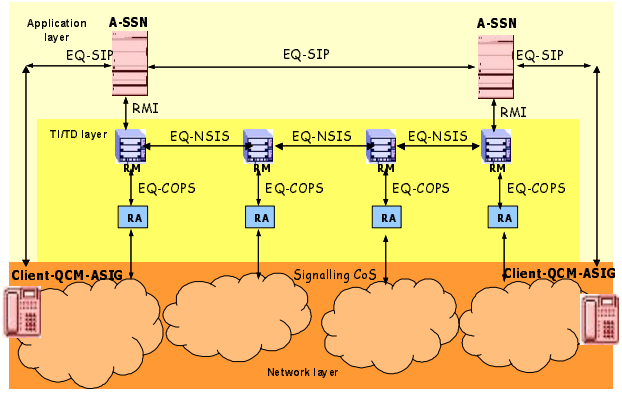

Figure 3 – A call setup in EuQoS – loose model

4.1 Resource Reservation Process

Figure 4 shows the horizontal (i.e., between RMs) and vertical (i.e., between an RM and an RA) signaling exchange for the successful setup of a call traversing two domains. RM1 receives a *reserve Commit* request from the A-SSN server. Next, it checks if an end-to-end path with the requested QoS exists (by looking at its EQ-BGP routing information base). Then it asks the CAC module in RA1 if enough resources exist to handle the new connection. If so, the requested resources are *reserved* (i.e., booked) and RA1 sends an OK to RM1. RM1 then forwards the QoS request to its peer RM2 and sends a request to RA2 to actually allocate the reserved resources in the associated access network equipment. RA1 does so by configuring the network equipment with packet marking, classifying, policing or shaping parameters, depending on the QoS request. When all these actions are successful, RA1 sends a confirmation to RM1. Finally, when RM1 receives the confirmation from both RA1 and RM2, it replies to A-SSN that the new call can be admitted. As we can see in Figure 4 the call handling scenario in RM2 and RA2 (i.e., in the egress domain) is the same as in RM1 and RA1 (ingress domain).

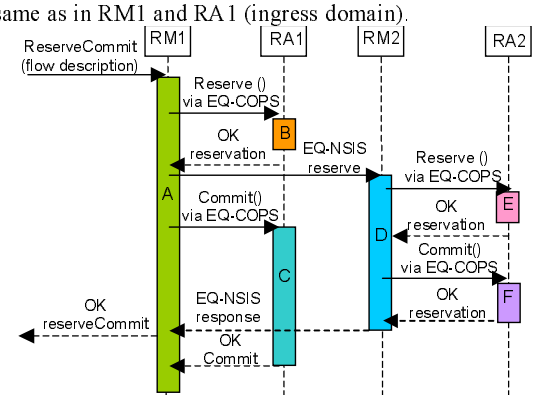

Figure 4 – Call scenario for two ASs, one direction

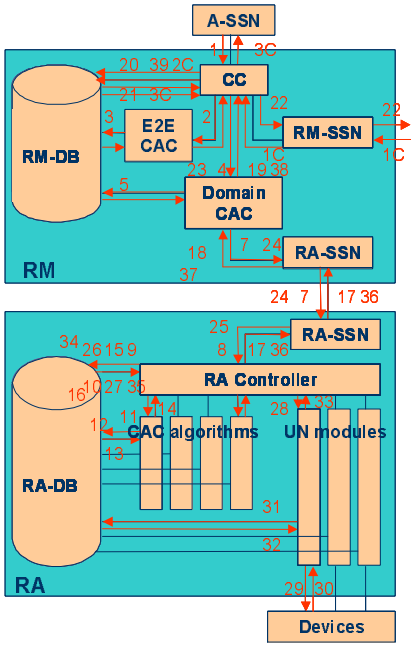

Figure 5 - RM and RA architecture and call handling scenario for ingress domain calls and confirmations, DB-Data Base, UN-Underlining Module, SSN-Signaling and Negotiation, C-confirmation task

A call may be routed along a path traversing more than two domains. In this case, transit domains are involved in the signaling, since resources have to be reserved in each domain. However, only the ingress domain RM has to check whether an end-to-end path with feasible QoS exists. For bi-directional calls, the call handling process is performed simultaneously during the setup phase. So, in order to calculate the setup delay for a successful scenario, it is sufficient to simulate it in one direction.

The detailed architecture of the RM and RA, together with the vertical signaling message exchange required for handling an ingress call, are presented in Figure 5. In the RM, the *Call Controller* (*CC*) receives QoS requests from the application level and controls all the CAC sub-modules. The *E2E CAC* is in charge of checking if an end to end path with feasible QoS characteristics actually exists. Then the CC asks the *Domain CAC* to check the operator policies and to look for an intra-domain path and to have the RA reserve the resources. The Domain CAC has two sub-modules, one for the intra-domain part and the other for the inter-domain link. Finally, the *RA Controller* receives the QoS request via the signaling module interconnecting the RM and RA (RA-SSN, *Signaling and Service Negotiation*). The RA Controller runs a different CAC algorithm for each CoS. The RA CAC algorithms check the amount of available resources – by querying the *RA Data Base* (*RA-DB*) – against the new request, and decide whether a new call is accepted or not. If the call is accepted, the RA Controller asks the appropriate access network UN module to configure its network devices for handling the new call. Obviously, different mechanisms must be configured depending on the type of network device (which can be an IP router, a LAN/Ethernet switch, a WiFi Access Point, etc...). Note that UN modules are involved in the call handling process only in *access* domains. For *transit* domains resources are allocated based only on a decision of the RA CAC algorithm, and there is no dynamic configuration of resources in inter-domain network devices.

4.1.1 Simulation model

In our simulation studies we model the RM and RA architecture of Figure 5 as a queueing network, shown in Figure 6. The simulation model is composed of infinite queues and servers chains. We distinguish eight servers: RM-CC (1), RM-DB (2), RM-RA link (3), RM-RM link (4), RA-C (5), RA-DB (6), RM-RA link (7), Devices (8), each one associated to an infinite-length FIFO queue.

We assume that new calls and confirmations of call acceptance arrive to RM independently according to a Poissonian process with given mean arrival rates. Service process times are deterministic, with the following values:

- RM-CC, RA-CC: $t_{RM\text{-}CC} = t_{RA\text{-}CC} = 100\mu s$;
- database access in the RM and RA: $t_{RM\text{-}DB} = t_{RA\text{-}DB} = 1ms$;
- RM-RA link, RA-RM link, RM-RM link: $t_{RM\text{-}RA} = t_{RA\text{-}RM} = t_{RM\text{-}RM} = 1ms$
- access to the device: $t_{DEV} = 600ms$ (WiFi) and $t_{DEV} = 100ms$ (IP).

The processing times for WiFi and IP devices are based on off-the-shelf network equipment (LINKSYS WiFi Access Point WRT54G, CISCO 1841 router or Linux router kernel 2.6.18 IMQ). For the others, we tried to infer the expected processing time based on today's high speed computers.

We consider three types of event in the simulation model: *new call arrival*, *new confirmation arrival*, *departure of a task from a server*. The arrival or confirmation invokes a sequence of tasks. The departure of a task from one server determines its arrival to another. Each event is represented by the quadruple <*Sequence ID, Task ID, Server ID, Processing time*>. The Sequence ID denotes whether the event is a *call* or a *confirmation* arrival. Furthermore, we need to distinguish call events based on whether

they are being processed at an *ingress*, *transit* or *egress* domains. Due to space limitations, we present only events related to an ingress domain, the latter being however the most significant case. Each type of event invokes a different sequence of tasks. The maximum number of tasks is performed when processing calls at the ingress, since the end-to-end CAC is checked. Fewer tasks are required when processing confirmations. The mean call arrival rates are $\lambda_{ingress}$, λ_{egress}, $\lambda_{transit}$, and the mean arrival rate of confirmations for ingress and transit domains are $\lambda_{conf\text{-}ingress}$ and $\lambda_{conf\text{-}transit}$.

Note that in this simulation model we do not model as the state machine of EQ-NSIS or EQ-COPS (see Figure 4). The detailed message exchange of EQ-NSIS is modeled in simulation studies of Signaling CoS (see section 4.2). For EQ-COPS protocol, we model its performances by $t_{RM\text{-}RA}$, $t_{RA\text{-}RM}$ times only. The above simulation model has been coded in a discrete event simulator written in C++.

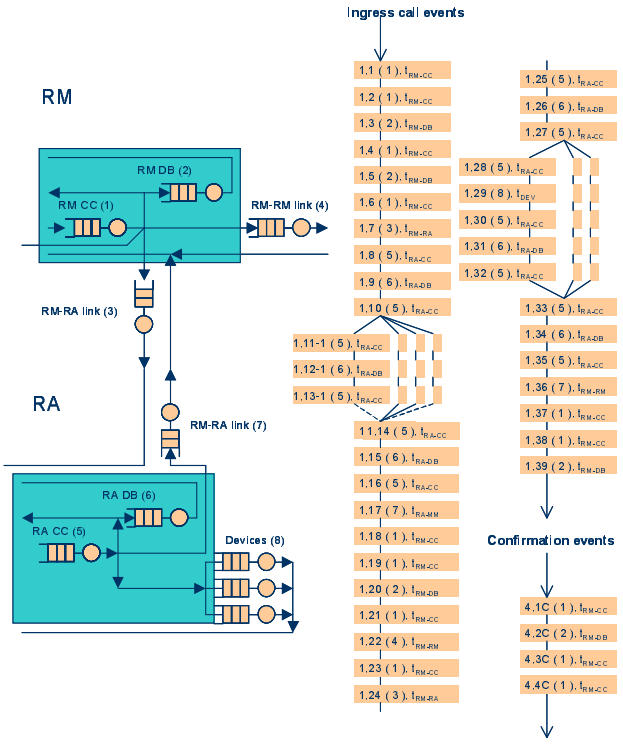

Figure 6 – RM and RA for the ingress domain - simulation model

4.1.2 Simulation results

We begin our studies with single domain performance evaluation. In particular, we simulate two scenarios. First, we assume that the single access domain handles both *ingress* and *egress* calls and confirmations. We simulate WiFi and IP access domains in isolation. In addition, we assume that $\lambda_{ingress}=\lambda_{egress}=\lambda_{conf}$. For each test we vary the call and confirmation arrival rates up to the 95% resource utilization of the RM-RA elements (in this case the bottleneck element is the device itself). In the next step, we analyze performances of single transit domain assuming that RM and RA handle only *transit* calls and confirmations (in this case the bottlenecks are RM-DB and RA-DB, it means access to the databases). Again, $\lambda_{transit}=\lambda_{conf}$.

Finally, based on the results obtained for single ingress/egress and transit domains, we estimate the percentiles of the setup delay for a multidomain call scenario as shown in Figure 7.

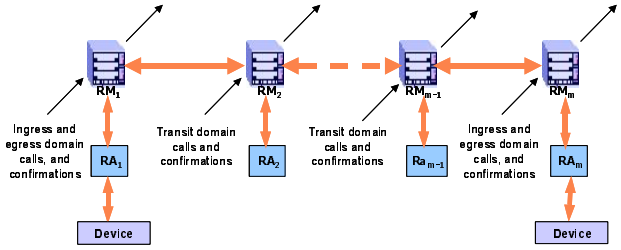

Figure 7 – Multidomain call handling scenario with m domains

We compute the 95th percentile of the setup delay for m=2, 10, and 20 domains. For this purpose we distinguish among the following sequences of tasks performed by the ingress domain RM and RA:

- $S1_{ingress}$: resource reservation tasks in RM1<->RA1 performed before the request is forwarded to the next RM;
- $S2_{ingress}$: tasks in RM1<->RA1<->DEV1 resource allocation tasks;
- $S3_{ingress}$: RM tasks invoked by confirmation arrival from neighbouring domain;

Thus, we define the target delay $T_{ti\text{-}td}$ for this scenario as follows:

$$T_{ti-td} = t^{S1}_{ingress} + \max\left\{t^{S2}_{ingress}, \left(n \cdot t_{transit} + t_{egress} + t^{S3}_{ingress}\right)\right\} \quad (1)$$

where $t^{X}_{ingress}$ is the 95th percentile of the delay introduced by sequence X, t_{egress} is the 95th percentile of the delay in the egress domain, n is the number of transit domains, $t_{transit}$ is the 95th percentile of the delay introduced by a transit domain.

Based on (1) we can compare the percentiles of setup delay for two resource reservation models, hard and loose, as described in section 2.4. Moreover, in our approach we calculate results for only two WiFi access domains or two IP access network domains. Based on the detailed simulation results obtained for single domain scenarios (for ingress/egress and transit) we can derive:

$$t^{S2}_{ingress} < t^{S3}_{ingress} + t_{egress} + n \cdot t_{transit} \quad (2)$$

By applying (1) and (2)the target setup delay $T_{ti\text{-}td}$ is calculated according to (3).

$$T_{ti-td} = t^{S1}_{ingress} + t^{S3}_{ingress} + t_{egress} + n \cdot t_{transit} \quad (3)$$

The 95th percentiles of setup delay for 2, 10 and 20 domains are presented in Figure 8-Figure 10. In particular, Figure 8 shows the results for two WiFi and IP access domains. For both scenarios the curves show a significant difference between the delays obtained in the two technologies. With reference to the 3.5s target for the 95th percentile of the setup delay, we observe that, in WiFi, $\lambda_{ingress}$, λ_{egress}, and λ_{conf} should not exceed about 0.6 calls/s. In an IP domain for the limit is about 4.75 calls/s. For scenarios with 10 and 20 domains we assume that two domains are access domains while the others are transit domains. Figure 9 and Figure 10 show the 95th percentile of the setup delays for WiFi and IP access domains, respectively. The transit call rates are 233 or 300 call/sec in each transit domain. The results show that the number of transit domains has a smaller impact on the setup delay than the transit call arrival rates. Moreover, the acceptable call arrival rates are larger for IP access domains than for WiFi.

Finally, based on the above results we can draw some conclusions for comparing call handling performances for loose and hard models. As far as resource reservation is concerned, the call setup scenario for a call traversing $N \geq 2$ domains in the hard model is the same as in the setup of a call in the loose model with just 2 access domains. Within the limits of these experiment, and with the processing times used, the resource reservation delay in the hard model is not so different from the one in the loose model.

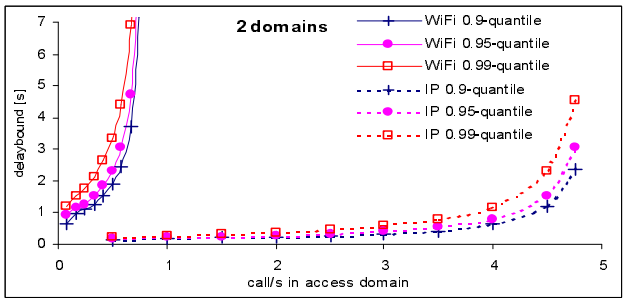

Figure 8 - Percentiles of $T_{ti\text{-}td}$ setup delay for 2 WiFi or 2 IP domains vs. $\lambda_{ingress}$ ($\lambda_{ingress}=\lambda_{egress}=\lambda_{conf}$)

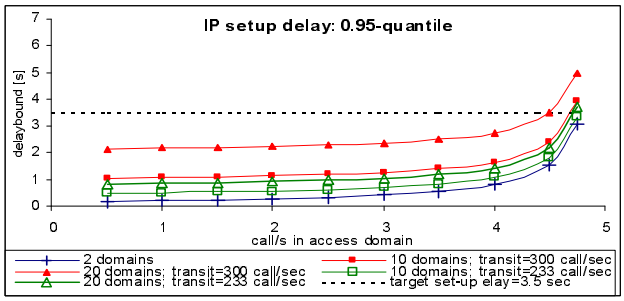

Figure 9 - 95th percentile of setup delay for 2 IP access domains vs. $\lambda_{ingress}$ ($\lambda_{ingress}=\lambda_{egress}$ and $\lambda_{ingress}=\lambda_{conf}$)

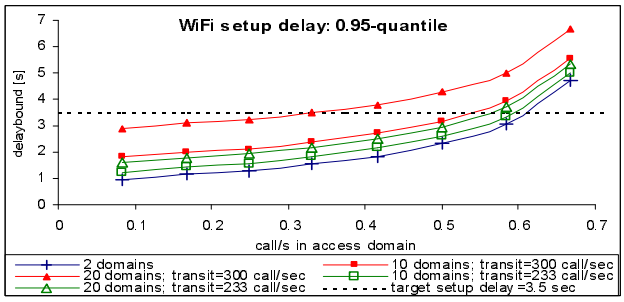

Figure 10 - 95th percentile of setup delay for 2 WiFi access domains vs. $\lambda_{ingress}$ ($\lambda_{ingress}=\lambda_{egress}$ and $\lambda_{ingress}=\lambda_{conf}$)

4.2 Signaling Class of Service

Within the network, signaling traffic is carried by a dedicated Signaling Class of Service (abbr. S-CoS). The delay introduced by the transit of signaling packets makes up for a part of the setup delay, which will be referred as "*transfer packets setup delay*" or briefly "*tp setup delay*".

Unlike the traffic from the other CoSs, signaling is not subject to admission control, as explained in [9]. Therefore, a correct resource provisioning is crucial in order to ensure tp setup delay constrains. In [12], we proposed an analytical method to provision bandwidth for the S-CoS, which computes the necessary bandwidth to ensure a 95th percentile tp setup delay. Such method is based on a worst-case approach: the traffic generated by each procedure is considered "in isolation", and resources are computed separately. Then, the overall bandwidth for the S-CoS carrying the aggregated traffic related to different procedures is computed as the sum of the resources calculated separately. This neglects the multiplexing gain and overestimates the amount of required resources.

In this section, we consider the multiplexing gain by simulating the setup procedure at the packet level, thus assessing the overestimation introduced by the analytical provisioning method.

4.2.1 Simulation model

Figure 11 shows the simulation model. We assume that the S-CoS is studied from the point of view of the signaling load generated by a number of users in *Access Network* 1 (callers) who want to establish connections with corresponding users in *Access Network* 3 (callees), through a *Transit Network 2*. For the purpose of dimensioning network resources, after calculating the necessary resources for one setup procedure, we consider N identical procedures from the users in Access Network 1. The capacity in S-CoS$_2$, will be N times the necessary capacity for one single setup procedure. The S-CoSs are provisioned in each domain separately, thus we refer to them by using the domain subscript.

As we can see in Figure 11, a single stage in the transit network supporting S-CoS is considered (say, S-CoS$_2$). Since EQ-SIP traffic related to the setup procedure is submitted by the EQ-SIP user entity (shown inside the terminal in Figure 11), we also model the S-CoS for the links between a terminal and the EQ-SIP proxy (in Figure 11 EQ-SIP proxies are in the Servers in Access Network 1 and Access Network 3). We expect that S-CoS$_1$, S-CoS$_2$ and S-CoS$_3$ will require a different provisioning, since the volume of signaling traffic submitted to them is different. For the same reason, the network resources for each direction of one S-CoS will be different (see Figure 11).

Each segment of the S-CoS is modeled as two separate queues, one for each direction, with reserved resources for the links (C_1-C_6) and for the buffers (B_1-B_6). In the simulations we do not consider the propagation delay on the links. The packet transmission time on the links between servers and routers is negligible, as those link rates are considerably larger than the rates of the other links.

We prepare the simulations as follows: we provision the S-CoS for N parallel setup procedures according to the analytical model. Table 1 presents the computed link resources for the network shown in Figure 11, assuming a target 95th percentile of the tp setup delay equal to 3s. The buffer sizes are large enough to ensure no losses. Then, we actually start $M \geq N$ simultaneous setup procedures: we compute the arrival rate r of setup procedures such that the system, on average, has M simultaneous setup procedures; assuming that one setup procedure lasts 3s, i.e. tp setup delay= 3s, then the rate r equals M/3 setup procedures/s. We consider that the number of users is large enough that a Poisson arrival process is a feasible model [13]. The transport protocol used is UDP.

Table 1 - Calculated link capacities in the S-CoS according to the provisioning method

Network	**Access Network 1**		**Transit Network 2**		**Access Network 3**	
Link	**C_1**	**C_2**	**C_3**	**C_4**	**C_5**	**C_6**
Capacity [kbps]	1 x 46.97	1 x 30.36	N x 35.87	N x 30.36	1 x 35.87	1 x 40.32

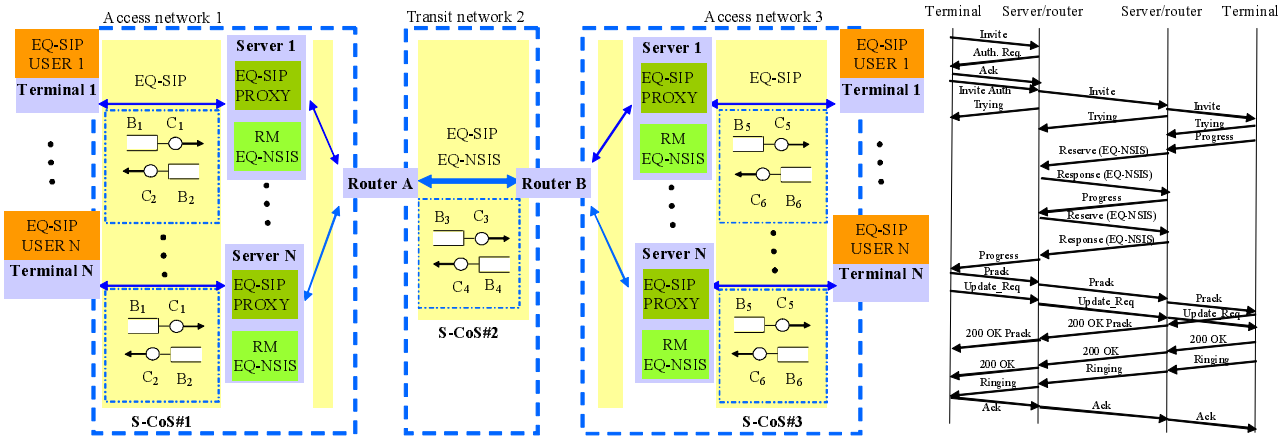

Figure 11 - Simulation model of the S-CoS

We simulate this process for various values of M until we reach the point at which the 95th delay percentile is equal to the target value of 3s. We repeat this simulation methodology for N=10, 20, 50, 100, 200, 500 and 1000 provisioned parallel setup procedures.

In Figure 12, we present the results of the maximum number of simultaneous setup procedures (M_{max}) for different values of N provisioned parallel setup procedures. We observe that the relation between them increases faster than linearly. This is because packets from different setup procedures order themselves after the first queueing in the Access network 1 and, afterwards, transit faster through the high capacity links. Thus, packets will only be delayed when they find packets from *new* setup procedures in the queues and, therefore, more setup procedures are allowed in the system. This effect increases with the link capacities and, thus, with N.

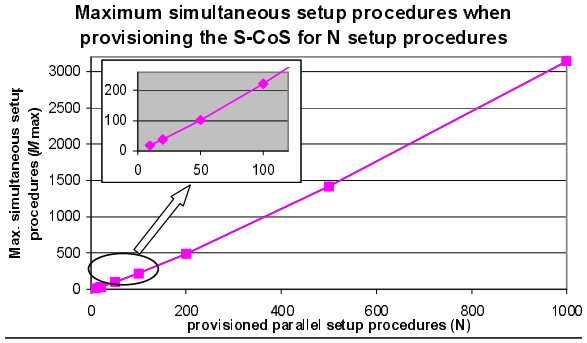

Figure 12 - Relationship between the number of simultaneous procedures and the provisioned resources

4.3 Inter-application Signaling

The simulation model for the inter-application signaling includes the different equipment involved in application signaling: a EuQoS client, a EuQoS application server and an authentication server (*SAAA, Service Authentication, Authorization and Accounting*), each one with its state machine. The latter include a round-robin task scheduler, which assigns work cycles to the processes/threads on the machine running the module. The time slice (which is, however, configurable) is set to 50 ms for the experiments.

Each module (client, server, SAAA server) has a similar behavior: messages received from the network interface, are sent to the appropriate process. Each process takes a given time to process a packet, and its output can be either a signal to another process or a new message for the next hop in the signaling chain.

This model requires the processing times of each message arriving to a module to be known in advance. Such data was obtained from measurements performed on the EuQoS testbed using a capture tool based on *tcpdump* [16]. The tool captures the EQ-SIP messages in all the entities involved in the signaling path, which are synchronized through the NTP protocol. The processing times are further obtained by calculating the elapsed time from the reception of a message to the sending of the subsequent message. The basic processing times have been obtained in a scenario with just one EuQoS call being established, on a EuQoS server running on a Pentium III 866 MHz computer with 256 MB RAM.

Taking a deeper insight into the simulator model, Figure 13 shows the architecture of the different entities involved.

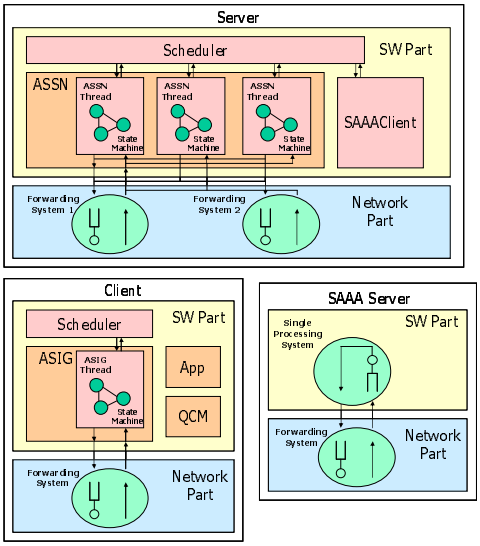

Figure 13 – Simulation model for the application components

Clients are composed of an Application (App), a Quality Control Module (QCM) and an Application Signaling (ASIG) modules. On the one hand, the processing time of the App and QCM modules have not been considered in the setup delay at the application level, as they lay at the beginning and at the end of the setup chain. On the other hand, the ASIG module has been modeled as a single-thread process implementing both the caller and callee EQ-SIP state machines. A EuQoS server is composed of an ASSN and

an SAAAClient (which communicates with the SAAA Server). The ASSN has been modeled as a multi-thread process, in which all threads share the CPU through the scheduler. Each thread implements the full state machine of an EQ-SIP proxy, making it possible to simulate a caller proxy, an intermediate or a callee one. The number of threads is configurable. If no threads are available when a call request arrives to ASSN, the call is rejected. The SAAAClient module has a synchronized interface with the ASSN, so that just one thread at a time can use it. The SAAA server has been modeled as a single processing system that receives requests and serves them at a constant rate. A delay uniformly distributed between 0 s and 0.5 s is added to each SAAA Server response to take into account processing times. Diameter is modeled as two-way exchange of messages between the SAAAClient and the SAAA Server. The resource reservation process, which is triggered by the EuQoS server after the reception of a "SIP Session Progress" message, is out of the scope of this simulator and was analyzed by the simulator described in Section 4.1. Since the purpose of this simulator is to evaluate the call setup delay, we modeled the resource reservation delay as a variable delay, uniformly distributed between 3.3 and 3.7 seconds, so as to reflect a reasonably large delay in that subsystem.

The above model was coded into ns-2 [14]. With this simulator, the following scenario has been configured in order to study the scalability of the system.

The simulation scenario is depicted in Figure 14 and consists of a number of clients trying to establishing a call with their peers through to the same EuQoS server, called *Main server* in the figure. The network links, including those with the two network aggregators, are provisioned so as to have a single bottleneck of the system, i.e. the EuQoS server. The interval between any two consecutive calls is drawn from an exponential distribution. In the scenario investigated, the system load, i.e. the average number of concurrent call attempts, is increased by decreasing the average interval between consecutive calls. Clients only try to establish a given call once, i.e. blocked requests are not re-iterated. A maximum number of 10 simultaneous calls, i.e. ASSN threads, are allowed in the EuQoS server.

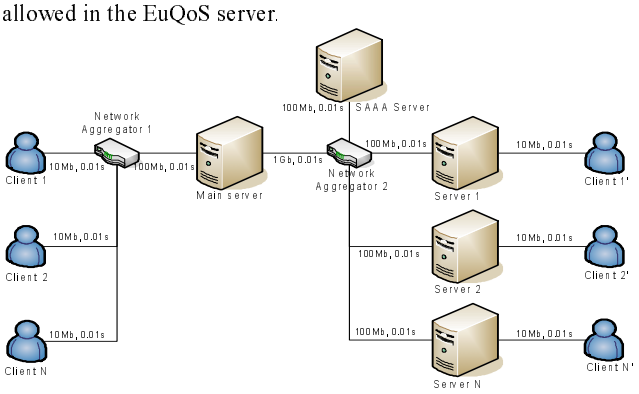

Figure 14 – Simulation scenario

We first analyze the setup delay, which is defined as the time between when a client issues a call request and when the call is established. The 95th percentile of the setup delay is shown in Figure 15 as the request rate increases from 0.001 to 1 calls/s, with different processing capabilities, labeled from CPU 1 to CPU 4. Specifically, CPU 1 refers to the results obtained by feeding the simulator with the processing times obtained with an Intel Pentium III at 866 MHz PC, and CPU *i* to the results obtained by dividing them by *i*. When the request rate is lowest, it is very unlikely that more than one request is handled at the same time by the server. Therefore, the latency is only due to the processing times. The setup delay target is 8s, since both the application signaling (4.5s target delay) and the resource reservation (3.5s target delay) have to be considered. As can be seen, CPU 1 cannot provide the required latency not even under such light load, since it is not guaranteed that 95% of the requests are served within 8 s. All the other CPUs, instead, meet this requirement at low loads. Specifically, when the system load increases the setup delay becomes greater, because the processing unit of the server is shared among concurrent call requests. With CPU 2, the maximum request rate that can be handled while satisfying the request quality of service is about 0.1 calls/s, while with CPU 3 and 4 up to 1 calls/s can be served with the 95th percentile of setup delay being smaller than or equal 8 s.

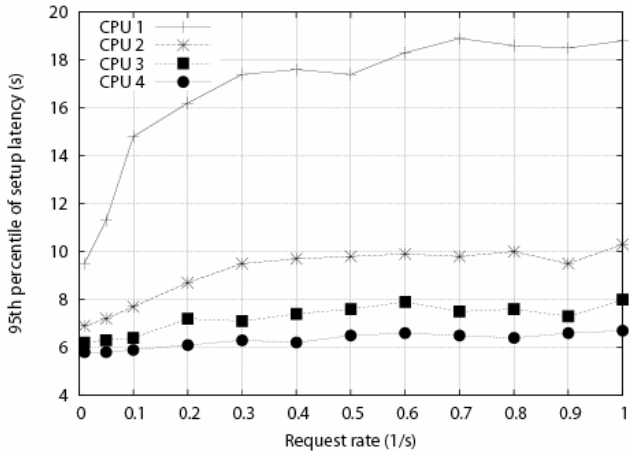

Figure 15 – Setup delay depending on the call request rate for different CPU processing capabilities

In Figure 15 all the curves become stable after a given request rate is exceeded, depending on the CPU capabilities. This is because of the increasing number of calls that are blocked by the server, due to exhausted threads for managing concurrent call request contexts. The average number of blocked calls, called *blocking probability*, is reported in Figure 16.

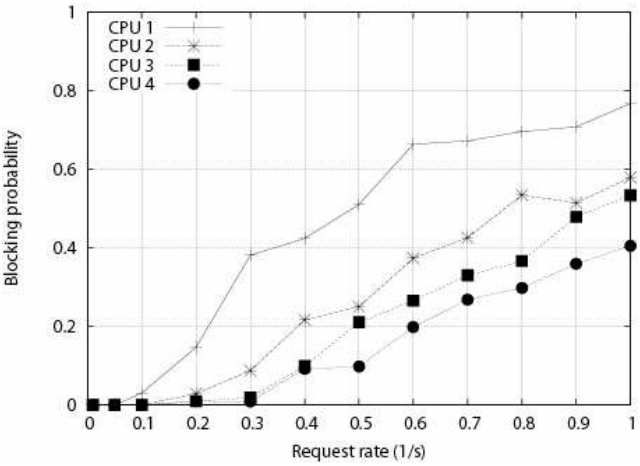

Figure 16 – Blocking probability depending on the call request rate for different CPU processing capabilities.

As expected, the lower the processing times is, the smaller the blocking probability becomes. Note that, while with CPU 4 all the request rates yield a feasible setup delay in 95% of the cases, still

a noticeable ratio of requests are blocked at high loads, i.e. with a request rate greater than 0.4 calls/s.

In summary, the simulations provide an operation point for the EuQoS server at the application level. A EuQoS server installed on a PC twice as fast as a Pentium III at 866 MHz can work with call rates of 0.1 calls/s, allowing ten simultaneous calls with negligible blocking probability. Both the setup delay and the blocking probability are affected significantly by the processing times in the EuQoS server. In a real deployment, with call request rates in the order of 8 calls/s, a tandem of CPU 4 servers with load distribution would be necessary.

5. CONCLUSIONS

In this paper we have presented the simulation activity carried out in the framework of the EuQoS project, aimed at providing support to the design of a complete multi-domain QoS system. We have presented the approach to simulation, which capitalizes on using different simulation models for events occurring at different timescales. Moreover, we have reported the simulation effort related to the modeling and evaluation of the EuQoS signaling subsystem. The latter provides useful insight on how to dimension all the components of the signaling subsystem (application servers, Resource Managers and Resource Allocators, and reserved bandwidth on the links) so as to verify setup delay constraints. The results show that the application servers are currently the main bottlenecks due to the overhead of the EQ-SIP protocol message processing. The system is able to handle about 300 call/s in transit domains. The values of setup delay obtained for 10 and 20 domains allows us to conclude that the proposed signaling system is scalable to a large scale networks. For now, we have considered commonly assumed traffic model for call arrivals as for PSTN network. The next step of our studies would be to consider traffic models at call level based on measurements for multiservice IP networks.

6. ACKNOWLEDGEMENTS

We would like to thank all the partners of the EuQoS consortium for their support and their work on developing the EuQoS system. Special thanks to all the partners involved in Work Package 2, "Validation of the EuQoS system".

7. REFERENCES

[1] The IST-EuQoS consortium, http://www.euqos.eu.

[2] C. Cicconetti, *et al.*, "Simulation Model for End-to-end QoS across Heterogeneous Networks", proc. of 3rd IPS MoMe 2005, Warsaw, 14-15 March 2005, pp. 79-89.

[3] X. Masip-Bruin, *et al.*, "The EuQoS System: A solution for QoS Routing in Heterogeneous Networks", IEEE Communications Magazine, Vol 45, Issue 2, February 2007, pp. 96-103.

[4] A. Beben, "EQ-BGP: an Efficient Inter-domain QoS Routing Protocol," Proc. 20th Int'l. Conf. Advanced Info. Networking and Apps., Vienna, Austria, Apr. 2006.

[5] L. Veltri, S. Salsano, D. Papalilo, "QoS Support for SIP Based Applications in a Diffserv Network", IEEE Softcom2003, October 7-10, 2003, Split/Dubrovnik (HR), Ancona/Venice (IT).

[6] Next Steps in Signaling (NSIS) IETF WG: http://www.ietf.org/html. charters/nsis-charter.html.

[7] L. Cordeiro *et al.*, "Hybrid on-Path Off-Path Approach for End-to End Signaling Across NSIS DOMAINS (HyPath)," IETF draft-cordeiro-nsis-hypath-00, Feb. 2006.

[8] D. Durham *et al.*, "The COPS (Common Open Policy Service) Protocol," RFC 2748, Jan. 2002.

[9] RFC 4594, Configuration Guidelines for DiffServ Service Classes, Aug. 2006.

[10] IST-EuQoS deliverable D1.1.1, "Business models and system design specification", 2005.

[11] IST-EuQoS deliverable D1.1.3, "Definition of Business, Communication and QoS models", 2005.

[12] J. Mongay Batalla and R. Janowski, "Provisioning dedicated class of service for reliable transfer of signaling traffic", 20th. Int.'l Teletraffic Congress, June 2007, Ottawa, CA, pp. 853-864.

[13] J. Brazio *et al.*, "Analysis and Design of Advanced Multiservice Networks Supporting Mobility, Multimedia, and Internetworking", COST Action 279 Final Report, pp. 29-30.

[14] The Network Simulator – ns2: http://www.isi.edu/nsnam/ns/.

[15] ITU-T Rec. E.721, "Network grade of service parameters and target values for circuit-switched services in the evolving ISDN", May 1999.

[16] http://www.tcpdump.org/ .

[17] Packet Transmission Level simulator, http://tnt.tele.pw.edu.pl/include/tools/sim-euqos-ptl.tgz .

[18] Signaling Class of Service Simulator, http://tnt.tele.pw.edu.pl/include/tools/sim-euqos-sig.tgz .

[19] EQ-SIP Call Level simulator, http://tnt.tele.pw.edu.pl/include/tools/sim-euqos-sip.tgz .

[20] ITU-T Rec. Y.1540, "IP Packet Transfer and Availability Performance Parameters", 2002.

[21] ITU-T Rec. Y.1541, "Network Performance Objectives for IP-based Services", 2002.

[22] ITU-T TR Q-Series Supplement 51 (12/04), "Signalling Requirements for IP QoS".

[23] ITU-T Rec. Y.2111, "Resource and Admission Control Functions in Next Generation Networks", 2006.

[24] ITU-T Rec. E.671, "Network Post-selection delay in PSTN/ISDN networks using Internet telephony for a portion of the connection", 2000.

[25] RFC 2216, "Network Element Service Specification Template", September 1997.

[26] RFC 3261, "SIP: Session Initiation Protocol", June 2002.

[27] RFC 4566, "SDP: Session Description Protocol", July 2006.

[28] O. Dugeon, *et al.*, "End to End Quality of Service over Heterogeneous Networks (EuQoS)", NetCon'05, Lannion, France, 14-18 Nov. 2005.