Project Number:	**IST-1999-10077**
Project Title:	**AQUILA** **Adaptive Resource Control for QoS Using an IP-based Layered Architecture**
Deliverable Type:	**PU – public**

Deliverable Number:	**IST-1999-10077-WP1.3-COR-1303-PU-O/b0**
Contractual Date of Delivery to the CEC:	
Actual Date of Delivery to the CEC:	**March 31, 2003**
Title of Deliverable:	**Traffic handling studies**
Workpackage contributing to the Deliverable:	**WP 1.3**
Nature of the Deliverable:	**O – Other**
Editor:	**Stefano Salsano (COR)**
Author(s):	**Christof Brandauer (SPU), Wojciech Burakowski (WUT), Marek Dabrowski (WUT), Peter Dorfinger (SPU), Thomas Engel (SAG), Simeone Mastropietro (COR), Eugenia Nikolouzou (NTU), Fabio Ricciato (COR), Stefano Salsano (COR), Petros Sampatakos (NTU), Halina Tarasiuk (WUT), Giacomo Troiano (COR), Evi Tsolakou (NTU)**

Abstract:	This deliverable collects the studies about the traffic handling mechanisms in the AQUILA architecture. Traffic handling in AQUILA is composed of four related mechanisms operating at different time scales: provisioning (days to weeks), resource pools (hours), admission control (seconds to minutes), traffic control (milliseconds).

Keyword List:	AQUILA, QoS, Internet, Traffic control, Admission Control, MBAC, Provisioning

Executive Summary

This deliverable collects the studies about the traffic handling mechanisms in the AQUILA architecture. Some of these studies were also helpful to improve or correct some specifications provided in D1302, as will be mentioned in the document.

Section 1 deals with the evaluation of distributed admission control schemes in Diffserv network. A distributed admission control scheme performs admission control only at the border of the network (ingress and/or egress), with no exact knowledge of the interior of the network. The approach followed by AQUILA and some variants are examined and compared with a reference scenario where hop-by-hop admission control is performed. The performance parameters are the admission control loss and the loss in the network due to wrong admission control decisions.

Section 2 deals with the evaluation of the mechanisms used in AQUILA to dynamically distribute resources: the resource pools in the intra-domain and the BGRP quiet grafting in the inter-domain. Both mechanisms have the goal to provide a scalable answer to the need of dynamic resource management. The performance of this mechanism in term of signalling processing reduction and network resource utilization are provided.

Section 3 deals with the analysis of packet level performance of the traffic classes proposed by AQUILA in a larger scale network with respect to the AQUILA test bed. Packet level performance measure like end-to-end delay are considered.

Section 4 deals with the analysis of BGRP Quiet Grafting mechanism in terms of reduction of the path length of BGRP messages and consequently of overall number of signalling messages.

Section 5 focused on the topic of rate assurance for TCP flows. A traffic conditioning model is developed with the goal to provide an assured goodput to TCP flows. The model is analysed by means of extensive simulations.

Section 6 analyses the performance of the traffic class TCL 3 specified in AQUILA, focusing in particular on the setting of the WRED (Weighted Random Early Discard) parameters. The traffic class TCL 3 is used for long lived, greedy TCP flow (AQUILA PMM network service).

Section 7 considers the performances of the traffic class TCL 4. The traffic class 4 is used for short lived TCP connection and for “bursty” TCP connections (AQUILA PMC network service).

Section 8 performs a comparison between declaration based and measurement based admission control (resp. DBAC and MBAC) for IP Diffserv traffic. The comparison is focused on the support of the AQUILA Premium VBR network service (i.e. AQUILA traffic class TCL 2). Scenarios with DBAC only, MBAC only and the coexistence of DBAC and MBAC are analysed by means of simulation.

Section 9 deals with a modified admission control rule for TCL 3 traffic. It tries to better fulfil the QoS requirements of the flows by taking into account the round trip time of the path and by proper setting of the token bucket parameters.

Section 10 deals with dual-homing solutions that are used at different levels to increase the resilience of IP networks. In particular the impact of these solutions on the AQUILA architecture and their interaction with RCL mechanisms is considered.

Section 11 presents a scalability analysis of the AQUILA architecture. Starting from data gathered in the AQUILA RCL performance testbed, a performance model is derived and it is used to evaluate the scalability of the system.

Table of content

1 Evaluation of distributed Admission Control schemes

The aim of the Admission Control in AQUILA is to limit the total amount of traffic that is injected in the network for each TCL. At every Reservation Request, the admission decision is taken by the ACA responsible of the ingress ER and eventually the egress ER, depending on some algorithms and on locally stored limits called *AC Rate Limits*. If the flow type is a p2a reservation style, the AC algorithm does not look at the final destination and the AC algorithms run on the ingress ER only. If the flow type is a p2p reservation style, the AC algorithm runs both at the ingress and egress ERs. The definition of the AC Rate Limits is a critical point. In AQUILA they represent the maximum amount of traffic that can be injected into an ingress ER or extracted from an egress ER regardless of the flows exit / entry points. The AC Rate Limits provide to the edge nodes synthetic information about the available resources in the network. This way the edge nodes do not need to maintain any explicit view of the network topology and state, which preserves simplicity and scalability. On the other hand, the cost of such a simple scheme is paid in terms of reduced *effectiveness* and *consistency* of the AC. We consider an AC to be poorly *effective* when it rejects traffic that could be admitted, and poorly *consistent* when it admits traffic that should be rejected.

In this contribution we want to evaluate the effectiveness and the consistency of the AQUILA methods. This is accomplished by a performance comparison with the classical link-by-link AC (referred to as Method 4 through the paper), taken as a reference. In fact in the link-by-link AC method the *effectiveness* and *consistency* are maximised, as the admission decision is taken by considering the actual current value of link utilisation for each link, i.e. in a regimen of *full information*. On the other hand, in the simplified approaches adopted in AQUILA the admission decision is taken on the basis of partial *information*, summarised in the local AC Rate Limit. This leads to a major simplification in the control plane (less information to be collected maintained, distributed, etc.), but coming at some price in terms of *effectiveness* and *consistency*. The goal of the work reported in this contribution was to analyse the performances of such distributed Admission Control approach and to evaluate the price that one has to pay for the simplicity of having edge-only aggregated admission control.

Moreover, we also compare the AQUILA methods with an alternative AC approach based on per ingress/egress pair. In this alternative scheme (referred to as Method 3) the admission decision is performed only at the ingress ER, but the flow egress point is taken into account. This is supported by redefining the AC Rate Limits on a per ingress/egress pair basis. This method would be particularly interesting in a network supporting MPLS, as the AC Rate limits can be straightforwardly mapped to LSPs.

As the distributed AC Rate Limits are computed in the provisioning phase, the choice of an AC method has impact on the provisioning scheme itself[1].

[1] Note that no provisioning scheme is implied by Method 4, where no AC Rate Limits are used

1.1 Overview of methods

According to the adopted admission control strategy, AC Rate Limits assume different meanings. Now, referring to the notations defined in [1], we illustrate from an architectural point of view different possible admission control strategies based on distributed *AC Rate Limits.* Further details are given in the appendices.

1.1.1 Method 1 – limiting ingress traffic at entry point

In this first method the admission control is performed only at the ingress ER, independently the egress point. For each TCL each ER is assigned a maximum amount of bandwidth, called AC Rate Limit, which represents the total amount of traffic that is allowed to enter in the network according to the desired QoS, independently on the egress point and the path of the traffic flows. We denote by r_i the AC Rate Limit for the generic *i*-th ER. Thus the Resource Control Layer (RCL) provides N limits, where N is the number of ERs.

In this approach the Admission Control Agent (ACA) does not take into account the egress point. When a new flow asks for admittance to ER *i*, the responsible ACA for that ER will simply check that the total amount of already admitted traffic plus the incoming flow does not exceed r_i. This scheme is shown in Figure 1. This method is used in AQUILA for the p2a reservations (currently supported by the PMC NS only).

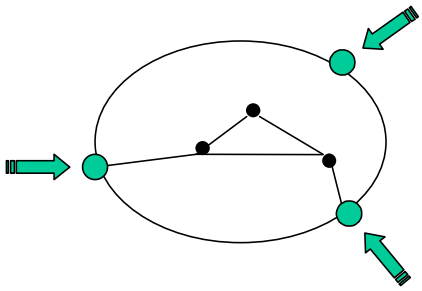


Figure 1 - Method 1 scheme

1.1.2 Method 2 – jointly limiting ingress/egress traffic at entry/egress points

In this second method the admission control is performed both at the ingress ER and at the egress ER. For each TCL each ER is assigned two amounts of bandwidth, called ingress AC Rate Limit and egress ingress AC Rate Limit. The ingress AC Rate Limit represents the total amount of traffic that the ER is allowed to inject in the network for the relevant TCL, independently on the egress point and the path of the traffic flows. The egress AC Rate Limit represents the total amount of traffic that ER is allowed to extract out of the network according to the desired QoS, regardless of the egress point and the path of the traffic flows. We denote by r_i the ingress AC Rate Limit and by e_i the egress AC Rate Limit for the generic *i*-th ER. Thus the RCL provides 2N limits, where N is the number of ERs.

When a new flow asks for admittance to ER *i*, the responsible ACA for that ER will simply check that the total amount of already admitted traffic plus the incoming flow does not exceed r_i. In an analogue way the ACA responsible for the ER *j* relevant to the egress point of the requesting flow check that the total amount of already admitted traffic plus the incoming flow does not exceed e_j. This scheme is shown in Figure 2. This method is used in AQUILA for the p2p reservations.

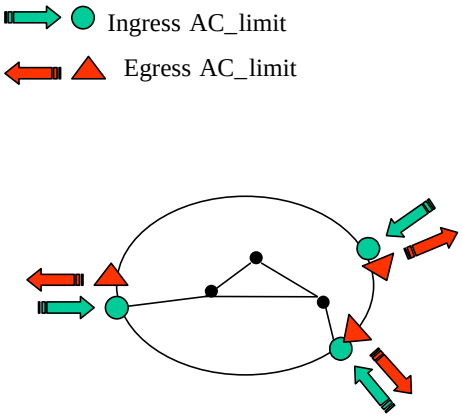


Figure 2 - Method 2 scheme

1.1.3 Method 3 – limiting ingress traffic at entry point on a per-egress basis

In this third method the admission control is performed only at the ingress ER, but the egress point is taken into account. For each TCL, each ingress ER is assigned a set of AC Rate Limits, *each one associated to a possible egress ER*. We denote by $r_{i,j}$ the AC Rate Limit at ingress ER *i* associated to the egress ER *j*. Thus $r_{i,j}$ represents the maximum amount of traffic that is allowed to cross the network from ER *i* to ER *j*. Accordingly, RCL must provide N·(N-1) limits, where N is the number of ERs.

In this approach the ingress ACA takes into account the egress point of each requesting flow. That means that the ingress ACA must somehow retrieve the information about the egress ER from the flow destination specified in the Reservation Request. Anyway this does *not* represent a major increase in the system complexity, as the so-called “egress ER discovery” is already performed in AQUILA. In facts for implementing the previously discussed Method 2, the ingress ACA must send a reservation request to the egress ACA on behalf of the requesting flow, in order to enforce AC at egress point: this is the procedure currently implemented in AQUILA for any p2p reservation.

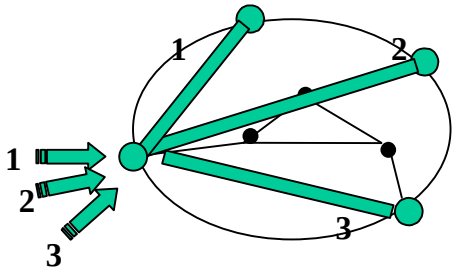


Figure 3 - Method 3 scheme

Remark that the AC Rate Limit defined for Method 3 has a completely different meaning from those relevant to the previous Methods 1 and 2. Recall that the AC Rate Limits are meant to provide a synthetic view of the available resources in the network. Clearly, the AC Rate Limit based on ingress/egress pairs as defined for Method 3 provide a more detailed and precise information network resources than those based on ingress or egress point only. One could also argue that, from a signalling point of view, Method 3 is also lighter than Method 2 as the end-to-end AC decision is done locally at one single point (i.e. the ingress ACA), rather than requiring co-ordination between two single entities (ingress and egress ACA). Also, as a further advantage, such scheme is well suited to be implemented in a MPLS environment where LSP tunnels, which are intrinsically point-to-point objects, can be naturally associated to AC Rate Limits based on source/destination pair.

1.1.4 Method 4 – classical link-by-link AC

The fourth approach is the reference case in which there is no provisioning at all and the AC Rate Limits are given link by link. In this case the admission control algorithm is run at each hop along the flow path, and must simply ensure that the total amount of traffic on each link does not exceed the assigned capacity portion for the TCL relevant to the requesting flow.

As detailed information about the current state of the network is needed to perform such a “precise” admission control: possible architectural implementations envision the presence of a centralised Bandwidth Broker (BB) or a specific control agent associated to each network router (included the internal routers). Such architectures are of course outside AQUILA: we have considered this method only because it provides reference bounds for the performance parameters relevant to the evaluation of the AC schemes.

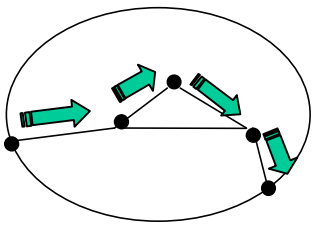


Figure 4 - Method 4 scheme

1.1.5 Discussion on the determination of the AC Rate Limits

The *AC Rate Limits* must be calculated off-line by the Initial Provisioning algorithm, before the network operation, and communicated to the ACAs. The AC Rate Limit computation for Methods 1 and 2 works as follows (details are given in Appendix 1.4.1).

Given the expected traffic matrix for a generic TCL, we derive the spatial distribution of traffic. More precisely, for each ER *i* we calculate the fraction of the expected total traffic that will be injected in *i*, and the share of such traffic directed to each egress ER *j*. In other words we derive a normalized traffic matrix T*. In order to discover the AC Rate Limits, the procedure is the following: first we increase the offered traffic according to T*, i.e. we inject a traffic equal to $m \cdot T^*$ with increasing *m*, until some link inside the network is saturated. This link will be denoted as the current bottleneck. When a bottleneck occurs, we fix the ingress AC Rate Limit for those ER injecting [and the egress AC Rate Limit for those ER extracting, in case of Method 2] traffic into the bottleneck.

Anyway if an ER is not injecting traffic that will cross the bottleneck, it is still possible to increase its AC Rate Limits. In methods 1 and 2 we mark those ER injecting into the bottleneck as *frozen*. We then increase the traffic injected into all but the frozen ERs according to their reciprocal proportions, until a new bottleneck appears. This procedure can be iterated until all ERs are declared frozen.

Anyway it is quite unlikely that some ER *x* exists with the property that the bottleneck link is not included in the path tree spanning all the ERs with which *x* exchange traffic. That means that the all the ER will likely be declared frozen in the first one or two iterations at most. This undesirable blocking behaviour, due to the fact that AC Rate Limits are egress-unaware, could be paid in terms of reduced utilisation efficiency.

In Method 3, the calculation of AC Rate Limits is done in a very similar way as for Method 1. The only difference is that when a bottleneck is discovered, not all the traffic injected into the ingress ERs is frozen, but only the AC Limits relevant to the ingress/egress pair whose paths include the critical link. So it is possible to inflate independently the AC Rate Limit relating to traffic relations (*i*,*j*).

Method 3 is not affected by the blocking behaviour evidenced above for Methods 1 and 2. This could represent an advantage of such method. In other words Method 3 offers a better precision in the allocation of bandwidth to aggregates, as egress-aware AC Rate Limits are used. From that it follows that for Method 3 the maximum total admissible traffic – bundled at any ingress ER and in the network as a whole - is greater than those relevant to the AQUILA methods[2]. Formally, for each ingress ER *i*:

[2] Note that this hold because of the adopted computation procedure for AC Rate Limits, which tries to increase the AC Limits not involved.

$$\sum_j r_{ij}[METHOD\_3] \geq r_i[METHOD\_1,2] \quad \forall i \qquad \textit{eq. 1}$$

$$\sum_i \sum_j r_{i,j}[METHOD\_3] \geq \sum_i r_i\ [METHOD\_1,2] \qquad \textit{eq. 2}$$

In fact in Method 3 the generic AC Rate Limit r_{ij} is constrained by the amount of traffic that saturates the bottleneck along the *actual* route (*i*,*j*), while in the AQUILA methods the generic AC Rate Limit r_i is constrained by the tightest bottleneck in the path tree spanning all the egress ERs that receive traffic from ingress ER *i*.

In Method 3, this positive effect has to be compared with the drawback effect derived by sharing the available bandwidth at each ER. The definition of AC mechanism is such that Method 3 suffers by the 'N squared' problem. Because the AC Rate Limits are defined for traffic relations' point-to-point, the admission control is sensible to low variations of the expected traffic. As we will explain later, it has to be studied which impact on performances has this effect in competition with the major precision of allocation.

Note that all the provisioning algorithms rely on the knowledge of an expected spatial traffic distribution.

As regards Method 4, we remember that no provisioning procedure is involved in it.

1.2 Evaluation approach

1.2.1 Performance metrics

The efficiency of a provisioning method is evaluated in terms of two parameters: *the admission loss* and the *overload loss*.

The **admission loss** represents the fraction of traffic that is rejected by the AC, and is related to the per-flow rejection probability.

The **overload loss** represents the fraction of traffic injected in the network (thus admitted by the regulatory AC) but exceeding the actual available resources (e.g. link capacity). More precisely we consider that an amount x^s_ν of traffic for TCL *s* is lost due to overflow at the generic link ν if the sum of the flows that enter the network for TCL *s*, and whose paths include link ν, exceed by x^s_ν the link capacity C_ν. The total overload loss is equal to the ratio between the sum $\sum_\nu x^s_\nu$ and the total amount of traffic entering the network.

Certainly a more precise model would consider the fact that a packet cannot be discarded more than one time along its path, and the computation of the *exact* amount of traffic lost would be far more complex. Anyway for the scope of our analysis such a precision is redundant, and the traffic lost to overflow is simply computed as described above, as our goal was to evaluate at *high-level* the consistency and the effectiveness of the AC approaches. The admission loss deals with the **effectiveness**, because it measures how much the requests are re-

jected. While the overload loss deals with the **consistency**, because it furnishes the allocation error given by the AC.

1.2.2 Evaluation strategy

In our analysis we avoid considering the dynamic behaviour of the traffic and we adopted a simple 'static' strategy, with a simple network model. In our work we adopted a bufferless network model, and a fluidic traffic model: this way we do not simulate any queue process to evaluate the overload loss. As we want to perform a static analysis, we do not consider that successive admission requests with finite size arrive randomly accordingly to some stochastic process. Instead, we simply represent every traffic relation between two edge nodes through a scalar number representing its bandwidth demand. The complete set of such demands represents a traffic matrix. On the basis of this matrix, and from the knowledge of the network topology, routing and AC Rate Limits, we compute both the overload loss and admission loss.

Considering that each bandwidth request has a finite size would mean introducing a finite *granularity* in the traffic characterisation. To explain what we mean by this term, we speak of "fine granularity" when the aggregated flow between two ERs is composed by many elementary flows of very small size, and of "coarse granularity" when it is composed by a few elementary flows of big size. Clearly the actual network loss performances depends on the assumed flow granularity, but for our purpose it is enough to study a rough model where the flows have a infinitely fine granularity, i.e. assume a fluidic traffic model.

1.2.2.1 Computation of admission loss

Some words are needed about the computation of the admission loss. Next we will refer to a single class, just to simplify the notations below. Consider the offered traffic matrix T, whose elements $t_{i,j}$ represent the amount of traffic from ingress ER i to egress ER j. Imposing a limit on the maximum amount of traffic injected by ingress ER i, as the ingress AC Rate Limit r_i in Methods 1 and 2, means imposing an upper limit on the sum over the i-th row of matrix T, i.e.

$$\sum_j t_{i,j} \le r_i \qquad \textbf{eq. 3}$$

On the other hand, imposing a limit on the maximum amount of traffic extracted out of egress ER, i.e. the egress AC Rate Limit e_j in Method 2, means imposing an upper limit on the sum over the j-th column of matrix T, i.e..

$$\sum_i t_{i,j} \le e_j \qquad \textbf{eq. 4}$$

In Method 1 and 2, when the condition eq. 3 does not hold for the offered matrix T, that means that some traffic is rejected at ER i, and precisely:

$$x_i = r_i - \sum_j t_{i,j} \qquad \textbf{eq. 5}$$

In Method 1 each traffic element $t_{i,j}$ is subject to eq. 3 only. Accordingly, it is reasonable to repartee the amount of rejected traffic x_i can proportionally shared among the single flows $t_{i,j}$, By denoting with G the matrix of admitted traffic, we will have

$$g_{i,j} = t_{i,j} - x \cdot w_{ij} \qquad \textit{eq. 6}$$

with

$$w_{ij} = \frac{t_{i,j}}{\sum_{j} t_{i,j}} \qquad \textit{eq. 7}$$

Eeq. 5 to eq. 7 express a simple transformation T → G, from the *offered* traffic matrix to the *admitted* traffic matrix.

For Method 2 things are not so simple: in fact in that case each traffic element $t_{i,j}$ is subject to *both* eq. 3 and eq. 4 at the same time, thus the proportional repartition done for Method. 1 is not meaningful. We will refer to this problem as the problem of *crossed constraints*. In general the determination of the admitted traffic matrix G from the offered traffic matrix T is not unique, and in practice depends on a number of dynamical factors including the arrival process, granularity, etc. As we are not considering the dynamical request process, we only are looking for a rough estimate of G. Thus we developed an heuristic procedure that computes G starting from T, based on iterated *small* cuts at the single elements $t_{i,j}$. Details of such procedure are given in Appendix. Further figures A1, A2 and A3 show that the obtained results are consistent with the values obtained by simulation.

Method 3 does not present the crossed constraint problem, because it checks only the condition:

$$t_{ij} \le r_{ij} \qquad \forall (i, j) \qquad \textit{eq. 8}$$

In a very simple way, if the condition in eq. 8 does not hold for the offered matrix T, that means some traffic is rejected at the traffic relation $i \to j$, and precisely:

$$x_{ij} = r_{ij} - t_{ij} \qquad \textit{eq. 9}$$

In Method 4, for each link *v* it is checked if traffic exceeds the pre-allocated capacity on link *v*. Consider the offered traffic matrix T, whose elements $t_{i,j}$ represent the amount of traffic from ingress ER *i* to egress ER *j*. Imposing a limit (equal to the pre-allocated capacity) on the maximum amount of traffic crossing the link *v* means imposing an upper limit on the sum over all the elements of matrix T whose paths cross the actual link *v*, i.e.

$$\sum_{(i,j) \in \textit{routing } n} t_{i,j} \le C_n \qquad \textit{eq. 10}$$

where C_v is the pre-allocated capacity on link *v*.

Each traffic element $t_{i,j}$ is subject to eq. 10 for all the links *v* which compose the path $(i \to j)$ at same time. Thus it rises the problem of *crossed constraints*. Also here we developed an heuristic procedure that computes the admitted matrix G starting from T, based on iterated *small* cuts at the single elements $t_{i,j}$.

The problem of crossed constraints appears also in the *overload loss* evaluation, where a single "limit" (the allocated capacity) constrains many requesting variables (the single traffic re-

lations). In order to overcome this problem and compute the overload loss we followed a similar approach based on small-iterated cuts.

1.2.2.2 Static vs. dynamic

As we do not take into account any dynamical process of arriving requests, we will refer to our scenario as the “static model”.

On the other hand we call “dynamic model” a process of traffic demands, in which the arrival instants, the size, and the duration of the reservation requests is considered. This implies that some choices (or better *guesses*) about the respective processes must be done, and greatly complex the simulations.

We choose the static approach, because of its simplicity. We are looking for a high-level evaluation and we don’t need having detailed performance results. So the static analysis fits our goal in a simple way, without requiring any guess about the underlying traffic processes.

Even if the static model represents a reduction of the dynamic one, the results that it provides can be considered as a valid approximation of what we can obtain from a detailed simulation of the dynamical model. In other terms, we claim that the static model delivers *substantially* the same information than the analysis of the dynamical model.

In order to verify this statement, we run a comparative study between our static model and a sample dynamical model, described in the following:

- Every edge device receives admission requests and performs the admission control by comparing those requests with its available resources.
- The arrival process of demands is assumed poissonian.
- The network model is bufferless.
- The duration of each reservation requests is fixed.

In the ingress-based admission control methods (Methods 1 and 3), the dynamic simulations show that in the static analysis the loss parameters present higher values than for the static model, i.e. in the dynamic model things go worse (see section 1.4.2 in Appendix for numerical resultant graphs). We run several dynamical simulations by varying the size of reservations, i.e. the granularity of the flows. The results for the loss values show that the difference with the static model decreases with the size of reservation. This is an expected behaviour, as increasing the flows granularity means approaching the fluid model that was assumed for the static model. Remark that in a core network scenario, as we are, it is reasonable to assume that the flow size is small compared to the link capacities. Accordingly the fluid model assumption is comforted.

1.2.3 Experimental scenario

Now we will give an overview about the off-line provisioning procedure calculating the admission control rate limits, and about the simulation scheme used to evaluate the efficiency parameters. The following general discussion applies to all the AC Methods 1, 2 and 3. The differences between the provisioning algorithms for each Method will be discussed later.

The provisioning algorithm is represented in Figure 5 along with its input and outputs.

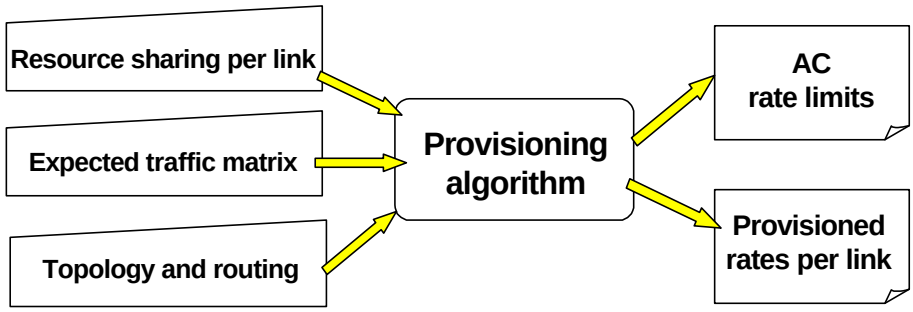


Figure 5 – The provisioning procedure

The first input is represented by topology. For the choice of topology we used both real topologies and randomly generated topologies ([2],[3]). An example of real topology is given by the vBNS (Very high speed Backbone Network Services) network, which is a North American backbone connecting several universities and research organisations. We have also tested other real topologies, but it is not often simple collecting complete data-sheets. Random topologies have already been used in a several previous works. For the topology generation we used GT-ITM software [5].

Once a topology and capacities of links are given, it is necessary to find the shortest path between every couple of nodes of the network. The routing-table that must be passed to the provisioning algorithm is provided by a routine that looks for the Shortest Path according to a fixed metric where each link has a cost inversely proportional to its capacity.

The provisioning algorithm must take into account some policy rules in the distribution of resources for the different TCLs. This is represented in Figure 5 by the Resource Sharing block.

Another input is represented by the expected traffic matrix. The traffic matrix is given apart a scale factor, i.e. in terms of traffic distribution between ingress / egress ERs. Indeed it is difficult to find in literature guidelines about the generation of a reasonable traffic matrix. For our work we used two different synthetic generators of traffic matrices, that are illustrated in section 1.4.3 of the Appendix: i) the Fortz-Thorup model and ii) a modified version of it. In the Fortz-Thorup model [4] the traffic is completely independent from the topology. The second model introduces some dependency of the traffic demand on the network topology.

The synthetic generator of traffic matrices provides the realisation of the expected traffic matrix. On the basis of the expected traffic matrix we run the provisioning algorithms calculating the AC Rate Limits. Likely the real traffic distribution could not exactly match to the expected one, so the aim of experimentations is to test the effectiveness and the consistency of the provisioning-based methods when the actual traffic demand derives from a perturbation of the matrix generated by the Fortz-Thorup model or by the derived one. On the expected traffic matrix we introduced a perturbation composed of two factors: the first one is a scale factor, which multiply the expected matrix; the second one is an additive gaussian noise. We consid-

ered the traffic demands of different classes independently (this restriction needs to be overcome in further studies).

Thus the actual incoming traffic, for every TCL, is given by:

$$[T] = k \cdot [F] + [N(c)]$$

where [T] = actual traffic matrix; [F] = expected traffic matrix; [N] = random matrix that represents prediction error; k = scale factor.

[N] is a matrix whose component n_{ij} is a realisation of a gaussian random variable. The mean is assumed null while the variance is such that every t_{ij} has constant rate of variation c, i.e.:

$$\bar{n}_{ij} = 0$$

$$s_{ij} = f_{ij} \cdot c$$

In Figure 6, it is shown on which model is led the simulation.

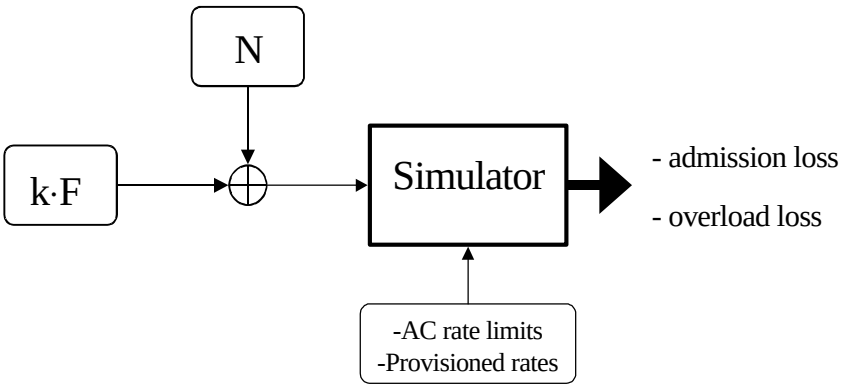


Figure 6 – Perturbation model

We tested several simulations varying K being c fixed, and varying c and being K fixed. The values of every sample are determined of the average of 100 realisations, so we checked network performances in a statistical way.

Now we want to stress that if we assume gaussian noise that overlaps traffic matrix, the mean and the variance loss at call level is, for Method 1 and 3, completely determined by mathematical analysis. For example, if we refer to Method 3, it can be shown that:

If r_{ij} is the actual AC Rate Limit, the mean *admission loss* is equal to:

$$E\{(x - r_{ij})^+\} = \int_{x=r_{ij}}^{+\infty} (x - r_{ij}) p_x dx \equiv \Pi_{ij} \qquad \textbf{eq. 11}$$

where the density function of probability is:

$$p_x = \frac{1}{s_x \sqrt{2p}} \cdot e^{-\frac{(x-m_x)^2}{2s^2}} \qquad \textit{eq. 12}$$

with

$m_x = K \cdot f_{ij}$

$s_x^2 = s_{ij}^2 = t_{ij} \cdot f_{ij}$

In a similar way we can compute the second order moment:

$$E\{((x-r_{ij})^+)^2\} = \int_{x=r_{ij}}^{+\infty} (x-r_{ij})^2 p_x dx \qquad \textit{eq. 13}$$

and the variance of admission loss:

$$s_\Pi^2 = E\{((x-r_{ij})^+)^2\} - E^2\{(x-r_{ij})^+\} \qquad \textit{eq. 14}$$

But when dependent constraints are in competition, we have to manage stochastic variables that are neither gaussian nor independent. So it is not possible to determine values in closed form.

1.2.3.1 Examples of topologies

Here we want to gives two examples among the studied topologies.

The first one is the vBNS core topology, on which we take results that we will show in next section. The second one is a topology derived from the vBNS, to confirm the results even increasing the hop-count distances among ERs.

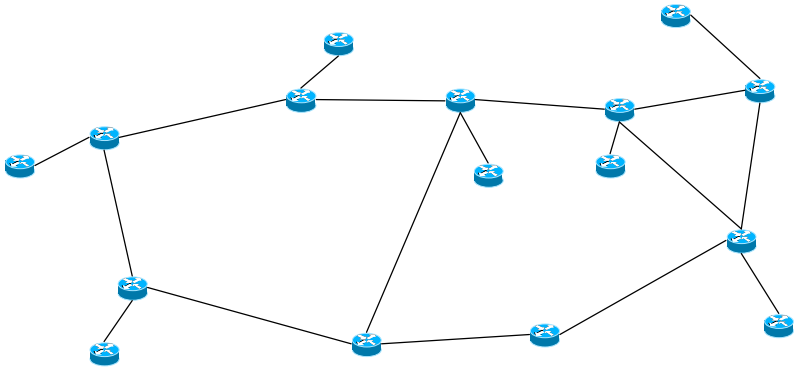


Figure 7 – vBNS topology

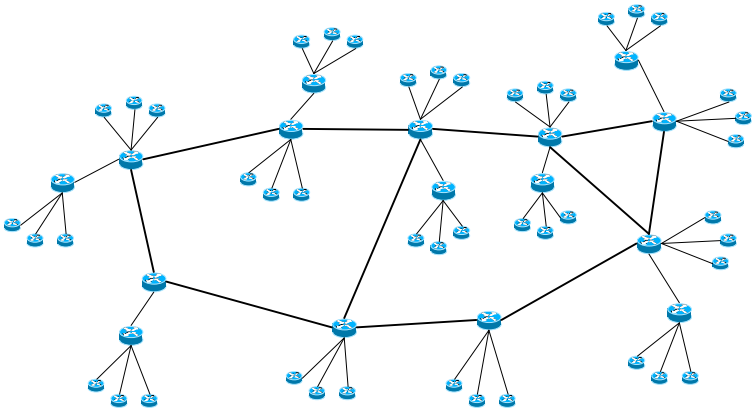


Figure 8 – vBNS-derived topology

1.3 Experimental results

In the next study we focus on a single TCL. Anyway that is not a limitation as far as condition $\Sigma C_i{}^s < C$ (see [1] for details) holds for each link, as it allows decoupling the analysis for different TCLs.

Let's now consider the *admission loss*.

Method 2 is stricter in comparison with Method 1, because in Method 2 there is also a control at egress ERs. Accordingly, we expect that the admission loss for Method 2 will be higher than for Method 1. This is verified to Figure 9 and Figure 10 as the curve relevant to Method 2 is always above the one relevant to Method 1.

Let's now compare Method 2 and Method 3. Two different phenomena are on the ground. On one hand, Method 3 suffers the "N squared" problem: as each ER has N-1 associated AC Limits (N is the number of ER in the network), it can be seen that the bandwidth on each links is repartee into a number of components on the order of N^2. In other words, we partition the overall network capacity into a large set (N^2 elements) of very small AC Limits. Intuitively, this could lead to a large inefficiency in the resource distribution. On the other hand, Method 3 offers a higher precision in the allocation of the quantity of bandwidth assigned to AC Rate Limits. (Recall that Method 3 is not affected by the fastidious blocking behaviour discussed in section 1.1.5). From that it follows that the maximum admissible (but not necessarily admitted!) traffic in the network is greater in Method 3 than in the AQUILA methods (see section 1.1.5 and eq. 1 to eq. 2). The objective of the experimental simulation is to find out how these phenomena drive the admission loss performance of Method 3 compared to Method 2.

We see from Figure 9 that for low perturbations (i.e. small values of parameters *s* or *K*, the curve of Method 3 stays above the curves relevant to the AQUILA methods). It means that Method 3 is loosing more than methods 1 and 2. In this region, the effect of "N squared" phenomenon is prevailing. Note that a small perturbation represents a small uncertainty in the

forecast of the actual traffic matrix. At the same time there is always a region where Method 3 gives better performances. For high perturbations, the curve of Method 3 stays below the curves relevant to the AQUILA methods. It means that Method 3 is loosing less than methods 1 and 2. In this second region, it is prevailing the effect of the higher precision in the allocation of the bandwidth to the AC Rate Limits.

All that is shown in the next Figure 9, where values of admission losses for the vBNS core network are represented, varying the rate of variation *c* (on the left figure) and varying the scale factor *K* (on the right figure). The traffic demand model refers to the Fortz-Thorup's model.

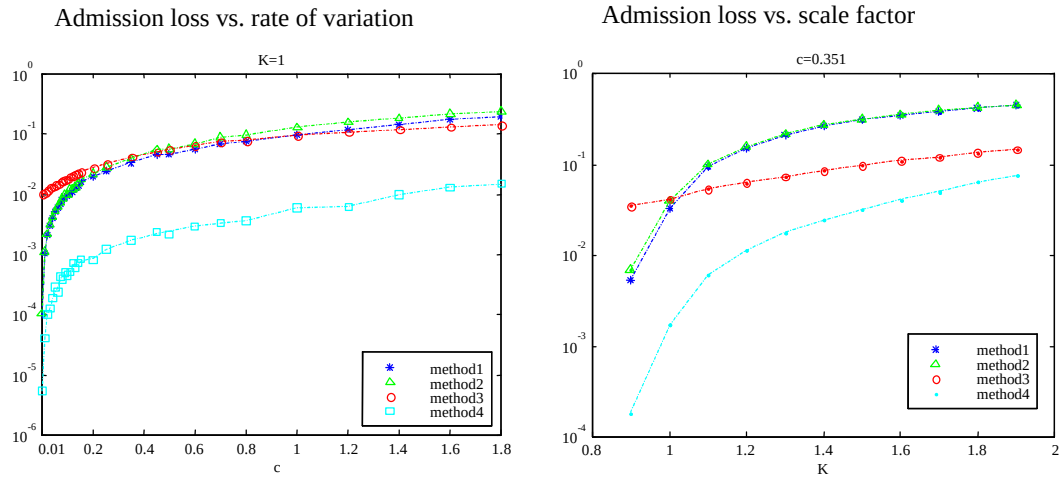


Figure 9 – admission loss related to Fortz-Thorup's traffic model

Let's consider the case in which the expected traffic matrix is generated according to the topology-dependent model. Note that the aim of this model rises from the necessity of following the behaviour of real networks. We mean that every network is dimensioned looking for serving the offered traffic: thus it is unlikely to have high capacity links that are almost unloaded. The topology-dependent model is described in detail in section 1.4.3.2 in the Appendix. In Figure 10 the curves assume the same aspects as in Figure 9, but the region of goodness of the Method 3 is less extended. The reason is due to the distribution of the links load in the network. In fact, if we define the *global weighted utilisation of the network links* as the average of links utilisation weighted by their capacity, we can say that the current model gives a higher value of this parameter in comparison with the Fortz-Thorup's model.

In comparison to Figure 9 (where it is assumed the Fortz-Thorup's model), another point to be noticed is that the performances of provisioning algorithms come nearer to the *admission loss* values of the centralised and adaptive Method 4.

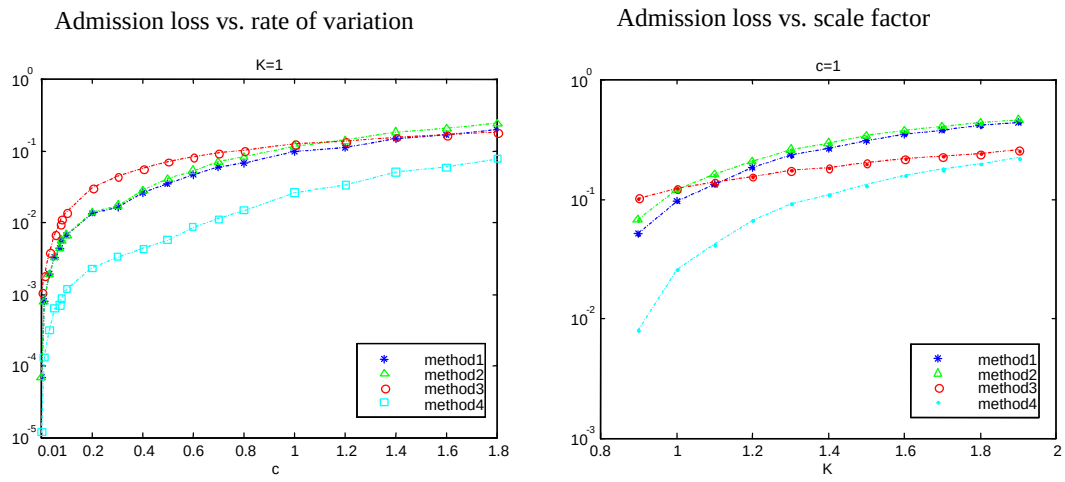


Figure 10 – admission loss related to topology-dependent traffic model

Let's go to consider the *overload loss*.

First of all, it should be noticed that in Method 3 there is no overload: it is possible due to the choice of the provisioning algorithm. In fact the AC controls point-to-point flows, so the provisioning can avoid overload losses. The second point is to explain how much the overload loss depends on the traffic distribution. In particular, the overload trouble in Method 2 is more or less critical depending on the distribution of network load; as we said previously, it results by assuming one or the other between the traffic demand models. In fact if the bottlenecks (links with high utilisation) are located near the ISP border, the admission control both at ingress ERs and at egress ERs overcomes overload troubles, what is not true if the distribution of links utilisation in the network is more uniform (like in the topology-dependent model of traffic demand).

Next we will show when Method 2 suppresses overload due to an efficient admission control, and when it can't.

Figure 11 refers to the vBNS network assuming the uncorrelated traffic model of Fortz-Thorup. It could be appreciated how low are the values corresponding to Method 2 (admission control both at ingress ERs and at egress ERs).

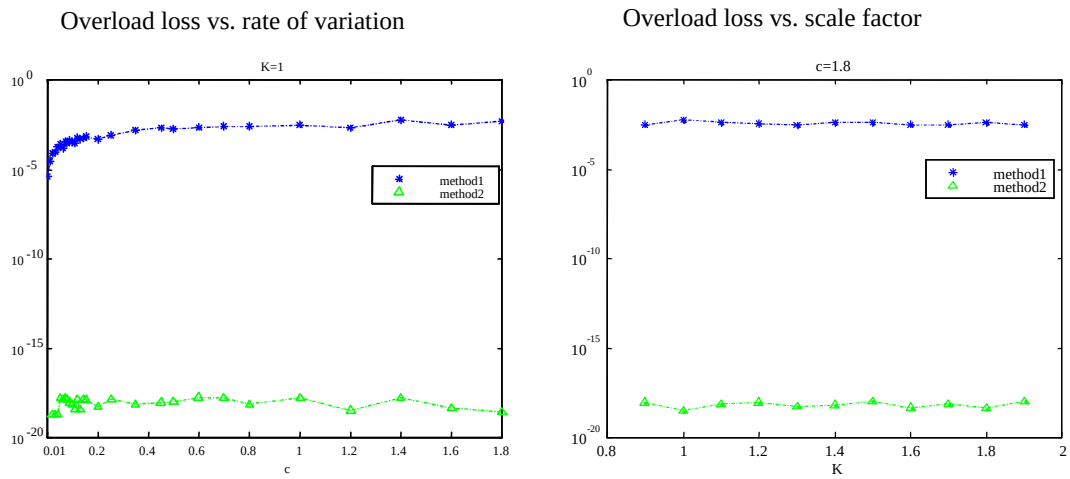


Figure 11 – overload loss related to Fortz-Thorup's traffic model

The curves referring to the correlated model they are shown in Figure 12. This time the overload loss in Method 2 is comparable with Method 1. This means that implementing an admission control at the egress points too doesn't improve the efficiency of the AC in terms of overload loss.

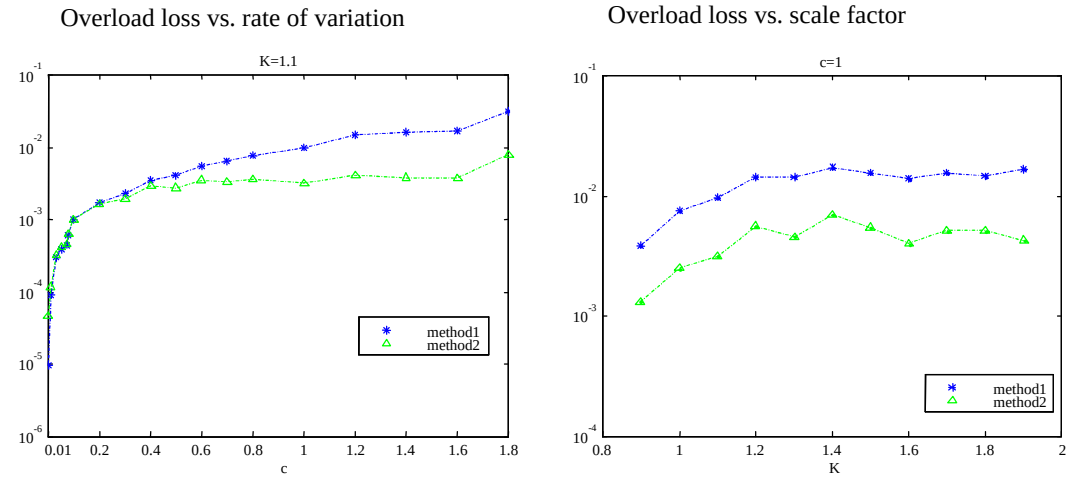


Figure 12 – overload loss related to topology-dependent traffic model

It can be said that a control on outgoing traffic is useful when it is probable the rising of bottleneck near to egress nodes. Method 1 and Method 2 control the total amount of traffic at ingress and/or egress, but they cannot fully avoid the rise of congestion on internal links since they cannot manage information about how flows are routed.

1.4 Appendix

1.4.1 Provisioning algorithms

Let's remember the notation in [1]:

b_i^s = fraction of total traffic of TCL *s* that will be injected at ER *i* according to traffic forecast, with $\sum_j b_i^s = 1$

$a_{i,j}^s$ = share of the ingress traffic of TCL *s* at ER *i* going to egress ER *j* with $\sum_j a_{i,j}^s = 1$

We also define:

C_n = bandwidth of link *v* of the considered topology.

C_n^s = share of bandwidth of link ν used for TCL *s* (that is resource partition for TCL *s*).

$d_{n,i,j}$ = share of the traffic going from ingress ER *i* to egress ER *j* that will cross link ν such that:

$d_{n,i,j} = 1$ if traffic from ingress ER *i* to egress ER *j* will cross link ν and

$d_{n,i,j} = 0$ else.

L^s represents the maximum amount of TCL *s* that can be injected into the network until a link is saturated

l_i^s is the share of the maximal amount of traffic of TCL *s* that can be injected into ER *i.*

Provisioning algorithms will calculate the following configuration parameters:

z_n^s = provisioned rates for all supported classes *s* and links *v* of an IP backbone. The provisioned rate z_n^s is the maximum amount of traffic of the traffic class *s* that should transit link *v* according to provisioning. Admission control, policing and queue management has to take care of that. Provisioned rates will be used to set the WFQ weights in the routers.

r_i^s (or $r_{i\ j}^s$) = AC Rate Limits for all ERs *i* at ingress (or for traffic relation *(i,j)*), that is the maximal amount of traffic that can be injected into the network from ER *I* (or for traffic relation *(i,j)).*

e_j^s = AC Rate Limits for all ERs *j* at egress, that is the maximal amount of allowed traffic at egress from ER *j.*

1.4.1.1 Method 1

Referring to the notation given before, we assume to know b_i^s, $a_{i,j}^s$, $d_{n,i,j}$, and we define the following quantity:

$$a_{n,i,j}^s = a_{i,j}^s d_{n,i,j} \tag{1}$$

$a_{v,i,j}^s$ represents the load of traffic relation between ERs *i* and *j*. We can demonstrate that the equation finding out the link that constitutes the bottleneck is given by the subsequent:

$$L^s = \min_n \left(\frac{C_n^s}{\sum_i b_i^s \sum_j a_{n,i,j}^s} \right) \tag{2}$$

We will denote this bottleneck link by ν_s. The limit amount of incoming traffic from each ER *i* is given by:

$$l_i^s = b_i^s L^s \tag{3}$$

Anyway, if the ER is not injecting traffic into the bottleneck, it can happen also that it is still possible to increase the traffic entering some ER. To find out this additional traffic, we can follow this procedure:

We classify as *frozen* the ERs injecting traffic into the bottleneck and we consider a new traffic matrix and a new topology, assuming:

- $\forall i \mid \quad ED \quad i \; is \; frozen: \quad b_i^s = 0$
- $\forall i \mid \quad ED \quad i \; is \; not \quad frozen: \quad b_i^{s'} = \frac{b_i^s}{\sum_{j \; not \; frozen} b_j^s}$
- $\forall n \quad : \quad C_n^{s'} = C_n^s - \sum_{i \; frozen} l_i^s \sum_j a_{n,i,j}^s$

Under this position we go and find the new bottlenecks and new $\tilde{l}_i^s$ so that

$$\tilde{l}_i^s \to l_i^s + \frac{b_i^s}{\sum_{j \; not \; frozen} b_j^s} I$$

where *I* is the total increment to L^s. This routine must be iterated until all ERs are *frozen.*

The final AC Rate Limit of ER *i* is:

$$r_i^s = \tilde{l}_i^s$$

This routine must be repeated for every TCL *s*.

1.4.1.2 Method 2

The second case goes beyond the preceding scheme and it foresees that every pair ER/TCL has two AC Rate Limits: one for the ingress and one for the egress. The routine for calculating ingress AC Rate Limit is exactly as explained in first method. So, as found before:

$$r_i^s = \tilde{l}_i^s$$

Furthermore, in this case also egress *AC Rate Limits* must be determined too. For ER *j*, the egress *AC Rate Limit* is given by the following:

$$e_j^s = \sum_{n \in Dest(j)} \sum_i r_i^s a_{i,j}^s d_{n,i,j}$$

where *Dest(j)* is the set of links that terminate on ER *j*.

The pair (r_j^s, e_j^s) will be used by each ER to drive two independent admission controls: one for incoming traffic and one for outgoing traffic.

1.4.1.3 Method 3

Another possible approach is providing ER with AC Rate Limits for pair source/destination: the admission control is performed only at ingress but based on destination.

As in the previous two cases we have:

$$a_{n,i,j}^s = a_{i,j}^s \cdot d_{n,i,j}$$

$$L^s = \min_n \left(\frac{C_n^s}{\sum_i b_i^s \sum_j a_{n,i,j}^s} \right)$$

$$\forall (i,j) \ : \quad l_{i,j}^s = (b_i^s \cdot L^s) \cdot a_{i,j}^s$$

Anyway it is still possible to increase independently all the other traffic relations routed elsewhere. We classify as *frozen* all traffic relations *(i,j)* that inject into the bottleneck and we consider a new traffic matrix and a new topology such that:

- $\forall \quad i,j \ \mid (i,j) \ is \ \ frozen: \quad a_{i,j}^s = 0$

- $\forall \quad i, j \;\big|\; (i,j) \quad is\ not\ \ frozen: \quad a_{i,j}^{s} = \dfrac{a_{i,j}^{s}}{\sum_{(i,j)\,not\ \ frozen} a_{i,j}^{s}}$

- $\forall n \;\; : \quad C_n^{s'} = C_n^{s} - \sum_i \sum_j l_{i,j}^{s} d_{n,i,j}$

Under this position we go and find the new bottlenecks and the new $\tilde{l}_i^{\,s}$ so that

$$\tilde{l}_i^{\,s} \to l_i^{s} + \frac{a_i^{s}}{\sum_{j\ not\ \ frozen} a_j^{s}} I$$

where *I* is the current increment to L^s. This routine must be iterated until all traffic relations (i,*j)* are *frozen.* The final AC Rate Limit of *(i,j*) is:

$$r_{i,i}^{s} = \tilde{l}_{i,i}^{\,s}$$

1.4.2 Comparison between static and dynamic models

We implemented a dynamic model of the process of traffic demand and then we compared it with the static model. A "Poisson" model represents the process of traffic demand. We assumed flows of assigned size (it is the flow *resolution*). The processes of arrival instants and time length of the flow requests are such as exponential distributions**.**

We tested some sample simulations to extract values of admission loss. Having fixed the size of elementary flows, we can collect data on refusals of the requesting flows. We observe the process of requesting flows and the respective AC decisions, and we calculate statistic parameters in the condition of stationarety of the process.

As we said in chapter 3, we should distinguish when there are dependent constraints in the AC mechanism and when not. In the ingress-based admission control methods, the static model simply compares the actual traffic matrix with the set of AC Rate Limits. Thus in Method 1 it is checked if the total amount of traffic which is requesting to enter at each ER (its total bandwidth is equal to the sum of the correspondent row in the actual traffic matrix) is less than the ER's AC Rate Limit. In Method 3 it is checked if the bandwidth of the requesting flow on the traffic relation (*i,j*) is less than the AC Rate Limit r_{ij}. The measurements given by the simulation model are closer to the loss values given by the static analysis, when the granularity of the requesting flows becomes finer.

It can be appreciated looking at the following figure, in which are depicted three values of *admission loss*. The static values are compared with the sample means given by the simulation measurements. When the granularity of the requesting flows becomes finer, the 'static' values are closer to the simulations values.

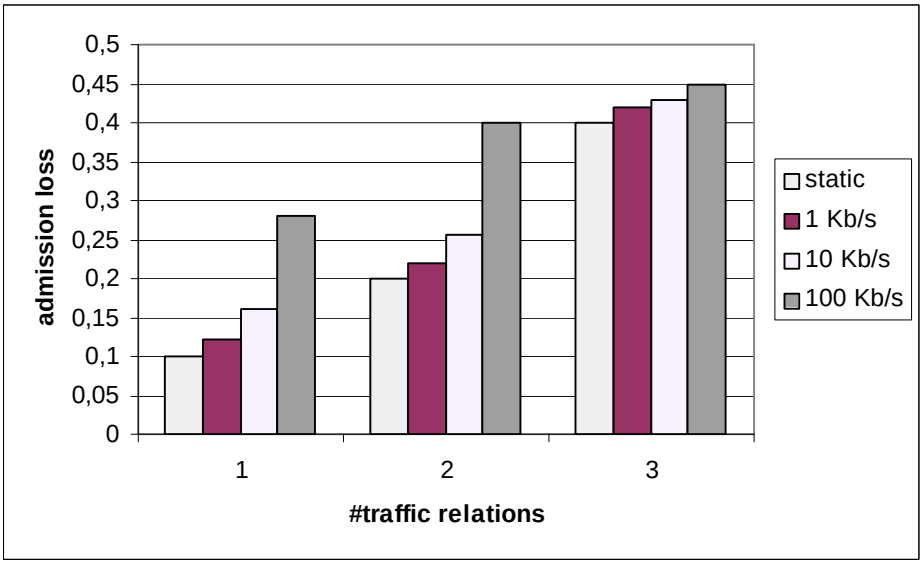


Figure A1 – Admission loss measurements varying resolutions

Now we want to discuss the case in which some constraints are dependent each other. For example consider the AC in Method 2. It can happen that the sum of the elements on a row of the actual traffic matrix is major than the correspondent ingress AC Rate Limit, and, at the same time, the sum of the elements on a column of the actual traffic matrix is major than the correspondent egress AC Rate Limit. In the static analysis this condition rises the question on the way to cut the excess of bandwidth among the matrix elements. Since there are independent controls on ingress and egress, every cut in ingress has a consequence at egress and vice versa. Our choice is to follow a fairness approach. Now we will show the heuristic algorithm used to calculate "at one shot" for ingress and egress how much traffic must be rejected to reach the minimum total loss with the fairest cut. It follows an explanation on Method 2, but the *fairness* policy is choose in every case of crossed constraints (in the calculation of admission loss in Method 4 and in the calculation of overload loss in AQUILA Methods).

[T] is the actual traffic matrix, r is the ingress AC Rate Limit vector, e is the egress AC Rate Limit vector, in a network with N edge routers.

$$T = \begin{pmatrix} 0 & t_{12} & .. & t_{1N} \\ t_{21} & 0 & .. & t_{2N} \\ .. & .. & .. & .. \\ t_{N1} & t_{N2} & .. & 0 \end{pmatrix}$$

$$r = \begin{pmatrix} r_1 & r_2 & .. & r_N \end{pmatrix}$$

$$e = (e_1 \quad e_2 \quad .. \quad e_N)$$

Consider that the sum of the elements on the *i*-th row of the actual traffic matrix is major than the correspondent ingress AC Rate Limit r_i, and, at the same time, the sum of the elements on the *j*-th column of the actual traffic matrix is major than the correspondent egress AC Rate Limit e_j.

$$\begin{cases} \sum_{j=1}^{N} t_{ij} > r_i \\ \sum_{i=1}^{N} t_{ij} > e_j \end{cases}$$

Having fixed *M* as the maximum number of iterations within the algorithm should converge, at the generic *k*-th step it is checked which rows and columns exceed the respective AC Rate Limits. For each of these it is cut an amount of bandwidth equal to the exceeding bandwidth multiplied by *k* / *M*. The aggregated flow to cut is distributed on the elements of the row or of the column in a way proportional to their bandwidths.

Thus at step *k* for the *i*-th row exceeding the AC Rate Limit r_i it could be calculated the generic element t_{ij}:

$$t_{ij}^{(k+1)} = t_{ij}^{(k)} - \frac{t_{ij}^{(k)}}{\sum_{j=1}^{N} t_{ij}^{(k)}} \cdot \left(r_i - \sum_{j=1}^{N} t_{ij}^{(k)} \right) \cdot \frac{k}{M}$$

where $t_{ij}^{(k)}$ is the actual flow request at step *k* on the traffic relation (*i*,*j*).

In the same way at step *k* for the *j*-th row exceeding the AC Rate Limit e_j it could be calculated the generic element t_{ij}:

$$t_{ij}^{(k+1)} = t_{ij}^{(k)} - \frac{t_{ij}^{(k)}}{\sum_{i=1}^{N} t_{ij}^{(k)}} \cdot \left(e_j - \sum_{i=1}^{N} t_{ij}^{(k)} \right) \cdot \frac{k}{M}$$

where $t_{ij}^{(k)}$ is the actual flow request at step *k* on the traffic relation (*i*,*j*).

If we collect the sample mean of the N edge routers by the dynamic simulation, we can identify the vector:

$$m_p = \begin{pmatrix} \overline{p}_1 \\ \overline{p}_2 \\ .. \\ \overline{p}_N \end{pmatrix}$$

Now we introduce a metric to evaluate how the static values of *call loss* fit to the dynamic measurements. **O** is the vector identifying the values given by the static model. Thus the metric wants to evaluate the distance between the two N-dimensional points **O** and $\mathbf{m_p}$.

The metric is:

$$\nabla = \sqrt{\sum_{i=1}^{L} \left(\overline{p}_i - o_i\right)^2}$$

where $\mathbf{o_i}$ is the *i*-th element of the vector **O**.

Then we evaluate the precision of the distance $\tilde{N}$ by its variance calculated in a simulative way. We assume that the stochastic variables given by every element of the sample mean are independent. Thus they are generated 10000 realisations of the variable "sample mean" as realisations of a N-dimensional gaussian variable, in which the N components are uncorrelated. Then for each of the 10000 realisations it is calculated the distance $\tilde{N}$ from **O**. In particular the standard deviation is known.

In that way we had the instruments to demonstrate that the 'static model' is an approximation of the dynamic process and it fits better becoming the resolution of elementary flows finer. In fact we collect many samples in which the metric ∇ decreases becoming the request granularity finer.

Next we will show two figures describing how measurements fit to the static values of admission loss. Given a fixed resolution of the elementary flows, each figure represents the static values in comparison with the sample means and their relevant confidence intervals.

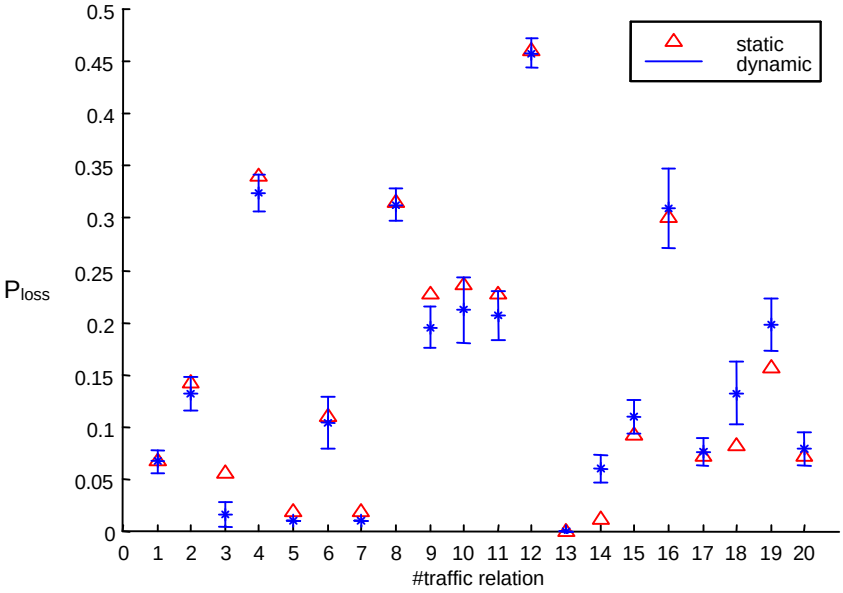


Figure A2 - comparison @ 100 Kb/s resolution

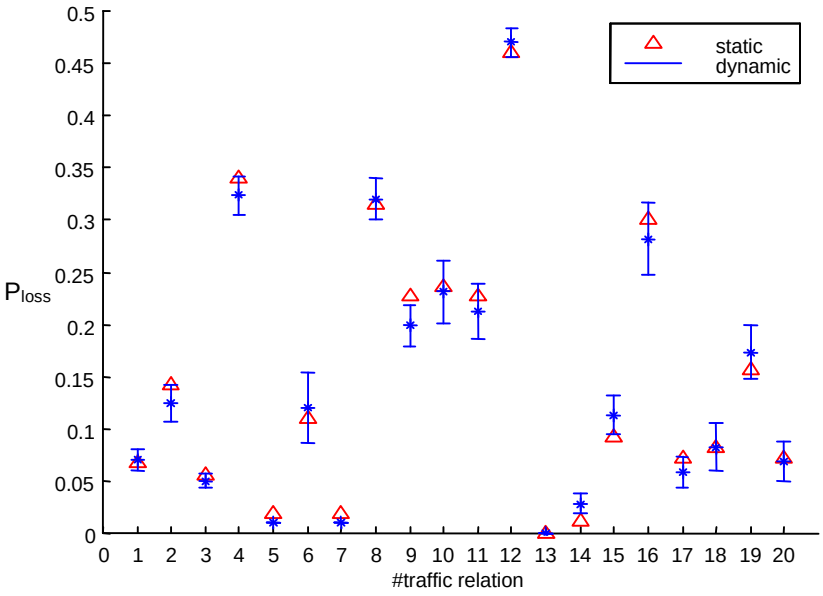


Figure A3 – comparison @ 10 Kb/s resolution

Resolution	Ñ	**Variance of** Ñ
1 Mb/s	0.43	0.000089
500 Kb/s	0.24	0.000107
100 Kb/s	0.073	0.000113
50 Kb/s	0.064	0.000132
10 Kb/s	0.051	0.000163
5 Kb/s	0.045	0.000357
1 Kb/s	0.043	0.000146
500 b/s	0.040	0.000126
100 b/s	0.033	0.000118

Table 1

1.4.3 Traffic matrix models

1.4.3.1 Fortz-Thorup model

The first model is derived from the model given in the article [4] developed by Fortz and Thorup.

The generic traffic relation, where *x* is the source node and *y* is the destination node, is calculated by:

$$a \cdot O_x \cdot D_y \cdot C_{(x,y)} \cdot e^{-d(x,y)/2\Delta}$$

O_x and D_y are stochastic variables uniformly distributed in the interval [0,1].

$C_{(x,y)}$ represent la correlation existing between the nodes *x* e *y*: it is a random number comprised in the interval [0,1].

δ is a matrix which those elements are the distances among ERs in terms of sums of OSPF costs.

1.4.3.2 Topology-dependent model

Now, in this model is introduced the correlation between traffic matrix and topology. So we should introduce, for every ER:

FAN_IN : is the sum of the incoming capacities connected to the actual ER

FAN_OUT : is the sum of the outgoing capacities connected to the actual ER

The generic traffic relation, where *x* is the source node and *y* is the destination node, is calculated by:

$$a \cdot O_x \cdot D_y \cdot C_{(x,y)} \cdot e^{-d(x,y)/2\Delta}$$

$C_{(x,y)}$ represent la correlation existing between the nodes *x* e *y*: it is a random number comprised in the interval [0,1].

δ is a matrix which those elements are the distances among ERs in terms of sums of OSPF costs.

O_x is a stochastic variable uniformly distributed in the interval [0.6·m_{Ox} ,1.4·m_{Ox}].

$$m_{Ox} = l_1 + l_2 \cdot \frac{FAN\_in - FAN^{\min}\_in}{FAN^{\max}\_in - FAN^{\min}\_in} \qquad l_1 \le l_2$$

The apices *min* e *max* refer to the maximum and the minimum values assumed by the FAN_in at the ERs; l_1 and l_2 are two constants indicating respectively m_{Ox} calculated for the ER at minimum FAN_in and for the ER at maximum FAN_in.

Symmetrically, D_y is a stochastic variable uniformly distributed, with mean m_{Dy} and in the interval wide $\frac{4}{5} m_{Dy}$.

$$m_{Dy} = l_3 + l_4 \cdot \frac{FAN\_out - FAN^{\min}\_out}{FAN^{\max}\_out - FAN^{\min}\_out} \qquad l_3 \le l_4$$

1.5 References

[1] Deliverable D1301, Specification of traffic handling for the first trial, AQUILA project consortium, http//www-st.inf.tu-dresden.de/Aquila/, September 2000.

[2] K. L. Calvert, M. B. Doar, E. W. Zegura, Modeling Internet Topology, IEEE Communications Magazine, June 1997

[3] E. W. Zegura, K. L. Calvert, S. Bhattacharjee, How to model an internetwork, IEEE INFOCOM 1996

[4] B. Fortz, M. Thorup, Internet Traffic Engineering by Optimizing OSPF Weigths, IEEE INFOCOM 2000

[5] E. W. Zegura, GT-ITM: Georgia Tech internetwork Topology Models (software), http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz

2 Evaluation of Resource Pools and BGRP

AQUILA uses dynamic resource management, i. e. continuous online demand measurements and resource allocations based on deduced demand forecasts, in order to reduce signalling traffic and to utilise network resource efficiently and flexibly. Dynamic resource management is applied intra-domain, implemented in the *Resource Pools* (RP), as well as inter-domain, implemented in the *quiet grafting* (QG) feature of BGRP (see [1]). In both cases, the question about scalability arise and performance evaluations were conducted using simulation models. This section presents results of these performance evaluations of AQUILA's dynamic resource distribution processes.

In order to give *Admission Control* (AC) the necessary information about how much traffic can be sent across a core network, two bandwidths values, called *AC Limits* (ACL), are provisioned for each traffic class (but best effort) to each ACA. One is for ingress AC and limits the ingress traffic. The other is for egress AC and limits the egress traffic. *Resource Pools* (RP), more specifically hierarchical sets of RPs, are used to provision ACLs dynamically to AC.

An individual hierarchical set of RPs distributes bandwidth in each traffic aggregation and distribution area close to the network border (see [1]). Each hierarchical set of RPs creates a resource distribution tree. This tree is used to dynamically distribute the network resources, i. e. the bandwidth that is given at the root RP, to AC in the ACAs which are the leaves. In case an ACA needs more resources, i. e. a higher ACL, a signalling message is sent to its RP in order to request additional bandwidth. If a RP is not able to serve the request, a further signalling message will be sent to the next higher level RP, which itself may send a further signalling message and so on. Signalling messages are used to return unused resources in a similar way. Altogether RPs build a dynamic resource distribution process, that adapts resource allocations to AC to online measured traffic load. Therefore, each RP has to manage reservation states and to process signalling messages and the question about scalability arise.

In AQUILA, inter-domain resource reservations are established in a very similar way like RPs distribute resources from a root RP to ACAs. BGRP uses sink trees which correspond to resource trees of RPs. BGRP agents aggregate received requests, that try to reserve resources of the same sink tree, into a single reservation towards the root of a sink tree, just like RPs aggregate incoming requests into a single reservation towards the next level RP. Here again the question about scalability arise.

Both, RPs and BGRP, reduce the number of signalling messages that have to travel across a resource tree respective sink tree. Both use the same resource management scheme to do this. In order to reduce the signalling load, local resource cushions are established that allow local responses to future requests (local AC). In order to use network resources efficiently, resource cushions are continuously adapted to online measured demand. Similar independent and local procedures for the management of resource cushions are used in both cases, intra-domain and

inter-domain resource management. A simulation model was developed to evaluate the performance of these dynamic resource distribution processes.

A common model for intra-domain and inter-domain resource management is a resource management tree that consists of a root node that determines the available resources, a number of resource distribution nodes that allocate resources and distribute them down-streams towards the leaves, and an number of leave nodes and may be intermediary nodes that receive resource requests and release messages that stimulate resource distribution across the tree.

The remainder of this section is organised as follows. The first subsection describes the performance measures that were used for performance analysis and summarises the results. Next subsection sketches the different dynamic resource management schemes that were evaluated. It follows a subsection that describes the simulation and traffic models. Finally, detailed performance results are presented in the last subsection.

2.1 Performance Measures and Summery

Together with scalability a number of performance measures were analysed:

- signalling reduction factor (SRF) (scalability indicator)
- cushion size (CS) and signalling reduction trade-offs (price paid for signalling reduction)
- fairness (of resource sharing)
- manageability

Signalling Reduction Factor (SRF)

The SRF is the ratio, $\frac{N_o}{N_i}$, of the number of signalling messages, N_o, that are sent by a resource management node (ACA, RP, BGRP agent) to the number of signalling messages, N_i, it received. The SRF indicates the fraction of signaling messages that cannot be answered locally but passes a resource management node, i. e. are sent to the next hop in a RP hierarchy or sink tree.

In summary, a reduction of the number of signaling messages by two orders of magnitude are reached in most scenarios. But, SRF depends on traffic load and control parameters of resource management. The higher the traffic aggregate the better the SRF, and even more than 2 orders of magnitude are possible. SRF depends on control parameters which influence cushion size as described below.

Cushion Size (CS) and Signalling Reduction Trade-off

The cushion size is the amount of unused resource that a resource management node (ACA, RP, BGRP agent) keeps to be able to stop propagation of future signalling messages, i. e. bandwidth that is allocated up-streams towards the root of a resource or sink tree but not reserved down-streams towards its leaves.

In summary, the larger the resource cushions the better are the signalling reduction factors. So, signalling reduction is accompanied with an utilisation trade-off. Reasonable cushions with a size of around 25% were used to reach reduction of the number of signalling messages by two orders of magnitude. But, cushions size depends like SRF on traffic load and control parameters of resource management. The higher the traffic aggregate the smaller the cushions. Cushion size can be controlled by different control parameters of resource management. With cushion size SRF is controlled implicitly.

Fairness

Resources, i. e. available bandwidth, should be distributed in a fair way between all competing nodes. We used the resulting blocking probabilities as a measure for fairness. Thus a dynamic resource management is fair, if the resulting blocking probabilities of all nodes competing about the same resources are almost equal.

In summary, some resource management schemes must be configured very carefully to avoid unfair resource allocations. Even worse, a good configuration in view of fairness depends on the traffic load. After we introduced an additional control loop for auto-adaptation, the resource management schemes worked fine.

Manageability

Manageability is a different performance measure that focuses on how easy it is to use a resource management schemes in a real environment and answers questions like: Is configuration easy? Is it robust to changing traffic conditions?

We evaluated a small number of different resource management schemes. All of them performed similar in view of the performance parameters above, but they are harder or easier to control. All have got control parameters that control the building and management of resource cushions, which determines the signalling reduction factor and fairness. Some need more careful selected control parameter values and more circumspectly parameter adaptations if an optimal configuration is searched.

2.2 Resource Management Schemes

2.2.1 Watermark Schemes

Different watermark schemes are defined in AQUILA. Basically, they use two different watermarks, a high watermark and a low watermark to decide whether more resources should be requested (from the next node closer to the root in a resource distribution tree) or unused resources should be released.

A resource request is forwarded, if down-stream needed bandwidth meets or exceeds the high watermark. Different strategies are used to determine bandwidth increments using the following parameters: additional bandwidth needed to accept the current request and a given block size which may be dependent on cushion size (larger cushions → lager blocks).

A portion of the unused resources is released, if down-stream needed bandwidth meets or falls below the low watermark. Different strategies are used to determine the portion of unused bandwidth that is released.

2.2.2 Leaky Shares

Parameters:

- a high watermark, given as fraction of up-stream allocated bandwidth
- increment block size (IBS)
- cushion holding time (CHT) and
- decrement block size (DBS)

A resource request is forwarded, if down-stream needed bandwidth meets or exceeds the high watermark. In that case, the maximum of the additional requested BW and the DBS is requested.

Unused bandwidth is released, if the resource cushion meets or exceeds the DBS and does not fall below that value for the CHT. Together, DBS and CHT define a leak rate with which unused bandwidth is lost.

2.3 Simulation and Traffic Model

The simulation model is a flow level (not a packet level) model. It consists of different kind of nodes that can be connected to resource distribution trees. The first kind of nodes, resource management nodes, model the different resource management schemes that are used in AQUILA. They are models for ACAs, RPs, and BGRP agents. With them, resource distribution across hierarchical RPs to intra-domain AC was modeled as well as inter-domain resource distribution across sink trees by BGRP agents. The second kind of nodes are traffic generators that can be connected to resource management nodes. They create resource request and release messages. Resource management nodes process these request and release messages from the traffic generators and may generate further resource request and release mes-

sages that are sent to higher level resource management nodes.

During each simulation we measured the ratio of the number of signaling messages forwarded and received by each resource management node. This ratio is the signaling reduction factor. It is 1 if every request and release is forwarded, e. g. by using BGRP without QG. Furthermore, we measured the bandwidth that was allocated up-streams (towards the root) and down-streams (towards the leaves) by each resource management node. The difference is the resource cushion. It is indicated as fraction of the up-streams allocated bandwidth.

The traffic generators model voice over IP calls requiring a single bandwidth unit u each. A Poisson traffic model was used, i. e. inter-arrival times and holding times were exponentially distributed.

Resource management nodes have got configuration parameters according to the modeled resource management scheme. They are used to regulate the cushion management.

2.4 Results

2.4.1 Scalability and Trade-offs

2.4.1.1 Simulation and Traffic Model

Three nodes were connected in a chain in the following sequence, see Figure 1:

- a traffic generator (TG_1)
- the investigated resource management node (N_1) using a LS
- a root resource management node (N_R) providing a unlimited amount of bandwidth.

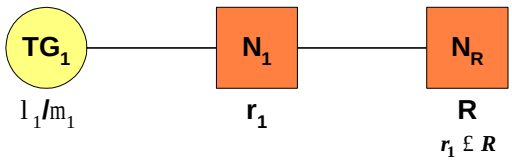


Figure 1 - Simulation Scenario

2.4.1.2 Measurements

SRF and CS over cushion holding time (CHT) for different load scenarios:

- L20: mean load = 20 flows
- L100: mean load = 100 flows
- L200: mean load = 200 flows
- L1000: mean load = 1000 flows

A mean call holding time of 180 sec was used in all cases.

2.4.1.3 Results

Figure 2 shows the measured SRFs on the left graph and CSs on the right graph.

SRF

- improves fast with growing CHT
- depends on aggregation, the better the higher the load: higher aggregation → better SRF

CS

- increases with growing CHT
- there is a trade-off: better SRFs require larger CSs
- depends on aggregation, the better the higher the load: higher aggregation → smaller CS

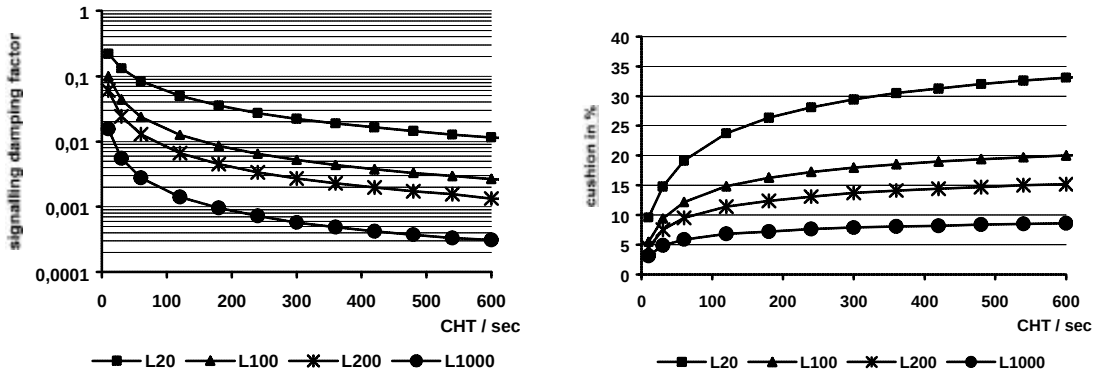


Figure 2: *Measured signaling reduction factors (log scale, left) and resource cushions (right) vs. cushion holding times (CHT) for different loads (L20 to L1000).*

2.4.2 Changing Traffic Loads

In order to stress the resource management:

(i) a limited amount of bandwidth which results in blocking probabilities greater than zero was to distribute between three competing nodes,

(ii) traffic generator with time-variant mean loads produced shifting loads.

Both conditions forced the resource management to continuously adapt resource distribution to changing traffic conditions through resource exchange via the root node.

2.4.2.1 Simulation and Traffic Model

Three nodes share the limited resources of a common root, see Figure 3:

- 3 traffic generator (TG_i)
  - TG_1: mean load = 100 flows
  - TG_2: the following load pattern was repeated until end of simulation:
    - increasing load from 20 to 45 flows on the average during 1 hour, exponential increase by a factor of 1,07 every 5 minutes
    - stationary load with 45 flows on the average for one hour
    - decreasing load to 20 flows on the average during next hour, exponential decrease by a factor of 0,935 every 5 minutes
    - stationary load with 20 flows on the average for 1 hour
  - TG_3: mean load = 20 flows for the first hour, afterwards load followed the same pattern described above for TG_2
  - all traffic generators used a mean call holding time of 180 sec
- 3 edge router performing admission control (AC_i) using a watermark scheme
- a common root resource pool (RP) providing a limited amount of bandwidth: 200 u

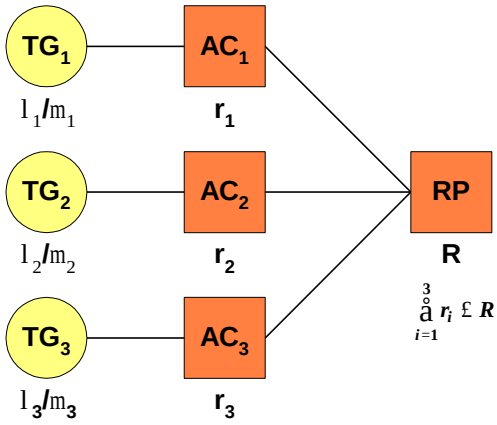


Figure 3 - Simulation Scenario

2.4.2.2 Results

Figure 4 shows offered load, Figure 5 dynamic resource distribution.

Resource distribution was fair with reasonable blocking frequencies:

- AC_1: 1%
- AC_2: 0,9%
- AC_3: 1%

Only 2,6 % of the requests of the three traffic generators arrived at the resource pool RP.

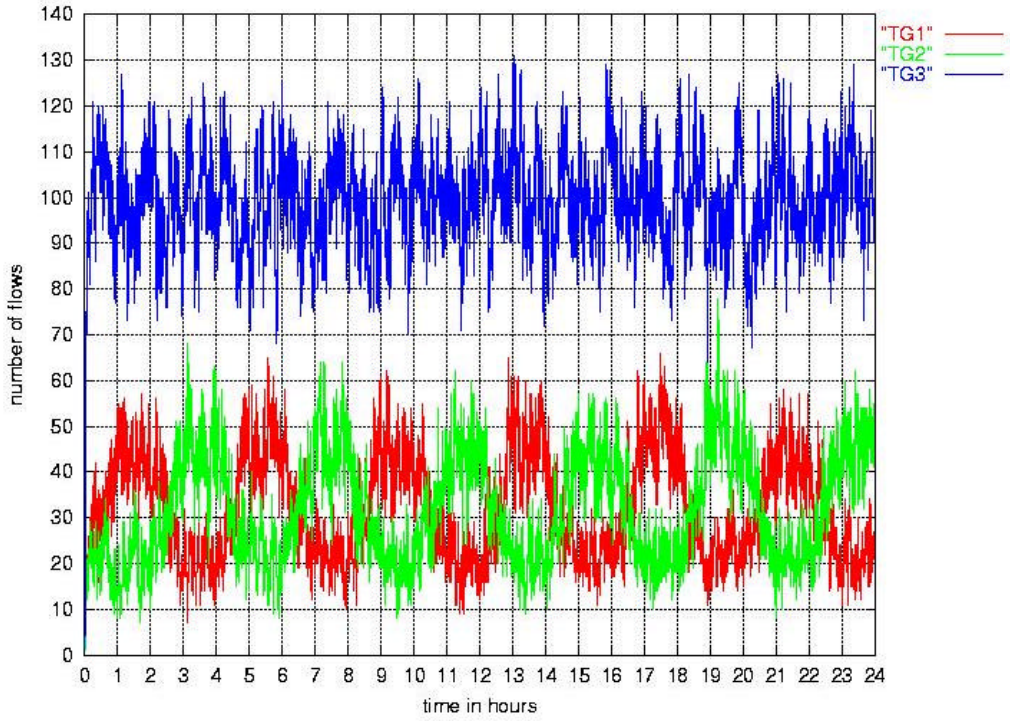


Figure 4 - Offered Load

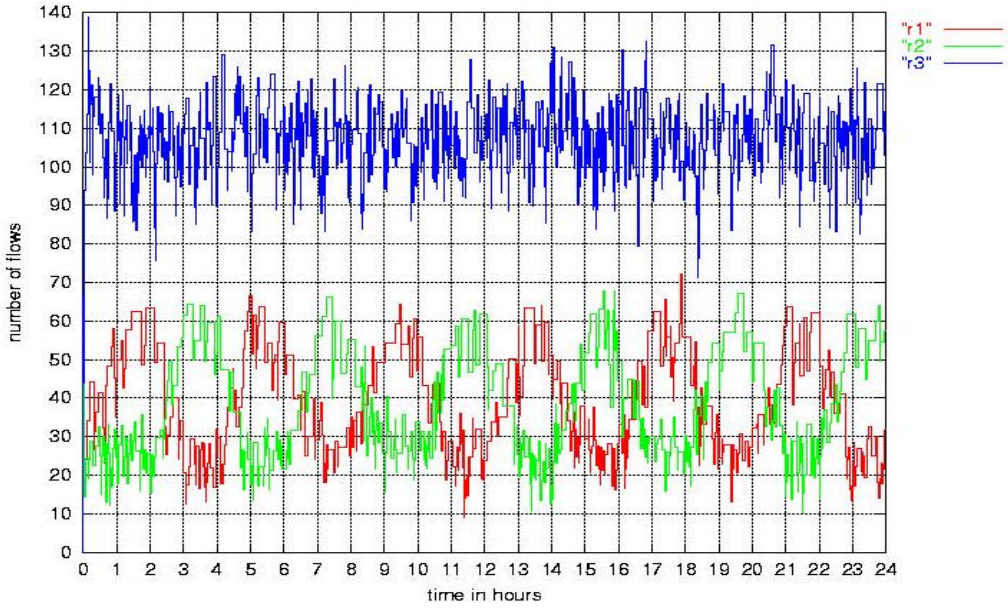


Figure 5 - Resource Allocations

2.4.3 Concatenated Traffic Aggregations

2.4.3.1 Simulation and Traffic Model

Sink-trees of a depth of three: 10 source domains, a common transit domain and a common destination domain, see Figure 6:

- 10 traffic generators, each connected to one of
- 10 egress border routers of source domains, using a LS, commonly connected to
- 1 egress border routers of a common transit domain, using a LS, connected to
- 1 ingress border routers of a common destination domain, providing a unlimited amount of bandwidth.

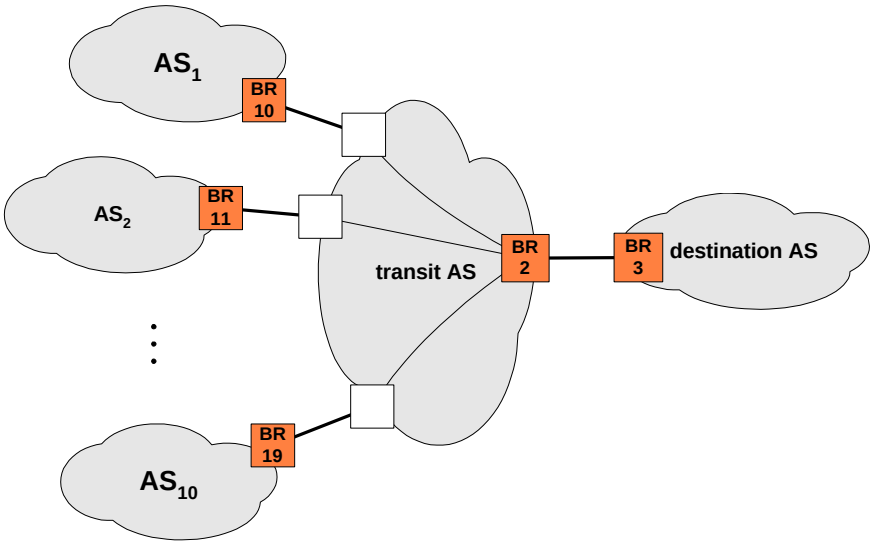


Figure 6 - Sink Tree Scenario.

Large load shifts were applied in order to stress the resource management. Mean load passed 3 phases in each source domain, see Figure 7:

- stationary load, mean = 50 flows
- stationary load, mean = 100 flows
- stationary load, mean = 50 flows

Begin of phase two, which was of equal length for each source domain, was shifted by a quarter of an hour from source domain to source domain.

A mean call holding time of 180 sec was used in all cases.

2.4.3.2 Results

SRF and cushions of source and transit domain nodes were measured.

The border routers of each source domains forwarded about 0,9% of incoming requests, while the border router of the common transit domain passed 2,3% of the received requests, corresponding to about 0,02% of the original requests.

The bandwidth cushions was 16% at the source domains and less than 6% at the transit domain, with 23% in total.

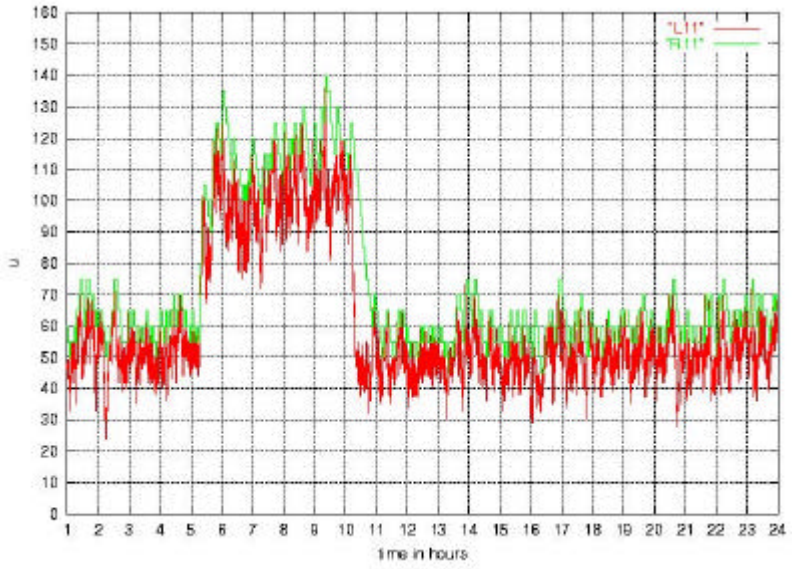


Figure 7 - Offered load and resource allocation of a source domain.

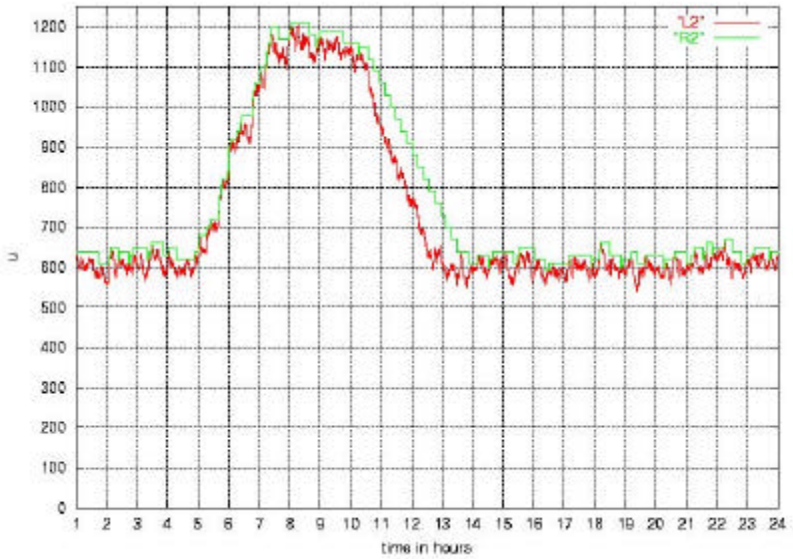


Figure 8 - Offered load and resource allocation of transit domain.

2.5 References

[1] AQUILA consortium, deliverable D1203, "Final system specification", IST-1999-10077-WP1.2-SAG-1203-PU-O/b0, http//www.ist-aquila.org, April 2002

3 Large Scale Scenarios for TCL1 and TCL2

3.1 Simulation Topology

3.1.1 Description of the Simulation Topology

The simulations were realized in a large-scale network topology. Large-scale networks are considered those consisted of a relative large number of interconnected routers, which belong to different networks, as depicted in Figure 1.

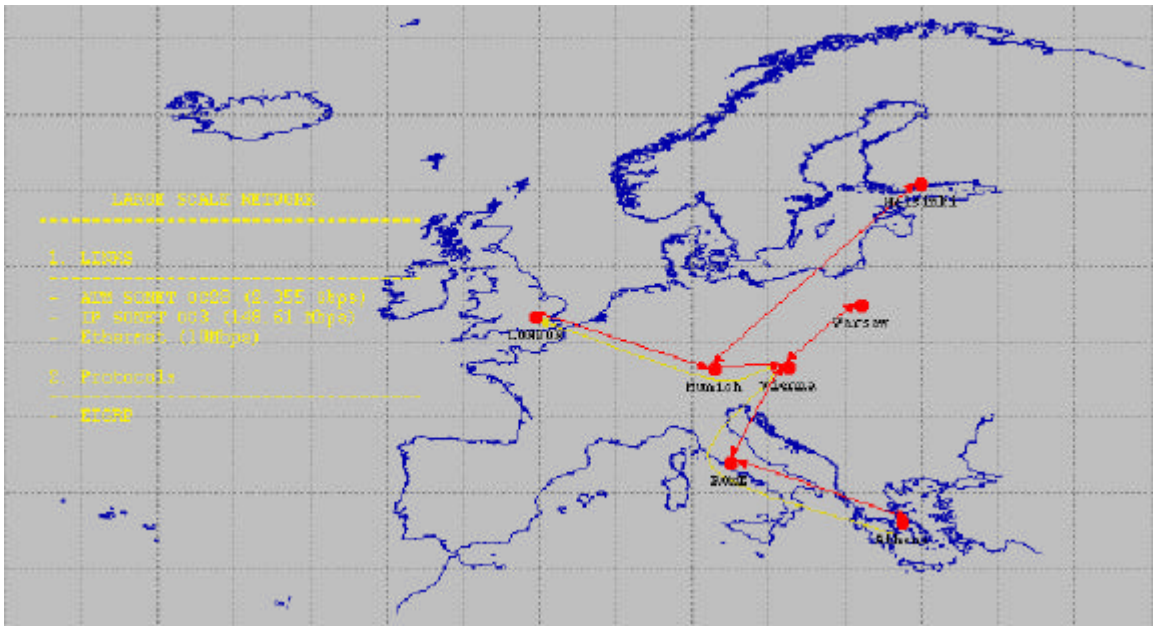


Figure 1 – Large-Scale Network

This topology consists of seven interconnected networks, which belong to seven cities of Europe. Three of them are considered transit networks, which are situated in Munich, Vienna and Rome. All links between the transit networks (between the border routers) are ATM-SONET OC3 with 150Mbps data rate. The links inside each individual network between the border routers and the core routers and between the core routers and the edge routers are considered of SONET OC3 Ethernet links. The traffic generators (hosts) and edge routers are placed in the networks Athens, Warsaw and Helsinki. The access links between the users and the edge routers have a data rate of 10Mbps (see Figure 2). The following figures present the topology for some sub-networks.

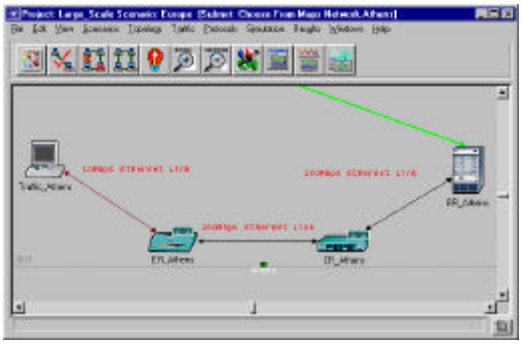


Figure 2 – Athens Sub-Network

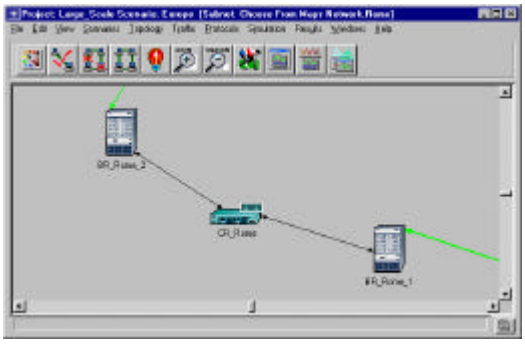


Figure 3 – Rome Sub-Network

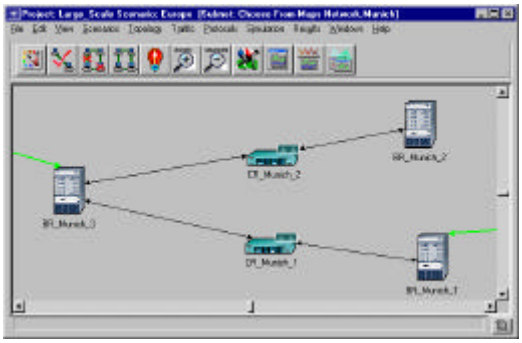


Figure 4 – Munich Sub-Network

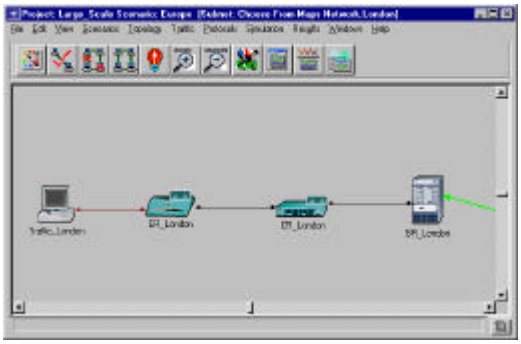


Figure 5 – London Sub-Network

Different paths are considered, with different number of hops, in order to study the impact of both distance and number of hops on the end-to-end delay. The proposed destination network is “London” and is the same for all hosts. The background traffic can be generated in all networks. The path Athens -> Rome -> Vienna -> Munich -> London is consisted of 15 hops, the path Warsaw -> Vienna -> Munich -> London of 12 hops and the path Helsinki -> Munich -> London of 9 hops. We have used in the simulation topology Cisco routers and traffic generators corresponding to applications profiles.

The EIGRP is considered as the routing protocol for the whole network.

3.2 Simulation Scenarios

We have used the referred simulation topology configured with the mechanisms used by the TCL1 and TCL2. We implemented the scheduling algorithm appropriate for these traffic classes (PQ-WFQ). In addition the kind flows used to produce the traffic are very closed to the nature of applications used for the trials. We use the path Athens -> London (this is the longest path and has the maximum number of hops) for our simulation because this is the worst case.

Assuming the 10Mbps access links, the recommended AC limits for each TCL as well as the maximum permitted traffic are configured as following: for TCL1, AC1=10% (1Mbps), for TCL2, AC2=15% (1.5Mbps), for TCL3, AC3=20% (2Mbps), for TCL4, AC4=5% (500kbps) and for TCL-STD is dedicated the rest of the link (5Mbps). Regarding the background traffic, the traffic parameters are depicted in Table 1.

Table 1 – Background Traffic Parameters.

TCL	Bit Rate	Packet Size
BT - TCL1	CBR 1Mbps	125B
BT - TCL2	CBR 1.5Mbps	500B
BT- TCL3	CBR 2Mbps	1000B
BT - TCL4	CBR 500kbps	1000B
STD BE	CBR 5Mbps	1000B

The routers compromising the end-to-end topology are depicted in Figure 6. Background generators are placed in different links and different domains, rising five different bottlenecks in the network. The test duration for all simulations was 200sec. The bottleneck occurs between Edge Routers and Core Routers (Athens, London) or Core Routers and Border Routers (Vienna, Munich and Rome). The Bottleneck links are set to 10Mbps.

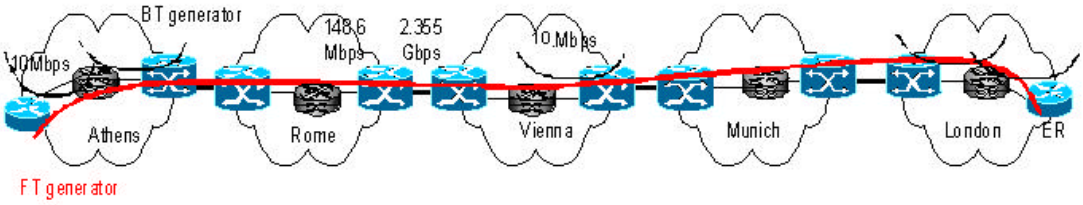


Figure 6 – Path between host and destination.

The PQ-WFQ scheduling scheme is considered with the default weight settings for high bandwidth links, as depicted in Table 2.

Table 2 – PQ-WFQ Configuration.

Traffic Class	Weight	Queue Size (pkts)
TCL1	-	5

TCL2	90%	5
TCL3	3.3%	30
TCL4	3.3%	10
TCL-STD	3.3%	59

3.2.1 TCL1 as foreground traffic

In the first scenario, the TCL1-PCBR is served as foreground traffic using voice flows. The background traffic is consisted of the BT-TCL2, BT-TCL3, BT-TCL4 and STD BE (Table 1). Our purpose is to investigate the QoS parameters of the Premium CBR service that should guarantee low packet delay and packet loss.

The buffers in the routers for TCL1 were set to 5 packets to guarantee low packet delay requirements. The performance of TCL1 was validated assuming target packet loss ratio (P_{Loss}) to be 10^{-2}. According to the specified admission control algorithm the maximum admissible load in this case is ρ=0.685, what is equivalent to 685kbps (for link bandwidth: 10Mbps). Therefore, 10 voice flows of 64kbps are used in the first case and 5 flows of 128kbps in the second. Therefore, the single TB for TCL1 was configured with PR = 64kbps or 128kbps and BSP equal to the packet size.

3.2.2 TCL2 as foreground traffic

In this scenario, the TCL2-PVBR class is served as foreground traffic and video flows are used for studying its performance the end-to-end delay is concerned. The background traffic is consisted of flows as depicted in Table 1.

Assuming that the AC2 is 1.5Mbps and the target packet loss equal to 10^{-4}, the effective bandwidth for each admitted flow is:

- 486kbps, when each flow is characterized by PR=450kbps, SR=250kbps and packet size 500bytes. Therefore, based on the AC algorithm the number of admitted flows is 3, while the effective bandwidth for each admitted flow is 486kbps and $\sum_{i=1}^{3} Eff_i = 1.46Mbps$. The dual TB was consequently configured for each flow, with PR=450kbps, BSP=1000B(2*M), SR=250kbps and BSS=5000B(10*M).

- 237.6kbps, when each flow is characterized by PR=220kbps, SR=125kbps and packet size 500bytes. Therefore, based on the AC algorithm the number of admitted flows is 6, while the effective bandwidth for each admitted flow is 237.6kbps and $\sum_{i=1}^{6} Eff_i = 1.43Mbps$. The dual TB was consequently configured for each flow, with PR=220kbps, BSP=1000B (2*M), SR=125kbps and BSS=5000B(10*M).

3.3 Results

3.3.1 TCL1 – PCBR

3.3.1.1 TCL1: Case 1- 64 kbps bit rate

In the first scenario the end-to-end delay for TCL1 was measured. TCL1 flows are transmitted by 64kbps bit rate, with different number of bottlenecks in the network. Every scenario was repeated several times having different packet sizes for TCL1. The results are depicted in Figure 1.

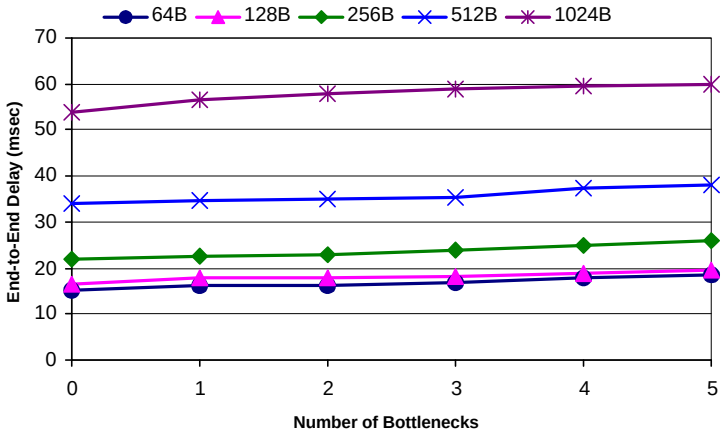


Figure 7 – Average End-to-End Delay of TCL1 flows with 64kbps bit rate.

Moreover, the delay variation of TCL1 flows for each case is shown in the Table 3.

Table 3 – Delay Variation of TCL1 flows with 64kbps bit rate.

Number of Bottlenecks	Packet Size				
	64B	***128B***	***256B***	***512B***	***1024B***
0	1.5E-08	4.3E-08	6.1E-08	1.0E-07	1.8E-06
1	1.9E-08	2.2E-07	2.5E-07	1.6E-07	6.0E-06
2	1.2E-07	4.2E-07	5.0E-07	1.8E-06	1.0E-05
3	1.7E-07	7.6E-07	7.8E-07	2.5E-06	2.1E-05
4	2.0E-07	8.6E-07	9.9E-07	4.0E-06	2.4E-05
5	2.1E-07	9.0E-07	1.3E-06	5.4E-06	2.8E-05

The results show that the end-to-end delay depends on the packet size and it is increased up to four times when the packet size is 1024B. Moreover, the value of end-to-end delay is growing up when background traffic is added in the network but for all cases it is much less than the target value (<<150ms).

3.3.1.2 TCL1: Case 2- 128 kbps bit rate

In this scenario, it was measured the end-to-end delay for TCL1, considering two different packet sizes having a sequentially increasing number of bottlenecks in the network - from one to five. The results can be seen in Figure 8.

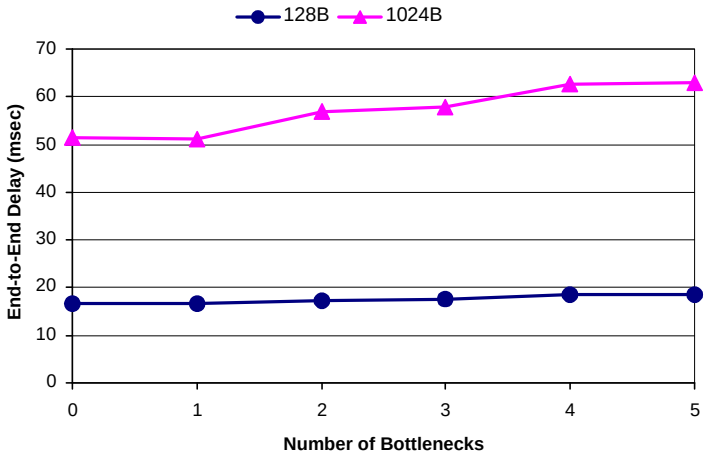


Figure 8 – Average End-to-End Delay of TCL1 flows with 128kbps bit rate.

Moreover, the delay variation of TCL1 flows for each case is shown in the Table 4.

Table 4 – Delay Variation of TCL1 flows with 128kbps bit rate.

Number of Bottlenecks	**Packet Size**	
	128B	***1024B***
0	5.3E-08	6.4E-06
1	3.3E-07	6.6E-06
2	4.9E-07	1.6E-05
3	8.5E-07	2.4E-05
4	9.0E-07	2.9E-05
5	9.4E-07	2.9E-05

The basic conclusion after all those different simulations for TCL1 was that the value of end-to-end delay is influenced very much by the packet size and it is growing up to 4 times when the packet size is 1024B. There is also a main difference in the delay variation; this value is increased up to 200 times when the packet size is 1024B. Therefore TCL1 flows must be characterized by small packet sizes. In addition, increasing the amount of Background Traffic results in increasing the maximum value measured of the end-to-end delay up to 30%. Even though, the end-to-end delay still remains less than 150msec.

3.3.2 TCL2 – PVBR

3.3.2.1 Case 1: 3 Flows x 250kbps bit rate

In this scenario 3 flows of TCL2 have been used transmitting each on average 250kbps. The minimum, maximum and average end-to-end delay for TCL2 with different number of bottlenecks in the network was measured as depicted in Figure 9.

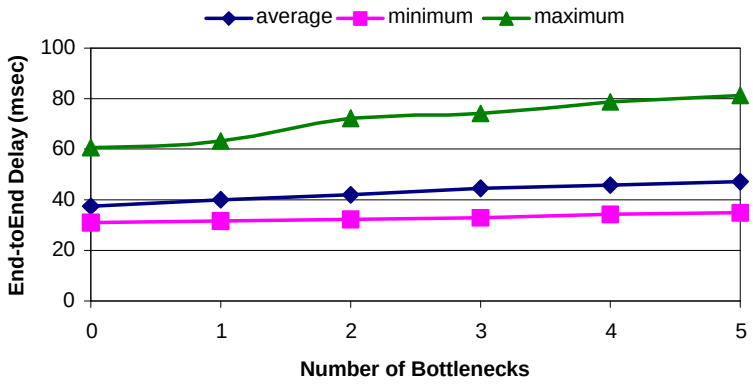


Figure 9 – End-to-End Delay of TCL2 flows for different number of bottlenecks.

In this case, the maximum end-to-end delay is getting bigger when bottleneck is occurred. Nevertheless this value is less than the maximum permitted value (<<250msec).

3.3.2.2 Case 2: 6 Flows x 150kbps bit rate

The same simulation scenarios were used for TCL2 in this section, as in the previous one but having 6 flows of TCL2. The minimum, maximum and average end-to-end delay for TCL2 with 0 up to 5 bottlenecks is depicted in Figure 10.

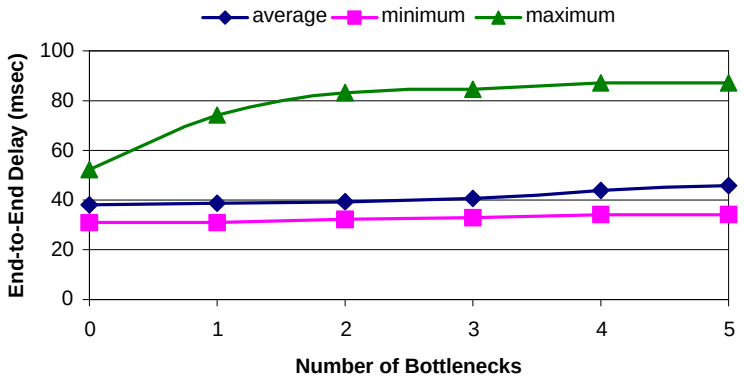


Figure 10 – End-to-End Delay of TCL2 flows for different number of bottlenecks.

As a final remark from the simulations, concerning TCL2, was that increasing the BT injected in the network, has as result the increase of the maximum observed value of end-to-end delay up to two times; though it still remains less than 250msec.

4 Evaluation of the Quiet Grafting Mechanisms

4.1 Requirements

As described in [1], the fundamental concept of Border Gateway Resource Protocol Plus (BGRPP) is the aggregation it performs on destination-based inter-domain reservations and thus resulting to the formation of sink trees. It is therefore evident that pure BGRP [2] addresses the problem of scalability to a certain extent, i.e. it significantly reduces the amount of control state information kept at routers in conjunction with the volume of REFRESH messages needed for the maintenance of the corresponding state. Of interest to us, however, is enhancing BGRP with additional mechanisms so as to achieve a reduction in the number of PROBE and GRAFT messages as well.

As stated in [1], in order a resource reservation request to be successfully established in the network, the PROBE message should be forwarded between BGRPP agents hop-by-hop along the BGP route until it reaches the destination domain. Each BGRPP agent forming the end-to-end path should check the resource availability. The last BGRPP agent, which corresponds to the root of the sink tree, can assign a sink tree identifier to the reservation, which uniquely identifies the sink tree the reservation belongs to. Then, a GRAFT message is generated containing this identifier. Consequently, signalling messages still have to travel the full path from the source to destination increasing the signalling overhead and utilising a significant amount of bandwidth. Aiming at reducing the signalling overhead, the quiet grafting mechanisms are introduced.

The quiet grafting mechanisms should provide an intermediate BGRPP agent with the necessary functionality in order to successfully answer a PROBE message, before the latter arrives at the destination domain. Towards this end, the BGRPP agent must be able to identify the sink tree, to which the reservation belongs and in addition must have pre-reserved resources for this sink tree, so that he can guarantee that resources are available on the path from the current point to the destination domain.

4.2 BGRPP Enhancements

Based on the above requirements, it is essential that a new reservation can be classified earlier to a sink tree and moreover that resources are available for that tree. The most adequate mechanisms for achieving these two goals have been the network layer reachability information (NLRI) labelling of the sink tree and the introduction of an efficient resource management algorithm. The first one will enable an early sink tree identification, i.e. it will classify a new reservation to its corresponding tree. This is mainly due to the fact that the nodes of a sink tree can be aware of the NLRI information of the destination domain, which serves the root of the tree. Therefore, when a new reservation request arrives (PROBE message), a node can verify if its destination is pertained to the domains that correspond to the NLRI label of a sink tree. However, it is not our intention to delve into the mechanisms that distribute this

kind of information since they have already been described in [1]. On the contrary, the effectiveness of the distribution of the NLRI information associated to the root of the tree will be studied.

Moreover, the resource management algorithm will perform hysteresis with respect to delayed bandwidth release so as to maintain available resources while achieving high resource utilization. The concepts of this algorithm will be examined in detail and its performance will be studied when applied to a real network topology.

4.2.1 Hysteresis for the creation of Resource Cushions

Irrespective of the identification of a sink tree, the quiet grafting of a new reservation will not be feasible if there are not enough resources so as to accommodate the new request. Therefore, it is required that BGRPP nodes are assigned with more bandwidth than currently reserved. To accomplish that, over-reservation, quantization and hysteresis techniques are likely to be employed.

The first two approaches are likely to introduce some limitations with respect to their impact on the Quiet Grafting probability and can inadvertently produce the opposite effects to the ones envisioned. Over-reservation, when performed without control, can lead to a reduction of the Quiet Grafting probability. This is due to the fact that future requests, coming from BGRPP nodes other than the ones with over-reserved resources, are likely not to be grafted onto the tree or not to be accommodated at all. Another impact of over-reservation requests can be PROBE messages that travel unnecessarily towards the root of the tree even if the requested resources are available, operating as before at the expense of Quiet Grafting.

Quantisation of the requested resources, i.e. rounding them up to a multiple of a chosen quantum, can lead to undesirable results as well. This is particularly true in the case of sink trees that consist of many leaf nodes (N) and are moreover incapable, due to lack of future reservation requests, of using the remaining amount of the reserved bandwidth. As a result, the root of the sink tree will end up having reserved at least N*quantum resources, far surpassing the actual needs. Thus, the adoption of the quantisation mechanism can only be justified for sink trees with a limited amount of leaves and a much greater amount of reservation requests.

In essence, given the aforementioned concerns coupled with the complexity induced by the adoption of the corresponding techniques, it is proposed that the BGRPP nodes do not perform over-reservation or quantisation upon receiving a new reservation request. Instead, hysteresis on the release of resources is more appropriate for the formation of resource cushions at the BGRPP nodes. The resource cushion mechanism that will be employed bases its resource release policy on the existence of the release period and thresholds. In the next section the corresponding algorithm is described in detail.

4.2.2 Resource Cushion Mechanism

When a BGRPP agent receives a REFRESH message indicating a smaller amount of resources than currently reserved and decides not to further forward this message downstream towards the root of the sink tree, then it allocates resources downstream, which are not in use upstream. These resources, which are reserved downstream but not upstream of a BGRPP agent due to retained REFRESH messages, are called resource cushions.

A resource cushion for a specific BGRP agent is tied to a sink tree. A BGRPP agent may build resource cushions for all of its sink trees. Building resource cushions has as an impact the reduction of the signalling load, since retained REFRESH messages reduce the signalling load of downstream domains. The use of resource cushions for arriving reservation requests further reduces the signaling load of downstream domains, when reservation requests are not forwarded but served from resource cushion immediately.

A BGRPP agent uses two parameters to control the size of its resource cushions. These parameters are:

- RBS: Release block size
- RP: Retain period

Both parameters together define a virtual release rate RR = RBS/RP. Whenever a resource cushion exceeds the RBS, a release timer is activated which runs down during the duration of a single RP. If a resource cushion shrinks to a size below RBS during a running release timer, then the release timer is cancelled. In the case where the release timer runs out without being cancelled, then the corresponding resource cushion is decreased by RBS and a REFRESH message releasing this amount of resources is generated and sent downstream to the next BGRPP agent on the sink tree. Thus, resource cushions are reduced with the release rate RR unless they are used to serve arriving resource requests. RBS and RP are the parameters that control the release rate. In this way released resources are not immediately forwarded towards the sink of the tree, but are used to form resource cushions. Additionally, those retained resources are released step-wise improving in this way the performance of the network.

4.3 Evaluation of the Quiet Grafting Mechanisms

4.3.1 Simulation goals

In this section, the gain of introducing the quiet grafting mechanism In a network will be evaluated, in terms of average hop number reduction that a PROBE (GRAFT) message has to traverse to reach its destination. Therefore, two simulation scenarios, aiming at a dynamic analysis, will be specified, whereas the topology, the traffic model and the parameters of the resource cushion mechanism will be defined.

Quiet grafting aspires to reduce the average number of traversed hops for a signalling message, and consequently the total number of messages in the network. As a result, the perform-

ance of the network with respect to CPU usage and bandwidth used for signaling messages will also be improved.

It is also worth mentioning that there is a trade-off between delayed-release and resource utilization. We are seeking a relation between the gain in terms of reduction of the average number hop traversed, and the utilisation of the reserved resources.

Both simulation scenarios aim at studying and evaluating the efficiency of the quiet grafting mechanism. The first scenario serves as a proof of concept for the early tree identification mechanism evaluating the gain on the message path length reduction.

The second scenario validates the applicability of the resource cushion algorithm and evaluates the performance of the proposed approach.

4.3.2 Scalability Issues

Our main goal is to evaluate the effectiveness of the Quiet Grafting mechanism in terms of the number of hops that a reservation request has to traverse in order to be accommodated into the corresponding sink tree. It is assumed that resources are always available, and therefore the effectiveness of the NLRI information distribution will be solely examined.

The topology for carrying out the simulations has been defined after taking into consideration the actual topology of the Internet. More precisely, the number of the AS domains that an inter-domain reservation can possibly traverse will not exceed the 9. In addition, for each transit AS domain, there will be at least two border routers that will forward the reservation request to the border router of the adjacent AS domain, upstream and downstream. Given the fact that there will be one BGRPP node corresponding to a border router of the AS domain, we can deduce that the path being traversed by a reservation request will consist of a maximum number of 18 BGRPP nodes.

Hence, the simulations performed are based on sink tree topologies that vary in depth (4 to 14) in order that they reflect the actual topology of the Internet. Moreover, in regard to the spreading of the sink trees, additional simulations targeted for the analysis of topologies varying in width (3 to 4 children for each node) have not produced a divergence on the results. Therefore, binary trees have been solely examined since more complex topologies are not considered to be of significant added value with respect to the goal of the simulations.

For identifying the effectiveness of the NLRI information distribution to the nodes of a sink tree, it is essential to examine how the number of populated nodes can contribute to the reduction of the path length. With the term "populated nodes" we denote the BGRPP elements that are aware of the NLRI information of the destination domain (root of the tree) and therefore can identify the destination (sink tree) of the impending reservation request. These nodes are assigned the task of intercepting a reservation request before reaching the root of the tree and silently grafting it onto the sink tree. It would be ideal if every node of a sink tree was populated but obviously, the "number" of populated nodes performing this identification intro-

duces a scalability problem. If this were true, then we would actually bring into scene again the problem that has been tackled by the BGP through the aggregation it performs on routes.

Given a particular sink tree, an "initial state" (number) of populated nodes needs to be created in relation to which, the mean path length of a potential reservation request is computed. This state is produced after performing a certain number of reservations towards the root of the tree. The nodes of the tree (not necessarily leaves) dedicated for initiating those reservations are randomly chosen. All the nodes that are traversed while the reservations make their way up to the root are transformed to populated nodes.

In this way, an initial state of populated nodes can be produced whose topology significantly mitigates the potential degree of homogeneity introduced by the binary tree. Our aim is to compute the impact of this state (number of populated nodes) on the average number of nodes that a reservation request has to traverse before striking a populated node. To that end, a certain amount of additional reservation requests needs to take place.

As before, randomly selected nodes initiate those reservations, which do not modify the initial state of populated nodes, i.e. they do not produce additional populated nodes, as is the case with the creation of the "initial state". Therefore, a mean value of the number of nodes traversed before striking populated one can be computed for a particular populated state.

4.3.2.1 Simulation Results

The simulations were carried out for a variety of binary sink trees and a variety of “initial states” (number of populated nodes) for each tree. Nonetheless, we have chosen to demonstrate how a particular sink tree of *depth 10* behaves to the augmentation of the populated nodes since it is considered as the most representative for an inter-domain reservation (4 transit AS domains). It can be seen from Figure 1 and Figure 2 how the number of populated nodes, which comprises a relatively low percentage (4%-20%) of the total number of nodes (2047), affects the average hop count of a PROBE (GRAFT) message. Having in mind that for a non-populated tree, where quiet grafting is not activated, the number of nodes traversed is equal to its depth, i.e. 10, it can be deduced that the percentage corresponding to the reduction of the actual hop count attains the values of around 47% to 77% for remarkably small percentages of populated nodes.

However, it can be seen from Figure 3, that in the case of sink trees of small depth (1 to 3 transit domains), the percentage reduction of the path length does not attain the same values. In fact, a comparison between 5 sink trees with depth values of 4, 6, 8, 10 and 12 is presented in terms of the path reduction for the same percentage of populated nodes. It is evident that the sink trees of small depth do not exhibit the same effectiveness on the reduction of the path length for a certain percentage of populated nodes. For example, a sink tree of depth 6 with 20% populated nodes will attain a 63% reduction of the path length whereas a sink tree of depth 12, for the same percentage of populated nodes, will achieve an 80% reduction of the path length.

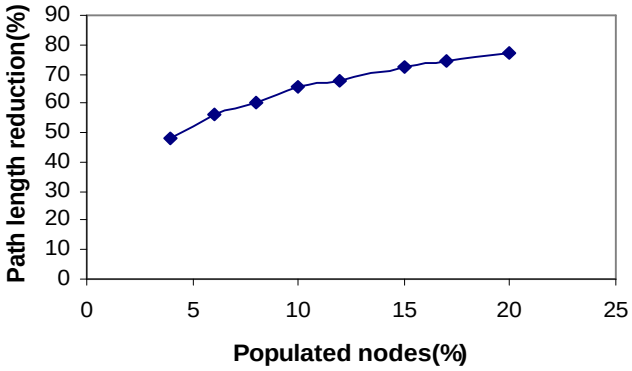


Figure 1 - Path length reduction vs. populated nodes

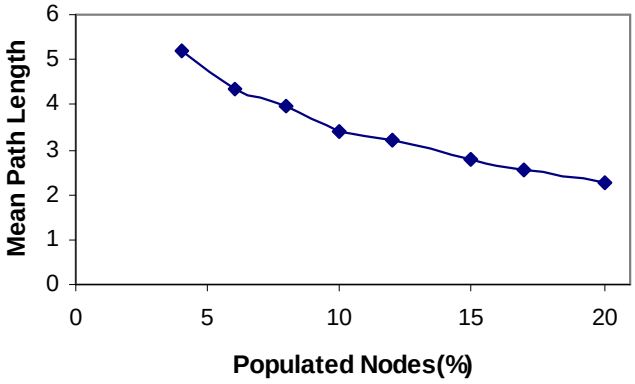


Figure 2 - Mean path length vs. populated nodes

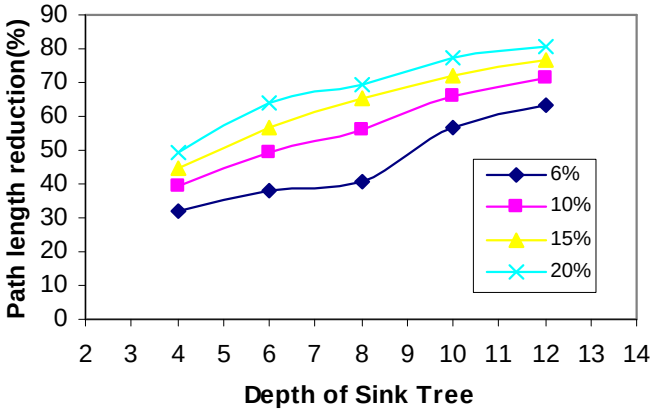


Figure 3 - Path length reduction for different sink trees

4.3.3 Performance evaluation of resource cushion mechanism

In order to evaluate the proposed quiet grafting mechanism, the resource cushion algorithm technique and the ability of early sink tree identification are used in conjunction. It is assumed that every BGRPP agent of the sink tree can identify the destination domain, and therefore can execute the following step; the BGRPP agent will check, whether it has enough resources to accommodate a new reservation request. If there are sufficient resources, the PROBE message will be terminated and a GRAFT message towards the corresponding source will be generated.

For that purpose, a simulation scenario is specified, which is based on the realisation of a sink tree reflecting a real network that is consisted of a number of inter-connected DiffServ domains. In fact, a simple binary and complete tree is proposed, since the properties of symmetry simplify the interpretation of the results. Each node of the sink tree will be a BGRPP agent capable of performing quiet grafting.

In order to have a relatively large number of nodes, while keeping at the same time complexity at a low level, the depth of the tree for this simulation scenario was chosen to be equal to 4. That results in a total number of N = 31 nodes. This scenario could represent a real network topology, consisting of a number of source domains, the destination domain and the transit domains in order to form the binary tree as depicted in Figure 4. We have also made the assumption that traffic is injected into the network from the leaves of the tree, which correspond to the source domains. Therefore, reservation requests are only generated at the edges of the network topology.

Concerning the traffic model used for the simulation scenario, a traffic generator that generates reservation requests with exponential distributed inter-arrival times and exponential distributed holding times is attached to nodes belonging to source domains. As regards the size of each reservation request, for the sake of simplicity, a homogeneous scenario is assumed where all the reservations have the same fixed size. It is assumed, without loss of generality that each request asks for one unit of bandwidth, since an infinitive bandwidth capacity is presumed for every node of the tree. Two different scenarios are considered with different traffic loads, in order to examine the effect of load on the performance of the quiet grafting mechanism. Therefore, two traffic conditions are examined with 20 and 100 flows average accordingly.

The simulation scenario implies that in an initially “reservation free” tree, traffic flows will be generated at the leaves of the tree contemporarily, following the traffic pattern previously described. During the initial phase, each reservation request (PROBE message) has to travel all the way up to the root of the tree in order that resources are reserved. After the initial phase, sequential requests can be potentially accommodated from the already reserved resources, due to the resource cushion algorithm given the fact that the sink tree has been successfully early identified.

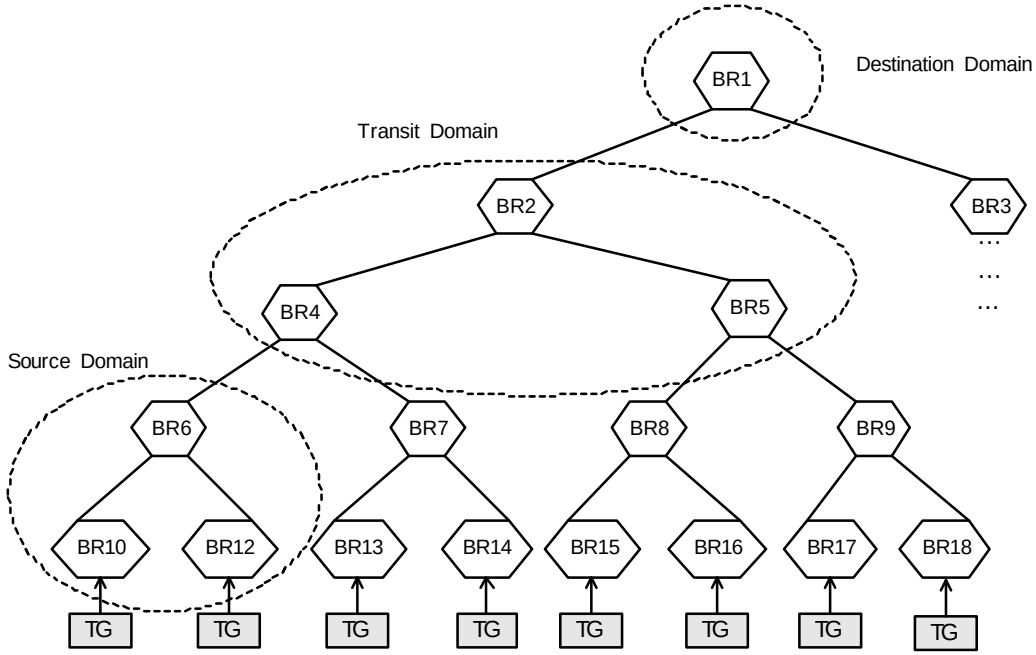


Figure 4 - The reference sink tree

Two performance measures were used to rate quiet grafting mechanism performance: average signalling overhead and average resource utilisation for the whole sink tree.

The average signalling overhead *S* for a sink tree with a number of *n* nodes, is defined as:

$$S = \frac{\sum_{j=1}^{n} r_{out}(j)}{\sum_{j=1}^{n} r_{in}(j)} * 100\,\% \qquad \textbf{eq. 1}$$

where $r_{in}(j)$ is the number of received requests of node *j* and $r_{out}(j)$ is the number of forwarded requests of node *j*. The signalling overhead indicates the percentage of received resource reservation requests (signalling messages) forwarded by the nodes of the sink tree.

The average utilisation *r* for the whole tree is defined as:

$$r = \frac{\sum_{j=1}^{n} R_{reserved}}{\sum_{j=1}^{n} R_{assigned}} * 100\,\% \qquad \textbf{eq. 2}$$

In other words, it comprises the average utilisation of the sink tree, which is defined as the bandwidth used for the active reservations ($R_{reserved}$) divided by the total amount of bandwidth assigned to the created sink tree ($R_{assigned}$).

4.3.3.1 Simulation Results

The objective of the simulations is to present the effectiveness of the introduced resource cushion mechanism on reducing the number of messages processed by each node, limiting in this way the number of signalling messages, and on improving the accomplished performance.

Based on the realised resource cushion mechanism, there are two parameters, which need to be appropriately tuned for achieving the desired network performance: the RBS and the RP. The guidelines for setting these parameters compromises a trade-off between achieved resource utilisation and signalling overhead. The results concerning the overall utilisation and signalling overhead of the sink tree in relation to RP and RBS are shown in the following figures (all simulation scenarios have lasted 10 hours).

Figure 5 and Figure 6 present the effect of the release period RP on the performance of the resource cushion algorithm. As expected, the overall signalling overhead percentage decreases, respectively to the increase of the release period. By increasing the RP parameter (which represents the time during which the free resources should exceed the RBS), resources are released less frequent. Longer retained resources may accommodate future requests, limiting in this way the number of requests forwarded to the next nodes. On the other hand, the increase of the RP has a negative impact on the average utilisation of the sink tree, since the resource status is checked less frequently, resulting in longer intervals between subsequently releases of resources. Notice that under heavy load conditions the signalling overhead is reduced to 2,2171%.

In Figure 7 and Figure 8, we can observe the impact of RBS parameter on the performance of the quiet grafting mechanism. As it can be seen, the effect of the RBS is not similar to the one of RP. We can observe that there is a maximum of request forwarding activity between small and large values of RBS. Moreover, the randomly changing but stationary load provokes a reallocation of the released bandwidth, which results in request forwarding.

Notice that there is an optimum RBS that follows changing demand best. We have to stress here that lower RBSs do not decrease allocated bandwidth fast enough, while larger RBSs cannot follow small temporary demand changes. With a very small RBS, resources are not released fast enough to follow the changing bandwidth demand. Moreover, more requests are forwarded with higher RBS values, because releases follow demand closer. Nevertheless, large RBS values do not allow following small demand changes. This decreases the number of forwarded requests to lower levels again beyond a certain RBS value.

This is justifiable if we consider that while the RBS is increased (but still gets small values), a greater amount of resources is released concluding in a higher level of utilisation. Moreover, since the release resources procedure is more frequently performed, a smaller amount of resources are retained for accommodating future requests resulting in a higher signalling overhead. Nevertheless, as the RBS rises up to really great values, the possibility of accumulating that amount of free resources declines. This significantly impacts the utilisation level, particu-

larly under low load conditions. Accordingly, it is anticipated that the signalling overhead will be reduced.

Concluding, it is obvious from the presented results that the algorithm's performance is much more sensitive to the RP than to the RBS parameter. Moreover, the algorithm's performance is significantly improved under heavy load conditions.

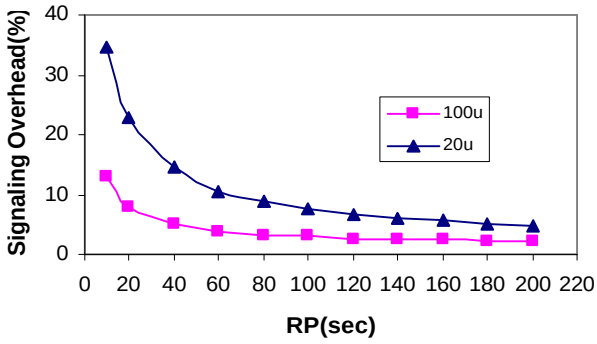


Figure 5 - Signalling Overhead vs. RP

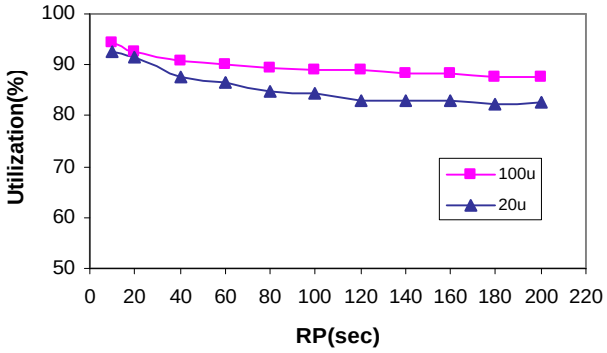


Figure 6 - Utilisation vs. RP

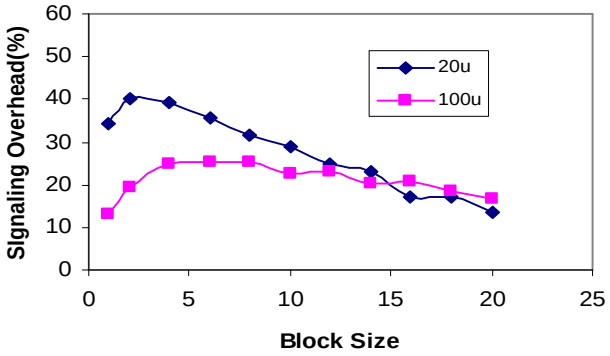


Figure 7 - Signalling Overhead vs. RBS

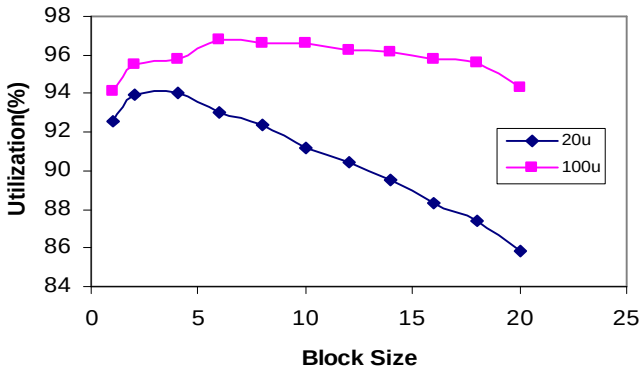


Figure 8 - Utilisation vs. RBS

4.4 References

[1] S. Salsano, V. Genova, F. Ricciato, M. Winter, B. F. Koch, L. Dimopoulou, E. Nikolouzou, P. Sampatakos, I.S. Venieris, G. Eichler, "Inter-domain QoS Signaling: the BGRP Plus Architecture", Internet Draft, May 2002.

[2] P. Pan, E. Hahne, H. Schulzrinne: "BGRP: Sink-Tree-Based Aggregation for Inter-Domain Reservations", Journal of Communications and Networks, Vol. 2, No. 2, June 2000, pp. 157-167, http://www.cs.columbia.edu/~pingpan/papers/bgrp.pdf.

5 A rate controller for long-lived TCP-flows[1]

The focus of this chapter is on the topic of rate assurance for TCP flows. Given that TCP is the number one transport protocol [5] in today's Internet, we believe that such a service could be of interest. The topic of assuring TCP rates has been investigated in several other publications, e.g. [5]-[7]. The goal of this work is to develop a traffic conditioning mechanism that can be used to assure a certain level of goodput to long-lived TCP flows. The proposed conditioner could be used in a DiffServ network to enable such a service class. We propose a TCP rate controller (TRC) that regulates the goodput of a TCP flow by controlling packet drops and the round trip time (RTT) of the flow's packets. The TRC is based on a model of TCP sending behaviour.

5.1 Related Work

The Capped Leaky Bucket (CLB) as proposed in [7] is an improved Leaky Bucket traffic conditioner. To take the behaviour of TCP into account it is tried to estimate the *RTT* by measuring the time between two bursts. If the input rate is higher than the target rate one packet each two *RTTs* is marked as out-profile. Simulations in [7] show performance that is not appropriate to give assurances and a bias against big reservations.

In [9] equations how to set the parameters of a token bucket marker for achieving a requested rate are proposed. The parameter setting depends on the requested rate (R_{req}), drop probability of out-profile packets (p_2) and the *RTT*. With the equations in [9] it is possible to make correct goodput assumptions for a known value of *RTT* and p_2 . But the crucial aspect in the application of this model is that the *RTT* and p_2 are not constant for different connections and also strongly vary over time due to changes in the level of congestion.

We therefore believe that it is very difficult, if not impossible, to assure TCP rates by employing a token bucket marker that is configured with a *static* parameter set. A very interesting adaptive marking algorithm has been proposed recently in [10].

5.2 Network Environment

The TCP rate controller essentially requires some conditions from the network. The first aspect is that at the network edge the TRC must exclusively control single long-lived TCP flows that are not multiplexed with a different kind of traffic (e.g. short-lived TCP flows or UDP flows). Second, some kind of admission control framework must ensure that the sum of requested rates over all accepted reservations is in fact available. This condition provides an

[1] The work presented here has been originally published in: *Lecture Notes Computer Science, Vol. 2515, Boavida, F. (Eds.): IDMS/PROMS 2002,* © Springer-Verlag Berlin Heidelberg. Springer-Verlag has generously provided permission to include the work for the deliverable at hand.

over-provisioned network. Further the used code-point can be different in different domains. In the core network the traffic can be multiplexed with any kind of traffic, but it has to be ensured that drop probability (p) is zero and the network RTT (RTT_{net}) is known and nearly constant. The TRC does not need any special queue management mechanism, because no packet should be dropped in the network. It has to be established that a few packets can be buffered in the queue. The receiver window has to be larger than the maximum congestion window (W_{max}) otherwise the achieved rate will be controlled by the receiver and not by the TRC. Further also TCP's Slow Start threshold (*ssthresh*) should be larger than W_{max} otherwise the performance during Slow Start will be worse. The TRC has to be placed at the ingress point of the network. Rate controlling has to be done on a per-flow basis, therefore every flow has to be extracted from an aggregate. Only one TRC can be applied to one flow, because if two TRCs are working on the same flow two times the packets needed to control the flow are dropped.

It has to be ensured that from the receiver to the sender there is no congestion, because RTT_{net} is assumed to be small and constant. Consequently also ACKs have to be marked with the same code-point than packets from the sender to the receiver.

5.3 TCP rate controller

5.3.1 Goal of the TRC

The goal of the TRC is it to provide the TCP sender with a goodput that was requested (by some means of QoS request) in a prior step. The TRC tries to achieve that goal by imposing well directed drops and delays on the flow's packets. The choice of drop probability p and (artificial) delay RTT_{TRC} is based on an analytical model of TCP sending behaviour. We performed our study using the well-known ns-2 simulator, version 2.1b6, and realised that existing TCP models [2] [3] do not accurately predict the sending behaviour of the TCP SACK implementation we used. In order to exclude errors in the TCP model and to focus on the feasibility of the TRC approach itself, we derived our own model which describes the TCP sending behaviour by equations eq. 1-eq. 4.

$$time\ per\ cycle = \frac{5}{2} + \frac{W_{\max}}{2} \qquad \textbf{eq. 1}$$

$$data\ per\ cycle = \frac{1}{p} = \frac{3}{8} * W_{\max}{}^{2} + \frac{5}{4} * W_{\max} - 2 \qquad \textbf{eq. 2}$$

$$W_{avg} = \frac{data\ per\ cycle}{time\ per\ cycle} = \frac{\frac{3}{8} * W_{\max}{}^{2} + \frac{5}{4} * W_{\max} - 2}{\frac{5}{2} + \frac{W_{\max}}{2}} \qquad \textbf{eq. 3}$$

$$BW = \frac{W_{avg} * MSS}{RTT} \qquad \textbf{eq. 4}$$

Thus the sending rate (*BW*) of the TCP sender depends on the average congestion window (W_{avg}) multiplied with the maximum segment size (*MSS*) divided by the *RTT*. The TCP model does not need to take into account timeouts, because losses are exclusively controlled by the TRC and do not force any timeout. From Equation eq. 2 and eq. 3 it is obvious that W_{max} and thus W_{avg} only depend on p and that *RTT* does not influence W_{max} .

```
For each packet arrival:
    if (packet since drop >= 1/p + E)
        drop packet
    else
        delay packet for RTT_TRC
```

Figure 1 – Pseudo code for each packet arrival

5.3.2 Principal idea of the TCP rate controller

Now, the basic idea of the TRC is that by controlling the amount of dropped packets and the *RTT*, the rate of the TCP flow can be pruned to the requested rate. The simple algorithm that has to be executed upon each packet arrival is shown in pseudo code in Figure 1. The TRC drops packets at the network ingress and thereby enforces the rate of the TCP flow to oscillate around the requested level. Consequently, assuming proper functioning of the resource control framework, TRC-controlled flows experience no sustained congestion but merely small and transient queuing delays inside the network. Therefore, the *RTT* is mainly comprised by RTT_{net} and can thus be well estimated. Besides dropping, the TRC can add an (artificial) delay RTT_{TRC} in order to control the achieved rate of the TCP flow. The total *RTT* can then be approximated as the sum of RTT_{net} and RTT_{TRC} .

Consequently for a known value of RTT_{net} the TRC exclusively controls *p* and *RTT*. Based on the underlying TCP model the TRC is thus able to make correct assumptions of the achieved rate of a TCP connection. For a given requested rate there exist several combinations of *p* and *RTT* which achieve the same rate. The tradeoffs in the choice of the two parameters are discussed in Section 5.3.3. The term *E* in Figure 1 is used to compensate the drop in the last cycle and has a value of 1. The TRC can however be equally operated in ECN mode which means that packets are not dropped but marked instead. In that case, the term *E* has a value of 0.

5.3.3 Tradeoffs in Parameter setting for each request

It has to be ensured that after a drop there are enough packets in the network to receive three duplicate acknowledgements and trigger further packet transmissions during fast retransmit. Therefore the maximum window W_{max} has to be at least 5 packets (1 loss, 3 duplicate ACK, 1 to trigger further transmissions). Clearly, the accuracy of the TRC is mainly influenced by the ability of the TCP model to accurately predict the flow's sending behaviour. The deviation between the model's prediction and the real sending rate increases with the drop probability. A W_{max} of 5 corresponds to a drop probability of 0.0735; we have seen that any *p* greater than this value results in unacceptably large deviations from the model. Even for some values smaller than 0.0735 (and thus W_{max} larger than 5) we noticed significant deviations in the ns-2

simulations. We tried to find values for p such that the corresponding W_{max} achieves at least the rate that's estimated by the model. We found that if W_{max} is set as an even number plus 0.5 this condition is fulfilled and that the achieved rate is at most 3% higher than estimated.

The second parameter which can be tuned is the RTT_{TRC} and consequently the *RTT*. As explained above there exist a lot of combinations of setting p and *RTT* to achieve a requested rate. One choice would be to fix p so that W_{max} is 6.5 and enable different requested rates by imposing different packet delays inside the TRC. This would mean that the greater the requested rate the smaller the *RTT* would be. For big requests this could lead to the problem that i) small deviations in RTT_{net} have a significant influence on the achieved goodput see Section 5.4.4 for details or ii) that RTT_{net} is even greater than the total *RTT* should be. To avoid this, a lower bound for the *RTT*, called RTT_{min} , has to be fixed. If the required *RTT* (RTT_{req}) is smaller than RTT_{min} the next greater value of W_{max} , i.e. W_{max} + 2 as discussed above, has to be used. To impose a delay of RTT_{TRC} , packets have to be stored in a buffer. The greater the delay for the same rate the greater must the buffer be. Thus on the one hand the delay should be kept high to keep the influence of a deviation the *RTT* small; on the other hand the delay should be kept small to keep buffer requirements low. This is a tradeoff that an operator must take into account when choosing *RTT* and p for a requested rate. It will be further discussed in the next section. Figure 2 shows the pseudo code of the algorithm that computes p and RTT_{TRC} upon each request.

```
For each request
     set RTT_min min(RTT*_min,RTT_net)
     calculate RTT_req for W_max=6.5
     if RTT_req > RTT_min
          set drop probability to achieve W_max of 6.5
          set RTT_TRC to (RTT-RTT_net)
     else
          set RTT to RTT_min
          calculate an appropriate W_max
          recalculate RTT based on appropriate W_max
          set drop probability to achieve appropriate W_max
          set RTT_TRC to (RTT-RTT_net)
```

Figure 2 – Pseudo code for parameter computation

5.3.4 Tradeoffs in network parameter configuration

An operator has to provide two parameters for the initial configuration of the TRC. One is called RTT_{dev} and denotes an operator's estimate on the maximum deviation between RTT_{net} and the real RTT. The second one is called $gput_{error}$ which is the error in the achieved goodput that should be compensated by the TRC. In order to be on the safe side, the rate that is requested by the user is increased by $gput_{error}$ +1 percent. The smaller RTT_{dev}, the smaller is RTT_{TRC} and thus the smaller the buffer can be. On the other hand, if RTT_{dev} is high, this requires a large buffer space due to the high RTT_{TRC} .

Due to the bursty sending behaviour of TCP sources a few packets will be queued at the bottleneck leading to a slight variance in *RTT*. This occurs especially in high load scenarios. Consequently, RTT_{dev} can generally not be zero. Equation eq. 5 can be used for determining

RTT^*_{min} . As an example assume that a 5% error in the achieved rate is taken into account for TRC configuration and the error in the *RTT* estimation is not greater than 10ms. In that case RTT^*_{min} would be 200ms. Throughout the rest of the paper this value is used for RTT^*_{min} .

$$RTT^*_{\min} = \frac{RTT_{dev}}{gput_{error}} \qquad \textbf{eq. 5}$$

Thus lower bounds for drop probability and for delay are fixed. The other aspect which has to be taken under consideration is the buffer size needed by each connection. The higher the requested rate and the higher the delay the higher the demand of buffer is. The buffer has to be able to store ceil(W_{max}) packets from each flow. To keep RTT_{dev} small not the whole available bandwidth can be sold. We propose that the network is over-provisioned by some amount. In our case using ns-2 it should be over-provisioned by at least 6%. Further the maximum error of the TCP model which is 3% has to be taken into account. To compensate a RTT_{dev} of 5% the achieved goodput is increased by 6%. The sum of requested rates has to be smaller than φ*BW. Where φ for the above case must not be greater than 0.85 (6% over-provisioned, 3% TCP model error and 6% compensate RTT_{dev}). In the core network the queue has to be able to store at least one half of the maximum window of the largest possible request. Because during Slow Start packets at the queue are arriving in a burst with a rate that is twice the bandwidth of the bottleneck. Further a few packets have to be stored when a burst of packets from two or more connections arrives at the same time. It is proposed that the buffer size is 50 packets plus one half of the largest W_{max} . This is a topic of further research.

Figure 3 shows the pseudo code of the initial parameterisation of the TRC done by each operator. Figure 4 shows examples how to set p and RTT_{TRC} for R_{req} between 50kbps and 600kbps. RTT_{net} is assumed to be zero and RTT_{dev} is assumed to be 10ms. RTT_{min} is set to 200ms.

```
For each network:
     Set RTT*min to RTTdev/ gputerror
     Set rate_increase to gputerror+1%
```

Figure 3 – Pseudo code for initial configuration

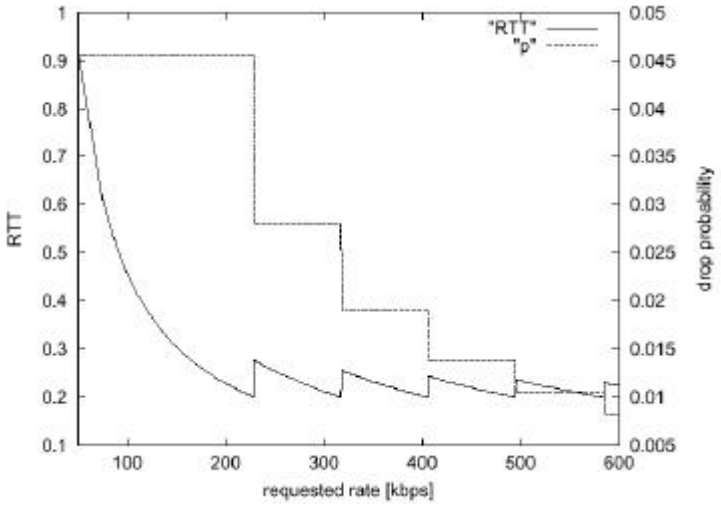


Figure 4 – Exemplary setting of p and RTT_{TRC} based on R_{req}

5.3.5 Control of Slow Start

Despite the TRC is designed for long-lived TCP flows the Slow Start phase can not be neglected in general. We try to control the Slow Start phase such that the achieved goodput is not significantly lower or higher than during congestion avoidance. At the beginning of Slow Start the window is small. This results in low goodput which has influence on the overall achieved goodput. If the connection stays too long in Slow Start the window will increase significantly above W_{max} . This will have the effect that there is more traffic in the network than estimated and packets of all other TRC-controlled connections might be dropped. Dropping packets inside the network would destroy the whole concept because drops would then be no longer exclusively controlled by the TRC.

Consequently an appropriate parameter setting for Slow Start would be to control packet losses so that the window does not exceed W_{max}. The RTT has to be controlled such that as long as the congestion window is smaller than W_{max}=2 the TCP sender sends with a constant rate which equals the requested rate. Analysing the window behaviour during Slow Start shows that when floor(W_{max}) packets are forwarded and then one packet is dropped the window will be ceil(W_{max}) when the loss is detected. Therefore the buffer for delaying packets has to be able to store ceil(W_{max}) packets. If both algorithms work adequately then the Slow Start phase does not significantly influence a connection's performance. This theoretical approach has some restrictions: first, it is not possible to reduce the *RTT* below RTT_{net} and for small windows it may thus be impossible to achieve the requested rate. Nevertheless this approach of controlling the Slow Start phase is superior to an approach where the Slow Start phase is not particularly taken care of. Second, the influence of the retransmission timeout (*RTO*) during Slow Start has to be evaluated, because the RTT_{TRC} is increased over time. And thus during Slow Start phase a *RTO* may occur yielding to a retransmit and a congestion window of one segment. Consequently the rate during Slow Start will be smaller than estimated and thus may influence the overall goodput. This is left for further study. When Slow Start is taken into account the code executed for each packet arrival is slightly modified and can be found in Figure 5.

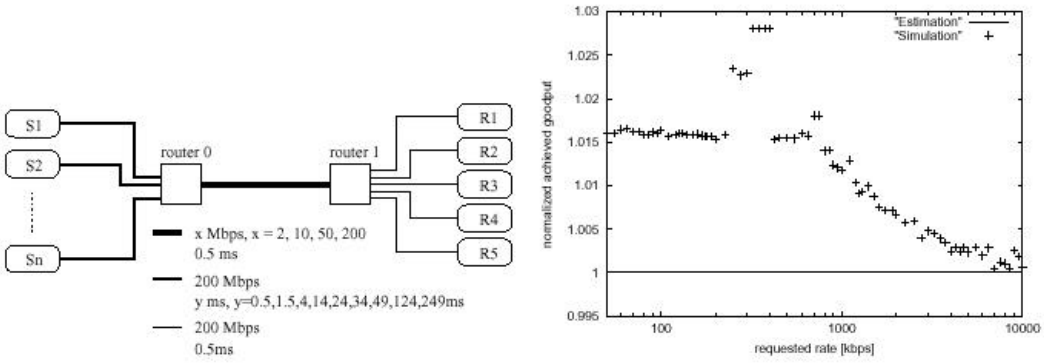
```
init_slowstart=1;
For each packet arrival:
     if init_slowstart //connection is in early Slow Start phase

          if packets since drop >= Wmax
                drop apcket
                init_slowstart=0
          else
                queue packet to achieve requested rate
     else
          code like during congestion avoidance
```

Figure 5 – Pseudo code of TRC with Slow Start

5.4 Simulation study

5.4.1 Simulation Topology

The behaviour of the TCP rate controller is studied by means of network simulations using ns-2 [1]. The simulations were run on a simple dump-bell topology shown in Figure 6(a). FTP senders start sending data from hosts S1-Sn at a random point of time within the first 10 seconds to hosts R1-R5. The bottleneck bandwidth, access delay and requested rate of the TCP senders are varied for the different scenarios to evaluate different effects. Both routers are using a DropTail queue. Traffic conditioning is done by the TRC on a per-flow basis at each sending host. Unless noted otherwise, simulations last for 1100 seconds where the first 100 seconds are not taken into consideration for the statistics. The purpose of the simulation study is it to investigate over a broad range of scenarios to what extent the TRC is able to provide TCP rate assurances. Especially the effect of a mixture of different *RTTs* and requested rates under maximum load is analysed. All simulations are run with and without ECN showing equal results.



(a) Simulation topology

(b) Simulation results for exact parameter setting

Figure 6 – Simulation topology and first results

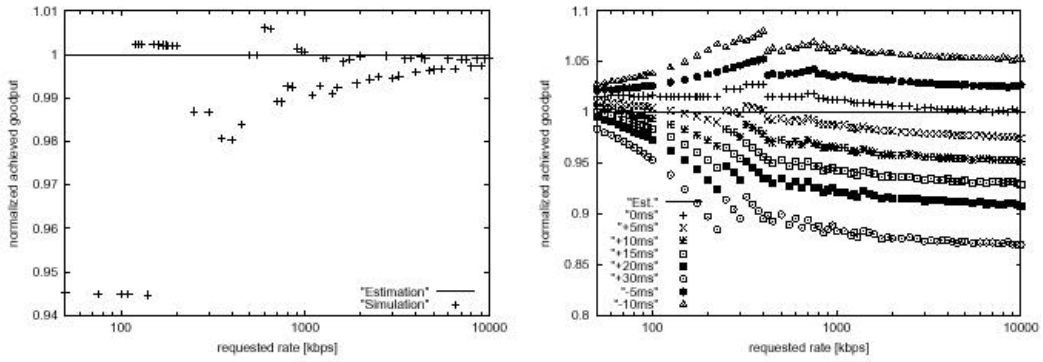
5.4.2 Performance evaluation for $RTT_{dev} = 0$

In this section we demonstrate that for an exact knowledge of RTT_{net} the TRC is able to give goodput assurances for a wide range of requested rates. Simulations are run on a 200Mbps bottleneck link so that queuing delay can be neglected. All links have a delay of 0.5ms consequently RTT_{net} is 3ms. The requested rates vary between $50 – 10^4$ kbps.

Figure 6(b) shows the normalised achieved goodput over the requested rate. Each flow achieves at least the requested rate. The maximum deviation between simulation and estimation is about 3%.

5.4.3 Slow Start behaviour

This section provides simulation results for evaluation of the mechanism introduced to control the Slow Start phase. Simulations are run on the topology shown in Figure 6(a) where all links have a capacity of 200Mbps and a delay of 0.5ms. Simulations are run until the second drop is detected by the sending host. This is chosen for two main reasons. On the one hand the transition from control of Slow Start to Congestion Avoidance should be shown. On the other hand if simulations were stopped after Slow Start only a few packets were transmitted. There are three packets (triggering 3 dupACKs) that have already arrived at the receiver but were not taken into account for statistics, because they are not acknowledged. These packets have great influence on the achieved goodput. The more packets are transmitted the smaller the influence will be. Figure 7(a) shows normalised achieved goodput for several requested rates. A few earlier discussed effects can be seen in this graphs. The marks in the lower left corner are flows that had a *RTO* during Slow Start because increasing the RTT_{TRC} during Slow Start does not take into account the value of the *RTO*. The marks above the estimated goodput come from the effect that for moderate loss rates the achieved goodput is underestimated by about 3%. Simulation results providing values between 0.98 and 1 show the influence of mainly two aspects. The one is that the three packets arrived at the receiver but not yet acknowledged ones, have influence on the achieved goodput. The second aspect is that for a large *W* in the early phase of Slow Start with small congestion window it is not possible to reduce the *RTT* so far that the requested rate is achieved. If RTT_{net} increases the achieved rate during Slow Start will decrease.



(a) Simulation Results during Slow Start

(b) Simulation results for deviations in RTT

Figure 7 – Simulation results

5.4.4 Simulations with RTT_{dev} greater than 0

Simulation results in this section should show the influence of a deviation in the *RTT* on the achieved goodput. Simulations were run on a 200Mbps bottleneck link consequently the queuing delay can be neglected. The access delay is varied to achieve different deviations. The TRC is configured with an RTT_{min} of 200ms. The parameters of the TRC are not adapted

to compensate the RTT_{dev}. Figure 7(b) shows the normalised achieved goodput over different requested rates for a few RTT_{dev} . RTT_{min} has direct influence on the goodput error for big requests. For an error of 10ms and a RTT_{min} of 200ms the deviation for big requests is 5%. For small requests the influence is smaller, because small requests have a much higher *RTT* than RTT_{min} . The maximum deviation between simulation results with no *RTT* error and with *RTT* error can be approximated by $RTT_{dev} = RTT_{min}$.

5.4.5 Simulations under maximum load

This section provides simulation results for maximum load scenarios. The network parameters were set according to Section 5.3.3 which means that φ is set to 0.85. The bottleneck bandwidth is set to 2Mbps, 10Mbps and 50Mbps respectively. The access delay is varied to achieve the desired network behaviour. Simulations are run 100 times. The first simulation study should analyse if the TRC has some bias against R_{req} . For evaluating the influence of requested rates the *RTT* is homogeneous and varied for different scenarios. Simulations are run for an access delay of 0.5, 1.5, 4, 24, 49, 124, 249ms. The requested rates are varied from 50kbps up to 10Mbps within one scenario. Table 1 provides simulation results for several selected scenarios. Each line shows results from one scenario. It can be seen that there is no bias against big requests. All flows achieve the requested rate.

The rate of a TCP sender is in general heavily influenced by the *RTT*. Flows with lower *RTT* are more aggressive. For evaluating the influence of different *RTTs* the access delays for link S1 router0 is set to 0.5ms up to 249ms for link S7 router0. Table 2 shows simulation results

BW [Mbps]	delay [ms]	50 [kbps]	100 [kbps]	250 [kbps]	500 [kbps]	750 [kbps]	1000 [kbps]	2500 [kbps]	5000 [kbps]	10000 [kbps]
2	1.5	53.229 53.222-53.237	105.64 105.633-105.647	X X	517.38 517.348-517.412	X X	1024.87 1024.81-1024.93	X X	X X	X X
10	1.5	53.518 53.516-53.519	106.353 106.345-106.361	265.035 265.019-265.051	526.374 526.343-526.405	786.818 786.756-786.88	1043.61 1043.53-1043.69	X X	5156.98 5156.4-5157.56	X X
10	124	53.475 53.472-53.478	106.238 106.23-106.246	266.73 266.716-266.744	525.35 525.316-525.384	787.662 787.6-787.724	1047.22 1047.14-1047.3	X X	5179.03 5178.28-5179.78	X X
50	1.5	53.746 53.743-53.75	107.234 107.217-107.251	268.824 268.753-268.895	533.844 533.690-533.998	799.138 798.9-799.376	1063.5 1063.18-1063.82	2638.91 2638.08-2639.74	5261.22 5259.42-5263.02	10495.9 1049.24-10499.4
50	124	53.718 53.713-53723	108.055 108.037-108.073	270.198 270.133-270.263	532.422 532.195-532.649	798.072 797.839-798.305	1061.36 1061.07-1061.65	2639 2638.23-2639.77	5263.32 5261.77-5264.87	10504.5 10500.8-10508.2

Table 1 – Simulation results for different requested rates

BW [Mbps]	R_{req} [kbps]	0.5ms [kbps]	1.5ms [kbps]	4ms [kbps]	24ms [kbps]	49ms [kbps]	124ms [kbps]	249ms [kbps]
2	100	105.231 105.179-105.283	105.225 105.172-105.278	105.225 105.172-105.278	105.225 105.172-105.278	105.222 105.169-105275	105.217 105.165-105.269	106.601 106.566-106.636
10	100	107.263 107.176-107.350	107.256 107.170-107.342	107.257 107.171-107.343	107.256 107-170-107.342	107.254 107.168-107.340	107.249 107.163-107.335	108.101 107.998-108.204
50	100	105.55 *99.901-111.198*	105.546 *99.898-111-194*	105.547 *99.899-111.195*	105.546 *99.898-111.194*	105.546 *99.898-111.194*	105.538 *99.890-111.186*	106.301 *100.612-111.190*

Table 2 – Simulation results for different RTTs

scenario	50k 500ms	100k 250ms	250k 100ms	500k 70ms	750k 50ms	1M 30ms	2.5M 10ms	5M 5ms	10M 3ms
achieved Rate [kbps]	53.715 53.711-53.720	107.22 107.2-107-24	268.794 268.705-268.883	533.786 533.592-533.98	799.047 798.751-799.343	1063.37 1062.97-1063.77	2638.58 2637.53-2639.63	5260.43 5258.21-5262.65	10494.9 10491.2-10498.6

Table 3 – Simulation results (The higher the requested rate, the higher the RTT)

scenario	50k 3ms	100k 5ms	250k 10ms	500k 30ms	750k 50ms	1M 70ms	2.5M 100ms	5M 250ms	10M 500ms
achieved rate [kbps]	53.719 53.712-53.725	107.115 107.086-107.144	268.21 268.078-268.342	532.809 532.531-533.087	797.508 797.078-797.938	1061.19 1060.62-1061.76	2631.05 2629.49-2632.61	5256.47 5252.28-5260.66	10337.9 10270.3-10405.5

Table 4 – Simulation results (The higher the requested rate, the lower the RTT)

for different *RTTs*. It can be seen that the TCP flows achieve nearly the same rate. Thus, the achieved rate of a TRC policed TCP flow has no bias against any *RTT* and the TRC is able to provide goodput assurances for a wide range of *RTTs*.

Further simulations are run to evaluate the influence of mixed *RTTs* and mixed R_{req} . Therefore the bottleneck bandwidth is set to 50Mbps. Table 3 shows simulation results where the parameters are set so that the higher the target rate the higher the *RTT* and Table 4 for a parameter setting where the higher the target rate the lower the *RTT*. Each combination of requested rate and *RTT* achieves the requested rate.

5.5 Summary

In this chapter a TCP rate controller (TRC) is proposed. The TRC seeks to achieve goodput assurances for long-lived TCP flows.

The task of the TRC is it to control the achieved goodput of a TCP connection by controlling a connections *RTT* and *p*. The TRC is based on a TCP model which predicts the sending behaviour for known values of *RTT* and *p*. The idea of the TRC is it to fix *RTT* and *p* of a connection. Therefore the TRC drops packets to control the window size of the TCP connection and delays packets to increase the *RTT*. Consequently the achieved rate of the connection is controlled. The TRC is a traffic conditioner which has to be placed at the ingress point of the network. A TCP model for ns-2 TCP SACK implementation was derived. Based on this model the TRC is constructed.

The TRC is evaluated by simulations using ns-2. It is shown that the TRC has no bias against any requested rate or *RTT*. The requested rate is achieved with a very high probability and confidence intervals for the achieved goodput are small. Overall concluding from the simulation results it seems promising to drop packets already at the ingress point of the network.

The whole concept of the TRC is based on simulations in `ns-2`. So the TRC has to be evaluated by real measurements in TCP/IP networks, because the accuracy of the TRC depends on the TCP model. In real TCP/IP networks there exist a lot of slightly different TCP implementations [8]. Consequently the applicability of the TRC in such an environment has to be evaluated.

The TRC rate controller does not need more suppositions as needed in QoS networks. The diversity of TCP implementations is a general problem of all attempts that try to control or estimate TCP rates based on a TCP model.

There are still open issues which have to be investigated in more detail. For example the dependence of RTT_{dev} on link bandwidth and requested rates has to be analysed.

5.6 References

[1] Network Simulator ns-2, see http://www.isi.edu/nsnam/ns/.

[2] Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The Macroscopic Behavior of the TCP Conges-tion Avoidance Algorithm. ACM Computer Communication Review **27** (1997)

[3] Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation. In: ACM SIGCOMM‘98. (1998)

[4] K. Claffy, Workload Characterization, http://www.caida.org/analysis/workload , June 2002

[5] Sahu, S., Nain, P., Towsley, D., Diot, C., Firoiu, V.: On Achievable Service Differentiation with Tocken Bucket Marking for TCP. In: Proc. of the ACM SIGMETRICS’2000 Int. Conf. on Measurement Modeling of Computer Systems, Santa Clara, CA, USA. (2000)

[6] Chait, Y., Hollot, C., Misra, V., Towsley, D., Zhang, H., Lui, J.: (Providing Throughput Differentiation for TCP Flows Using Adaptive Two-Color Marking and Two-Level AQM) to be presented at INFOCOMM 2001.

[7] Elizondo-Armengol, A.: TCP-Friendly Policy Functions: Capped Leaky Buckets. In: Seventeenth International Teletraffic Congress (ITC17). (2001)

[8] TBIT The TCP Behavior Inference Tool, see http://www.icir.org/tbit/.

[9] S. Sahu, P. Nain, D. Towsley, C. Diot, V. Firoiu, "On Achievable Service Differentiation with Token Bucket Marking for TCP", Proc. ACM SIGMETRICS'00 (Santa Clara, CA, June 2000)

[10] Chait, Y., Hollot, C., Misra, V., Towsley, D., Zhang, H., Lui, J.: (Providing Throughput Differentiation for TCP Flows Using Adaptive Two-Color Marking and Two-Level AQM) to be presented at INFOCOMM 2001.

6 An implementation of the AQUILA PMM service class

6.1 Introduction

The AQUILA architecture offers a network service called Premium Multimedia (PMM). It is intended for greedy TCP-based applications that require a minimum sending rate. Example applications are premium FTP data transfers or TCP-based streaming media applications. The network operator implements a network service by means of admission control, traffic conditioning, queue management, and scheduling.

In order to be able to perform the testbed / real-user trials we strive for an approach that could be implemented with the router equipment available in the project. In this paper we investigate the feasibility of a PMM implementation on the basis of *token bucket marking* and *preferential dropping*.

The foundation of the approach is an analytical model [4] for the TCP sending rate when a TCP flow is subject to a token bucket marker and preferential queue management. The model shows that there are conditions where the sending rate of the TCP flow can be regulated by the configuration of the token bucket parameters. The goal of the PMM implementation is to permanently enforce these conditions by means of admission control and queue management.

If an application wants to utilize the PMM service it sends a reservation request to the AC entity. The request contains the requested rate *R*. The AC entity decides whether or not the request can be accepted. If the request is accepted, the requester's ingress edge device is reconfigured which involves the setup of a classifier and the token bucket marker. In any case, the AC decision is signalled back to the requester using the signaling facilities provided by the AQUILA framework.

6.2 Related Work

There exist several works [5-8] that enforce TCP rate control by relying on an “invasive” mechanism where TCP header fields are modified. Packeteer [9] seems to have originally come up with this concept. Their TCP rate controller [10] modifies the receiver window and acknowledgement number and additionally modulates the rate of acknowledgements.

Several “non-invasive”' mechanism based on packet marking algorithms have been investigated in the context of Differentiated Services. Many of these algorithms use a static marking profile [11-17]. Adaptive marking algorithms are proposed in [18-21].

Unlike these works the AQUILA architecture explicitly acts upon the assumption of an admission control entity limiting access to the offered service classes. The TCP rate controller (TRC) for long-lived TCP flows proposed in [22] essentially requires an admission control entity. It operates as a traffic conditioner at the edge of a domain. Given a requested rate and an estimation of the domain's delay it computes a target packet drop probability and a target

delay on the the basis of a TCP model [24]. By enforcing the drop rate and introducing artificial delays the TCP flows are trimmed to the requested rate. The TRC is shown to be unbiased to the requested rate and RTT.

6.3 Token bucket marker

The authors of [4] model the impact of token bucket marking on greedy TCP flows. On the basis of a model for TCP sending behavior [24] they develop an analytical model for determining the sending rate of a TCP flow when edge-routers use token bucket marking (with statically configured token bucket parameters) and core routers employ active queue management with preferential packet dropping. Using this model (we denote it as *TBM model* in the following) it is shown that there exist conditions where it is not possible to influence the service achieved by a TCP flow through a marking profile. For a different set of conditions it is, however, feasible to achieve a requested sending rate. In that case the sending rate A of a greedy TCP flow is given by eq. 1

$$A = \begin{cases} R - \dfrac{3}{2R_{P_2}T^2} & R \le \dfrac{3W}{2T} \\ \dfrac{4}{3}\left(R - \dfrac{3}{2T\sqrt{2}}\sqrt{Z + \dfrac{1}{P_2}} \right) & R > \dfrac{3W}{2T} \end{cases} \qquad \mathrm{W} = \sqrt{2(\mathrm{Z} + 1/\mathrm{p}_2)} + 2\sqrt{2\mathrm{Z}} \qquad \textbf{\textit{eq. 1}}$$

Table 1 describes the parameters involved. The necessary condition is a so *called under-subscribed* scenario which is defined as $p_1 = 0$ and $p_2 > 0$.

The results of [4] make the TBM model a promising candidate for a PMM implementation. The model, which is analytically derived from an accurate TCP model, provides closed-loop formulae for the computation of the appropriate marking profile required to achieve a target sending rate. In a simulation [4] the TBM model is shown to be very accurate over a wide range of values for p_2, T, and R. Given the accuracy of the model and the possibility to practically implement the token bucket marking approach using today's routers we construct the PMM class on the basis of the TBM model.

The goal is to operate the service class in the region where the achieved TCP rate can be regulated through the configuration of the token bucket parameters. We seek to establish the required under-subscribed scenario by combining the token bucket marking with adequate admission control and queue management.

Parameter	Meaning	Unit
A	Token bucket rate	packet/s
Z	Token bucket size	packet
R	Requested rate	packet/s
p_1	packet drop probability for in-profile packets	-
p_2	packet drop probability for out-profile packets	-
T	round-trip-time (RTT)	s

Table 1 - Token bucket parameters

The goal is to operate the service class in the region where the achieved TCP rate can be regulated through the configuration of the token bucket parameters. We seek to establish the required conditions by combining the token bucket marking with adequate admission control and queue management.

6.4 Admission control

The goal of any admission control (AC) functionality is to limit the amount of traffic admitted to a particular service class such that the QoS objectives are reached for all admitted flows. At the same time, the service class utilization should be maximized.

For PMM a declaration based approach is employed: the requested rate R is part of the reservation request sent to the AC entity. Using the TBM model, the AC entity computes the token bucket rate A according to eq. 1. Now, if $A > R$, the flow requires at least an available bandwidth of A in order to obtain R. If $A \leq R$, an amount of R resources must be available.

On this basis a simple AC rule can be formulated. The resources required by a single flow are expressed by the greater value of the token rate A and the requested rate R. The inequality in eq. 2 ensures that the bandwidth required by the aggregate traffic submitted to the PMM class is smaller than r times the reserved capacity C, where r is a (tunable) over-provisioning factor; it denotes the fraction of reserved capacity that is at most allocated to resource requests.

$$\sum_{i=1}^{N} \max(A_i, R_i) \leq rC \qquad \textbf{eq.2}$$

The number of flows in the PMM class, including the new one if being admitted, is denoted by N. Note that eq. 2 implicitly assumes that the aggregate PMM traffic is able to fully utilize the reserved capacity C. It is discussed in below why this is a valid assumption. In the AQUILA framework the bandwidth is allocated to the different network services by means of WFQ-based scheduler at each router output port.

It follows from the TBM model that in an under-subscribed scenario each TCP flow achieves a minimum sending rate R_0 when the token bucket marks all packets as out-profile (A=Z=0).

$$R_0 = \frac{1}{T}\sqrt{\frac{3}{2P_2}} \qquad \textbf{eq.3}$$

A reservation request for a rate $R < R_0$ can be either principally denied (because a rate as small as R cannot be achieved) or it has to be handled in the following way:

- the token bucket parameters are set to: $A = Z = 0$.
- for that particular request, R is replaced by R_0 in the computation of the sum in eq. 2 to take into account that the flow will in fact consume R_0 bandwidth.

6.4.1 Queue management

In order to be combineable with the TBM model, the queue management mechanism must be able to enforce an under-subscribed scenario, i.e. $p_1 = 0$ and $p_2 > 0$. To achieve this goal, we derive a quantitative model for setting the parameters of two-color RED queue management. For easier reading the detailed derivation of the model is moved to chapter 7.

6.5 Parameterization

The TBM model requires the expected drop probability for out-profile packets p_2 as an input parameter. The value of p_2 must thus be estimated a priori through a constant value. Clearly, p_2 can fluctuate heavily as it depends on the level of congestion (number of flows) and the out-share. The out-share itself is influenced by:

- the size of the requested rates:

  it is a general property that TCP flows with smaller window sizes exhibit more "aggressiveness" than flows with larger windows. As a consequence, flows with lower bandwidth requests produce more out-profile packets than flows with higher bandwidth requests. This general effect is reinforced if the token bucket rate A is computed according to the TBM model (eq. 1), as in the TBM model (A - R) is strictly monotonic increasing with increasing R.

- the portion of reserved capacity allocated to accepted requests:

  The PMM class is utilized by greedy TCP flows which always fully utilize the available capacity - independently of the requested rate. Clearly, the token bucket parameters A and Z limit the amount of packets that are marked as in-profile but the amount of out-profile packets is only limited by the portion of unallocated capacity.

Unfortunately it is not possible to make a worst-case estimation of p_2. In the one case, if p_2 is estimated too low, the sending rates of the TCP flows are over-estimated and the resulting to-

ken rates are too small. For those requests where $A > R$, the max(A, R) is smaller than the amount of bandwidth that would in fact be needed by that flow. Thus, in general, too many flows would be admitted.

In the other case, if p_2 is estimated too high, the resulting sending rates of the TCP flows are under-estimated and the computed token rates are too high. This again leads in general to a situation where too many flows are admitted because the real TCP rates are higher than the ones used for the admission control algorithm.

The TBM model requires an estimate T of the RTT. It could be estimated as the propagation plus transmission delays plus the average queuing delay at the WRED bottleneck. It is however difficult to know in advance the number of bottleneck routers in a flow' s path. Additionally, different flows generally have different destinations with different RTTs.

In lack of an analytical model that captures the behaviour of the PMM implementation we try to discover parameter configuration dependencies by executing a large amount of packet level simulations. The goal is to discriminate between configurations which enable a successful PMM operation (if possible at all) from those configurations where the probability for a sending rate smaller than R is much larger than zero. Moreover, we investigate the influence of deviations in the parameter estimation on the usability of the various traffic control components.

6.6 Simulation study

We use the well-known packet-level network simulator ns-2 from [35] and extend it by admission control functionality. The AC entity decides on the basis of eq. 1, 2, 3 whether the request can be accepted.

The simulated topology is shown in figure 1. Applications are distributed over hosts S1-Sn and send to hosts R1-Rm, where n and m are chosen such that drops occur only at the output interface of B1. The AC entity controls access to the bottleneck link between routers B1 and B2. The bottleneck link has a capacity of 10 Mbps and a propagation delay of 20 ms. We simulate only PMM traffic and thus the full 10 Mbps are reserved for PMM. All other links have a capacity of 100 Mbps. The propagation delay of the links between B2 and hosts R1-Rm is either set as 70 ms for all links in the RTT_{equal} scenarios or set as 40ms (B1-R1), 70 ms (B2-R2), and 100 ms (B2-R3) in the RTT_{var} scenarios. This {40 , 70, 100} cycle is repeated until Rm. The RTT_{var} setup enables the study of competing flows under different RTTs.

We use the ns-2 built-in FTP application to simulate greedy, TCP based bulk-data transfers. The application life time is uniformly distributed between 50 and 150 seconds. While this is arguably not a choice matching real-world observations, we are initially not interested in very long transfers. The goal of the study is to first get an understanding what parameter configuration is useless / useful for the operation of TCL 3. Once this large parameter space can be reduced, more detailed studies can be performed and we argue that a configuration that optimizes transfers of FTP flows lasting for 150 seconds will also be beneficial for longer flow durations.

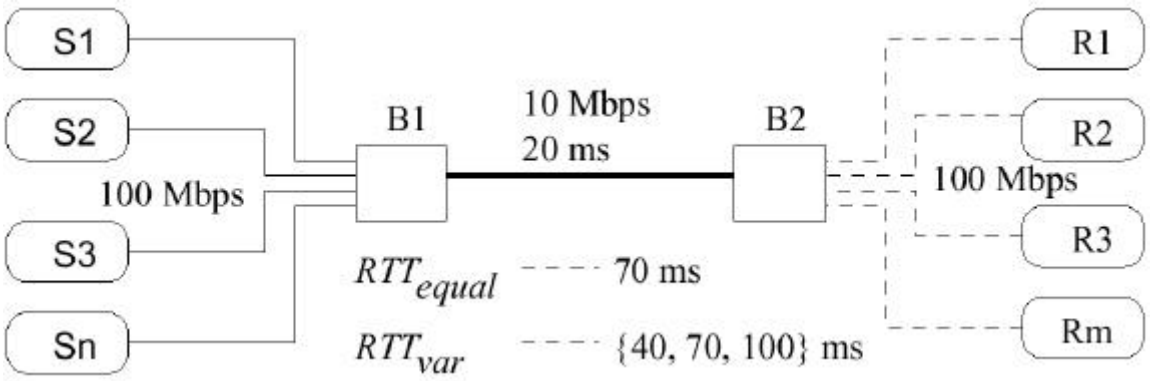


Fig. 1 Simulation topology

We have seen in initial simulations that the higher the ratio of the highest and smallest requested rate, the more difficult it is to provide the requested sending rates. This is a consequence of the weak estimation of certain parameters like p_2 or T. In fact, flows with a smaller requested rate tend to "steal" bandwidth from flows with a higher rate. To study this effect, we define a requested rate factor rF as $rF := R_{max}/R_{min}$ and include rF into the list of parameters that are varied between simulation runs.

We put an additional restriction on the rates that can be requested by allowing only a finite set of rates within the range $[R_{min}...R_{max}]$. The difference between R_{i+1} and R_i must be equal to the requestable rate distance rD. Like rF, rD is also varied over the simulation scenarios. In total, the requestable rates are the ones shown in table 2.

rF = 3	
rD = 100:	$R \in \{200, 300, 400, 500, 600\}$
rD = 200:	$R \in \{200, 400, 600\}$
rD = 300:	$R \in \{200, 500\}$
rF = 5	
rD = 100:	$R \in \{200, 300, 400, 500, 600, 700, 800, 900, 1000\}$
rD = 200:	$R \in \{200, 400, 600, 800, 1000\}$
rD = 300:	$R \in \{200, 500, 800, 1100\}$
rF = 8	
rD = 200:	$R \in \{200, 400, 600, 800, 1000, 1200, 1400, 1600\}$
rD = 300:	$R \in \{200, 500, 800, 1100, 1400\}$
rD = 400:	$R \in \{200, 600, 1000, 1400\}$

Table. 2 Requestable rates

Resource reservation requests are generated according to an exponentially distributed inter-arrival time with a mean on 2 seconds. This results in a high-load scenario where at almost any time all resources are allocated to flows and the probability that a new request has to be denied is high. Although such a request blocking probability may be unrealistically high in an operative network, it is a worst case scenario where flows cannot consume unallocated bandwidth and thus more easily reach the required sending rate.

We introduce the term traffic template which represents exactly one possible combination of requested rate R and RTT T. When a new resource request is generated, one such traffic template is randomly (uniform) chosen and a request for the template's requested rate R is sent to the AC entity. Besides choosing T according to the RTT_{equal} and RTT_{var} scenarios, respectively, we also vary the error in RTT estimation, called T_{dev}, as {0%, 25%, 50%, 75%, 100%}. This enables us to study the influence of a wrong RTT estimation. A T_{dev} of 0% means that T is set as a value that closely matches the RTT of the simulation scenario (propagation delays plus transmission delays plus average WRED queuing delay). If T_{dev} is larger than zero, the real RTT is around $T*(1+T_{dev})$, i.e. T underestimates the real RTT. Underestimation is the more difficult case as this overestimates the TCP sending rate.

Concerning the WRED model, we investigate two approaches for setting the number of flows N. In one case, called N_{high}, we set $N=rC/R_{min}$. This estimation assumes that all requests are for the minimum rate R_{min} and thus N_{high} is generally an overestimation. In the other case, called N_{low}, we set $N=rC/R_{avg}$, where $R_{avg}=(R_{min}+R_{max})/2$. Due to an unfairness on the request level - smaller requests have a higher probability of being accepted - N_{low} is thus generally an underestimation.

Finally, we use values of {0.7, 0.8, 0.9} for the r parameter of the AC formula (eq. 2). Some parameters have been fixed for all simulations. The packet size is set to 1500 bytes. The WRED model is configured with *adapt*=0.5 (see chapter 7). For the TBM model, the p_2 value of 0.1 is configured. This choice is the result of a prior study not shown here.

r of admission control	$\in \{0.7, 0.8, 0.9\}$
number of flows for the WRED model	$\in \{N_{low}, N_{high}\}$
link delays in topology	$\in \{RTT_{equal}, RTT_{var}\}$
error in RTT estimation T_{dev}	$\in \{0\%, 25\%, 50\%, 75\%, 100\%\}$
requested rate factor rF	$\in \{3,5,8\}$
requested rate distance rD	$\in \{100,200,300,400\}$

Table 3 Summary: variation of input parameters

The input parameter space is summarized in Table 3. In order to exhaustively explore this input space we simulate all 540 possible combinations of input parameters. Each simulation is run for 50000 simulated seconds. As the request blocking probability increases with larger

requested rates, this long simulation time is needed to get enough results for traffic templates which request a rate of R_{max}.

6.7 Simulation results

The primary QoS goal of the PMM class is to achieve a sending rate that is at least as high as the requested rate. We therefore define a success value s_t for each traffic template t in the following way:

$$s_t := \frac{card\{flow \in t : rate \geq R\}}{card\{flow \in t\}} \qquad \textbf{eq.4}$$

A whole simulation run is characterized by one success parameter *S*, where $S=\min(s_t)$. Interestingly, the resulting success values vary between 0% (i.e. there is no traffic template where all flows reach at least *R*) and 100% (i.e. all flows in all templates reach at least R).

To evaluate the optimality of the different parameter configurations we represent each simulation with a 7-dimensional vector: one dimension for the success *S* of the simulation and 6 dimensions due to the input parameters (see table 3).

In order to gain insight on how to ideally configure the PMM service we select those simulation vectors with a success $S \geq 0.99$. Such high success values can only be obtained if the input parameters are chosen among the ones shown in table 4.

r of admission control	$\in \{0.7\}$
number of flows for the WRED model	$\in \{N_{high}\}$
link delays in topology	$\in \{RTT_{equal}, RTT_{var}\}$
error in RTT estimation T_{dev}	$\in \{0\%, 25\%, 50\%, 75\%, 100\%\}$
requested rate factor rF	$\in \{3\}$
requested rate distance rD	$\in \{100,200,300\}$

Table 4 Summary: input parameters resulting in high success S

We subsequently discuss the results and thereby look in detail at the influence of each parameter involved. The following paragraphs provide guidelines on how to optimally configure the PMM service class.

The choice of $r = 0.7$ provides enough safety margin (unallocated capacity) to compensate for the imperfections of the PMM traffic control combination. Among these imperfections are the impossibility to perfectly estimate changing parameters by a single static value and deviations between analytical models and the traffic under control.

Concerning the WRED model, better results can be achieved if the number of flows is set as N_{high}. The convergence behaviour of the average queue size is indeed according to the objective of the WRED model. This justifies the correctness and usability of the WRED model even in an environment where the input parameters are not exactly known.

Another positive outcome of the simulation study is the result that under the restricted choice of *r*, *rF*, and *rD* as shown in table 4, differing RTTs (RTT_{var}) as well as wrongly estimated RTTs ($T_{dev} > 0$) do not present a major problem. This is a convenient property as finding a good estimate for the RTT is a difficult task.

Providing requestable rates over a broad range seems an impossible objective with the design as investigated in this study. In fact, good results can only be achieved if *rF* is not larger than 3. The distance *rD* between requestable rates has no impact on the success S.

For the further analysis the focus is on simulation scenarios where success $S \geq 0.99$. We fix the input parameters to $r = 0.7$, N_{high}, $rF = 3$, $T_{dev} = 50\%$ and take a detailed look at the distribution of sending rates for the RTT_{equal} / RTT_{var} scenarios and different values of rD. The sending rates are classified into bins of 10 kbps. Figure 2 shows the relative frequency of sending rates for each requestable rate in the RTT_{equal}, rD = 100 scenario. The requestable rates are listed in table 2.

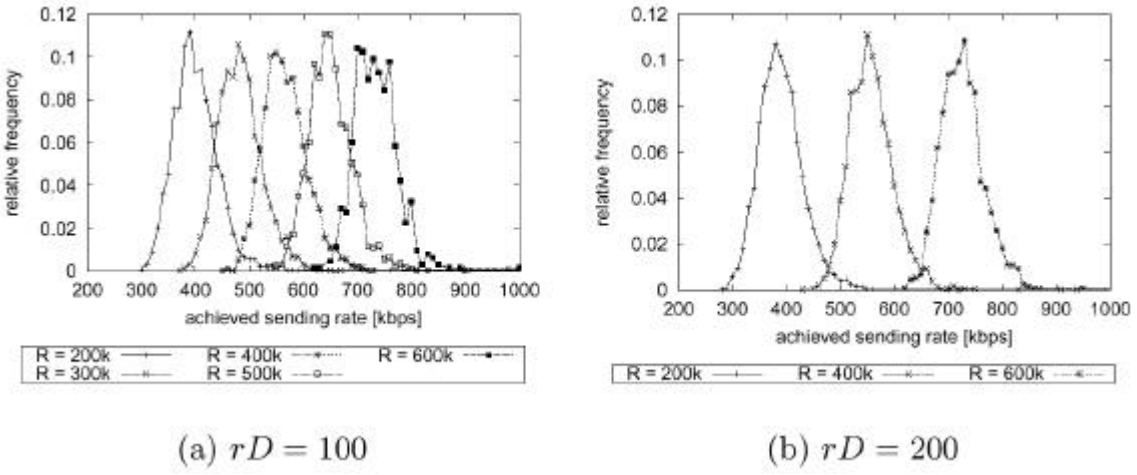


Fig. 2 Relative frequency of achieved sending rates in the RTT_{equal} scenario

As can be seen in figure 2, each flow achieves at least the requested sending rate, i.e. the success S of both simulations is 100%.

As far as the service differentiation within the PMM class is concerned, the scenario with *rD*=100 (subfigure 2(a)) shows a suboptimal behaviour. The user is offered a high number of requestable rates. However, the resulting service curves are significantly overlapping and the service offerings are thus not clearly distinguishable.

In the *rD* = 200 case (subfigure 2(b)) the service curves are hardly overlapping. This provides for a clear distinction of the service delivered for different requests. The service distinction is even more pronounced in the *rD* = 300 scenario where only two rates (200 kbps, 500 kbps) are requestable.

Looking at the shape of the service curves it makes sense to offer only a discrete set of requestable rates instead of a continuous range between [R_{min} ... R_{max}]. By offering a finite set of rates the operator can tune the service differentiation within the PMM class. This approves the usefulness of the discrete rates concept. A choice of *rD* = 200 results in a reasonable trade-off between the number of requestable rates and a clear service distinction.

In the RTT_{var} scenario, there are flows with different RTTs within each requested rate. Compared to the RTT_{equal} scenario, the resulting sending rates fluctuate more and the curves in figure 3 are thus slightly broader and lower. For *rD* = 200 the service differentiation within the PMM class is still well pronounced.

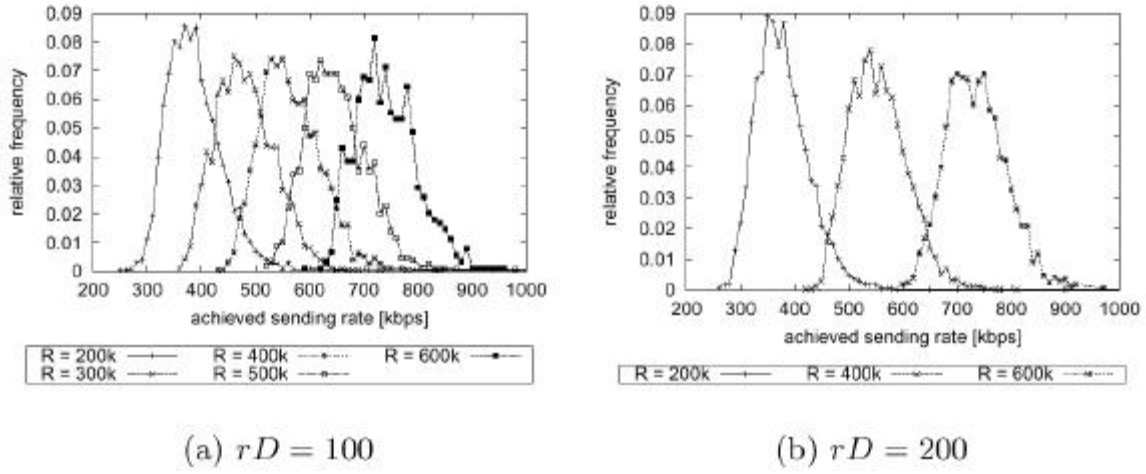


Fig. 3 Relative frequency of achieved sending rates in the RTT_{var} scenario

6.8 Conclusion

In this chapter we report on an attempt to establish a QoS class for long-lived, bulk-data TCP flows that require a minimum rate from the network. The approach is based on a model for TCP flows subject to token bucket marking at the network edge-device and preferential dropping in the core network. This model is combined with an admission control functionality and a model for the parameterization of multi-RED queue management. The goal of the additional components is to enforce conditions under which the sending rate of the flows can be regulated through the token bucket marking profile.

The difficulty of finding a proper parameter set for the various input parameters of the service class is discussed. In a large simulation study a broad spectrum of the input parameter space

is explored in order to identify inter-parameter dependencies and discriminate between useless / useful service class configurations.

From these results guidelines on how to configure the PMM class are derived. While the models derived for admission control and queue management can be generally applied in the context of long-lived TCP flows and token bucket marking, the guidelines only apply to the PMM implementation studied in this paper.

Although ideal traffic handling is not feasible with a *static* marking profile the simulation results encourage the practicability of a real-world implementation. Despite simulations were run at a very high service class utilization (and thus an unrealistically high service request blocking probability) a set of configuration parameters that enables a successful operation could be identified. The QoS objectives can be more easily reached under a lower service utilization where more unallocated resources are available.

Initial testbed measurements have been performed but they were heavily influenced by the maximum queue size that could be configured in the router when the full AQUILA scheduling approach is employed (the combination of Priority Queueing and WFQ). This restriction did not allow the use of the WRED model as discussed in the next chapter and consequently led to throughput degradations and unfairness issues. We plan to repeat the measurements with the PMM class only as this allows the use of a FIFO scheduler where our equipment does not have the mentioned buffer size restrictions.

6.9 References

1. Adaptive Resource Control for QoS Using an IP-based Layered Architecture (AQUILA), IST-1999-10077, http://www.ist-aquila.org/.

2. Nichols, K., Blake, S., Baker, F., Black, D.: RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (1998)

3. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: RFC 2475: An architecture for differentiated services (1998)

4. Sahu, S., Nain, P., Towsley, D., Diot, C., Firoiu, V.: On Achievable Service Differentiation with Tocken Bucket Marking for TCP. In: Proc. of the ACM SIGMETRICS'2000 Int. Conf. on Measurement Modeling of Computer Systems, Santa Clara, CA, USA. (2000)

5. Narvaez, P., Siu, K.Y.: An Acknowledgment Bucket Scheme for Regulating TCP Flow over ATM. In: Proceedings of IEEE Globecom. (1997)

6. Koike, A.: TCP flow control with ACR information (1997) ATM Forum/97-09989.

7. Kalampoukas, L., Varma, A., Ramakrishnan, K.K.: Explicit Window Adaptation: A Method to Enhance TCP Performance. In: Proceedings of INFOCOM '98. (1998)

8. Satyavolu, R., Duvedi, K., Kalyanaraman, S.: Explicit rate control of TCP applications (1998) ATM Forum Doc. Number 98-0152R1.

9. (Inc., P.) http://www.packeteer.com/ (Feb. 2003).

10. Karandikar, S., Kalyanaraman, S., Bagal, P., Packer, B.: TCP rate control. ACM Computer Communications Review (2000)

11. Elizondo-Armengol, A.: TCP-Friendly Policy Functions: Capped Leaky Buckets. In: Seventeenth International Teletra_c Congress (ITC17). (2001)

12. Heinanen, J., Guerin, R.: RFC 2698: A Two Rate Three Color Marker (1999)

13. Fang, W., Seddigh, N., Nandy, B.: RFC 2859: A Time Sliding Window Three Colour Marker (TSWTCM) (2000)

14. Lin, W., Zheng, R., Hou, J.C.: How to make assured service more assured. In: ICNP. (1999) 182+

15. Seddigh, N., Nandy, B., Pieda, P.: Bandwidth assurance issues for tcp ows in a differentiated services network (1999)

16. Makkar, R., Lambadaris, I., Salim, J., Seddigh, N., Nandy, B.: Empirical Study of Buffer Management Scheme for Diffserv Assured Forwarding PHB. In: Proceedings of Ninth International Conferance on Computer Communications and Networks, Las Vegas, Nevada. (2000)

17. Azeem, F., Rao, A., Kalyanaraman, S.: A TCP-friendly traffic marker for IP differentiated services (2000)

18. Feng, W., Kandlur, D., Saha, D., Shin, K.: Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated Services Internet. In: IEEE/ACM Transactions on Networking. Volume 7. (1999) 685{697

19. Nandy, B., Seddigh, N., Pieda, P., Ethridge, J.: Intelligent traffic conditioners for assured forwarding based differentiated services networks. In: IFIP High Performance Networking (HPN 2000). (2000)

20. El-Gendy, M.A., Shin, K.G.: Equation-based packet marking for assured forwarding services. Infocom (2002)

21. Chait, Y., Hollot, C., Misra, V., Towsley, D., Zhang, H.: Providing throughput differentiation for TCP flows using adaptive two color marking and multi-level AQM. In: Proceedings of Infocom 2002. (2002)

22. Dorfinger, P., Brandauer, C., Hofmann, U.: A rate controller for long-lived TCP flows. In: Joint International Workshop on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems (IDMS/PROMS). (2002) 154-165

24. Padhye, J., Firoiu, V., Towsley, D., Krusoe, J.: Modeling TCP throughput: A simple model and its empirical validation. Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication (1998) 303-314

25. Seddigh, N., Nandy, B., Pieda, P., Salim, J.H., Chapman, A.: An Experimental Study of Assured Services in a Diffserv IP QoS Network (1998)

26. Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking 1 (1993) 397-413

27. Floyd, S.: The RED Web Page (1997) http://www.aciri.org/floyd/red.html.

28. Ziegler, T., Brandauer, C., Fdida, S.: A quantitive Model for Parameter Setting of RED with TCP traffic. In: Proceedings of the Ninth International Workshop on Quality of Service (IWQoS), 2001, Karlsruhe, Germany. (2001)

29. Brandauer, C.: Web interface to the RED model developed in [28] (2000) http://www.salzburgresearch.at/~cbrand/REDmodel.

30. Firoiu, V., Borden, M.: A study of active queue management for congestion control. In: Proceedings of IEEE Infocom, Tel Aviv 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00). (2000) 1435-1444

31. Clark, D., Fang, W.: Explicit allocation of best effort packet delivery service. IEEE/ACM Transactions on Networking 6 (1998) 362-373

32. Salsano, S. et al: Traffic handling studies (2003)

33. Brandauer, C.: Web interface to the WRED model (2001) http://www.salzburgresearch.at/~cbrand/WREDmodel.

34. Network Simulator ns-2, see http://www.isi.edu/nsnam/ns/.

7 A Queue management model for TCL 3

This chapter describes the effort to develop a queue management mechanism that enforces an under-subscribed scenario as required by the TBM model. To enable easier reading the detailed derivation of the model was moved from section 6.4.1 to this chapter.

7.1 WRED queue management

In order to be combineable with the TBM model, the queue management mechanism must be able to enforce an under-subscribed scenario, i.e. $p_1 = 0$ and $p_2 > 0$. First of all, this requires the ability to distinguish between in-profile and out-profile packets, respectively. We employ a two-color extension of RED [11] queue management. There is one parameter set for in-profile packets $\{minth_i, maxth_i, maxp_i\}$ and one set for out-profile packets $\{minth_0, maxth_0, maxp_0\}$. A single average queue size is calculated over all arriving packets and depending on the color of the packet the corresponding set of parameters for that color is used. This approach is generally referred to as Weighted RED (WRED).

In order to optimally support the PMM class, the following queue size behavior should be enforced:

- the average queue size converges within the control range for out-profile packets, i.e., between $minth_0$ and $maxth_0$.
- the amplitude of oscillation of the average queue size is bounded and significantly smaller than the difference between $minth_0$ and $maxth_0$.
- the instantaneous queue size is (mostly) greater than zero and smaller than the buffer size.

Such a behaviour is clearly beneficial for the PMM class: besides establishing the required under-subscribed scenario, the available link capacity can be fully utilized. First, this provides for optimal resource usage. Second, the predictability of bandwidth utilization is an important input for the AC algorithm. In fact, the AC rule has to take into account the amount of bandwidth the aggregate traffic stream is able to consume - not of the capacity reserved for that traffic class.

Moreover, due to the enforced queue size behaviour the controlled TCP flows experience a rather constant drop probability for out-profile packets and should be able to handle these drops without resorting to timeouts. Consequently, the sending behaviour of the flows is rather smooth.

It must be noted that a careful selection of WRED parameters is required to achieve the above described performance. If the queue management parameters are chosen rather incidentally the average queue size will generally not exhibit a similar behaviour. If $maxp_o$ is chosen too

high, the average oscillates around $minth_o$ which increases the probability of link-underutilisation. On the other hand, if $maxp_o$ is set too low, the average converges towards $maxth_o$ and all out-profile packets are dropped. The TCP flow might then experience several consecutive packets dropped and have to recover by means of timeout and slow start. This causes a burstier sending pattern which we consider unfavourable.

If the difference between $maxth_o$ and $minth_o$ is not adapted to the bandwidth*RTT product and the number of flows, the amplitude of average queue size oscillation may be improperly high (causing link under-utilisation and forced drops of out-profile packets) or even unnecessarily low (imposing unnecessarily high queuing delays and buffer requirements). See [3] for an in-depth analysis of these effects.

7.2 Implementation

The WRED model is an extension of the quantitative RED model [3]. This RED model calculates the RED parameters *minth*, *maxth*, maxp, *wq* , and the buffer size as a function of the scenario parameters bottleneck bandwidth, RTT, and number of TCP flows. The RED model has been developed by assembling an accurate analytical model of TCP sending behaviour [5], an analytical model for setting *wq* [6] and an empirical model providing the required difference between *maxth* and *minth*. It is shown in [3] that under the load of long-lived TCP flows, RED's average queue size converges between *minth* and *maxth* and the amplitude of average queue size oscillation is around one third of the difference between *minth* and *maxth*.

The idea of the WRED model is to achieve a similar convergence behaviour in a two-color environment. We derive the WRED model directly from the equations of the RED model [3]. In fact, the method to calculate the parameter set for out-profile packets is based on the RED model. The reader is referred to [3] for the details of how the equations are established in order to avoid a duplication of results. Differing from [3] we assume a homogeneous RTT scenario for the sake of simplicity. However, the model can be easily extended to a scenario with multiple RTT classes. With (W)RED queue management the long term average queue size is mostly dependent on the maximum drop probability *maxp*. This parameter determines the aggressiveness of dropping packets when incipient congestion is detected. In order to establish optimal support for services trying to provide throughput assurances, the dropping of in-profile packets should be avoided whenever possible [2]. The goal is to enable the desired convergence behaviour by dropping only out-profile packets.

If in-profile packets are excluded from the dropping process the WRED parameter $maxp_o$ must be higher than the RED parameter *maxp* to achieve the same overall drop probability. Therefore to adapt *maxp* correctly, some knowledge of the expected ratio of out-profile packets is required.

We define the number of out-profile packets divided by the total number of packets that arrived at the queue as the out-share. The out-share is influenced by many factors and may vary strongly over time. The out-share is estimated through a parameter called *adapt*.

$$L = \frac{N}{RTT\sqrt{\frac{b * maxp_{WRED}}{3}} + RTO * \min\left(1,3\sqrt{\frac{3b * maxp_{WRED}}{16}}\right)\left(\frac{maxp_{WRED}}{2} + 4(maxp_{WRED})^3\right)} \qquad \textit{eq. 4}$$

Equation eq. 4 is similar to the RED model with the one change that *maxp* is substituted by $maxp_{WRED} = maxp_o * adapt$ to (approximately) equalise the overall packet drop probability. The equation is based on the TCP model from [6] and the assumption that the aggregate of *N* flows can fully utilise the link capacity. The packet drop probability is set to $(maxp_o * adapt)/2$ because the goal is to make the average queue size converge to $(minth_o + maxth_o)/2$. The parameters involved in the equations are described in Table 2.

parameter	meaning	unit
C	Link capacity	bit/s
N	Number of flows	--
RTO	estimated retransmission timeout	s
b	number of packets acknowledged by an ACK (equals 2 if delayed ACKs are used, 1 otherwise)	--
d	total propagation delay	s
psize	Average packet size	bit
a	constant 0.01	--
adapt	constant. 0.5	--

Table 2 – Description of input parameters

Equations eq. 5 and eq. 6 are the same as for the RED model: Equation eq. 5 is the result from an empirical model providing the necessary difference between $maxth_o$ and $minth_o$ to limit the amplitude of average queue size oscillation. The values for the parameter $c_1 - c_2$ are shown in Table 3. Equation eq. 6 is a somewhat arbitrary way to set $minth_o$.

$$maxth_o - minth_o = c_1 * C * RTT + c_2 * N + c_3 \qquad \textit{eq. 5}$$

$$minth_o = \frac{maxth_o - minth_o}{3} \qquad \textit{eq. 6}$$

packet size [Byte]	c_1	c_2	c_3
250	0.02739	0.7324	17
500	0.02158	0.5670	85
1000	0.01450	0.3416	46
1500	0.01165	0.09493	85

Table 3 – Constants for maxth-minth model

The Equation system eq. 4 - eq. 6 has to be solved numerically for $minth_o$, $maxth_o$, and $max\text{-}p_{out}$ with the help of Equations eq. 7 - eq. 9.

$$L = \frac{C}{psize} \qquad \textbf{eq. 7}$$

$$RTT = 2d + \frac{minth_o + maxth_o}{2L} \qquad \textbf{eq. 8}$$

The term $(minth_{out} + maxth_{out})/2$ matches the average queuing delay at the bottleneck.

$$RTO = RTT + 2\frac{minth_o + maxth_o}{L} \qquad \textbf{eq. 9}$$

In the same way that is implemented by TCP, the retransmission timeout *RTO* is computed as the RTT plus four times the RTT variance which is approximated by the average queuing delay at the bottleneck. The buffer size is calculated according to eq. 10.

$$buffer\ \ size = \frac{3maxth_o}{2} \qquad \textbf{eq. 10}$$

Due to the assumption of a resource control framework, convergence of the average queue size within the control range for out-profile packets is feasible. This eliminates the need to drop in-profile packets for the sake of congestion avoidance/control. Thus, in order to minimise the drop probability for in-profile packets, we recommend to set the WRED parameters $minth_i$ and $maxth_i$ to the total buffer size and $maxp_i$ to 1. Note that with this approach there is practically no difference between RIO and WRED.

Finally, *wq* is computed with the model proposed in [6]. We have implemented a Web interface to the WRED model which can be accessed under [7].

7.2.1.1 Parameter dependencies

The model requires an estimate of the out-share as an input parameter. The out-share can oscillate significantly. For the WRED model in order to be practically applicable it is thus crucial to achieve the desired properties even if the real out-share differs significantly from the

estimated value *adapt*. The model's sensitivity on a correct estimation of the out-share is investigated in the next section.

Another critical input parameter needed for the WRED model is an estimation *N* of the number of flows. This dependency cannot be avoided as the RED behaviour is intrinsically influenced by the number of flows. In order to compute a WRED parameter set it is therefore required to estimate the expected number of flows. As it will change over time there is no correct value for *N*. Clearly, there is a lower bound (zero or more flows) as well as an upper bound for the number of flows that can be active at any time in the system. Within TCL 3, the upper bound is determined by the finite reserved capacity and the smallest acceptable requested rate.

7.2.2 Simulations

We studied the behaviour of the WRED model by means of network simulation using ns-2 [8]. The simulations are based on the simple dumb-bell topology depicted in Figure 1. All FTP senders are distributed among hosts 1-4 and start sending to hosts 5-8 at a random point of time within the first 10 seconds. Traffic is multiplexed at `router0` where the WRED queue management is executed and monitored. Packet drops happen solely at the output port of `router0`. A token bucket marker is attached to each TCP sender so that the traffic conditioning is realised on a per-flow basis. The token bucket size is set constant to 40 packets in each scenario. The token rate is varied in order to achieve the desired out-share. The TCP segment size is set to 1500 bytes. Simulations last for 500 seconds; the initial 100 seconds are not considered for the statistics.

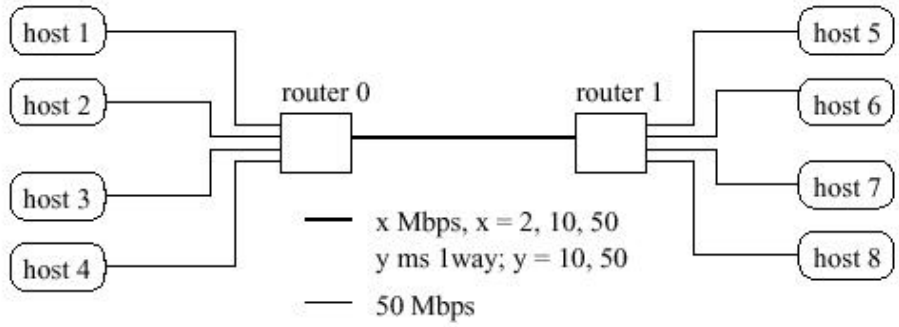


Figure 1 – Simulation topology

The main focus of the simulations shown here is to evaluate the convergence behaviour of the WRED queue. Particular attention is paid to the out-share parameter. The model is calculated for an estimated out-share of 50%, i.e. *adapt* = 0.5, because this should roughly minimise the deviation to the real out-share. The same number of flows N is chosen as input for the model and configuration of the simulation. In the simulations the token bucket rates are adapted to produce three different scenarios with out-shares of 10%, 50%, and 90% respectively. Each scenario is simulated with different bottleneck bandwidths (2, 10, 50 Mbps) and different propagation delays (10, 50 ms).

scena-rio	ban-dwidth [Mbps]	delay [ms]	N	token rate [kbps]	$minth_o$ [pkt]	$maxth_o$ [pkt]	$maxp_o$	buffer size [pkt]	w_q
M1	2	1	10	100	32	129	1/27	194	1/268
S1	2	5	10	100	32	129	0.0361	194	0.004551
S2	2	10	10	100	33	131	0.0341	196	0.004334
S3	2	50	10	100	33	133	0.0264	199	0.003446
S4	10	10	12	400	33	133	0.0372	200	0.003690
S5	10	50	16	300	37	148	0.0239	222	0.001865
S6	50	10	25	1000	38	152	0.0429	228	0.002023
S7	50	50	62	400	57	228	0.0328	342	0.00064

Table 4 – WRED parameter configuration

The WRED parameter configuration for several scenarios is shown in Table 4. Note that $minth_i$ and $maxth_i$ are set equal to the buffer size; $maxp_i$ is set to 1. The Figure 2(a)–(c) plot the evolution of the average and instantaneous queue size over time for scenario S4 with 3 different out-shares. The dotted level lines indicate $minth_o$ and $maxth_o$. The instantaneous queue size is plotted as a thick dark solid line, the average queue size is plotted as a thin white line so that it can be distinguished from the instantaneous queue in a black/white printout. Note that the average is fully encased by the instantaneous queue size. The figures show that the average oscillates around a value between $minth_o$ and $maxth_o$ and that the amplitude of oscillation of the average is significantly smaller than the difference between $maxth_o$ and $minth_o$. There are no forced drops of out-profile packets. The buffer occupancy remains well below the queue limit and there is spare buffer space that can absorb bursts of incoming traffic. No in-profile packets are dropped. As the buffer never drains, the output link is fully utilised. These are all preferable properties which provide a sound foundation for a service that seeks to deliver throughput guarantees to long-lived TCP flows.

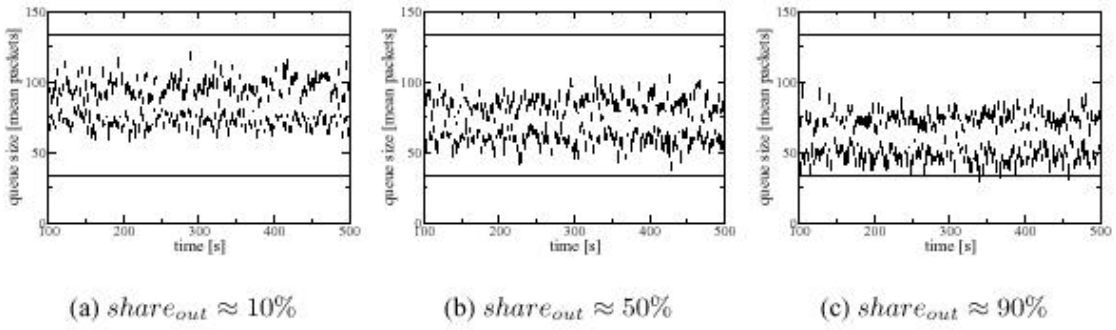


Figure 2 – WRED queue size over time behavior

Figure 2 indicates that it is not crucial to have an exact estimation for the out-share when calculating the WRED model. It can be seen that the long-term average queue size decreases from subfigure Figure 2(a) to Figure 2(c). In figure Figure 2(a), where the out-share is significantly lower than assumed in the calculation (10% instead of 50%), there are less out-profile packets than prospected and thus $maxp_o$ is set too small. This results in a somewhat too unaggressive dropping of out-profile packets and a higher queue size than in Figure 2(b). In Figure 2(c) the opposite effect occurs. In any case the convergence behaviour is satisfying.

Simulations for the other scenarios named S* in Table 4 exhibit a very similar convergence tendency. Table 5 provides numerical results for each scenario – again simulated with 3 different orders of out-share. Column 4 indicates the value around which the average queue size oscillates; column 5 relates to the amplitude of oscillation.

scenario	out-share [%]	pdrop IN [%]	pdrop OUT [%]	mean of avg q [pkt]	stddev of avg q [pkt]
M1	92.4	0	0.73	54.9	13.3
M1	52.4	0	1.02	61.5	10.9
M1	13.2	0	2.80	72.9	5.57
S1	90.1	0	1.35	62.7	6.28
S1	50.4	0	1.78	76.4	8.92
S1	12.4	0	5.76	91.7	9.15
S2	90.1	0	1.23	64.8	6.76
S2	50.4	0	1.66	77.7	8.51
S2	11.8	0	5.85	93.5	9.22
S3	90.1	0	0.94	65.4	7.98
S3	50.3	0	1.30	81.0	9.66
S3	11.8	0	4.20	98.0	10.48
S4	90.5	0	1.15	61.1	7.07
S4	52.4	0	1.61	72.4	7.97
S4	12.3	0	6.54	85.2	8.63
S5	90.5	0	0.68	66.7	9.78
S5	52.3	0	1.00	81.3	11.78
S5	16.2	0	2.86	101.4	13.23
S6	90.1	0	1.15	65.3	9.07
S6	50.5	0	1.81	82.3	10.18
S6	12.2	0	7.69	97.2	9.32
S7	90.2	0	0.82	96.3	16.64
S7	50.7	0	1.33	121.4	16.72
S7	13.2	0	5.33	150.5	16.89

Table 5 – Simulation and measurement results

7.2.3 Measurements

This section presents first results from testbed measurements. Similar to the simulations the goal of the measurements is to evaluate the convergence behaviour of the WRED queue under long-lived TCP flows. As there is no way of directly obtaining the instantaneous queue size at a router's output port the following approach is taken: we measure the one-delay of packets from the sender to the receiver host and approximate the router's queue size from these delay values as *queue size = (delay *bandwidth)=MTU*. Propagation and processing delays are small enough to allow for negligence.

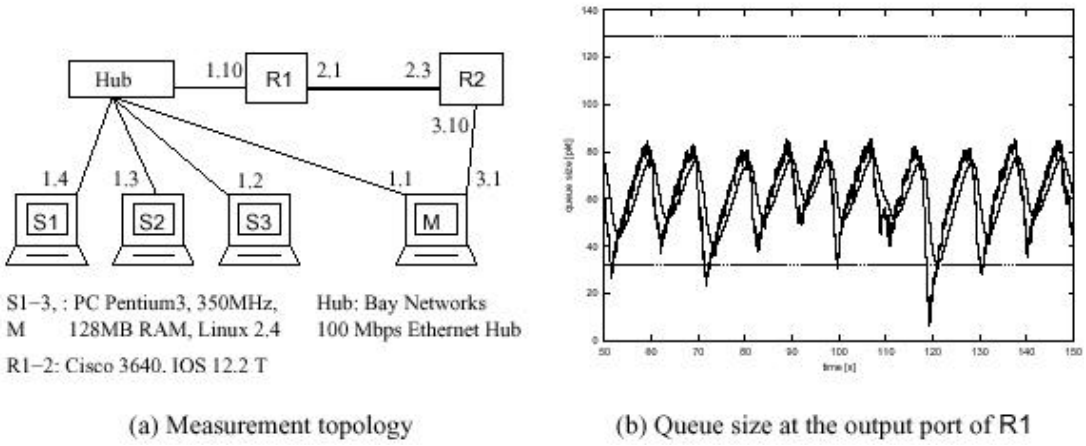


Figure 3 – Measurements

To establish these measurements the topology shown in Figure 3(a) is used. All IP addresses have the prefix `192.168`. and a netmask of `255.255.255.0`. The second portion of the IP address is shown in the figure. In the `192.168.1.0` net there are the three traffic generating hosts `S1`– `S3`, one interface of the router `R1`, and one interface of the master host M. These interfaces are connected by a hub. The number of collisions on the ethernet is extremely small and does not influence the results. The routers `R1` and `R2` are connected via a serial link with a clock rate of 2 Mbps. The router `R1` is configured according to scenario M1 in table 3. On the "receiver side" there is again the master host `M` with the `3.1` interface. In the host `M` the routes are configured such that all traffic is sent via the `3.1` interface even if the destination host is on the `1.0` net.

The `sock` [9] program is used to generate a total of 10 TCP flows from hosts `S1` (4 flows), `S2` (3), and `S3` (3) to host M. The TCP stack employs delayed ACKs and SACK. The packets of each flow are marked at `R1` by a per-flow token bucket conditioner. The token rate is varied over the measurements to get different out-shares. The flows start sending at a random point of time within the first 10 seconds after the beginning of the measurement and last for about 200 seconds. The statistics are gathered between 50 and 150 seconds. In order to enable an accurate one-way delay measurement the measurement points have to be time-

synchronised. We do not use a GPS clock for that task; instead sender side traffic as well as receiver side traffic is both traced at the same host M at the 1.1 and 3.1 interface, respectively. The capturing is done by the tcpdump [10] program. After the measurement the dump files are post-processed to calculate the one-way delay of packets and the instantaneous queue size at the router output port as described above. In order to verify the measurement infrastructure we successfully ran several tests prior to the WRED scenarios. Figure 3(b) shows the evolution of the instantaneous and average (calculated like RED's EWMA) queue size for one measurement with a medium out-share (approx. 50%). The behaviour of the queue is quite similar to simulation results obtained for the same scenario: convergence within the control range for out-profile packets, bounded oscillations, full link utilisation and no drop of in-profile packets. See Table 5 for numerical results. Additional measurements with smaller / higher out-shares resemble Figure 3(b) but the differences in the long-term average queue size are not as distinct as in the simulations.

7.2.4 References

[1] Deliverable D1301, Specification of traffic handling for the first trial, AQUILA project consortium, http://www.ist-aquila.org/, September 2000.

[2] S. Sahu, P. Nain, D. Towsley, C. Diot, V. Firoiu, "On Achievable Service Differentiation with Token Bucket Marking for TCP", Proc. ACM SIGMETRICS'00 (Santa Clara, CA, June 2000)

[3] T. Ziegler, C. Brandauer, S. Fdida, "A quantitative Model for Parameter Setting of RED with TCP traffic.", in Proceedings of the Ninth International Workshop on Quality of Service (IWQoS), 2001, Karlsruhe, Germany, June 2001

[4] S. Sahu, D. Towsley, J. Kurose " A quantitative study of differentiated services for the internet" Tech. Rep. C;PSCI 99-09, University of Massachusetts, MA, Sep 1999

[5] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and ist empirical validation," Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 303–314, Aug. 1998.

[6] Victor Firoiu and Marty Borden, "A study of active queue management for congestion control," in Proceedings of IEEE Infocom, Tel Aviv 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00), Mar. 2000, pp. 1435–1444.

[7] Christof Brandauer, "Web interface to the WRED model," 2002, http://www.salzburgresearch.at/~cbrand/WREDmodel.

[8] Network Simulator ns-2, see http://www.isi.edu/nsnam/ns/.

[9] R. Stevens, *UNIX Network Programming: Networking APIs: Sockets and XT*I, vol. 1, Prentice Hall, Second edition, 1998.

[10] V. Jacobson, C. Leres, and S. McCanne, "tcpdump - dump traffic on a network (man page)," http://www.tcpdump.org/.

[11] 26. Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking 1 (1993) 397-413

8 A proposal for TCL 4 queue management

8.1 Status Quo

In the first specification of the AQUILA architecture [1] it is proposed to implement TCL 4 queue management by means of WRED with two sets of (minth,maxth, maxp) – one for in-profile and one for out-of-profile packets. The choice of parameters is based on the quantitative RED model proposed in [ZFB01]. This model has been developed to optimize the behavior of RED with bulk-data TCP flows. To enable a distinction between in-/out-of profile packets, the RED model has been extended to a WRED model in [1] and this WRED model is used to determine a parameter set for the WRED queue of TCL 4. The configuration of the TCL 4 queue is sketched in Figure 1.

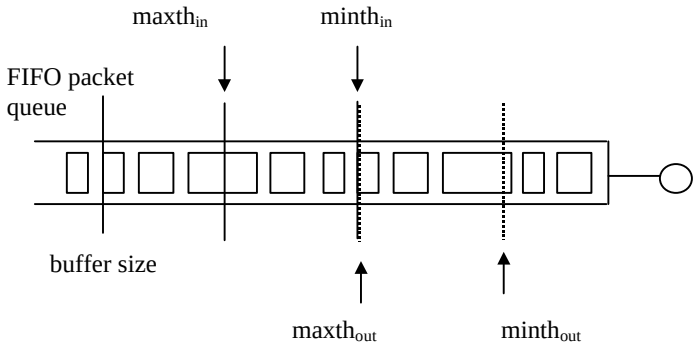


Figure 1 – Configuration of TCL 4 queue – D1301 spec.

8.2 Drawbacks of the first proposal

The first specification of TCL 4 queue management has some drawbacks. In the following we want to give some background information on the relevant traffic control aspects.

8.2.1 Short summary of RED

The RED [3] algorithm has been designed to detect **incipient** stages of congestion as this enables early notification of traffic sources. Therefore, RED calculates an average queue size using a low pass filter and bases its decision whether or not to signal congestion to data senders on this average. Signalling congestion usually means dropping a packet but it could also translate into marking a packet, e.g. if ECN [2] is employed.

The main design goals of RED are to provide **congestion avoidance** by controlling the average queue size, avoid global synchronization and avoid a bias against bursty traffic.

RED compares the average queue size to two thresholds: a minimum threshold *minth*, and a maximum threshold *maxth*. If the average is below minth, no packets are dropped. If the average exceeds maxth, all arriving packets are dropped. If the avg is greater than minth but smaller than maxth, an arriving packet is dropped with a probability. This probability increases linearly from minth where it is zero, to maxth where it equals maxp. Figure 2 illustrates this dropping logic of RED.

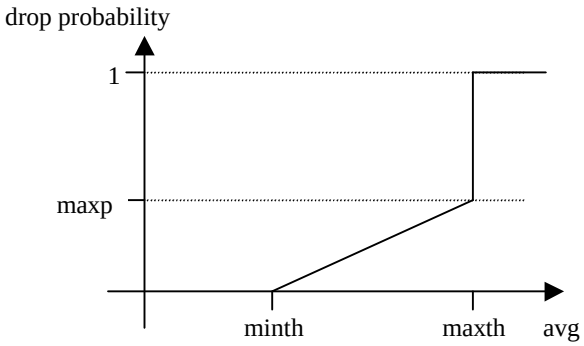


Figure 2 – RED packet dropping probability depending on the average queue size

The idea of dropping packets early (before the buffer overflows) is that signalling incipient congestion to responsive sources should require only few packet drops, while eliminating sustained congestion usually requires several consecutive packet drops. Note that RED may drop packets although the (instantaneous) queue size is smaller than the available buffer size.

It has been shown in [4] that if bulk-data TCP flows send through a properly configured RED gateway, the RED average queue size converges between minth and maxth which is a desireable behavior. This is valid for long-lived TCP connections which always have data to send. Such connections reach an equilibrium state after several RTTs. From then on, they reside mostly in congestion avoidanc. It has become clearer in recent discussions that TCL 4 traffic does not consist of such type of traffic and we can not directly apply those results on a one-to-one basis.

8.2.2 TCL 4 traffic

TCL 4 traffic is supposed to be comprised of

- applications that generate short-lived flows which typically send a few packets (maybe <5) and finish within a few RTTs (maybe <3) or
- applications which generate non-greedy, low bandwidth flows that may live for a long time (up to several hours).

The main QoS requirement of TCL 4 applications is an extremely low loss probability for in-profile packets and a low queueing delay to retain the feeling of interactiveness.

The short-lived connections exhibit a very different sending behavior compared to long-lived connections. When a connection starts sending, it performs an initial "slow start", where the initial transmission window is small (usually 1 or 2 segments) but increases rapidly (it roughly doubles every RTT). The TCP sender will hardly ever reach the congestion avoidance phase – simply because it does not have enough packets to send. Such short-lived TCP senders are much harder to control because their sending pattern is very bursty and heavily influenced by user behavior.

The long-lived, non-greedy flows again show a different behavior: they usually don't fully utilize their transmission window – simply because they don't have data to send. The transmission pattern depends mostly on user behavior.

8.2.3 Random dropping and TCL 4

In this section we want to enumerate some reasons, why RED is rather contraproductive than supportive for TCL 4 queue management.

8.2.3.1 Theoretical traffic control considerations

- As far as TCL 4 IN packets are concerned, *persistent congestion* is a state that must never occur in our QoS network! In AQUILA, this state is achieved by means of provisioning and admission control. As the primary objective of RED is to provide congestion avoidance and control, it is not of much use here.

- Due to the bursty nature of TCL 4 traffic, there are periods of *transient congestion*. With respect to the TCL 4 QoS requirements, it is important that these burst can be fully buffered without having to drop packets because this would increase the drop probability and in turn the delay due to the need for retransmissions. As RED drops packets before the maximum buffer size is reached it is contraproductive in this context.

- Short-lived TCP connections are mostly in slow start. If they notice a packet drop, they usually have to wait for a timeout and will then slow start again (fast recovery won't be triggered because the window size is too small). Counterproductive.

  In total, early signalling of incipient congestion increases the probability for packet drops (compared to conventional TailDrop) which is very undesireable for TCL 4.

- RED should avoid global synchronization. However, this effect is known to exist only in lightly-loaded scenarios with bulk-data TCP connections and TailDrop queue management. It is not relevant for bursty traffic sources.

8.2.3.2 Service protection

Another issue that must be taken into consideration (for each TCL) is the susceptibility to denial of service through (a) misbehaving user(s). Please note that this term does not only refer to a human being with destructive intentions. It may, e.g., also apply to a flow that was (for whatever reason) put into TCL 4 although it should have actually gone into TCL 1, 2 or 3. Such a mis-classified flow has the potential to damage the QoS of all other flows that are otherwise properly serviced within TCL 4.

A strong protection against misbehaving users is, of course, desired. It is thus an important quality criterion that must be taken into consideration when evaluating the appropriateness of a queue management strategy for any of the AQUILA TCLs. It is shown below in section 8.4 that the current TCL 4 proposal [1] exhibits only a weak service protection whereas the new proposal is a strong performance boost in this respect.

8.2.3.3 Simulation results

Finally, we made simulations to see if we could find evidence for the above theoretical arguments. However, as long as our traffic sources stayed within their contracted traffic descriptor (hardly any OUT packets) and we did not increase the number of flows beyond what is allowed by TCL 4 admission control it was impossible to show any drawbacks of the WRED scheme. This is simply due to the conservative AC. In order to obtain a better understanding of the dynamic behavior of the new proposal (see section 8.3 below) and to compare with the original proposal, we increased the number of flows beyond the AC limit. Some important simulation results are summarized in section 8.4.

8.3 New proposal for TCL 4 queue management

As argued in Section 2 it is reasonable to replace the RED mechanism. We propose to use a simple TailDrop scheme for handling TCL 4 traffic.

TCL 4 traffic is subjected to a dual bucket conditoner where packets are marked as IN or OUT of profile. There must, of course, be some mechanism, to differentiate between IN / OUT packets at the TCL 4 buffer of the router output port. The following goals should be met:

- very low packet drop probability for IN packets
- low delay for IN packets
- forwarding of OUT packets in times of sufficient capacity
- strong protection against OUT packets in the sense of

- unacceptably high queueing delay for IN packets
- unacceptably low buffer space remaining for IN packets which would result in an increased dropping probability for IN packets

To reach these goals, we propose to use the following queue management mechanism (see Figure 3):

- FIFO queue
- 2 different drop thresholds, one for IN packets, another one for OUT packets
- the drop threshold for OUT packets is very low.
- the drop threshold for IN packets equals the total buffer size.
- dropping logic:
  - arrival of an OUT packet: if the (instantaneous) queue size exceeds the drop threshold for OUT packets, the arriving packet is dropped; otherwise it is enqueued at the tail of the queue.
  - arrival of an IN packet: if the (instantaneous) queue size exceeds the drop threshold for IN packets (= total buffer size), the arriving packet is dropped; otherwise it is enqueued at the tail of the queue.

Figure 3 sketches the design of the new TCL 4 proposal:

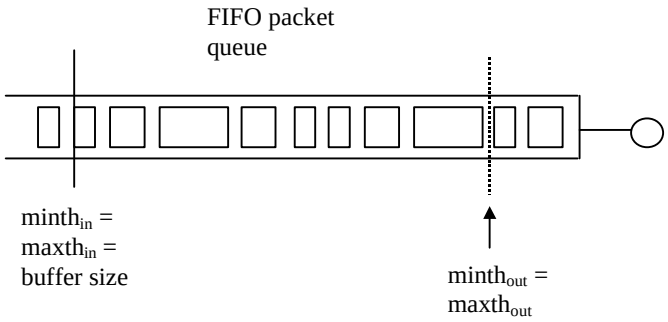


Figure 3 – Proposal for new TCL 4 queue management

The value for the OUT drop threshold should depend on the total buffer size and determines the ratio of buffer space that may at most be occupied by OUT packets. It will probably be in the range of a few packets. The total buffer size should be high to enable large bursts to be buffered without packet loss.

It is possible to enable such a queue management behavior with the equipment that is currently available within our project. Therefore, one has to (ab-)use the WRED mechanisms with the following parameter set:

- threshold_out = minth_out = maxth_out, maxp_out = 1
- threshold_in = minth_in = maxth_in = buffer size, maxp_in = 1
- wq=1

Note that such a configuration effectively eliminates any randomness in the dropping behavior!

It has been noted recently that with the currently used Cisco equipment it is not possible to set minth equal to maxth. This does not impose a problem because one simply has to set maxth to the respective value and minth to maxth-1 in order to achieve the desired effect.

8.4 Simulation results

In the simulations shown below the number of exponential flows is increased from 10 (the AC limit) to 90 or 100. For each scenario the queue management specification of [1] is compared to the approach proposed in the previous section. The results focus on dropped IN packets and queueing delay as these are of major interest for TCL 4 traffic. Simulations are run for 5500 seconds – the first 500 seconds are not taken into respect for the statistics.

8.4.1 Topology

Simulations are based on the following topology:

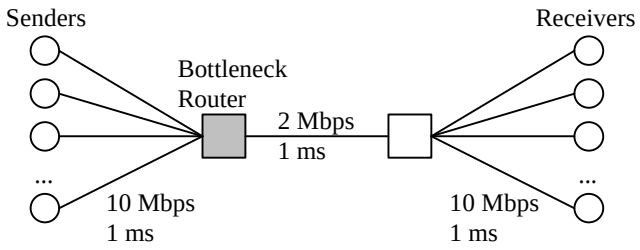


Figure 4 – Simulation topology

8.4.2 Traffic description

PMC traffic is simulated through ON-/OFF sources with exponentially distributed ON/OFF times. During ON time the source sends with a fixed rate. The following configurations for traffic sources and per-flow conditioners are used:

Traffic source	
average on time	4 s
average off time	36 s
Rate	200 kbps
packet size	1000 Bytes
=> avg. Sending rate	20 kbps

Conditioner	
Peak Rate	PR = 200kbps BSP = 2000 Byte
Sustained Rate (BSS is set to 4 times avg burstlen)	SR = 20 kbps BSS = 400000 Byte

TCL 4 Admission Control	10 flows

Table 8-1: Traffic configuration

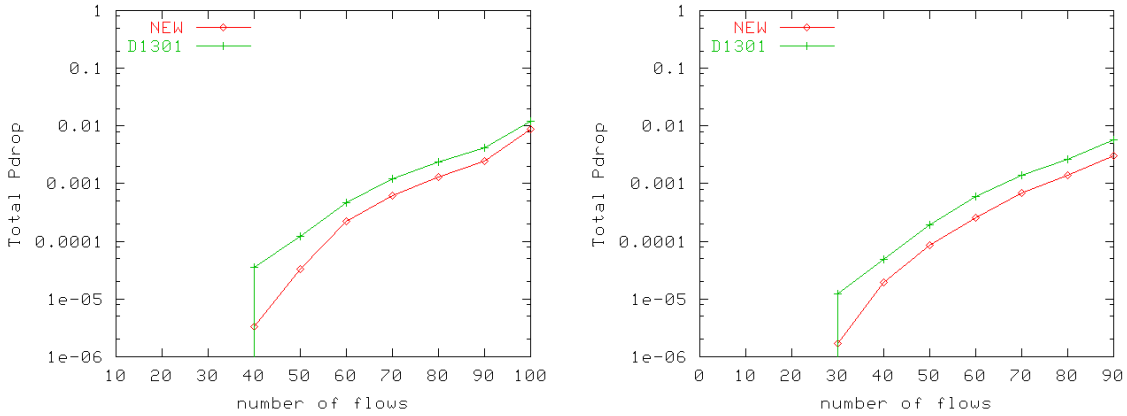
In order to evaluate the susceptibility to denial of service caused by misbehaving users, we repeated each scenario with the addition of 5 FTP flows. These flows send the same traffic descriptor and are thus policed in the same way as the exponential flows.

8.4.3 Achievement of QoS goals

8.4.3.1 Packet drops

Figure 5 shows the packet drop probability of IN packets for the **whole aggregate** of exponential flows (we are currently not able to measure the packet drops per flow). The newly

proposed approach is superior under all simulated load scenarios. The AC limit of 10 flows seems overly conservative.



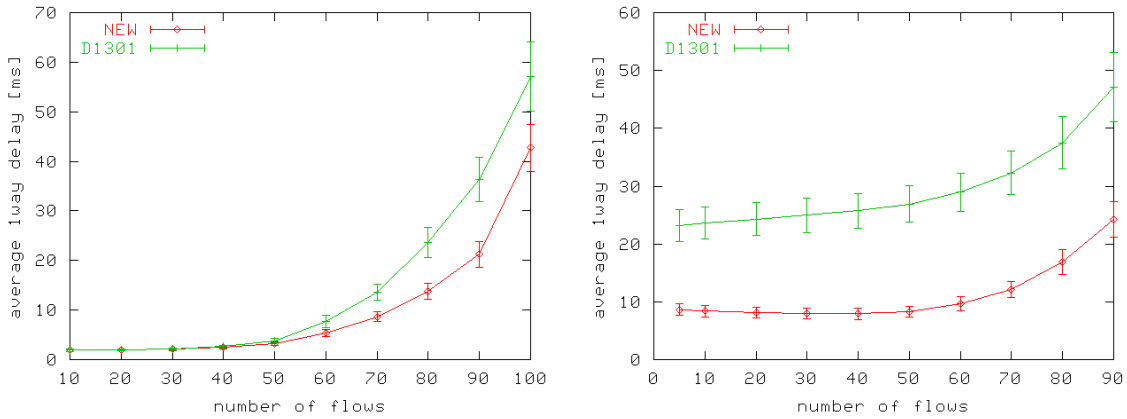
a) Exponential flows

b) Exponential + 5 FTP flows

Figure 5 – Packet drops

8.4.3.2 Queuing delay

As far as the queuing delay is concerned, there is hardly any difference between the proposals up to a load of 50 well-behaving sources (Figure 6 a)). This is simply due to the fact that the queue is empty almost all the time. For higher loads, the average "D1301 queue size" is higher because the new approach uses a lower threshold for OUT packets.



a) Exponential flows

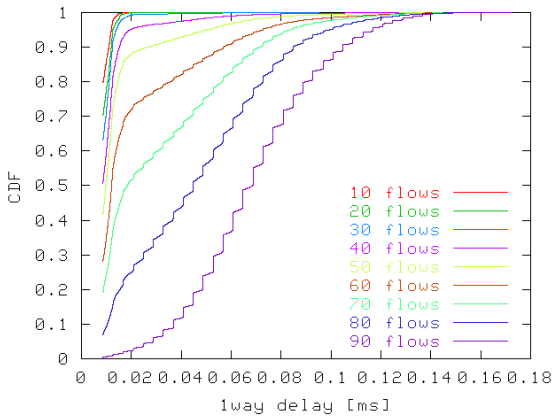
b) Exponential + 5 FTP flows

Figure 6 – Average queuing delay

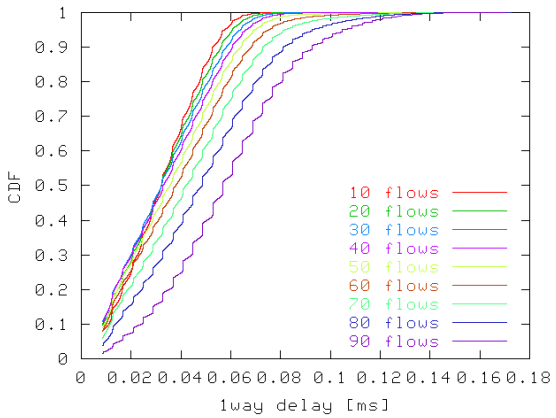
One of the main advantages of the new approach, namely the service protection ability, can be seen in the Figure 6 b). The average queueing delay remains very low even in the presence of 5 misbehaving flows which flood the queue with many OUT packets. The maximum queueing delay that OUT packets can impose on IN packets is configurable through the threshold for OUT packets.

It seems contradictionary that the red curve in Figure 6 b) decreases slightly as the number of exponential flows increases from 5 to around 40. Our analysis showed that as the load increases, the TCP agents of the FTP flows get more timeouts. This results in a burstier sending behavior and a somewhat smaller average queue size.

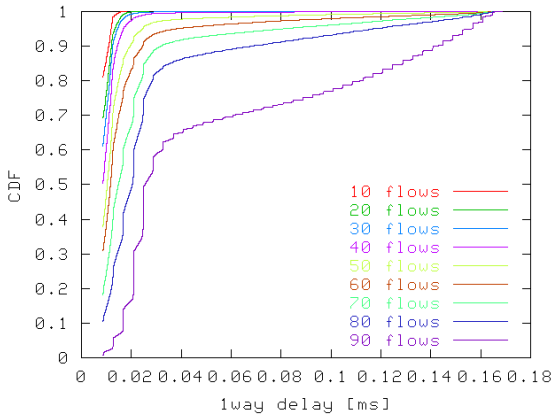
The empirical cumulative distribution function of the queueing delay is depicted in Figure 7. The superiority of the new approach is clearly visible.



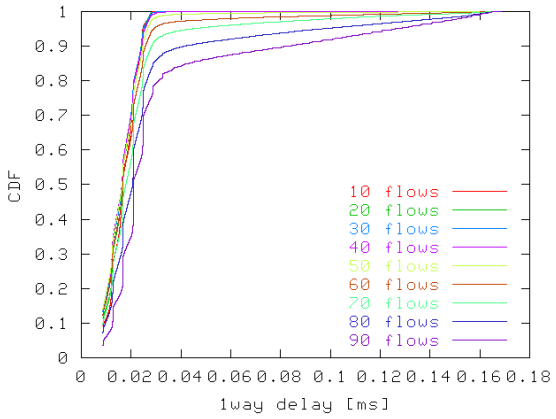
a) Exponential flows – D1301



b) Exponential flows + 5 FTP flows – D1301



c) Exponential flows – NEW



d) Exponential flows + 5 FTP flows – NEW

Figure 7 – CDF of queuing delay

8.5 Summary

In this document the queue management strategy for the TCL 4 traffic is revised. It is argued by theory and simulation that the first specification as made in [1] is suboptimal. The new approach performs better in terms of QoS guarantees and provides much stronger protection against misbehaving users. Moreover it is simpler to configure and can be employed with the equipment that is currently used in the AQUILA project.

8.6 References

[1] Deliverable D1301, Specification of traffic handling for the first trial, AQUILA project consortium, http://www.ist-aquila.org/, September 2000.

[2] K. Ramakrishnan and S. Floyd. RFC 2481: A proposal to add Explicit Congestion Notification (ECN) to IP, Jan. 1999.

[3] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, 1(4):397–413, Aug. 1993.

[4] T. Ziegler, S. Fdida, C. Brandauer, "A quantitative model for parameter setting of RED with TCP traffic", In Proceedings of the Ninth International Workshop on Quality of Service (IWQoS), 2001, Karlsruhe, Germany

9 Comparative studies of declaration- and measurement-based admission control algorithms for IP QoS networks

The objective of a well-designed AC algorithm is to admit as many flows as possible for getting link utilisation at reasonable level, while still maintaining the target QoS. The AC methods can be grouped in two general categories [3]: (1) methods that take into account only traffic declarations submitted by the users during the connection set-up phase, called DBAC (*Declaration-Based Admission Control*) and (2) methods that are additionally supported by some measurements of real carried/offered traffic, called MBAC (*Measurement-Based Admission Control*).

We discuss the problem of appropriate AC algorithms for the purpose of PVBR network service in AQUILA network. The streaming traffic associated with this service is generated by applications like voice or live video, which require information transfer in real (or near real) time. For this network service so-called REM [2], [3] (*Rate Envelope Multiplexing*) scheme with bufferless link model (in fact, with small buffer required only for absorbing simultaneous arrivals of packets) is recommended. Let us assume that source *i* submits its traffic to the network with instantaneous rate $X_i(t)$ (see Figure 9-1). $X_i(t)$, $i=1,\ldots,n$, is thus a set of independent random variables, following in general case an arbitrary distribution. In fact, assuming that the considered stochastic process has stationarity properties, it is sufficient to omit the time index *t* and deal with the stationary distributions of X_i.

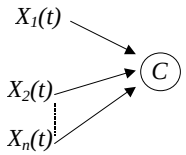


Figure 9-1. Buferless link model (n sources submit traffic to the link of capacity C)

Furthermore, let us define overflow as such event, that the total instantaneous rate of all active sources exceeds the link capacity. So, the probability of overflow is a probability of such event, that

$$S_n = X_1 + X_2 + \ldots + X_n \geq C \qquad (1)$$

As the overflow event will likely cause packet losses, we assume that packet loss probability P_{loss} is approximately equal to the overflow probability. In order to develop AC algorithm for bufferless multiplexing, appropriate traffic model must be chosen (described in form of parameters understandable for the user and sufficiently precise for the purpose of traffic control). Then, based on this model, probability of an overflow event (which constitutes a meaningful QoS parameter) should be calculated as a function of a number of admitted flows. All needed computations must be simple enough to enable implementation of the AC algorithms in a scalable architecture.

We evaluate the selected algorithms for DBAC and MBAC implementations for supporting Premium VBR service in the IP QoS network. In this case, the recognized DBAC algorithms usually operate on the notion of effective bandwidth assuming dual token bucket traffic characterization. On the contrary, for MBAC algorithms a simplified traffic description based on single token bucket is sufficient, but some additional traffic measurements should provide valuable information about the carried/offered traffic. The algorithms discussed in the paper are: the Hoeffding bound [10] for MBAC and the approach suggested by Lindberger [2] for DBAC. It appears that none of these methods is ideal and all have advantages and breakdowns. Therefore, the approach based on co-existence of DBAC and MBAC is proposed.

9.1 DBAC method

In a DBAC [2] method for Premium VBR service, the admission decision is performed only based on traffic descriptors submitted by the users at the beginning of each connection. The user traffic is characterised by the parameters of *dual token bucket* algorithm (with declared peak rate *h* and sustainable rate *r*).

9.1.1 DBAC algorithm based on effective bandwidth

The effective bandwidth represents the amount of resources (link capacity in this case), which must be reserved for a given flow in order to serve it with assumed QoS level. The value of effective bandwidth can be calculated for example using the formula developed by Lindberger and Tidblom [2].

$$Eff(h,r) = \begin{cases} a \cdot r(1+3z(1-r/h)) & if\ \ 3z \le \min(3, h/r) \\ a \cdot r(1+3z^2(1-r/h)) & if\ \ 3 < 3z^2 \le h/r \\ a \cdot h & otherwise \end{cases} \qquad (2)$$

where

$$a = 1 - \frac{\log_{10} P_{loss}}{50} \quad and \quad z = \frac{-2\log_{10} P_{loss}}{C/h} \qquad (3)$$

New flow is admitted only if the sum of its effective bandwidth and cumulative effective bandwidth of running flows does not exceed the link capacity. Otherwise, the flow is rejected.

9.1.2 Practical limitations of DBAC: difficulties of tuning token bucket parameters

It appears that it is rather difficult for a user to precisely specify the values of traffic descriptors at the beginning of the connection. Declaring lower values of traffic descriptors than the submitted traffic can cause undesired traffic losses due to the policing mechanism. On the other hand, values of traffic descriptors greater than submitted traffic simply lead to network under-utilisation. Since the user is usually uncertain about the values of parameters character-

ising his traffic, he chooses rather greater values than are really needed. The values of token bucket parameters needed for MPEG video source are presented in Figure 9-2. In order to limit packet losses, P_{loss}, caused by the policing mechanism to a level of 10^{-4}, a user should declare the value of *r* (corresponding to token bucket rate) about 3 times larger that the mean rate of submitted traffic.

Even in the case that the user is able to fix his traffic descriptors in accurate way, the characterisation of traffic by token bucket assumes so called worst-case traffic pattern, which can occur sporadically in real submitted traffic. Therefore, one can expect that the DBAC methods are rather conservative, in most cases leading to network under-utilisation.

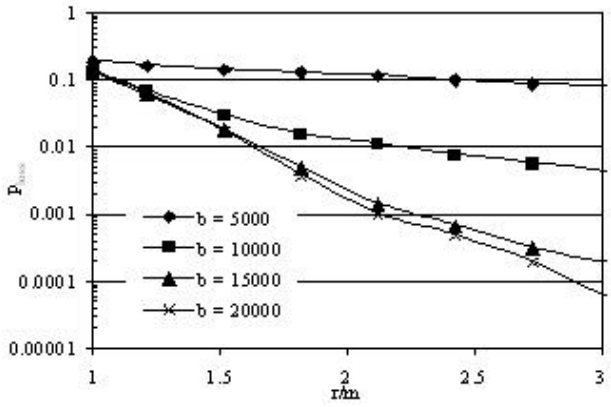


Figure 9-2. Tuning token bucket parameters for real video source. r – token bucket rate (sustainable bit rate), b – token bucket size, m – mean bit rate of the submitted traffic

9.2 MBAC method

The MBAC algorithms were developed to take into account not only the user declaration, but also the real traffic, which is carried in the network.

We can distinguish between two general types of MBAC algorithms:

- Methods based on link load measurements [9, 10, 11, 12]: parameters corresponding to load generated on a given link by individuals flows or their aggregates are measured. The AC decision is separated from the measurement process. In principle, these algorithms are similar to the DBAC methods in that they use some parameters of the offered traffic to calculate resource requirements of the flows. The main difference is that they obtain traffic descriptors from measurements rather than from declarations.

- Methods based on probing [13, 14, 15]: active measurements (flow probing) are performed in order to evaluate QoS experienced by packets transmitted in the network. Previously admitted traffic can be probed, or additional artificial flows can be set-up only for the purpose of evaluation of current state of the network. These algorithms

are inherently based on distributed measurements and the AC decision is strictly related with the measurement process.

The methods based on traffic probing are well suited for distributed environment, but they are more complicated in implementation. The probing agent either has to be built into the user application (which may lead to the security concerns), or has to be implemented by additional hardware co-located with the edge device. Considering this, rather the methods based on link-oriented measurements are considered for practical implementation in AQUILA network [6].

By applying effective MBAC algorithms, we can expect the following:

- To simplify the traffic declaration. Usually it is difficult for the user to specify accurate parameters other than the peak rate.
- To take into account real amount of traffic carried by the network. Since the real traffic can be quite far from the declared values, this should result in better network utilization (more accepted flows).
- To capture stochastic nature of the user traffic more accurately than it is possible with DBAC (which assumes the description of traffic by deterministic parameters). The mean rate, for example, is the most important parameter for traffic characterisation, but cannot be exactly captured by the token bucket mechanism.

Summarizing it is expected, that MBAC algorithms should be more efficient comparing to DBAC. Especially, the advantages of MBAC should be significant in the case of essential differences between traffic declarations and this what is observed in the network.

9.2.1 MBAC algorithm based on the Hoeffding Bound

MBAC algorithm based on the Hoeffding upper bound for the tail of probability distribution of sum of independent random variables was widely studied in the literature [9, 10]. Below, we recall the main properties and assumptions of the Hoeffding bound and present their implications for the performance of MBAC scheme.

According to the Chernoff formula (see for example [4]), upper bound for the probability that the sum S_n of arbitrarily distributed random variables X_i exceeds the value C, is given by:

$$P[S_n \geq C] \leq e^{\inf_s [M(s) - sC]} \tag{4}$$

where $M(s) = \sum_i M_i(s)$ is the sum of logarithmic moment generating functions of random variables X_i:

$$M_i(s) = \log E\left[e^{sX_i}\right] \tag{5}$$

The function $M_i(s)$ depends on the probability distribution of random variable X_i. Therefore, in order to calculate the upper bound for the sum S_n, we must make some assumptions about

the probability distribution of random variables X_i. Choosing appropriate probability distribution function, which effectively models the traffic generation process of the "real" source, is quite difficult. Two approaches are usually adopted for this purpose.

First approach assumes that the source submits traffic according to the ON-OFF model, with peak rate *h* and mean rate *m*. The probability distribution function of X_i is then:

$$X_i = \begin{cases} h \text{ with prob. } p \\ 0 \text{ with prob. } 1-p \end{cases} \tag{6}$$

where $p=m/h$. Calculating the logarithmic moment generating function of (6) and substituting it into equation (4) leads to the formulation of MBAC algorithms based on the Chernoff bounds [11]. Because this MBAC method requires per-flow measurements of mean rate, and moreover the calculation of Chernoff bound is in this case computationally demanding, the practical implementation of this algorithm can be quite difficult.

The second approach [9] assumes that random variable X_i follows the Normal probability distribution $N(m,s^2)$, with mean value *m* and variance s^2. The logarithmic moment generating function of such random variable is equal to:

$$M_i(s) = s \cdot m_i + \frac{1}{2} s^2 s_i^2 \tag{7}$$

Now, let us assume for the moment, that the peak rate *h* is the only known parameter characterizing the traffic source. It can be shown, that variance of such source is greatest, when the traffic pattern is of ON-OFF type and its mean rate is equal to *h/2*. We can substitute the worst-case variance resulting from such model $s^2=h^2/4$ into the formula (7). Now, rewriting formula (4) and calculating the minimum on the right side of the equation leads to (8) - Hoeffding bound for the tail of probability distribution of sum of bounded random variables. If we further demand, that the overflow probability should not exceed some target value, say $P(S_n \geq C) \leq e^{-g}$, we can write the following inequality:

$$P(S_n \geq C) \leq e^{-2\frac{\left(C-\sum_i m_i\right)^2}{\sum_i h_i^2}} \leq e^{-g} \tag{8}$$

The second inequality can be stated as:

$$\sum_i m_i + \sqrt{\frac{g}{2}\sum_i h_i^2} \leq C \tag{9}$$

If inequality (9) is satisfied (with a given set of sources characterized by the values of h_i and m_i), the overflow probability is not greater than e^{-g}. The measurement based admission control based on Hoeffding bound works as follows. The value of $\sum_i m_i$ (let us denote it as *M*) is estimated by the measurement of aggregated traffic load on the link. The user declares

peak rate h_{new} of the new traffic flow. If inequality (10) is then satisfied (where h_i are the peak rates of previously accepted flows, and P_{loss} is the target overflow probability), the new flow is admitted. Otherwise, the flow is rejected.

$$h_{new} + M + \sqrt{\frac{-\ln P_{loss}}{2} \sum_i h_i^2} \leq C \qquad (10)$$

The analysis presented above suggests, that two important assumptions taken in the derivation of the Hoeffding bound formula may have impact on the efficiency of considered AC schema:

- Assumption that the instantaneous rate of each source is modelled by the random variable with Normal probability distribution function
- Assumption of the worst-case variance, which is the variance of ON-OFF source with mean rate equal to ½ of the peak rate

When real traffic submitted to the network significantly differs from this model (for example its variance is smaller than the assumed worst-case), the efficiency of AC can be quite low. This is illustrated in Figure 9-3 and Figure 9-4. The theoretical value of overflow probability was calculated for different traffic conditions (different number of sources admitted to the system). The considered sources are of ON-OFF type, with the following parameters: *h=2Mbps*, *m=1Mbps* in case#1, and *h=6.1Mbps*, *m=0.66Mbps* in case#2. In both cases, the link capacity is equal to 100Mbps.

The line denoted as “binominal” represents the value of overflow probability calculated assuming the binominal distribution of the number of sources being in ON state. This can be treated as the reference scenario, because for ON-OFF sources such approach gives the exact result. The “Chernoff bound” line represents the upper bound for the overflow probability calculated using the Chernoff formula (4), with X_i function given by (6). Note that in this case the real source model is consistent with the traffic model assumed for the derivation of the Chernoff bound. The “Hoeffding bound” line is calculated using the Hoeffding bound (8). It is clearly visible, that Hoeffding formula constitutes a conservative upper bound for the overflow probability.

In case#1 (see Figure 9-3), the overflow probability was calculated for sources with peak rate *h=2Mbps* and mean rate *m=1Mbps*. Note, that the worst case variance ($h^2/4=1$) assumed for the derivation of Hoeffding bound is equal to the variance of such source ($s^2=1$). Therefore, the value of Hoeffding upper bound is quite close to the Chernoff bound.

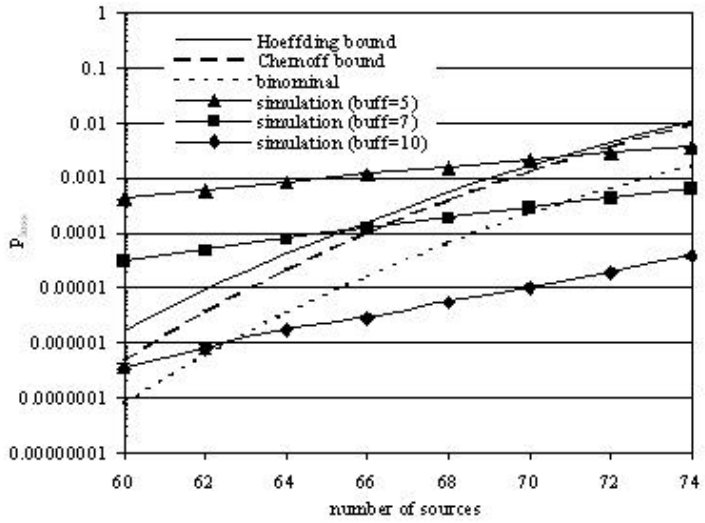


Figure 9-3. Probability, that the total rate exceeds the link capacity C=100Mbps in case #1; h=2Mbps, m=1Mbps

In case#2 (see Figure 9-4), the real variance of the considered source ($s^2=3.59$) is much smaller than the assumed worst-case ($h^2/4=9.3$). Therefore, the Hoeffding bound is now much higher than the Chernoff bound and significantly overestimates the real value of overflow probability. This overestimation can lead to unnecessary conservativeness of MBAC schema.

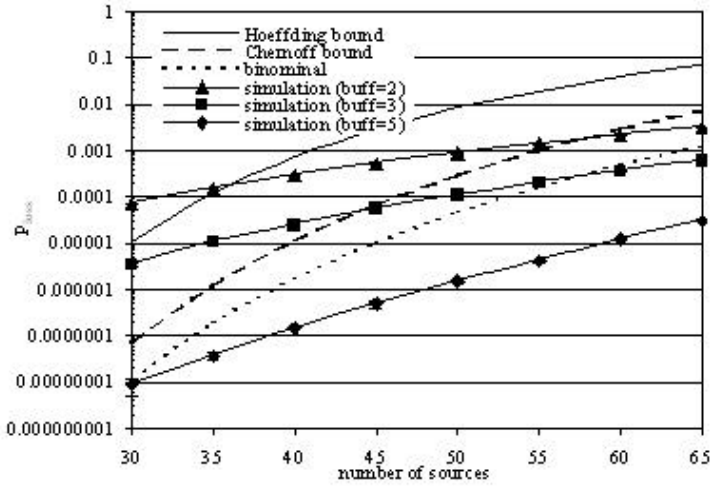


Figure 9-4. Probability, that the total rate exceeds the link capacity C=100Mbps in case #2; h=6.1Mbps, m=0.66Mbps

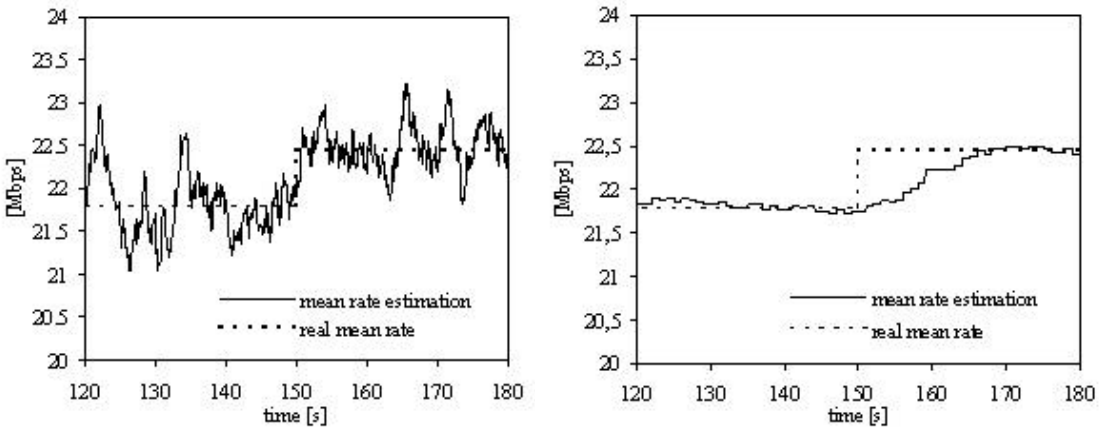
In both cases, larger difference between the two bounds is observed when the number of sources is rather small. The reason for this is the fact, that the Normal distribution effectively approximates superposition of random variables only in the case when the number of independent variables is sufficiently large.

The remaining graphs correspond to the simulation results. Simulations were carried out with exponential ON-OFF traffic sources. The experiments were repeated with different values of buffer size (5, 7, 10 packets in case#1 and 2, 3, 5 packets in case#2). A small buffer is needed in both cases for absorption of simultaneous packet arrivals. Anyway, when the applied buffer is larger (e.g. more than 5 in case#2), we can see that AC methods developed for bufferless multiplexing tend to be rather conservative.

9.2.2 Practical limitations of MBAC: measurement of the mean rate

The efficiency of MBAC strongly depends on the method, which is used for the link load measurements and mean rate estimation. Usually, the window-based mean estimation algorithms are used for this purpose. The number of bits transmitted by the link is measured in small time intervals (called sampling intervals, T_s). Then, the estimated mean rate is calculated as average value of rate measured in W latest sampling intervals (the value of W is the length of measurement window).

Length of the sampling interval is a critical parameter for the quality of mean rate estimation. This is illustrated by simulation results presented in Figure 9-5. The time plots show the value of mean rate estimator, compared to the real total mean rate of all admitted sources (the source model is exponential ON-OFF with *h=6.1Mbps* and *m=0.66Mbps*). The number of sources is 33. At time 150s, a new flow arrives to the system. When T_s is set too small (T_s*=100ms* with *W=10* in Figure 9-5a), the algorithm is not able to smooth out the traffic variability and we observe undesired oscillations of mean rate estimation. On the other hand, if T_s is set too large (T_s*=1s* with *W=10* in Figure 9-5b), the effect of mean rate of newly arriving flow is fully observed after about 20s, which is the length of the measurement window. Such a long time needed for the update of mean rate estimation can cause errors in the admission decisions.

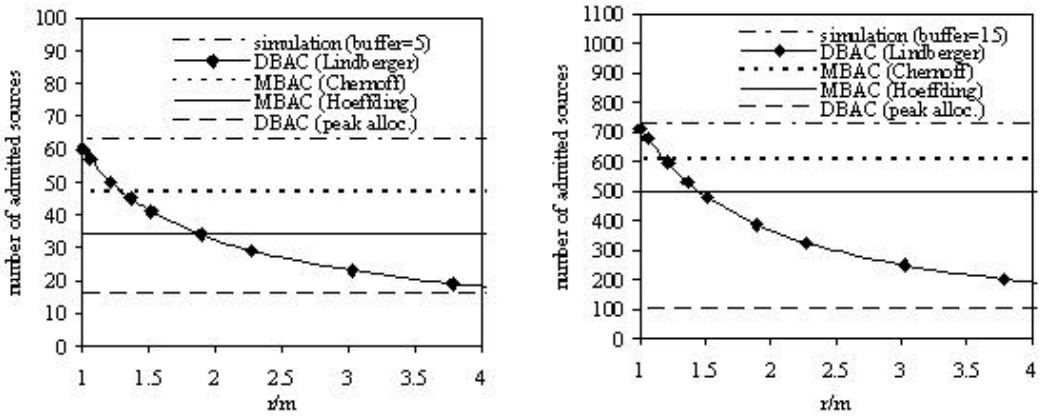


a) b)

Figure 9-5. Estimated mean rate of aggregate traffic; length of sampling interval is equal to a) 100ms, and b) 1s.

9.3 Evaluation of performance of DBAC and MBAC algorithms

Figure 9-6 shows the maximum number of sources admissible with different AC algorithms. The homogenous case is assumed and the parameters of the sources are: *h=6.1Mbps* and *m=0.66Mbps*. The number of admissible flows is compared with the simulated admission region, which is equal to 63 sources on the 100Mbps link with 5 packets buffer size, and 725 sources on the 620Mbps link with 15 packets buffer size. The value *r/m* denotes the level of over-declaration, i.e. how much the declared value of sustainable rate *r* is greater than the mean rate *m*. It is assumed, that the level of over-declaration is the same for all traffic sources.



a) b)

Figure 9-6. Efficiency of different AC algorithms; link capacity is equal to a) C=100Mbps, and b) C=620Mbps.

We observe, that the differences between AC algorithms are significant. It should be noted, that when a user properly declares his traffic parameters, the DBAC method based on the effective bandwidth performs well and allows for quite high link utilisation. When the level of over-declaration is high, MBAC methods (which attempt to estimate the traffic parameters from measurements of real traffic load), allow for admitting much more flows than DBAC. The MBAC based on Chernoff bound seems to be more effective than the Hoeffding bound, but as it was stated before, its practical applicability can be limited due to high implementation complexity.

As it was expected, the efficiency of Hoeffding bound MBAC is much better when the link capacity is larger. On the 100Mbps link (Figure 9-6a), the Hoeffding bound MBAC gains advantage over the Lindberger's effective bandwidth approach, when the level of over-declaration of *r* is about 2. On the 620Mbps link (Figure 9-6b), this happens when the level of over-declaration is only about 1.5. The important conclusion is that MBAC allows for exploiting high multiplexing gain on large capacity links.

9.4 Co-existence of DBAC and MBAC

The surprising conclusion from the previous section is that in some situations MBAC algorithms can be more conservative than the DBAC. This conservativeness is not justified by the fact, that admitting additional flows would cause QoS degradation. It rather results from some simplifications and assumptions taken in the derivation of the Hoeffding bound formula. Therefore, the most promising approach seems to be a co-existence of DBAC and MBAC algorithms.

When a new flow arrives to the system, first the DBAC rules are checked. If the DBAC decision is that the flow should be admitted, it is admitted. In other case, the flow is not necessarily rejected. The MBAC rules are applied first and if its decision is positive, the flow is admitted. Therefore, the new flow is admitted if either DBAC or MBAC decision is positive.

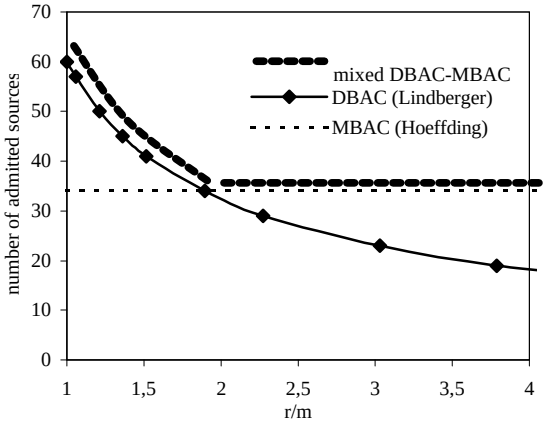


Figure 9-7. Co-existence of DBAC and MBAC algorithms

The effect of proposed modification on the maximum number of admitted flows is illustrated in Figure 9-7. We can observe, that the overall system utilisation can be improved thanks to using DBAC or MBAC, depending on which method allows for better link utilisation in a given traffic conditions. Note, that in all the cases the admissible region is below the simulated value of 63, which assures, that the target QoS guarantees are met.

9.5 Summary

A comparative study of DBAC, MBAC and mixed DBAC/MBAC methods of admission control for Premium VBR service in AQUILA network was carried out. The DBAC algorithms (e.g. Lindberger's effective bandwidth approach) seem to perform quite well when the declared traffic parameters are close to the real traffic submitted to the network. Anyway, difficulties with tuning the proper values for the traffic descriptors cause a serious problem for the effective use of this type of methods. On the other hand, the MBAC methods (e.g. Hoeffding bound algorithm) are designed to take into account not only declarations, but also real traffic submitted to the network. It was shown, that in some cases the MBAC method can be less effective that the DBAC. Thus, the co-existence of DBAC and MBAC algorithms seems to be the most promising schema for admission control.

9.6 References

[1] S. Blake et al., *An Architecture for Differentiated Services*, RFC 2475, December 1998

[2] J. Roberts, U. Mocci, J. Virtamo, Final Report COST 242, *Broadband network teletraffic: Performance evaluation and design of broadband multiservice networks*, Lectures Notes in Computer Science 1155, Springer, 1996

[3] P. Tran-Gia, N.Vicari, Final Report COST 257, *Impact of New Services on the Architecture and Performance of Broadband Networks*, compuTEAM, Wuerzburg, 2000

[4] L. Kleinrock, *Queueing Systems*, Addison Wesley, 1976

[5] A. Bak,W. Burakowski, F. Ricciato, S. Salsano, H. Tarasiuk, *Traffic handling in AQUILA QoS IP network*, Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001

[6] AQUILA Project Consortium, *Deliverable D1302, Specification of traffic handling for the second trial*, http//www.ist-aquila.org, December 2001

[7] W. Burakowski et al, *AQUILA network architecture: first trial experiments*, to appear in Special Issue of Journal of Telecommunications and Information Technology, Warsaw 2002

[8] L. Breslau, S. Jamin, S. Shenker, *Comments on the performance of Measurement-Based Admission Control Algorithms*, IEEE Infocom 2000, March 2000

[9] F. Brichet, A. Simonian, *Conservative Gaussian models applied to Measurement-based Admission Control*, IWQoS'98, USA, May 1998

[10] S. Floyd, *Comments on Measurement-based Admissions Control for Controlled-Load Services*, Technical Report, Lawrence Berkeley Laboratory, July 1996

[11] R.J. Gibbens, F.P. Kelly, *Measurement-based connection admission control*, 15th International Teletraffic Congress, June 1997

[12] M. Grossglauser, D.Tse, *A Time-Scale Decomposition Approach to Measurement-Based Admission Control*, IEEE Infocom 1999, March 1999

[13] L. Breslau et al., *Endpoint Admission Control: Architectural Issues and Performance*, In proceedings of ACM SIGCOMM 2000

[14] J.W. Roberts et al, *Integrated Admission Control for Streaming and Elastic Traffic*, Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001

[15] I.M. Ivars, G. Karlsson, *PBAC: Probe-Based Admission Control*, Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001

[16] W. Burakowski, M. Dabrowski, *Wielouslugowa siec IP QoS: architektura i praktyczna weryfikacja w instalacji pilotowej*, Przeglad Telekomunikacyjny, nr 5/2002 (in polish)

10Admission control based on advertised window setting for TCL3 class

The TCL3 class is designed for effective and guaranteed service of traffic generated inside admitted TCP-controlled connections. A candidate for using this TCL is e.g. the FTP application. The TCL3 should give some guarantees/assurances for throughput/goodput. In particular, using FTP a user wants to transfer the file in a foreseeable time duration, possible short. However, the simulation studies indicated that the variability of file transfer times could be significant even for TCP connections admitted according to the implemented AC algorithm as described in [7] . These results motivated for reviewing the applied AC rules. In this part we describe a new, we believe, more efficient AC algorithm. The innovative in this algorithm is that it takes into account the token bucket features, TCP behaviour and RTT (Round Trip Time).

10.1 Factors influencing AC

Before dealing with the details of the proposed AC algorithm for TCL3 class, we start with discussing around consequences of the impact of TCP traffic control mechanisms and limitation of traffic description by token bucket on AC. We argue that these two elements should be taken into account for designing of efficient AC.

There have been several implementations of TCP protocol. The most important modifications was the introduction of congestion and avoidance techniques. This version is usually referred to as Tahoe TCP. Tahoe TCP regulates the number of packets and sends by inflating and deflating a window according to the network requirements. In order to do this, the TCP sender uses the cumulative acknowledgements sent by the receiver. If no packets are lost, TCP keeps inflating the window in the three main phases: slow start, congestion avoidance and maximum window [1,16,17,18], Reno, New-Reno and Sack were designed to improve the performance of Tahoe TCP when packets are lost. However, when no packet loses occur they behave like Tahoe TCP. Thus, in the ideal case of no packet loses all these different implementation should have the same performance.

From the point of view of discussed AC algorithm, the most important issue is to control the changes (if any) of transmission window size. In the TCP case, there is rather difficult to predict TCP behaviour when the packet losses are not under control. Therefore, for the purpose of AC this is reasonable to consider ideal case of TCP behaviour, as discussed in [10]. Assuming ideal TCP behaviour, we can foresee the transfer time for given file size, just what we expect.

The modelling issues of TCP behaviour in best effort Internet network taking into account impact of packet losses on received throughput were discussed by many authors, e.g. [13,15]. Additionally, in [15] the authors pointed out that using token bucket marking mechanism is not sufficient way for getting ideal TCP service differentiation. Anyway, these studies as-

sumed the overload network conditions, which are not adequate when AC algorithm is employed.

10.1.1 Consequences of applying token bucket mechanism

Ideal AC should guarantee that accepted TCP connection would get throughput/goodput close to requested rate (RR) value. Let us remind that token bucket mechanism operates on deterministic parameters while TCP sends packets in a non-regular way accordingly to current sending/transmission window and round trip time (RTT). As a consequence, one can observe that the traffic produced by a single TCP source has a tendency to follow rather statistical behaviour than deterministic. Therefore, we argue that this what could be policed by the token bucket should be closely related to the value of TCP sending window size, like maximum guaranteed/requested value of TCP sending window size, W^{req}. Let us also remind, that token bucket mechanisms is dimensioned for policing the worst case traffic, which could occur sporadically that one can observe in TCP traffic. Therefore, the value of W^{req} should be specified for traffic contract (in direct or non-direct way). Starting from this point, we deduce for the token bucket parameters (SR – token accumulating rate, BSS – bucket size) the following relations:

$$L^{in} * BSS /(L^{in} - SR) = W^{req} \qquad \textbf{eq. 1}$$

, where L^{in} denotes the rate of input link (in bps).

The expression eq. 1 shows the way for dimensioning bucket size BSS on the basis of assumed W^{req}. The W^{req} denotes the maximum packet burst size (ON period), counted in bits, which could be emitted by the TCP source. Therefore, the time duration of ON period, T_{ON}, in this case is:

$$T_{ON} = \frac{W^{req}}{L^{in}} \qquad \textbf{eq. 2}$$

$$\frac{BSS}{SR} \geq T_{ON} + T_{OFF} \qquad \textbf{eq. 3}$$

, where T_{OFF} is the interval between consecutive packet bursts (OFF period). From eq. 3 we can obtain additional constrains referring to the minimum value of RTT for TCP connection, RTT^{min}. The condition for RTT^{min} results from eq. 2 and eq. 3, and is the following:

$$RTT^{min} = T_{ON} + T_{OFF} \leq \frac{BSS}{SR} \qquad \textbf{eq. 4}$$

The condition eq. 4 for RTT^{min} results from the fact that the token bucket size should be fulfilled to the maximum value (=BSS) before starting new ON period. Notice that less rigorous traffic pattern (not the worst case) is generated when current RTT is greater than RTT^{min}.

Summarising, from the point of view of the token bucket mechanism the worst case traffic, which a TCP-controlled stream could produce is obviously of ON/OFF type, as depicted on Figure 1.

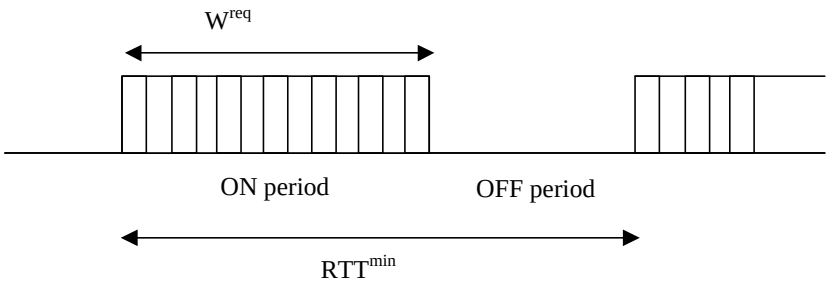


Figure 1 - The worst case traffic produced by a TCP-controlled stream from the point of view of the token bucket mechanism

In fact, the token bucket mechanism has ability for distinguishing between in- and out-of-profile packets. The in-profile packets are in accordance with the traffic declarations while the out-of-profile packets excess the policed profile. Therefore, there is reasonable to consider out-of-profile traffic as an additional, non-controlled traffic, which is not taken into account by AC. So, the only traffic for which we can give some guarantees corresponds to the in-profile traffic. As a consequence, in further part of this contribution we focus only on AC for in-profile traffic.

The requirement for the AC is to give guarantees that the requested by a user volume of bandwidth (throughput), RR, will be provided. Taking into account that we use token bucket mechanism for traffic contract policing, with above mentioned limitations of this mechanism, we argue that additional information about RTT^{min} is needed. The knowledge of the RTT^{min} allows us for proper tuning of the token bucket parameters (see eq. 4). Anyway, the guaranteed RR will never exceed the assumed token accumulating rate SR and this results in straightforward way from the token bucket concept. The relations between RR and SR will be discussed in the further part of this document. The BSS value strictly depends on RTT^{min}; if RTT^{min} increases, consequently BSS should also increase (assuming given SR).

10.1.2 Forming TCP traffic

The desirable shape of sending window size evolution for any admitted TCP connection is shown on Figure 2. In this scenario we assume that no packet losses are observed as long as the send TCP window does not exceed the W^{req}. However, this requires setting W^{req} for advertised window size in the receiver.

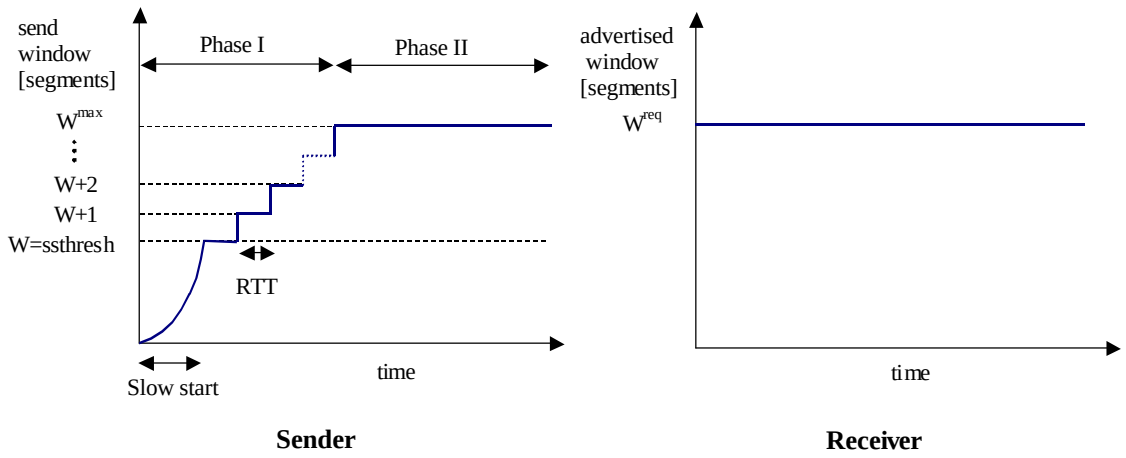


Figure 2 - Ideal TCP behaviour.

Observing the TCP connection rate, one can distinguish between two phases: (I) the beginning of the connection when slow start and congestion avoidance mechanisms govern the TCP behaviour, and (II) the time, when the send window size is stabilised at the value W^{max} equal to advertised window (set in the receiver to the W^{req}). Notice that the RR corresponds to the phase II.

10.2 Traffic description

For the purpose of discussed AC, we assume that traffic offered by a single TCP source is described in the form of token bucket parameters. Such approach allows also us for dimensioning token bucket for in-profile traffic policing. The starting point for the token bucket dimensioning method constitutes the TCP connection demands, which are in this case expressed in the form of:

- the targeted requested rate RR;
- and, the information about minimum round trip time, RTT^{min}.

The next step is to calculate the parameters of associated token bucket (SR, BSS) and advertised window size W^{req} on the basis of (RR, RTT^{min}). For phase (II) of the ideal TCP behaviour the following relation takes place:

$$RR = \frac{W^{req}}{RTT^{avg}} \qquad \textbf{eq. 5}$$

,where RTT^{avg} denotes average round trip time.

The SR corresponds to the worst case of traffic, which can occur when the round trip time reaches the minimum value: RTT^{min}. Therefore, SR is greater than RR. One can write the following relation:

$$SR = \frac{W^{req}}{RTT^{\min}} > RR \qquad \textit{eq. 6}$$

Then,

$$W^{req} = SR * RTT^{\min} \qquad \textit{eq. 7}$$

Substituting eq. 7 to eq. 5 we receive the relation between SR and RR, which is:

$$SR = RR\frac{RTT^{avg}}{RTT^{\min}} \qquad \textit{eq. 8}$$

As one can observe, the evaluation of RTT^{avg} is needed for correct calculating of SR. Notice that RTT^{avg} depends on traffic carried by the network. In the contents of AQUILA architecture, for evaluating RTT^{avg} we may do simplified assumption that variable part of RTT, caused by buffering packets in the routers, is mainly the packet waiting times in the edge routers. The core network is over-dimensioned and, as a consequence, the packet waiting times in core routers are negligible. We can write:

$$RTT^{avg} = RTT^{\min} + D \qquad \textit{eq. 9}$$

,where D is the variable part of RTT^{avg}. Now, we can concentrate on evaluation of D parameter. The proposed approximation for D, which gives accurate results, was derived by assuming M/D/1 queuing system with ρ=RR/SR and constant packet service time equal MTU/SR, where MTU is the maximum transfer unit (in bits). The final formula for SR is:

$$SR = RR + \sqrt{A/(RR + \sqrt{A/(RR + \sqrt{A/RR})})} \qquad \textit{eq. 10}$$

, where

$$A = \frac{RR^2 MTU}{2RTT^{\min}} \qquad \textit{eq. 11}$$

The BSS is calculated in straightforward way from expression eq. 1 by:

$$BSS = \frac{W^{req} * (L^{in} - SR)}{L^{in}} \qquad \textit{eq. 12}$$

The formula eq. 12 is used for BSS calculating at ingress point. From the AQUILA architecture point of view the L^{in} is unknown at egress point, in this case the BSS is simply equal to W^{req}:

$$BSS = W^{req} \qquad \textit{eq. 13}$$

Notice that W^{req} is an upper bound of the BSS.

10.3 Advertised window setting

The advertised window size is a function of maximum TCP segment size (MSS). MSS is usually shorter than MTU. Therefore, for proper setting of advertised window size, we introduce W^{adv} parameter (in bits).

$$W^{adv} = N^{seg} * MSS \qquad \textit{eq. 14}$$

Where N^{seg} is the number of TCP segments and is calculated by:

$$N^{seg} = \frac{W^{req}}{MTU} \qquad \textit{eq. 15}$$

W^{req} is evaluated using eq. 7.

10.4 Declaration based admission control algorithm

Using the above described token bucket dimensioning method we can receive the proper value of the SR as well as BSS parameter. The following expression gives the bandwidth value required by the aggregated TCL3 stream:

$$B_3^{pa}(.) = \sum_{i=1}^{N_{TCL3}} SR_i \qquad \textit{eq. 16}$$

where SR_i is the token rate of the i-th flow, N_{TCL3} is the number of flows in TCL3 class (including the new one).

For buffer requirement, the following condition should be satisfied:

$$\sum_{i=1}^{N_{TCL3}} BSS_i \le Buf_{TCL3} \qquad \textit{eq. 17}$$

where Buf_{TCL3} denotes buffer size dedicated for TCL3 class.

The traffic generated to TCL3 class is:

$$R_3^{pa}(.) = R_3^{db}(.) = \sum_{i=1}^{N_{TCL3}} SR_i \qquad \textit{eq. 18}$$

The input parameters for the TCL3 class DBAC algorithm are presented in Table 1.

Table 1 - DBAC input parameters for TCL3

Parameter	Description	Units
SR_i	Sustainable rate of i-th flow	bps
RR_i	Requested rate of the i-th flow	bps
BSS_i	Bucket size of the i-th flow	bits
RTT^{avg}_i	Average round trip time of the i-th flow	s
RTT^{min}_i	Minimum round trip time of the i-th flow	s
W^{req}_i	Maximum window size of the i-th flow	bits
N_{TCL3}	Number of flows in the TCL3 class (including new one)	-
L^{in}_i	Input link rate of the i-th flow	bps
Buf_{TCL3}	Buffer space for TCL3	bits

10.5 Numerical examples

This section contains simulation results showing the effectiveness of the investigated AC algorithm. In particular, we present the results for TCP connections differ in RR and RTT^{min}. For this purpose, the bottleneck network topology is considered as depicted in Figure 3. In

this configuration two edge routers are connected by the link of 2 Mbps. Furthermore, 5 PCs are connected by LAN Ethernet to Router 1, while 2 servers (Server 1 and 2) are attached to Router2. The TCP connections are established between given pair PC-Server. In addition, we can regulate the RTT^{min} by introducing additional constant delay in the link connecting routers as well as in the links between Router 2 and servers.

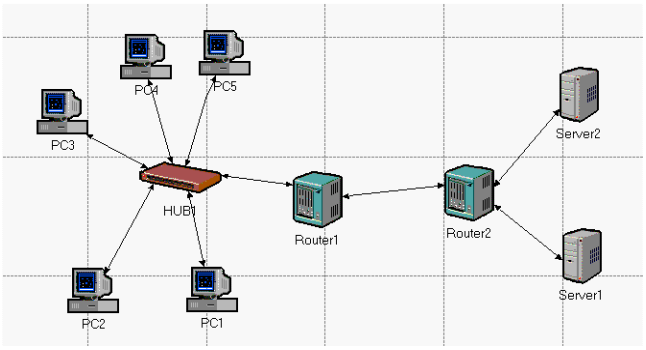


Figure 3 - Tested network topology.

Table 2 shows the results corresponding to the case, when conditions for all running TCP connections are the same. The system is uploaded to the AC limit (link of 2 Mbps and minimum sufficient buffer size) and serves different number of TCP flows depending on their requested rate. In particular, we consider the test cases of 5, 4, 2 and 1 TCP connections. For instance, in the scenario with 5 connections, each flow has SR value fixed as 400 kbps, while in the scenario with only single connection the SR value is 2000 kbps. In addition, all cases were simulated for RTT^{min} equals 0.1 as well as 0.2 sec.

Table 2 - Throughput characteristics (with 95% confidence interval): homogenous TCP connections.

Number of flows	**5**	**4**	**2**	**1**
		RTT^{min}= 0.1s		
SR (kbps)	400	500	1000	2000
W^{req}(bytes)	5000	6250	12500	25000
BSS (bits)	38400	47500	90000	160000
RTT^{avg}(s)(anal)	0.1388	0.1347	0.1245	0.1109
RTT^{avg} (s)(sim)	0.116	0.114	0.113	0.114
RR(kbps)	288.0	371.1	803.1	1 704.6
Throughput (kbps)	332.7-354.9	387.7–411.0	818.4-846.7	1 721.8-1 746.3
		RTT^{min}= 0.2s		
SR (kbps)	400	500	1000	2000
W^{req} (bytes)	10000	12500	25000	50000
BSS(bits)	76800	95000	180000	320000
RTT^{avg}(s) (anal)	0.2548	0.2490	0.2346	0.2245
RTT^{avg} (s) (sim)	0.2164	0.2165	0.2163	0.2144
RR (kbps)	313.9	401.5	852.3	1 781.7
Throughput (kbps)	335.6-357.9	428.5-452.5	892.5–926.1	1 835.3–1 860.9

One can observe that the obtained TCP throughput is a bit greater than the requested rate for all tested cases. Additionally, the simulation results of average round trip time (RTT^{avg} (sim)) confirm the correctness of applied approximate formulas (RTT^{avg} (anal)).

All tested TCP connections are of greedy type, sending packets of constant size, MTU =1500 bytes. The simulation results are obtained using OPNET software, assuming TCP Reno version, with advertised window size equals W^{req}.

The obtained results showing received TCP throughput characteristics, referring to the cases with different number of running TCP connections, requested bit rates and minimum round trip times are presented in Table 2 and Table 3.

Table 3 presents the simulation results for the case of heterogeneous TCP connections differing in both RR as well as RTT^{min}. The system is again uploaded to the AC limit. The received throughput is also close to the requested rate.

In all tested cases the ideal behaviour of TCP was observed, as it was expected (no packet losses).

Table 3 - Throughput characteristics (with 95% confidence interval): heterogeneous TCP connections.

TCP connections	**SR (kbps)**	**W^{req} (bytes)**	**BSS (bits)**	**RR (kbps)**	**RTT_{min} (sec)**	**RTT^{avg} (anal)**	**RTT^{avg} (sim)**	**Throughput (kbps)**
PC1-Server1	400	5000	38400	288.0	0.1	0.1388	0.1178	309.7 - 331.4
PC2-Server1	400			288.0				
PC3-Server2	600	15000	112800	490.3	0.2	0.2447	0.2194	514.8-539.5
PC4-Server2	600			490.3				

10.6 Advertised window setting: implementation aspects

For applying the proposed AC in QoS IP network in effective way some changes in TCP applications are required. These changes should allow applications to adapt their behaviour to QoS requirements. There are some socket API mechanisms allowing TCP applications to change send and receive buffer sizes. This can be done by setting proper value of e.g. *SO_RCVBUF* parameter (receive buffer size). This parameter determines value of advertised window size for a TCP connection established by application.

Let us explain the rules how a TCP application could interact with the proposed AC algorithm. This can be achieved as follows. A user defines RR value with the aid of the end user application toolkit. For the purpose of admission decision the ACA calculates W^{req} and the token bucket parameters (SR, BSS). W^{req} can be sent jointly with the admission decision from ACA to called site. Next, W^{adv} calculated on the basis on W^{req} can be used by TCP application to set value of *SO_RCVBUF* parameter. In this case user application has to inter-work with

AC mechanism. In addition, some mechanisms have to be implemented for sending W^{req} from ACA to application. No additional signalling between client and server is needed. From TCL3 class point of view only information about value of W^{req} from ACA is required. Setting of receive buffer size is done by running application (e.g. ftp). The application can set *SO_RCVBUF* parameter during TCP connection set-up. Such a test has been done for an adapted ftp application running at Linux environment. In this case ftp client could set *SO_RCVBUF* equal to the calculated W^{req} using ftp command line. The following lines in ftp client code have been added:

printf("\nBuffer size: ");

scanf("%d", &r_b_size);

setsockopt(data,SOL_SOCKET, SO_RCVBUF, &r_b_size, sizeof(r_b_size));

Notice that for all applications running in Linux system, the TCP received buffer size is limited by the system values of the following parameters: *net.core.rmem_default* and *net.core.rmem_max*. These parameters can be changed by Linux root using *sysctl* function.

10.7 Conclusions

New admission control algorithm based on advertised window setting for handling greedy TCP connections in TCL3 class have been proposed. The QoS objectives are expressed in the form of requested bit rates. For this purpose, the ideal TCP behaviour is maintained during the connection thanks to the appropriate setting of advertised window size in the receiver. The submitted parameters by a TCP source are: (1) requested rate, RR, and (2) minimum round trip time RTT^{min}. These parameters are mapped into the form of parameters corresponding to the single token bucket, which constitutes the base for admission control algorithm. The included simulation results show that the effectiveness of the proposed AC is satisfied. The requested bit rate by each TCP connection is always guaranteed, even if a mix of TCP connections differing in rate requests and round trip times share the same network resources.

11 References

[1] M. Allman, V. Paxson, W. Stevens, *TCP Congestion Control*, Internet RFC 2581, April 1999.

[2] A. Bak, W. Burakowski, F. Ricciato, S. Salsano, H. Tarasiuk, *Traffic handling in AQUILA QoS IP Networks*, Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001.

[3] M. Benameur, S. Ben Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. W. Roberts, *Integrated Admission Control for Streaming and Elastic Traffic*, Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001.

[4] S. Blake et al., *An Architecture for Differentiated Services*, Internet RFC 2475, December 1998.

[5] Y. Bernet et al., *An Informal Management Model for Diffserv Routers*, Internet Draft, draft-ietf-diffserv-model-06.txt, February 2001.

[6] Cisco web pages, Weighted Random Early Detection on the Cisco 12000 Series Router.

[7] Deliverable D1301, *Specification of traffic handling for the first trial*, AQUILA project consortium, http://www.ist-aquila.org, September 2000.

[8] Deliverable D1302, *Specification of traffic handling for the second trial*, AQUILA project consortium, http://www.ist-aquila.org, December 2001.

[9] Deliverable D3201, *First trial report*, AQUILA project consortium, http://www.ist-aquila.org, March 2001.

[10] H. ElAarag, M. Bassiouni, Performance evaluation of TCP connections in ideal and non-ideal network environments, Computer Communications 24 (2001), pp. 1769-1779.

[11] S. Floyd, V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, August 1993.

[12] M. Handley, J. Padhye, S. Floyd, *TCP Congestion Window Validation*, Internet RFC 2861, June 2000.

[13] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, Proc. ACM SIGCOM'98, August 1998.

[14] J. Roberts, U. Mocci, J. Virtamo eds., Final report COST 242, *Broadband network teletraffic: Performance evaluation and design of broadband multiservice networks*, Lectures Notes in Computer Science 1155, Springer 1996.

[15] S. Sahu, P. Nain, D. Towsley, C. Diot, V. Firoiu, *On Achievable Service Differentiation with Token Bucket Marking for TCP*, Proc. ACM SIGMETRICS'00, Santa Clara, CA, June 2000.

[16] W. R. Stevens, TCP/IP Illustrated, *Volume1: The Protocols*, Addison-Wesley Publishing Company, 1996.

[17] W. R. Stevens, TCP/IP Illustrated, *Volume2: The Implementation*, Addison-Wesley Publishing Company, 1996.

[18] W. R. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, Internet RFC 2001, January 1997.

[19] T. Ziegler, C. Brandauer, S. Fdida, *A quantitative Model for the Parameter Setting of RED with TCP traffic*, The Ninth International Workshop on Quality of Service (IWQoS'2001), Karlsruhe, Germany, June 2001.

[20] R. Mortier, I. Pratt, C. Clark, and S. Crosby, *Implicit Admission Control*, IEEE Journal on Selected Areas in Communications, Vol. 18, No. 12, December 2000.

[21] Deliverable D1202, *System architecture and specification for the second trial*, AQUILA project consortium, http://www.ist-aquila.org, September 2001.

[22] F. Brichet, M. Mandjes, M. F. Sanchez-Canabate, *Admission control in multiservice networks*, Proceeding of the Mid-Term Seminar, 1999, Villamora, Portugal.

11 Dual-Homing and Load-Sharing

Hosts, edge routers (ER), border routers (BR), and autonomous systems (AS) may be dual-homed in order improve resilience, i.e. a host may be connected to two different edge routers, an edge router may be connected to two different core routers (CR), a border router may be connected to two different CRs, respective a network may be connected via two different peering points to neighbouring networks. Dual-homing improves resilience, because in case of a (single) link failure, the host, ER, BR, respective network is still connected via the alternative connection. The capacity of the two dual-homing connections can be exploited with load-sharing as long as both connections are alive. Then, traffic load is partitioned into two shares, one for each of the two connections. With load-sharing using both dual-homing connections simultaneously, in total carried traffic can exceed the capacity of each single connection.

Dual-homing and load-sharing require counterparts in the AQUILA RCL, i. e. some components of it need to be aware of dual-homing and load-sharing and need additional features to be able to cope with this kind of connections. In case of dual-homing of hosts, the appropriate ACA has to be found, because an individual ACA is used for each ER. Dual-homing of ERs requires adaptations in admission control (AC) and use of resource pools (RP) which dynamically assign network resources to AC based on online demand measurements. Dual-homing of BRs requires the same adaptations. Dual-homing of networks is transparent for the AQUILA RCL.

This section sketches RCL problems caused by dual-homing and load-sharing, and briefly discusses possible solutions.

Here, we concentrate on dual-homing, which is a special case of multi-homing in general. This does not restrict the problems to be discussed nor the solutions.

11.1 Dual-Homing Scenarios

There are different network elements that can be dual-homed in order improve resilience: hosts, ERs, BRs, and networks.

Dual-homing of Hosts

Hosts can be connected to two different ERs, see Figure 1. Notice that this is a simplified picture which omits the access network between the host and the ERs. Because a different ACA is used to control each of these two ERs, a dual-homed host has to select the right ACA, or more precisely, because a host does not communicate with an ACA directly but via an EAT, either a host has to select the right EAT or a single commonly used EAT has to select the right ACA.

Dual-homing of Edge Routers

ERs can be connected to two different CRs, see Figure 2. Because AQUILA's joint admission control uses the capacity of the link between an ER and the CR which is the next hop, and because resources of different RPs may be use via the different CRs, AC needs to be prepared to dual-homing of ERs.

Dual-homing of Border Routers

BRs can be connected to two different CRs, see Figure 3. This imposes the same problems as dual-homing of ERs.

Dual-homing of Networks

Networks can use two different peering points to neighbouring networks, see Figure 3. This is transparent to the AQUILA RCL. The ingress ACA asks the BGP routing system anyway to determine the egress ACA, thus will discover the right network exit. There is no difference between single-homed and dual-homed networks.

Altogether there are two scenarios that require further analysis: dual-homing of hosts and dual-homing of ERs.

For a more comprehensive description of these scenarios see section 3.1.2.2 of [1], where Figure 1, Figure 2, and Figure 3 are taken from.

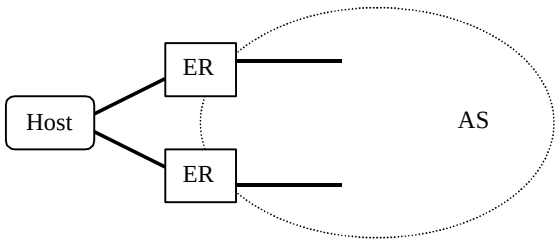


Figure 1 - Dual-homing of hosts, also called dual-hosting.

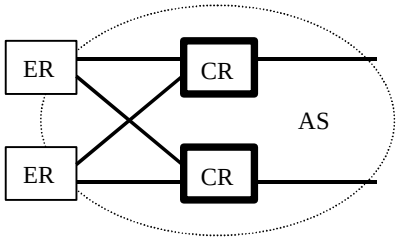


Figure 2 - Dual-homing of edge routers.

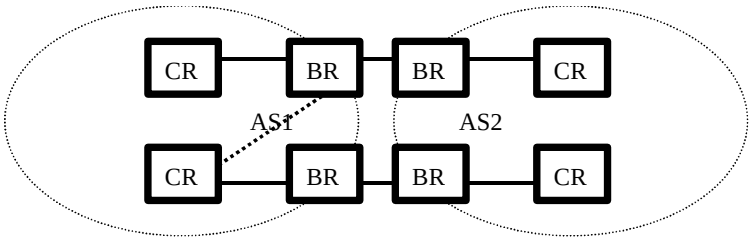


Figure 3 - Dual-homing of border routers.

11.2 Load-Sharing

For a first rough analysis of load-sharing we analysed a dual-homed ER, see Figure 4. We assumed:

- preservation of packet order, this means that some fields of the IP header like source address, destination address, source port, destination port, and protocol number are used to determine the egress link of an ER, and this will preserve the packet order of a flow (at a certain granularity),
- a fixed load-sharing target, which fixes how much % of the traffic should be transmitted via each of the two possible egress links,
- that the respective flow of each reservation is mapped to the two possible next hops statistically independently with probabilities equal to the load sharing target,
- that each flow requires the same amount of resources of the link capacity: one generic bandwidth unit u,
- given link capacities C_1 and C_2 summing up to a total forwarding capacity of $C = C_1 + C_2$, all expressed in integer multiples of the generic bandwidth unit u introduced above, see Figure 4.

We calculated the probability that one of the two egress links will be overloaded, if up to C flows are offered to an dual-homed ER which maps them to the links statistically independently as described above. The binomial distribution was used for these calculations.

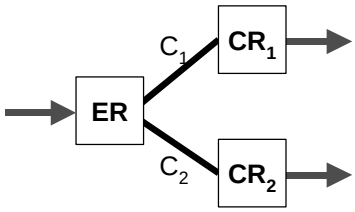


Figure 4 - A dual-homed ER splitting ingress traffic to two links.

Figure 5 shows the result for two scenarios with equal load shares. It depicts the probability that the capacity of one of the two links, that are used for load-sharing, will be exceeded over the utilisation of the in total available egress link capacity given on the x axis. For example, the probability that one of the two links cannot carry the offered load is more than 10%, in case that 79 flows are offered to two links which can carry 50 flows each. Figure 5 shows that the probability that AC must block a request is high, even if offered traffic is well below the in total available link capacity.

This is even worse, i.e. overload probability is higher, in case of unequal load sharing as shown in Figure 6.

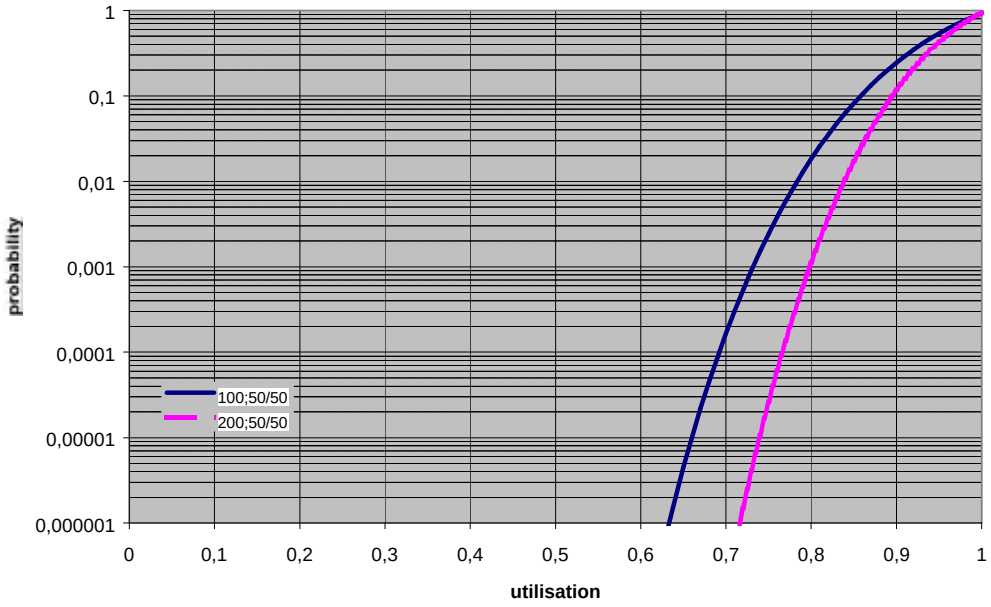


Figure 5 - Probability to exceed the capacity of one of two links used for load-sharing. First number given in the legend is link capacity, second and third are load share targets in %.

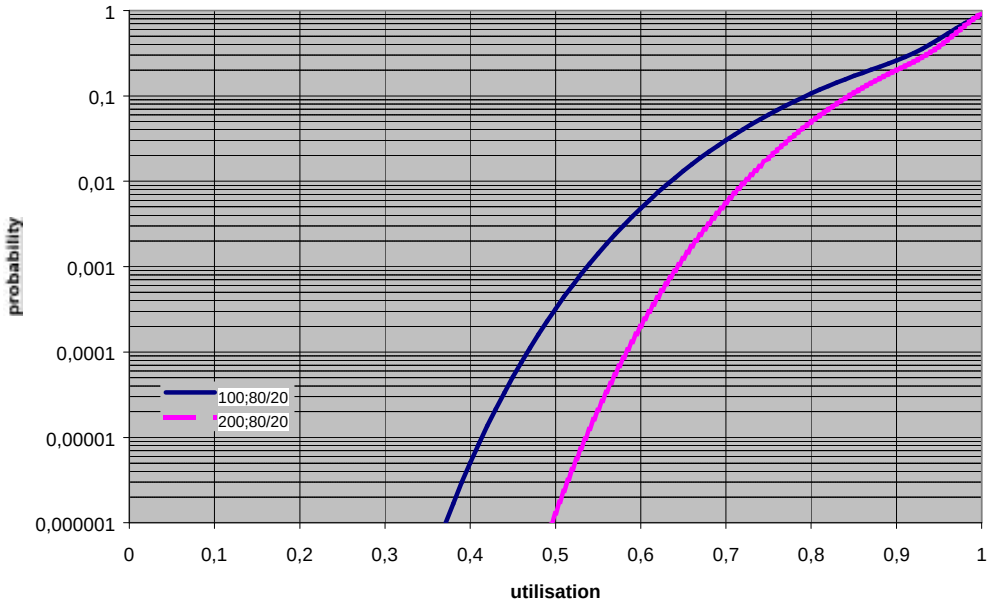


Figure 6 - Probability to exceed the capacity of one of two links used for load-sharing. First number given in the legend is link capacity, second and third are load share targets in %.

11.3 Dual-Homing and Load-Sharing Counterparts in the RCL

Dual-homing and load-sharing require counterparts in the RCL for ER selection, AC and usage of RPs.

If load-sharing exploits more capacity than is available on a single link, some reservations have to be released in case of a link failure in order to keep QoS targets. How flows are selected and released is not discussed here.

11.3.1 Dual-Homing of Hosts

In case of dual-homed hosts, the signalling layer has to follow the ER selection of the transport layer. It has to determined which ER will be used by the flow, a reservation request is asking resources for, at one of the hops in the signalling chain: host → EAT → ACA → ER. Because dual-homing and load-sharing is transparent to hosts at the transport layer, determination of the ER should not be done by the host at signalling layer.

There are two possible host to ER mappings. Either ER selection is done per host, via con-

figuration of a default gateway for example, or ER selection is done per IP packet using some header fields like the destination IP address for example.

A good solution in the first case is to use two EATs, one for each of the two groups of hosts which are determined by the host to ER mapping. The attachments of hosts to EATs and EATs to ACAs have to be carefully aligned with the attachment of hosts to ERs. This is a configuration and synchronisation problem. A synchronisation mechanism is required that triggers a re-attachment of the concerned EAT to the ACA of the remaining ER in the event of a link failure. Therefore, a notification mechanism has to be established between the load-sharing system and the EAT.

In the second case, ER selection by IP packet header fields, a decision system is required that knows how the transport layer decides. Here too, a synchronisation mechanism is required to keep both systems synchronised in the event of a link failure.

11.3.2 Dual-Homing of Edge Routers

For its decision on whether to accept or block a resource request, depending on whether there are sufficient free resources, AQUILA's joint admission control uses two parameters that depend on the next hop. These are the capacity of the link between the ER and the CR which is the next hop and the admission control limit (ACL) that represents the available resources in the core network and is dynamically assigned via a resource pool (RP) (QoS_check and Policy_check use the link capacity, ACL_check uses the ACL given by a RP, see [2]). Therefore in case of dual-homing, AC has to map reserved and requested resources to the two available links. There are two kind of solutions. There are solutions that map resource requirements to links explicitly. And, there is a solution that avoids this. The latter solution uses features of the AQUILA RCL that have been developed for MBAC (measurement based admission control).

11.3.2.1 Explicit Mapping of Requests to Next Hop

Ask the Routing System

The ACA can ask the routing system of the controlled ER for each resource request. Using the values of the IP header that will mark the IP packets of the considered flow, the routing system of the controlled ER can return the egress interface (via CLI in current implementation). This egress interface resolution has to be applied one time for each request.

Most of the AQUILA traffic classes use ingress and egress AC (called p2p AC) and require that all packets leave a network at the same egress ER. But a subscriber may ask for resources for an aggregated flow using blocks of source or destination addresses or blocks of ports instead of single values. The load-sharing system may not send all packets of an aggregated

flow via the same link. Even if all possible IP headers of an aggregated flow are mapped to an egress link one by one, it is still unclear how much resources are required by which of these sub-flows, whether all sub-flows will occur, and whether the resource split to sub-flows will be constant in time. Furthermore, there is an AQUILA traffic class which uses ingress AC only and allows a single reservation to use multiple exits. Here again, the load-sharing system may not send all packets of a flow via the same link.

If aggregated flows are used, egress link resolution through asking the routing system does not work.

Share Load-Sharing Strategy

The ACA could know the load-sharing strategy, i.e. include a module that maps appropriate IP headers to egress links in the same way, instead of asking the ER.

This requires a synchronisation that adapts this mapping in case of a link failure.

The same problems with compound flows as described above arise here.

11.3.2.2 Measurement Based Mapping of Requests to Next Hop

Here, AC estimates resource utilisation at each link as described below. Using this knowledge, AC accepts a reservation request, if the required resources are available at both links and in both RPs. This avoids a mapping for resource request to links.

In case an ACL is too small, the respective RP is requested for more resources. Based on the knowledge about link utilisation, free resources are returned to each RP independently just in the same way as it is done for single-homed ERs.

In order to estimate the resource utilisation of each link, link utilisation measurements are used in the same way as for MBAC. Therefore, the link utilisation has to be measured periodically for each egress link. This, together with the decision strategy described above, completely avoids explicit mapping of resource requirements to links.

Here, the compound flow problem is solved, too.

11.4 Conclusions

Dual-homing of host and dual-homing of ERs and BRs arise two selection problems: mapping of requests to the appropriate ER and mapping of requests to the appropriate egress link of an ER.

The ER selection problem of dual-homed hosts is hard to solve in the control layer, if IP

packet header fields are used for this decision by the IP layer. This problem would be avoided, if an on-path reservation protocol like RSVP was used where signaling messages follows the path of data packets until they arrive at the ER automatically.

The egress link selection problem of dual-homed ERs can be avoided in the current AQUILA architecture using features of MBAC.

11.5 References

[1] AQUILA consortium, deliverable D1202, "System architecture and specification for the second trial", IST-1999-10077-WP1.2-SAG-1202-PU-O/b1, http//www.ist-aquila.org, December 2001

[2] AQUILA consortium, deliverable D1302, "Specification of traffic handling for the second trial", IST-1999-10077-WP1.3-COR-1302-PU-O/b1, http//www.ist-aquila.org, September 2001

12 A scalability analysis for intra-domain AQUILA architecture

This section reports on a scalability analysis related to the AQUILA architecture. The scope of the analysis is focused on the intra-domain aspects. The aim of the scalability analysis is to discuss how the AQUILA system could scale in a realistic scenario, starting from the experiences that have been collected in the development and performance test of the AQUILA prototype. In particular the results of the Resource Control Layer (RCL) performance analysis, reported in [1] have been considered as starting point.

The desired output is to have an idea of the relation between the used demand (of QoS services) and the needed resources in Resource Control Layer (i.e. RCL equipment). The typical questions are: “how many users can be served with a given set of equipment” or “what equipment is needed to support this set of users”.

We clarify that this study is only a first step for the analysis of the scalability of dynamic IP QoS mechanism and we want to make the reader aware of the main limitations. The model parameters are based on the performance results collected in the AQUILA testbed. The prototype was not built with the aim of optimising performance, therefore we believe that there is room for a large performance improvement. On the other hand, only the control plane signalling load was considered, the impact of managing user profiles, collecting usage data, and all management tasks related to authorization, accounting and so on has not been considered. These tasks could even have a higher impact on the scalability of dynamic IP QoS than control plane aspects. Despite these (and others) limitations, we thought it was worth sharing some experience that we had on prototyping our IP QoS architecture.

The scalability analysis is based on a simple model. Some key parameters for the model are derived from the performance analysis of the AQUILA RCL [1] as will be described in section 12.1 and in section 12.2. Another important part of the model is the demand model, which estimates the user demand for QoS services. This model is described in section 12.3. We assume that the reader is familiar with AQUILA architecture [3], [2] and traffic handling mechanisms [4], [5].

12.1 Short report on the RCL performance results

The performance analysis of the Resource Control Layer performed in the AQUILA test-bed is fully described in [1]. For convenience, we will provide some information about it hereafter. The goal of the performance analysis was to evaluate the reservation setup time and the signalling load of the control procedures in the AQUILA architecture. The test environment consists of five routers and of some workstations that contains the RCL elements (End user Application Toolkits, Admission Control Agents, Resource Control Agent), as shown in Figure 1. In this test environment it was possible to execute sequences of reservations re-

quests coming from EAT and evaluate their total processing time and to split the processing time into the different architectural elements.

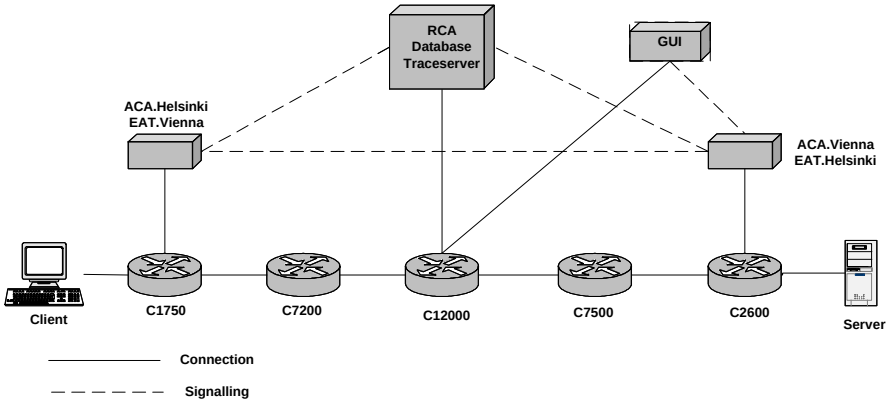


Figure 1 – Test scenario for RCL performance analysis

The measured setup and release times for reservations (averaged over a number of tests) are reported in Table 1. Measurements are reported for the four different AQUILA QoS traffic classes, which are based on different Admission Control algorithms. Anyway, for the purpose of our analysis, we will not consider a differentiation between the behaviour of the different traffic classes, but we will consider an average case. One thing that is not shown in Table 1 (see [1] instead) is that there is a large difference between the setup time of the first reservation (i.e. after that the system is initialised) and the subsequent reservations. The first one takes much longer as all the components of the Distributed Processing Environment (CORBA in our architecture) needs to be initialised. For this reason Table 1 only considers the subsequent reservations.

Table 1 – Total Processing times for reservation setup and release

	Setup Time [s]		**Release Time [s]**	
	Average	**Deviation**	**Average**	**Deviation**
TCL 1 Reservation	**0,90**	**0,026**	**0,46**	**0,065**
TCL 2 Reservation	**1,12**	**0,114**	**0,66**	**0,088**
TCL 3 Reservation	**0,97**	**0,071**	**0,50**	**0,011**
TCL 4 Reservation	**1,04**	**0,065**	**0,72**	**0,060**

In order to derive a performance model, it is needed to split the setup time into the different components. From the analysis of the time-stamps of the log files, it was evident that a large part of the time is spent in the communications between the elements of the RCL and the routers. This communication is needed for two main reasons. The first reason is that the ACAs ask the router for the BGP information needed to identify the egress border router given the destination IP address. The second reason is that when the reservation for a flow is accepted (or released), the configuration of the classifier/marker/policer in the ingress router must be updated. These communications are handled via telnet connections from the RCL elements to the routers and using the Command Line Interface (CLI) of the CISCO IOS. The second important component that was considered is the interaction between the ACA and the

RCA for the handling of the resource pools. This component represents the interaction with the "centralized" element of the AQUILA architecture, therefore it is important to consider it for scalability reasons. Table 2 shows the processing time that can be assigned to these two components with respect to the total processing time.

Table 2 – Router and resource pool contribution to total processing times

	Setup Time (ACA) [s]			Release Time (ACA) [s]		
	Total	**Router**	**RP**	**Total**	**Router**	**RP**
Initial	**4,508**	**1,188**	**0,199**	**0,676**	**0,464**	**0**
Subsequent	**1,102**	**0,754**	**0**	**0,665**	**0,435**	**0**

12.2 Building a model

The measurements described in the previous section allowed to build a simple model where 4 processing delay components are considered for the setup and two for the release of a reservation. The delay components and the corresponding times measured in the test-bed are reported in Table 3.

Table 3 – Processing delay components for reservation setup and release

SETUP		
BGP Route lookup (telnet to the router)	**350 ms**	**(3)**
Flow configuration (telnet to the router)	**700 ms**	**(3)**
Other RCL processing and communications with logging (without logging)	**350 ms (200 ms)**	**(1)**
Resource pool interaction	**100 ms**	**(2)**
RELEASE		
Flow configuration (telnet to the router)	**400 ms**	**(3)**
Other RCL processing and communications with logging (without logging)	**200 ms (120 ms)**	**(1)**

These delay components are used as mean processing times for a rough queuing model represented in Figure 2. The activities of the Resource Control Layer, are conceptually grouped into three tasks. The first task (1) includes all the communication and processing in the RCL distributed elements, excluding the interaction with the centralized RCA. The second task (2) includes the communications and processing in the RCL related to the interaction with the centralized RCA. The third task (3) includes the communication between the RCL (in particular the ACA) and the router. Of course this model is just a rough approximation of the real system, most of the assumptions are conservative with respect to the processing load, for example it assumes that all the activity in the task (1) is concentrated in a single element, the ingress ACA. In the real system as these activities are distributed between different elements (EATs, egress ACA), the ingress ACA has spare processing capacity with respect to what is modelled. Anyway the queuing model will be used (section 12.4) to make a bottleneck analysis, from which it results that the bottleneck of the system is the task (3) "telnet to the router". As the bottleneck behaviour dominates the overall performance, the approximations in modelling the task (1) bear little relevance. As for the modelling of task (3), the assumption here is that the processing activity of task (3) is located in the router which has to process the con-

figuration requests and the route lookup requests. The requests from the ACA to the router are serialized, therefore it is realistic to consider that the processing delay components measured for task (3) correspond to a time in which the system can exclusively process a single request. Table 4 reports the processing time for the three different tasks that have been considered in the model. The setup, the release and the total processing time are included. For the tasks (1) and (2) the processing times which are measured in the test-bed includes a lot of logging activities, which will not be present in the real system. Therefore the processing times with no logging have also been inferred from other measurements and reported in Table 3 in brackets. These processing times without logging have been considered in the scalability model. The total processing time required to process a flow is considered, i.e. including both the setup and the release of the flow.

Table 4 – Total processing time for the different tasks

		Setup	Relase	Total
(1)	**RCL communication and processing excluding RCA-ACA**	**Ts_1 = 200 ms**	**Tr_1 = 120 ms**	**T_1 = 320 ms**
(2)	**RCL communication and processing: RCA-ACA**	**Ts_2 = 100 ms**	**Tr_2 = 100 ms**	**T_2 = 200 ms**
(3)	**ACA-router communication and router processing**	**Ts_3 = 1050 ms**	**Tr_3 = 400 ms**	**T_3 = 1450 ms**

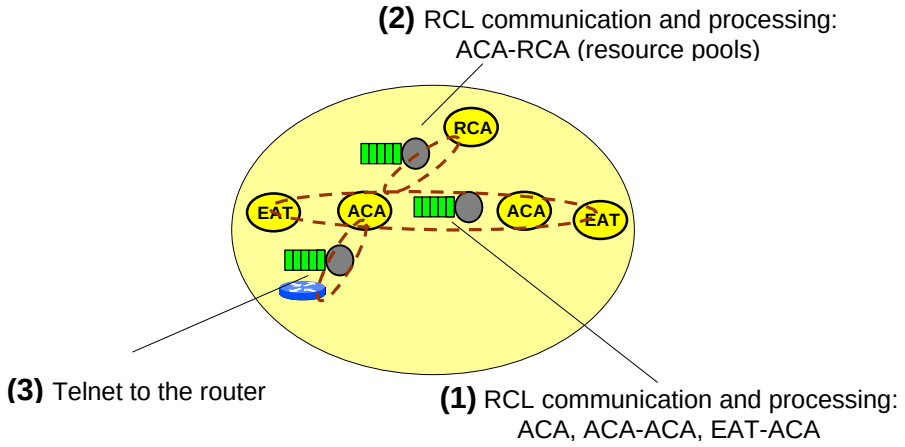


Figure 2 – Rough queuing model for RCL activity

From Table 4 we can derive the indication of how much the AQUILA components are loaded for a single reservation request / release. Let λ be the rate of reservation requests coming to an ACA. Then the load of the three components is:

$$r_1 = l T_1$$
$$r_2 = aK l T_2 \qquad \textit{eq. 1}$$
$$r_3 = l T_3$$

Note that in this model ρ_1 and ρ_3 represent a load of the distributed elements of the architecture (they can be mapped in the ACA and in the Access Router) while ρ_2 represents the load of the centralized RCA.

K represents the number of ACAs in the AQUILA domain that are controlled by the RCA. A simple architecture with one RCA and a set of ACAs (one for each border router) is considered, i.e. hierarchies of RCAs are not considered. Therefore all the request coming from all the ACAs will load the single RCA.

α represents the fractions of reservation requests arriving to the ACA that trigger an interaction with the centralized Resource Control Agent. In fact, thanks to the AQUILA resource control mechanism implemented in the RCL, most of the requests can be handled at the ACA level with no need of interaction with the centralized RCA. An analysis of the considered resource control mechanisms (named Resource Pool mechanisms) is described in section 2 of this deliverable, where α is called "signalling damping factor". See also [6] for a performance analysis of this mechanisms.

Equations (eq. 1) can be used to derive a limit for the reservation request rate taking into account a maximum value for the system utilization. Obviously this maximum value will be dimensioned taking into account the need to limit the response time to reservation request. This will be done in section 12.4.

What is still missing now is how to relate the reservation request rate to the number of users in the AQUILA system. The demand model utilized for this purpose is described in the next section.

12.3 Demand model

The purpose of the demand model is to relate the population of users to the rate of requests that must be processed by the Resource Control Layer. In order to build such model we should know which are the services available to the users and which will be the user demand for these services. Of course we can only guess a possible service scenario. In particular, we assume that the following three reference services are available: 1) real-time video communications, 2) video-downloads, 3) on line games. These three services map into three of the four AQUILA traffic classes. In particular the video communication uses the PVBR (Premium Variable Bit Rate, which is meant for inelastic UDP traffic), the video-download (which actually could be any type of long file download) uses the PMM (Premium Multimedia, which is meant for greedy elastic TCP traffic) and the on-line gaming uses the PMC (Premium Mission Critical, which is meant for non-greedy TCP traffic).

In order to assess the "control plane" load (i.e. the load on the AQUILA Resource Control Layer) we have to evaluate the service request rate (*SRRi*) originated by a single active user for each of these services (i=1,2,3). Let us consider the "activity" A_{ui} of a user in a service *i*, i.e. the fraction of the time in which a user is participating in an instance of the service *i*. We denote U_i the number of user that participates to an instance of the service i and SD_i the average duration of a service instance.

$$A_{ui} = SRR_i \cdot U_i \cdot SD_i \qquad \textit{eq. 2}$$

From the eq. 2 we can derive *SRRi* based on assumptions on user activity and service duration. Taking into account how many flow setups are included in a service session (*Fi*), the rate of reservation requests per user per service l_{ui} and total per user l_u are given by:

$$\begin{aligned} l_{ui} &= SRR_i \cdot F_i \\ l_u &= \sum_i l_{ui} \end{aligned} \qquad \textit{eq. 3}$$

(Note that we have counted the reservation requests, there will be a corresponding reservation release for each request.) Now, let N be the number of users which are attached to an ACA, the total rate reservation requests coming to an ACA λ is given by:

$$l = N l_u \qquad \textit{eq. 4}$$

It is interesting to take also into account the use of transport resources by considering the "user plane" bandwidth requirements for the various services. In particular we will consider the bandwidth in the "download" direction (from the ACA to the user). For each service *i* the user will receive J_i flows characterized by a rate R_{ij} (j = 1 to J_i). The average bandwidth requirements per user per service B_{ui} is given by:

$$B_{ui} = A_{ui} \cdot \sum_{j=1}^{J_i} R_{ij} \qquad \textit{eq. 5}$$

The total average bandwidth requirement per user B_u is obviously:

$$B_u = \sum_i B_{ui} \qquad \textit{eq. 6}$$

The values of the parameters for the services are reported in Table 5. For the service 1, a two way real-time video communication with reservation for audio and video flows is considered. The bandwidth of video and audio are around 500 kb/s and 25 kb/s respectively, while the duration of the conference in the order of 30 minutes. For service 2, a video download corresponds to a single flow and therefore a single reservation. The bandwidth of 500 kb/s and the duration of 20 minutes corresponds to a file size of 75Mbytes. For the on-line games, we assume for example that a session is composed of four users and that a full mesh of 6 TCP connection interconnects the users. Note that in the AQUILA architecture for a bi-directional reservation two reservation requests needs to be setup in the Resource Control Layer. This is also true for a single TCP connection if we want to reserve resources in both direction. Therefore, 12 flow setups are needed for the 6 connections. In this case the duration of a game has been assumed 20 minutes, while the bandwidth of each flow is 5 kb/s.

Table 6 reports the parameters related to the user activity. Two reference cases has been considered: a "highly demanding" user and a "normal" user. It is very difficult to imagine the real behaviour of the user, therefore the two cases simply represent two working hypothesis. The

"highly demanding" user is involved in a video communication for 50% of his time, it downloads three large files in an hour and it plays on-line game for 20 minutes per hour. Therefore he represent a sort of upper bound for the user demand. The required bandwidth in the download direction (see Table 7) is 767 kb/s… a very high value. The "normal" user has one half or the activity with respect to the previous one for video conference and one third for the other two services. (In our opinion, this is still an overestimation of the activity of a user). The average required bandwidth in this case is 300 kb/s.

Table 5 – Parameters for the services

	Service	**Users / Service session**	**Flow setup / Service session**	**Bandwidth per flow in the downlink**	**Service Duration**
i		*Ui*	*Fi*	*Ri*	*SDi*
1	**Real-time video communication**	**2**	**2**	**R_{11} = 500kb/s R_{12} = 25kb/s**	**30 min**
2	**Video- (file-) downloads**	**1**	**1**	**500kb/s**	**20 min**
3	**On-line games**	**4**	**12**	**5 kb/s**	**20 min**

Table 6 – User activity parameters (Activity A_{ui})

	"Highly demanding" user	**"Normal" user**
Real-time video communications	**50 %**	**25%**
Video- (file-) downloads	**100 % (3 req/h)**	**33% (1 req/h)**
On-line games	**33%**	**11%**

According to eq. 2 - eq. 6, we evaluated the control plane and user plane requirement per user. The results are reported in Table 7.

Table 7 – Control plane and user plane requirements

	Control plane: Reservation request rate l_u (1/h)	**User plane: Bandwidth B_u (kb/s)**
"Highly demanding" user	**8**	**767,5**
"Normal" user	**3**	**300**

12.4 Scalability results

Combining the model of the RCL described in section 12.2 and the demand model presented in section 12.3, it is now possible to consider the scalability analysis. We will evaluate a limit for the reservation request rate taking into account a maximum value for the system utiliza-

tion. The maximum utilization is dimensioned taking into account the response time to reservation requests. According to Table 4 the reservation setup time T_{setup} under unloaded conditions is:

$$T_{setup} = T_{s1} + aT_{s2} + T_{s3} = 1260\ ms \qquad \textbf{eq. 7}$$

We assume for the reservation damping factor α a value 0.1 (i.e. 90% of the requests are handled locally, while 10% triggers the intervention of the Resource Control Agent). The chosen target is that the average reservation setup time should be less than 2 seconds. Taking into account the load of the system, we can generically express T_{setup} as:

$$T_{setup} = f_1(r_1)\cdot T_{s1} + f_2(r_2)\cdot aT_{s2} + f_3(r_3)\cdot T_{s3} \qquad \textbf{eq. 8}$$

The bottleneck of the system dominates the setup delay. From eq. 1,

$$\begin{aligned} r_1 &= T_1/T_3 \cdot r_3 = 0{,}22\, r_3 \\ r_2 &= aKT_2/T_3 \cdot r_3 = 0{,}0138 \cdot K \cdot r_3 \end{aligned} \qquad \textbf{eq. 9}$$

Therefore the bottleneck of the system is task (3) as long as 0,0138 K <= 1. This is a condition on the number of ACAs that can be controlled by a Resource Control Agent. Considering that T_{s2} is much smaller than T_{s3} we can accept that the load of task (2) becomes equal to the load of task (3) so that we actually have two bottlenecks. Therefore from 0,0138 K = 1 we derive an indication of the number of ACAs that can be controlled by a RCA: K = 72,5.

If we consider an M/M/1 queuing system model, then $f_i(r_i)=1/(1-r_i)$. When the bottleneck load is $r_2=r_3=0{,}4$, $f_2(r_2)=f_3(r_3)=1{,}67$ and we find that the total average setup delay is below our limit:

$$T_{setup} = 1{,}082 \cdot T_{s1} + 1{,}67 \cdot aT_{s2} + 1{,}67 \cdot T_{s3} = 1938\ ms \qquad \textbf{eq. 10}$$

Once we have fixed the maximum load $r_3=0{,}4$, we can use eq. 4 and eq. 1 to derive the maximum number of users per ACA N:

$$N = l/l_u = r_3/T_3/l_u \qquad \textbf{eq. 11}$$

We can also evaluate the needed user plane bandwidth at the ACA $B=N\cdot B_u$ considering the requirements reported in Table 7. The results for the two types of users are reported in Table 8: N ranges from 124 to 331 users. Note that these are the *active* users that can be controlled by an ACA, the number of *subscribed* users can be higher according to a user activity factor to be considered. We notice that the user plane bandwidth is nearly constant for both types of users. The minor difference is due to the fact that we slightly changed the proportion between the service usage: the “Normal” user has a higher usage of Video Communications, which is a Bandwidth consuming service.

Table 8 – ACA capacity(users and bandwidth)

	Number of users per ACA *N*	**User plane Bandwidth** *B (Mb/s)*
"Highly demanding" user	**124**	**95**
"Normal" user	**331**	**99**

Table 8 shows that a single ACA can control a QoS bandwidth of around 100 Mb/s, for the considered service mix. Let us assume that QoS bandwidth is around 40% of access bandwidth, the remaining part being used by best effort services, then we can evaluate how many active QoS users N_{ER} can be served by an Edge Router with given access capacity L: N_{ER} = 0,4 L/ B_u. Then from the control plane requirement we derive how many ER can be controlled by a single ACA: $ER/ACA = N/N_{ER}$. The result is reported in Table 9. Note that the fact that a single ACA controls more that one Edge Router could even enhance the system capacity. This is due to the fact that ACA could handle more connections in parallel and the communication with the router would no longer be a single bottleneck. Anyway in this analysis we made the conservative choice to stick to the model described above.

Table 9 – ACA capacity(number of Edge Routers)

	"Highly demanding" user		**"Normal" user**	
ER Access capacity	**Number of Edge Routers per ACA** *ER/ACA*	**Active QoS Users per ER** N_{ER}	**Number of Edge Routers per ACA** *ER/ACA*	**Active QoS Users per ER** N_{ER}
10 Mb/s	**23,8**	**5,2**	**24,8**	**13,4**
34 Mb/s	**7**	**17,7**	**7,3**	**45,4**
155 Mb/s	**1,6**	**78,2**	**1,7**	**200,3**

12.4.1 Considerations about signalling bandwidth

The bandwidth occupation of signalling messages is another concern in the deployment of dynamic IP QoS. In the AQUILA testbed, the amount of signalling traffic has been evaluated. Note that the AQUILA system is based on a Distributed Processing Environment (CORBA) and that no attempt has been made during the system design and development to reduce the signalling traffic. This is justified by the fact that the implemented system is only a demonstrator. The measured amount of signalling traffic for each reservation setup and release was in the order of 50000 bytes. This must be multiplied by the reservation setup rate for an ACA λ in order to obtain the total signalling rate related to an ACA: 110 kb/s. This must be compared with the user plane QoS bandwidth per ACA *B* (~100 Mb/s). The signalling bandwidth is in the order of 0,1% of the QoS bandwidth, therefore it has a negligible impact.

12.4.2 A feasible scenario for the AQUILA prototype

This section describes a feasible scenario that could be realized with AQUILA prototype according to the scalability analysis. The "normal" user model has been considered. An ISP network could include a single RCA that can control 70 ACAs. 35 ACAs could be devoted to

business customers, each ACA could be connected to 24 customer ERs with 10 Mb/s links. Each ER supports 13 active QoS users: in total there are 840 customer ER and at most 10900 active QoS users. The other 35 ACAs could be devoted to residential customers. In this case each ACA supports 300 active QoS users, therefore there is a capacity of 10500 active residential users. Considering an over-subscription factor of 2 to 3, this corresponds to a total of 40000-60000 “QoS” subscribers.

12.5 Conclusions

The scalability analysis of AQUILA architecture has been quite promising. Even in its prototype version, which was not optimised for performance, a large number of users and the corresponding request rate seems to be supported. The analysis was limited to control plane (signalling) scalability aspects. Management issues (like handling of user profile, Authentication-Authorization-Accounting) have not been considered. These issue could be a threat for system scalability and should be carefully studied to assess the overall scalability of IP QoS.

Throughout the analysis, we used a rough engineering approach to queuing networks… we do not claim that the results are mathematically proved. Our decision was to use a simplified model and some coarse assumptions in order to have an idea of the system scalability.

12.6 References

[1] Deliverable D3202, “Second Trial Report”, AQUILA project consortium, IST-1999-10077-WP3.2-TPS-3202-PU-R/b0, http//www-st.inf.tu-dresden.de/Aquila/, February 2003.

[2] Deliverable D1202, “System architecture and specification for the second trial” AQUILA project consortium, IST-1999-10077-WP1.2-SAG-1202-PU-O/b1, http//www.ist-aquila.org, December 2001

[3] Deliverable D1203, “Final system specification”, AQUILA project consortium, IST-1999-10077-WP1.2-SAG-1203-PU-O/b0, http//www-st.inf.tu-dresden.de/Aquila/

[4] Deliverable D1301, “Specification of traffic handling for the first trial”, AQUILA project consortium, IST-1999-10077-WP1.3-COR-1301-PU-O/b0, http//www-st.inf.tu-dresden.de/Aquila/

[5] Deliverable D1302, “Specification of traffic handling for the second trial”, AQUILA project consortium, IST-1999-10077-WP1.3-COR-1302-PU-O/b2, http//www-st.inf.tu-dresden.de/Aquila/, March 2003

[6] T. Engel, F.Ricciato, S.Salsano, M. Winter, “Resource Management in QoS Enabled IP Networks with the AQUILA RCL” 10th International Conference on Telecommunication Systems, Modeling and Analysis, October 3-6, 2002, Monterey CA, USA