TR-AC-0053

003

A Distributed Resource Allocation Scheme over Diffserv, towards End-to-end QoS Assurance

Andreas Alexelis
山崎 達也　蓮池 和夫

2001. 2.28

ATR環境適応通信研究所

A Distributed Resource Allocation Scheme over Diffserv, towards End-to-end QoS Assurance

Andreas Alexelis, Tatsuya Yamazaki, Kazuo Hasuike

Authors are with the Adaptive Communications Research laboratories, Advanced Telecommunications Research, Japan

Abstract

In recent years, the concept of Quality of Service (QoS) has gained attention in the network research community and it is generally agreed that a crucial factor for the longlivity and usability of Internet will be it's extension to deliver QoS level assurance to QoS-sensitive applications. The Differentiated Services Architecture (DiffServ) proposes a general framework for differentiated treatment of traffic aggregates in the core network. DiffServ does not extend to end-to-end QoS assurances. Most of the complementary approaches proposed (RSVP, circuit-based QoS Broker etc.) have several drawbacks concerning scalability, time complexity, network utilization and administrating range. In this paper, we propose a simple resource-broking scheme operating over an hierarchically connected internetwork of administratively-independent, DiffServ-capable domains, a simplified model of Internet. The proposed approach uses an asynchronous, multi-agent recourse broking scheme that operates locally but avoids conflicts globally. In this way, traffic control reflects the underlying structure of the internetwork, introduces only localized complexity thus scaling up well and permits independence of policy between interconnected domains.

Keywords

QoS, Internet, DiffServ, traffic control

I. INTRODUCTION

Together with the wide-spread grow of Internet and the proliferation of applications in use, the best-effort type service of the present Internet is reaching its limitations. Since real-time applications like Internet telephony or stream audio/video applications have appeared and started gaining popularity, the service delivered by a fairly congested best-effort IP is unacceptable for all practical purposes. It is generally agreed that supporting Quality of Service (QoS) level assurance is a critical factor for the survival of Internet.

QoS inherently implies a form of categorization of traffic according to transmission characteristics, consequently a QoS-aware network should treat different traffic categories in an appropriate manner.The Differentiated Services Architecture (DiffServ or DS)[1] provides a simple, general framework for classification, aggregation and differentiated treatment of traffic, aimed at the global Internet. The DS framework concentrates on one part of the QoS concept, namely the semantics of extending current Internet to a QoS-aware direction, but does not cover management or assurance of QoS in order to reach all the way to the end-user.

The QoS concept is an end-to end concept, in the sense that the end user, the driving force behind the Internet-related information industry, sits at the end of the network and perceives only the final product of the communication process. Although the end-user is not aware of it,

this communication process depends heavily on the underlying network, it's structure, protocols and congestion state.

Since QoS is an end-to-end concept, at first it seems reasonable to try to establish an end-to-end connection for every traffic flow between source and destination, reserving network resources at every network element on the path. In fact, this is the only currently available solution in order to provide *strong guarranties* about the level of service that a particular flow will receive while propagating across the network. In case of private or medium-scale public networks this approach has given good results in many occasions.

In the case of Internet, however, this approach has several drawbacks[3] of which the most serious are the prerequisite for central administration and the proliferation of state information at the backbone together with prohibitive cost for connection setup/tear-down per traffic flow.

QoS management, apart from the two communicating hosts, involves also all network elements (e.g. links, routers, switches etc.) on the path between them. In contrast to private networks, where the whole network belongs to one administrating authority, Internet consists of many[1] interconnected independent networks, each with its own administration authority, scope and objective. Therefore a QoS scheme applicable to Internet has to reflect (or at least consider) this administrative diversity.

On the other hand, space requirements to store connection state information proliferates when moving from the leaves of network subtrees towards the backbone. Applying the same rationale that governs the design of most Internet routing protocols, it is evident that it is counter cost-effective to keep end-to-end state information per flow for all flows with QoS requirements on a large-scale network. On the contrary, examining the *hierarchical* divisioning utilized by routing protocols, state concentration points can be identified at the traffic aggregation elements of subnets, clusters and domains that Internet consists of.

At the same time, the cost for setting up and tearing down a connection for any particular traffic flow is prohibitive, due to the statistical properties of individual traffic flows. The average rate, duration and number of traffic flows varies significantly between applications and in the case of a large-scale network with long round-trip times the time required to set up a connection is many times the duration of transmission itself. On the other hand, setting up and tearing down

[1] According to [3] "...in 1995 the *average* off-site Internet conversation crossed 14 different administrative boundaries..."

a connection for an *aggregate* of traffic flows is much more cost efficient. Aggregates do not have the same statistical properties as individual traffic flows, their average varies less with time and, in general, have a more predictable behavior.

In this paper we argue that, to be able to scale up together with Internet, a QoS scheme must reflect the structure of Internet, utilize it's hierarchical nature to localize state information and deal with network resource allocation/deallocation at traffic aggregate level. We argue that if such QoS management is combined with rational admission policies that control incoming traffic at boundaries of adjacent hierarchical layers, it is possible to provide global *soft QoS assurance* for all traffic classes with QoS requirements.

Specifically, in this paper we propose a particular scheme having the above characteristics. Our scheme takes advantage of the hierarchical structure of Internet to employ network resource broking agents distributed across all state concentration and traffic aggregation layers. Each agent executes the policy of the respective domain and is responsible for the allocation/deallocation of network resources at domain level. Agents of adjacent domains communicate using a simple local resource allocation protocol and exchange policy information *locally* in a way that conflicts are avoided *globally*. We call our QoS resource broking agents scheme *the QB scheme*.

The rest of the paper is organized as follows. In the next section, we present our proposed framework in detail and derive the basic principles for its implementation. We then present our proposed scheme and examine the dynamics of it's operation. Next, experimental results taken from simulation using the proposed scheme are given, together with analysis of network behavior. Finally we conclude with some considerations regarding future directions of research.

II. The proposed framework

A. *Internet topology considerations*

The applicability of a traffic control scheme depends heavily on the underlying network structure; therefore, traffic control schemes should reflect the characteristics of network infrastructure.

Internet is the biggest existing internetwork. The strongest peculiarity of Internet is that there is no central administrating authority. Rather, Internet consists of a number of interconnected

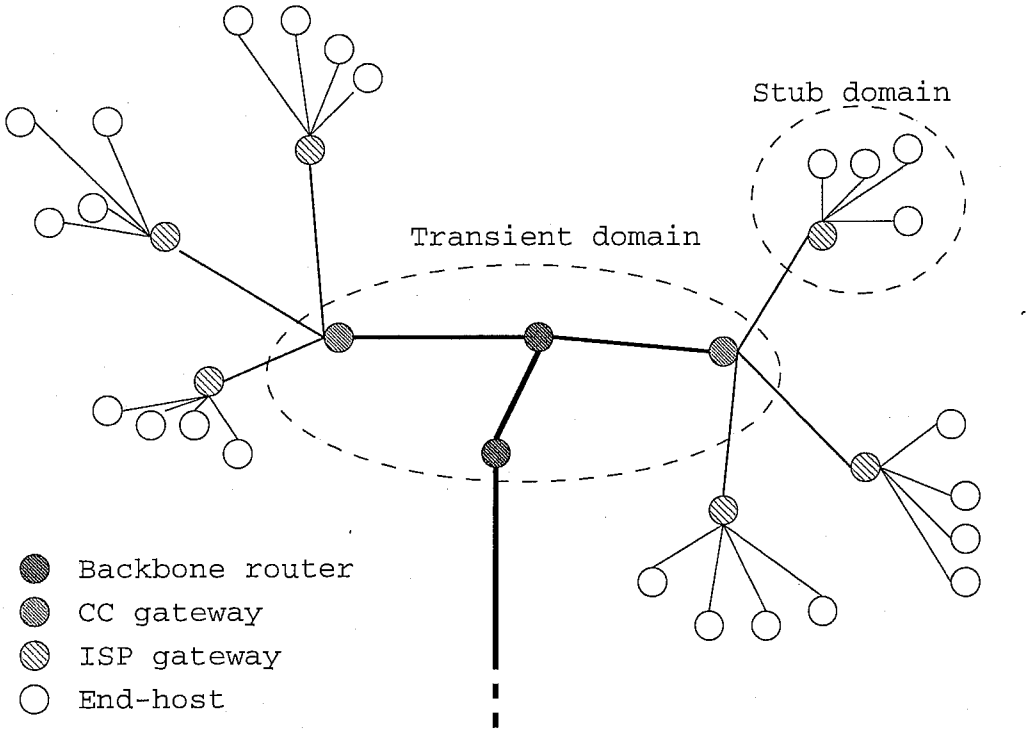

Fig. 1. The network hierarchy up to the backbone.

domains[2]. Each domain is a subset of the network under a single routing and policy administration.

In [2] domains are classified depending on whether traffic originates/terminates inside them (*stub* domains) or traffic just passes through (*transient* domains). In this paper, we rather consider a *hierarchy* of domains, in the sense of adjacent domains with increasing scale of resources. To make things concrete, we consider a large number of end-user hosts connected with low-bandwidth lines to various Internet Service Providers (ISPs). ISPs are connected with relatively higher bandwidth connections to Common Carrier companies (CCs), which in turn are connected to a high bandwidth inter-domain backbone, as depicted in Fig.1. In other words, a hierarchy is defined by the amount of resources owned and/or administrated by domains. If we use the stub-transit domain terminology, end hosts and ISPs form the stub domains while CCs and inter-domain backbone nodes form the transient domains.

The main characteristic of this hierarchy is that every entity uses recourses provided by the entities directly above them, for ex. end-hosts use (usually pay for the submission) network resources provided by ISPs, ISPs push the traffic through lines leased by the CCs, etc. In the

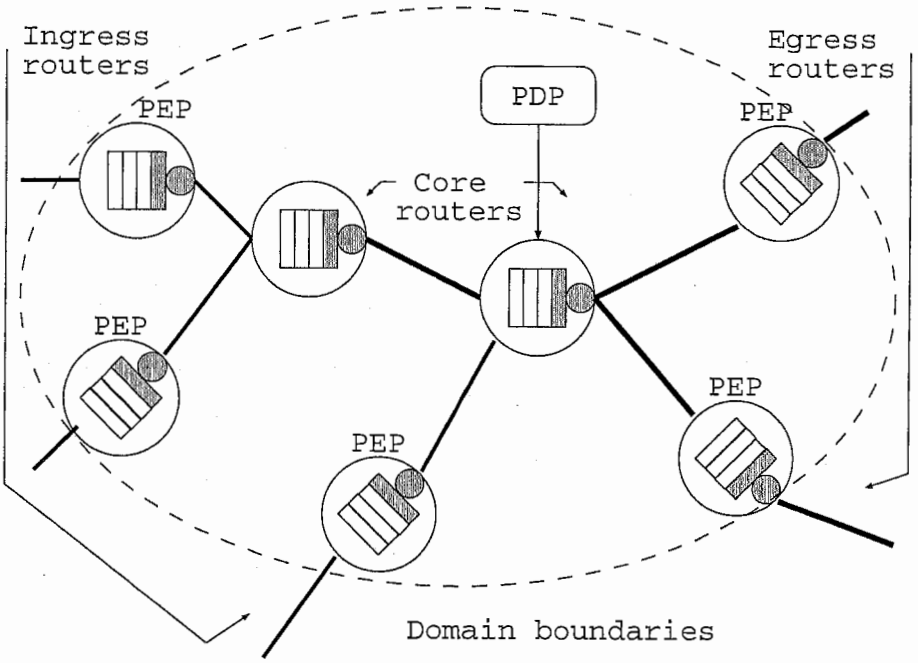

Fig. 2. The internals of a domain.

opposite direction, all entities (except the end-hosts) *provision* the entities directly bellow them with the resources they "own", for ex. ISPs use their lines' capacity to provision end-hosts, CCs use their access to the backbone to provision ISPs etc.

Fig.2 shows the internals of an autonomous administrating domain. Every such domain, has a Policy Decision Point, which functions as "recourse market". Policy decides how will domain's resources be allocated, for ex. what amount of the capacity will be allocated to premium class or best effort services (and usually, at what price) etc. Client entities of the layer bellow go to the "resource market" to negotiate resource allocation. In some cases , for instance in the case of a Common Carrier monopoly, which resource market will an entity negotiate with is decided by geographical or infra-structural necessity. In other cases, for instance choosing an ISP, it is a matter of choice.

The objective of communications industry, as a whole, is to maximize the usability and utilization of the internetwork. Domains are free to independently decide their policy; on the other hand, it is evident that, up to certain extend , policy information exchange and common understanding of policy objectives are necessary to achieve inter-operability.

The above arguments suggest that a viable QoS scheme should be a distributed system with policy information exchange and inter-operation capabilities.

B. Network resource allocation considerations

Network traffic originates (mostly) at the periphery, is carried by the inner part of the network and terminates at some other part of the periphery. Using the analogy from physics found in [5], network traffic obeys the *conservation principle*. Conservation principle states that for any part of the network that does not contain sources or sinks, the net sum of traffic entering and traffic leaving equals to the amount of traffic stored inside that part of the network. Since traffic stored inside a part of network that does not have sources or destinations signifies development of congestion, for such parts of the network, it is desirable that time average of the net sum of incoming and outgoing traffic is zero. It follows that any domain should not allow in more traffic than it can outsource, otherwise queues will built up and packets will be dropped, damaging network performance.

In the current best-effort Internet, applications inject traffic into the network at will, without any previous notification or resource allocation. Therefore, performance is dominated by statistical characteristics of the traffic sources, network topology and design and current level of congestion at network elements. In fact, there is no way to provide any kind of assurance about the level of service that any particular flow will receive.

The first step to providing QoS assurances to traffic flows with QoS requirements is to control the amount of incoming and outgoing traffic at each inter-domain boundary. Traffic control generally involves resource allocation, Connection Admission Control (CAC) and traffic policing. Every domain must grant requests for network resources to traffic flows coming in the domain, through the ingress links, only up to the total amount of resources allocated for traffic going out of the domain, through the egress links. Requests that, if granted, would overload the network should be rejected. Traffic should also be policed to make sure that it complies to the committed allocation. An example is given at fig.3.A CC with an output capacity of $5Mbps$ has allocated $2Mbps$ and $3Mbps$ two the two adjacent ISPs. If a total of more than $5Mbps$ enters the CC, congestion will start to occur so traffic from ISPs must stay within the committed transmission rate limits. End-hosts 1,2,3 have requested $1Mbps$ each; therefore, since the total is less or equal to the available $3Mbps$, these requests were granted. The request of end-host 4 was rejected be-

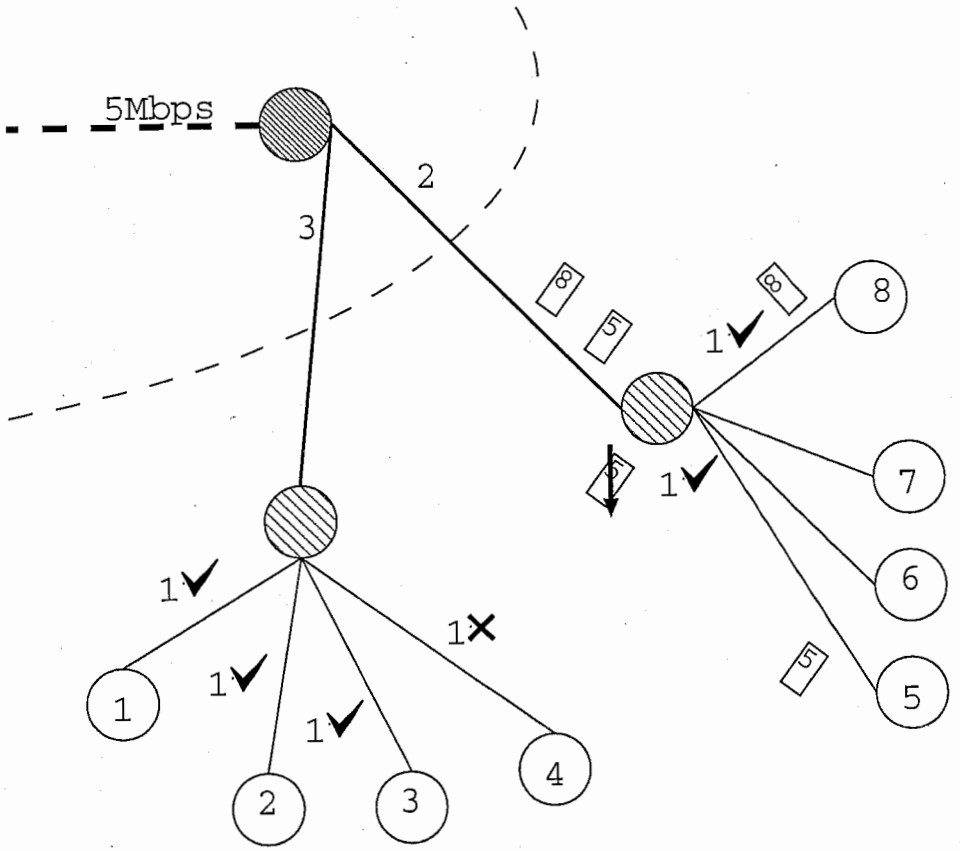

Fig. 3. Example allocations at ISPs.

cause of lack of resources[2], therefore end-host 4 is not allowed to inject traffic until resources become available. End-hosts 5 and 8 were both granted $1Mbps$ but end-host 5 attempts to inject $2Mbps$. The ISP, protecting compliant end-host 8 from performance degradation, drops half of end-host 8 traffic before entering the domain.

In order to achieve a QoS level that can accommodate evolving Internet applications, various classes of application traffic should be supported. The allocation scheme presented above holds also in the case of sharing the available capacity between classes. Needless to say, classes are served according to their relative priority, in this case. We have given an example regarding only bandwidth allocation but the example can be easily extended to include buffer space, queue service priority etc.

The main point here is that if allocations agree, in the sense that they don't contradict each

[2]We consider first-come-first-served (FCFS) media access strategy, without prior reservation of access.

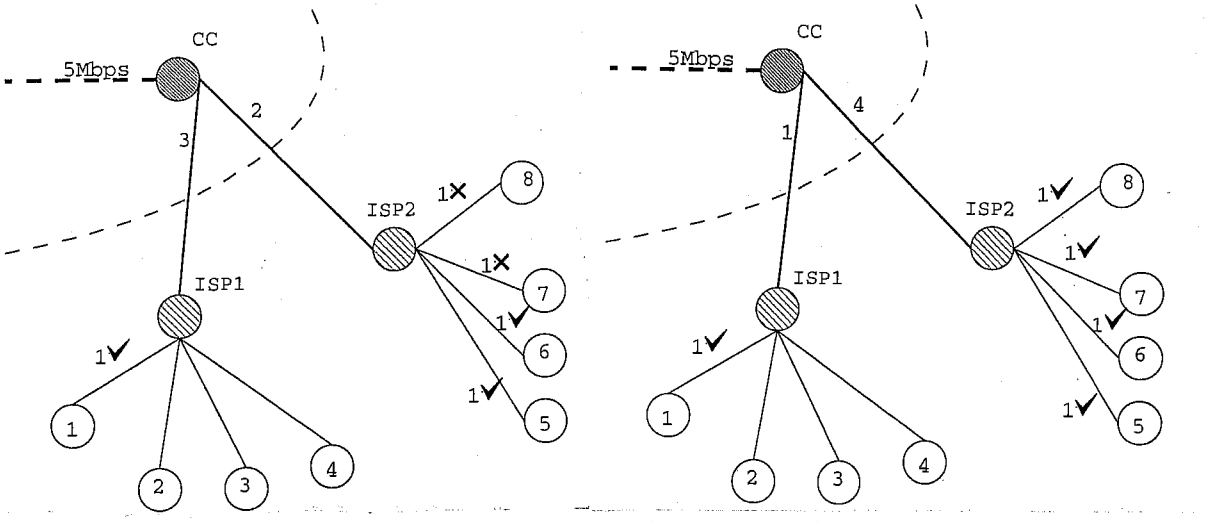

Fig. 4. Example of adaptative allocation at CC layer.

other, and agreements are honored at all layers, then traffic rate is controlled, therefore performance standards are met *at least* up to the last level of aggregation i.e. the backbone, providing therefore grounds for assurance of the level of services[3].

C. *Policy adaptation considerations*

The resource allocation scheme described in the previous section, if applied in a *fixed* manner usually leads to network underutilization. The left side of fig.4 depicts the situation; ISP1 has allocated resources that stay unused, while at the same time ISP2 has more demand than allocated resources. Although the network has enough capacity to carry all 5 connections, only 3 of them are allowed. The network is clearly underutilized.

Even though domains are free to chose their policy independently, occupying unused resources is not cost efficient, neither for ISP1, probably being charged by CC for $3Mbps$ using only $1Mbps$, nor for CC that has the link to the backbone underutilized at total $3Mbps$ out of $5Mbps$ leased, probably being charged as well. ISPs monitor their traffic and if the allocation for the aggregate is not representing the current traffic average, ISPs release or allocate more (if available) resources to accommodate traffic level. Therefore, if situation at ISP1 stays the same for a while, ISP1 releases the unused $2Mbps$ back to CC.

On the other hand, ISP2 needs more allocated resources to satisfy traffic demand. During

[3]It is a work in progress to examine the impact of *de-aggregation* of traffic, i.e. when backbone traffic is distributed to the terminating subtrees, on the overall performance

initial allocation period, ISP2 requests are being rejected by CC since there are no resources available for allocation. However, once the unused $2Mbps$ of ISP1 are released back to CC, ISP2 request is granted and can now accommodate the traffic demand. The left side of fig.3 depicts the network state after adaptation.

The choice of feedback mechanism becomes important here, as it is subject to the well-known performance vs. overload tradeoff. A crucial factor to this choice is that statistical characteristics of aggregates of large number of connections change slowly and quite predictably. Therefore, network can dynamically adapt to the current load average and achieve good utilization level. At the same time, service level commitments to the traffic flows can be kept, without overloading the network with a lot of control packets, as would be the case for per-connection schemes.

III. The proposed scheme

A. *The DiffServ framework and terminology*

Since the scheme proposed in this paper preassumes a network with DiffServ capabilities to operate, it is orderly that we give a brief description of Differetiated Services Architecture and introduce the terminology which we will follow in the rest of this paper.

Differentiated Services (DS) architecture is based on categorizing traffic into behavior aggregates (BAs) by marking packets with a DS codepoint (DSCP) to receive a particular forwarding treatment, or per-hop-behavior(PHB), at each network node. There is a small number of standard PHBs, namely expedited forwarding(EF)[6], assured forwarding(AF)[7] and best effort(BE). The main characteristics of the three classes are given in tbl.I.

A DS network consists of adjacent domains each of which has administrating, routing autonomy and decides it's own policy at the Policy Decision Point (or points). Policy is a wide concept but in the context of DS, it refers to how much traffic, belonging to which classes, originating from which sources, under which conditions will be allowed to enter the domain. Policy is enforced at the Policy Execution Points, which are the ingress routers of the domain.

Finally, since all domains exist in interdependence, there must be a common understanding and exchange of policy information regarding inter-domain use of the DS codepoints, per-hop-behaviors and policy enforcement. As argued in the previous section, cooperation regarding policy information exchange between independent domains and rational (without contradictions

Class	Alloc.	Prio.	In-Prof	Out-Prof
EF	Yes	High	Guarr.	Drop
AF	Yes	Medium	Assured	DPF set
BE	No	Low	-	-

TABLE I

THE MAIN CHARACTERISTICS OF DIFFSERV PHBS

) policies sets the ground for a high degree of end-to-end QoS assurances over inter-domain connections .

B. The QB Agent hierarchy

In our proposed scheme, allocation and deallocation of network resources is performed by a hierarchy of network resource broking agents, called QB Agents, thus following and reflecting the hierarchical structure of the network, from end-hosts towards the inter-domain backbone.

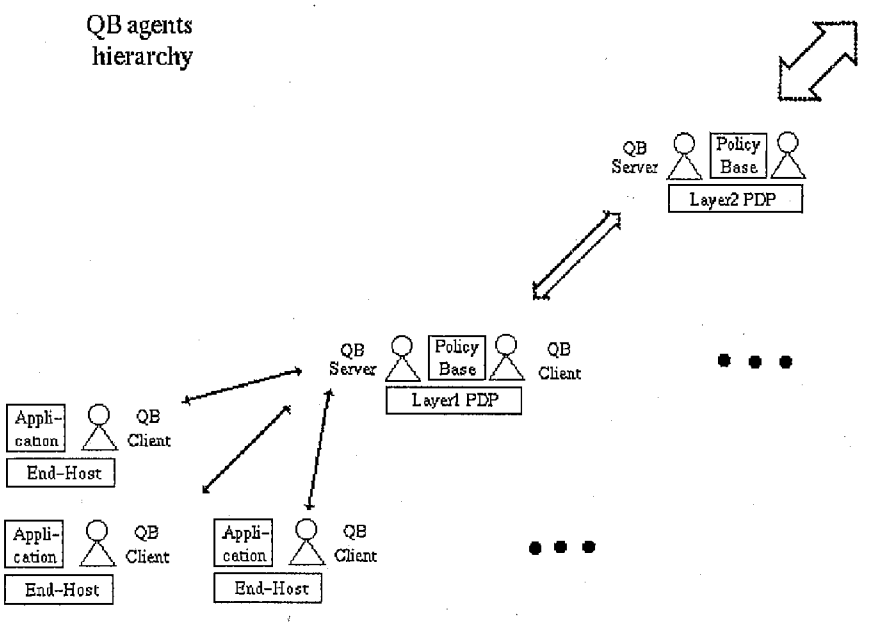

Fig. 5. Hierarchical server-client organization of QB agents

QB Agents follow the *hierarchical server-client paradigm*, illustrated at fig.5. Hierarchical server-client organization essensialy means that the agents of layer n of hierarchy act as servers towards the agents of layer $n-1$, while at the same time act as clients towards the agents of layer $n+1$. Thus, at every layer, QB agents allocate network resources from the layer directly above and make these resources available to the layer directly bellow. For simplicity we will refer to QB agents of layer n as *QB servers*, when they act as servers towards the agents of layer $n-1$, and as *QB clients*, when they act as clients towards the agents of layer $n+1$.

QB client and server agents communicate in order to negotiate resource allocation/deallocation. Negotiation is restricted between agents from adjacent layers within the same subnet, that is, client agents of layer $n-1$ interact only with server agents of layer n and server agents of layer n interact only with client agents of layer $n-1$ that of the same subnet. In this way, negotiation that involves state information stays local and does not proliferate towards the backbone, into the entire network.

QB agents of layer n, "hide" individual transactions with their layer $n-1$ clients from the servers of layer $n+1$, negotiating network recourses for the *aggregate* of those transactions. In fact, QB agents of layer n, request allocation/deallocation of network resources from layer $n+1$ and grant/reject requests to layer $n-1$ obeying to the specific policy of the domain. Every domain is free to choose it's own policy about the amount of resources that should be allocated and the way these resources are made available to clients of the layer bellow. However, since layer n is provisioning layer $n-1$, the *demand* from layer $n-1$ is a dominant factor in deciding the policy at layer n. In our proposed scheme, we assume a rational policy of optimizing resource utilization at every layer.

QB Agents can communicate using any resource allocation protocol that allows localized negotiation of network resources, as long as the protocol can support the basic DiffServ framework. In this paper, we use a simplified protocol to model the transactions between QB agents, called the *QB protocol.* QB protocol is a bare-bones resource allocation protocol developed specifically for localized negotiations. The internals of a QB message are shown in tbl.II. Header consists of information about the QB agent sending the meesege and the host it resides. Payload consists of one or more structures representing transactions.

The type of transaction made using the Qb protocol can be one of the following:

Header	
trans_num	trans_type
src_addr	dst_addr
c_p	buy_own | sel_avl
[...]	

TABLE II

THE STRUCTURE OF A QB PACKET

- {**NOTIF_REQ, TRANS_REQ**} which are issued by a QB client,
- {**NOTIF, TRANS_SUCC, TRANS_FAIL**} which are issued by a QB server.

NOTIF_REQ is a request for notification about the current state of allocations at server's database. It is used to update information about what kind and what amount of network resources are currently available. The server responds with a **NOTIF** message, currying the requested information. As the network state changes dynamically, clients update their allocation information asking for notifications in a *asynchronous* manner.

With a **TRANS_REQ** a QB client requests a transaction from the QB server. The characteristics of the transaction are stated in the fields that follow. The server decides on the request based on the current allocation table; if the request is feasible, the server commits the transaction and replies with a **TRANS_SUCC**, at the same time informing the client about the state after the commitment of the transaction. If the request is not achievable the server responds with a **TRANS_FAIL**, similarly informing the client about the current allocation state.

Requests and replies refer to a quantification of QoS resources being requested/granted. The issue of quantifying QoS and extracting negotiatable parameters is an open issue in the area of communication networks; to this moment, there is no general consensus about how QoS resources or guarranties should be formed. Any set of tangible parameters that can be quantified in a negotiatable form can, theoretically, be used inside these fields. However to make things concrete, in our current work we have considered transmission bandwidth as the single negotiatable parameter, as it is the dominant QoS parameter for which relatively agreed-upon, simple allocation methods (Weighted Class-Based queueing etc.) exist.

C. Dynamics of the QB scheme

In this subsection we examine the events taking place during network operation of the QB agent hierarchy equipped with the QB protocol.

When a *QoS-aware* application is about to begin a transmission, it first contacts the local QB client agent and requests for resources. The QB client at the end host sends a **TRANS_REQ** message carrying these requests to the QB server responsible. The server examines the requested resources against the resource allocation database kept at the Policy Decision Point of the domain.

If the request is deemed feasible, server commits the transaction. This means that a profile is set at the conditioner of the ingress router from which the client will inject traffic. The profile describes the characteristics of traffic that will be allowed in the domain by the conditioner, that is, codepoint and transmission rate. The current allocation state is notified to the client with a **TRANS_SUCC**. From that point, the application is free to transmit it's traffic marked with the DSCP for the particular class for which the allocation was done. As long as application's traffic stays within this profile, performance should meet the QoS level of the particular class. What happens to out-of-profile traffic is implementation specific. For the purposes of this paper, out-of-profile EF traffic is dropped and out-of-profile AF traffic has its Drop Precedence Flag set by the conditioner at the ingress of the domain.

If the request is deemed infeasible, server rejects it and responds with a **TRANS_FAIL** together with an update of the current allocation state. If application has *compromisable* requirements, that is can do with less resources than initially requested, the QB client re-initiates a transaction request asking for the compromised version of application's requirements, and the cycle is repeated. If even the last compromised request is not granted, or application's request was not compromisable in the first place, application is not allowed to transmit and the particular connection is considered lost to the network[4].

Server monitors the demand on various classes of traffic and when dictated by domain's policy, takes the client role and requests for allocation/deallocation of resources from the server at the next level of hierarchy in a similar fashion as the end-host client case described above. The

[4]The number of lost connections is the main evaluation criterion for the performance of the proposed scheme, as described at the experiments section.

difference is that the resources requested are meant for the aggregate traffic of the domain, and will consequently be allocated to end-host clients in order to accommodate traffic flows from applications. The point here is that aggregation of state is achieved by keeping the balance between incoming-outgoing trafficat aggregate level. As long as the deal is kept at the upper layer, all allocations at lower layers are automatically satisfied and performance standards are met. On the other hand, if rejections have to be made, it is more efficient to be made at source level so that excess traffic does not overload the network and degrade performance.

Similar dynamics take place at every intermediate layer up to the backbone QB servers for inter-domain transmission. Since there is no higher layer, backbone servers never play the client role, but only perform load balancing between the subtrees of the network, optimizing the overall performance.

IV. Experiments

A. Experimental setup

We have implemented our simulation model on a ns-2 simulator[8]. In our experiments we considered random tree-like hierarchical topologies, derived by a tree topology generator. Topologies have 4 layers of nodes, namely end-hosts, ISPs, CCs and backbone servers (this forms up 3 levels of hierarchy; LANs,MANs and a WAN)

The capacity of the links was set as follows. All access links were set to 1Mbps. Each link from an ISP to a CC was set to the 60% of the total capacity of the access links of the ISP. Links from CCs to backbone servers were set to 80% of the total capacity of the ISP links, and the backbone was set at 80% of the total incoming capacity. This kind of allocation practically means that an average load of more than 0.8, injected in the network via the access links, is enough to congest the network at every layer.

We have modeled application behavior by the following parameters:

- *average rate, size and duration* are the intrinsic characteristics of the traffic created by the application. "Call"-like applications have rate and duration defined, while "transfer"-like applications have size defined.
- *class and transmission rate* are the negotiatable QoS parameters of the application. Both class and rate might be compromisable or non compromisable, depending on the application.

We have considered 3 *application profiles*, namely "Netmeeting" (non-compromisable 1Mbps CBR of EF), "Chat"(compromisable 1Mbps exponential on-off of AF), "FTP"(BE).

End-host behavior was modeled using the following parameters:

- *connection utilization* which practically means the percentage of time that the source will inject traffic into the network. Adjusting the connection utilization parameter of end-hosts we can adjust the average load of the experiment.
- *session density* big session density yields many,short sessions while small session density yields few,long sessions.
- *application probability* vector, which gives the probabilities the a session will be of a particular application

We have set connection utilization at 0.8, session density at 1 and considered 3 end-host *profiles*: "mostly EF", "mostly AF" and "mostly BE" type. We have extensively studied mixtures of EF and BE traffic, and have not completed the study on mixtures of all three classes.

QB Agents of all layers obey the respective domain policy. End-host agents obey the intrinsic "policy" of applications, i.e. allocate bandwidth when needed for a session and deallocate when session is over. Backbone server agents do load balancing, by adjusting the allocations in the subtrees to meet the total demand. Agents at intermediate nodes can follow any allocation policy they please; however, for the experiments we have assumed a policy of adaptive allocation to meet end-host demand.

B. Effectiveness and scalability of the scheme

Figures 6 and 6 illustrate the scalability of our proposed system. $x-$axis depicts the number of end-hosts injecting traffic into the network while $y-$axis depicts the percentage of successful EF and BE sessions.

Theoretical curves are derived from first-order approximation of traffic characteristics. Specifically, according to the experimental setup described in the previous chapter, n end-hosts are attached to the network through 1Mbps links each, while the backbone is set at $n \times 0.6 \times 0.8 \times$ 0.8Mbps. From this capacity 10% is kept to accommodate best-effort traffic so the number of *maximum possible simultaneous 1Mbps sessions* of EF is given by

$$\lfloor 0.6 \times 0.8 \times 0.8 \times 0.9 \times n \rfloor$$

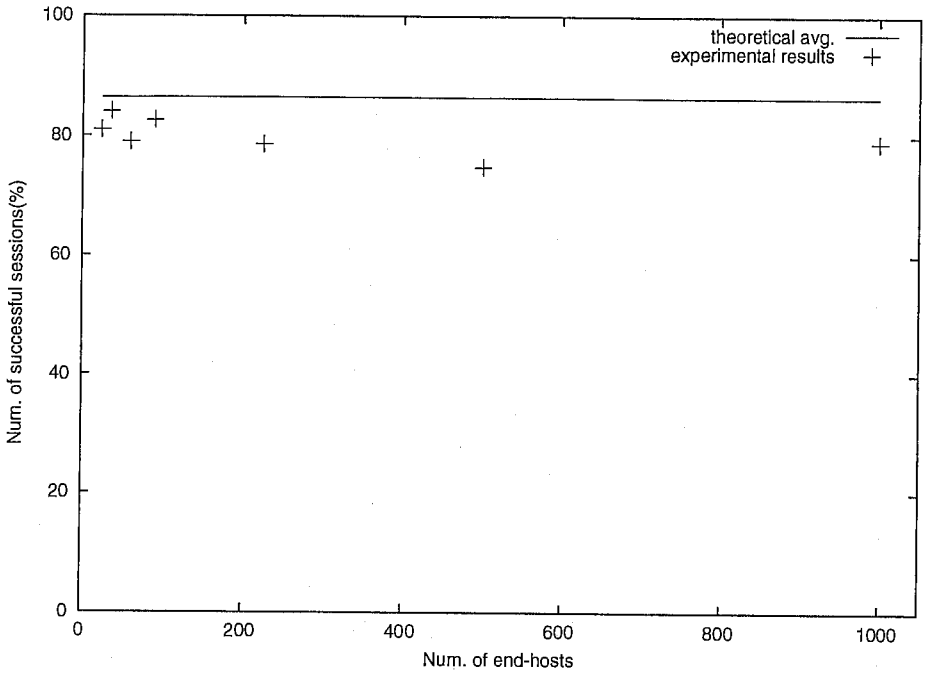

Fig. 6. Succesfull EF sessions vs. number of end-hosts.

On the other hand, n sources set at 0.8 connection utilization means that on the average there is demand for $0.8 \times n$ simultaneous sessions. Since host profiles were randomly selected between "mostly BE" and "mostly EF", on the average half of these sessions will be BE and half EF, which yields a demand for average

$$0.5 \times 0.8 \times n$$

simultaneous EF sessions.

The ratio of maximum possible simultaneous EF sessions over the average demand for simultaneous EF sessions gives the theoretical EF session success rate of this setup. The theoretical curve for BE throughput follows the same lines.

It is clear that the performance of Qb scheme scales up well to 2 orders of magnitude. Considering the fact tha Qb is a distributed system, i.e. decisions about accepting/rejecting EF sessions are taken *locally* at the ISPs, Qb scheme's performance stays close to the ideally expected behavior.

All successful EF connections got all of 1Mbps of transmission throughput in all topologies[5].

[5]There were occasional drops of EF packets in large-scale scenarios, but simulation traces shown it to be related with queueing. Since queueing was not extensively studied in this paper, we consider these drops irrelevant to the performance of the scheme.

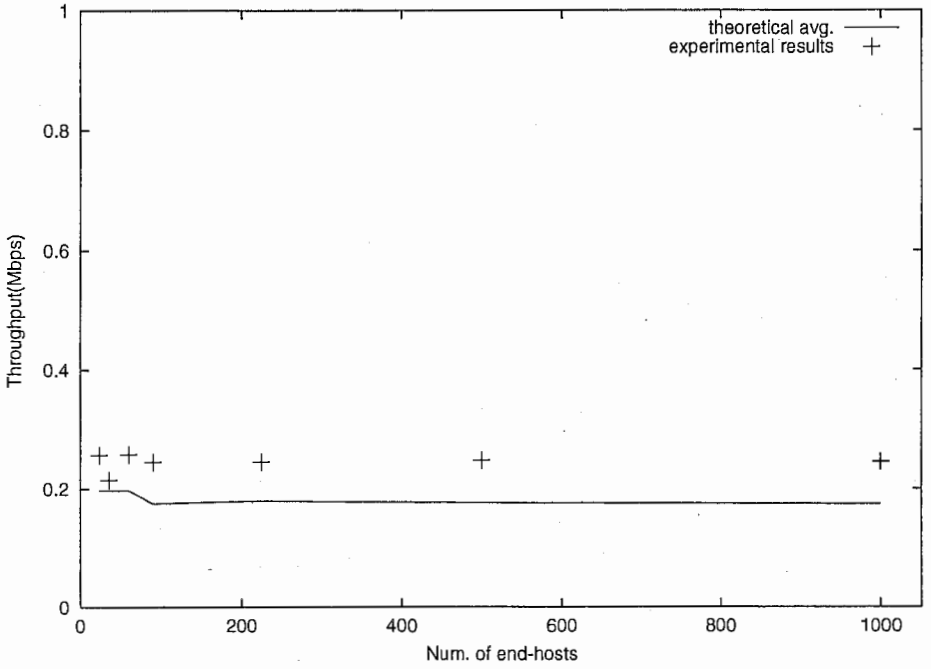

Fig. 7. BE throughput vs. number of end-hosts.

BE traffic was benefited by the fact that all EF connections that will not make it through the backbone are not initiated in the first place, and consequently received 125% of allocated capacity on the average. Therefore, traffic control of EF sessions contributes to high total (all classes) backbone utilization as well.

C. *Effectiveness of congestion controls*

For this experiment, except for the backbone capacity that is an input parameter, a similar topology of 24 end-hosts was used. Backbone capacity is easily identified as the global bottleneck. What is important is, how this bottleneck capacity effects the overall performance of network. The results from the experiments in this subsection illustrate the impact of congestion on performance, in the cases of best-effort network, diffserv-capable network without connection access control or feedback mechanisms and diffserv-capable network utilizing Qb scheme.

In the case of Qb scheme, sessions that are bound to fail are never allowed to initiate. In both best-effort and plain-diffserv networks, since no access control exists, all connections initiate but congestion causes degradation of the level of service. To make grounds for a comparison, in the cases of best-effort and plain-diffserv networks we consider a session successful, if it has

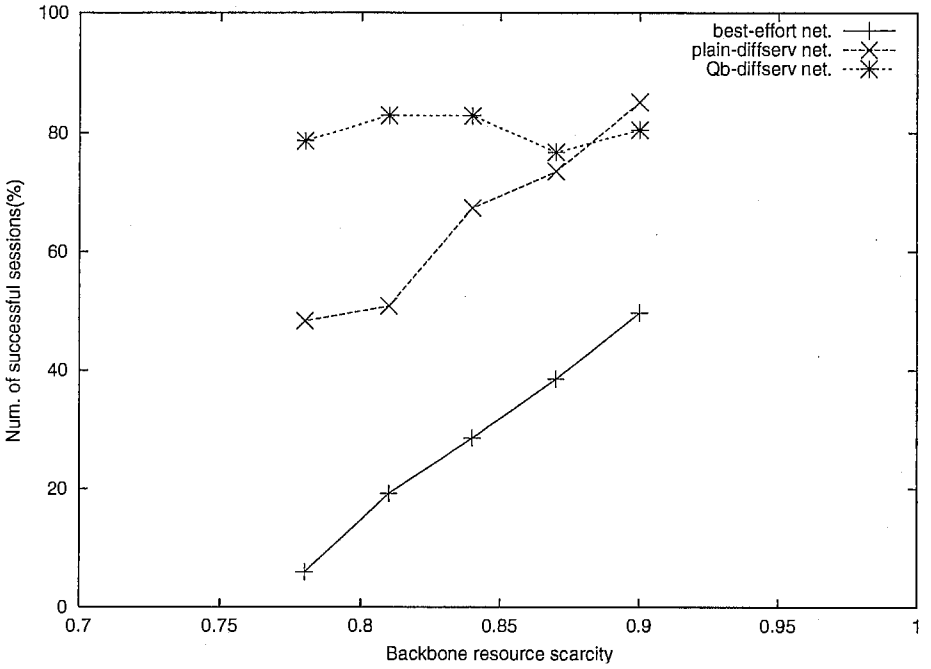

Fig. 8. Successful EF sessions vs. backbone resource scarcity.

suffered less than 5% of packet drops[6].

Figures 8 and 9 illustrate the impact of congestion on performance for the three systems. $x-$axis depicts the resource scarcity (provisioning) at the backbone while $y-$axis depicts the percentage of successful EF sessions and the average throughput of BE sessions.According to the rationale given in the previous subsection, in all cases, at the backbone there is room for a maximum of only 8 simultaneous 1Mbps sessions.

In this case, impact of unsuccessful connections on system performance is obvious. In the case of both best-effort and plain-diffserv networks, performance of both EF and BE degrades significantly with the resource scarcity. The performance of Qb scheme is dominated only by allocatable amount of resources, therefore by not initiating more EF sessions than there is room for, avoids congestion thus delivering good performance both to EF and BE traffic.

These results depict clearly the need for connection access control and feedback methods, in order to be able to give service assurances to applications with QoS requirements. Connection access control is needed as a basic form of congestion avoidance mechanism. Best-effort traffic,

[6]5% is actually a very optimistic bound. Usually audio/video streams and interactive sessions are not acceptable under such degradation[9].

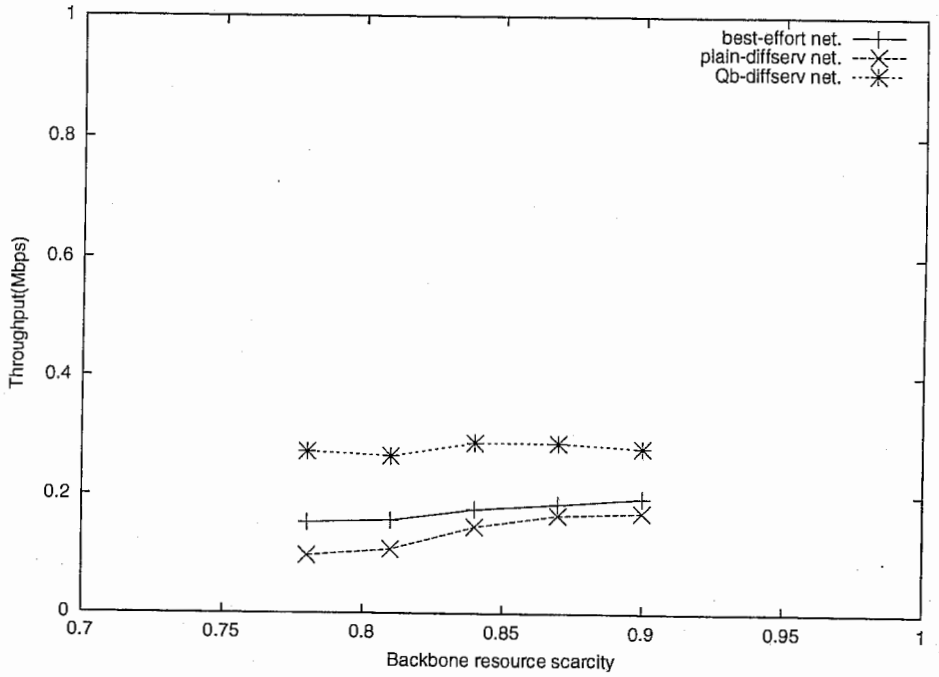

Fig. 9. BE throughput vs. backbone resource scarcity.

usually carried by TCP, has it's own congestion avoidance-control mechanism, which is very effective indeed. However, best-effort style avoidance is based on the assumption that a source can *alter* its transmission rate according to network congestion state, an assumption that does not hold true for most applications with QoS requirements. An effective mechanism for QoS-sensitive traffic, should be based on connection admission control. The results show that the strategy proposed in the Qb scheme is actually effective.

At the same time, since bottlenecks may develop far from traffic sources, a feedback mechanism is needed. It was already argued that this feedback mechanism should relate to *aggregates* and not individual connections, the main problem becomes how to perform effective resource allocation for these aggregates. Qb scheme effectively applies the conservation principle over the hierarchical structure of the network to do resource allocation/deallocation at domain boundaries.

D. Protocol overload on network performance

Qb scheme was designed to maintain scalability, and achieves this by letting agents negotiate resources related to the *aggregation level* of the hierarchical layer that they reside. At the end-

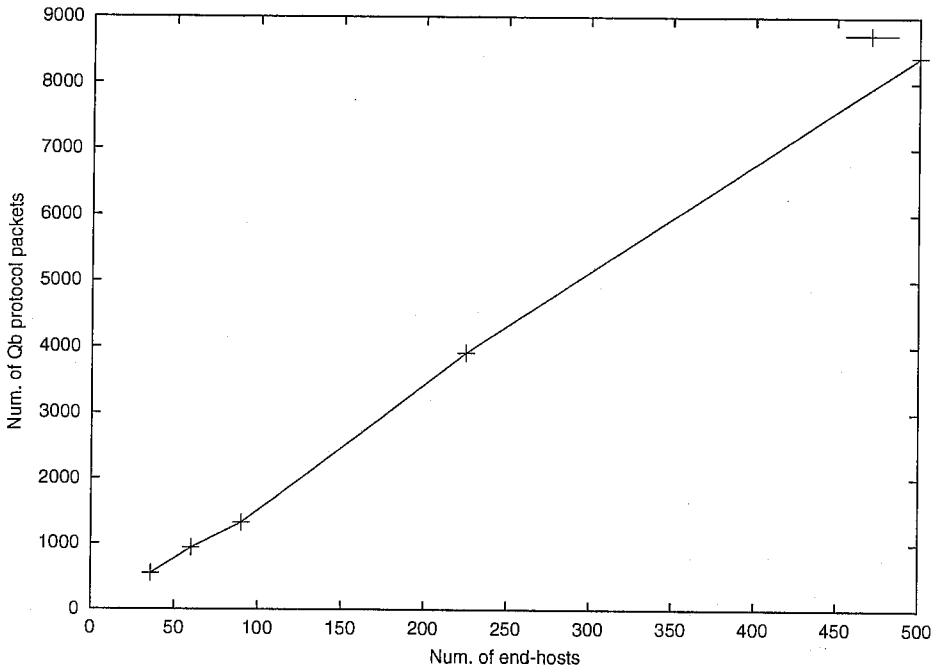

Fig. 10. Exchanged Qb packets vs. number of end-hosts.

host layer, Qb Agents negotiate about individual connections, At ISP layer, Qb Agents negotiate about LAN's aggregate resources and at CC's layer about MAN's aggregate resources. As aggregation degree gets higher, the statistical characteristics change slowly and there not many negotiations are needed. The result is very cost effective; as it is obvious from figure 10 the number of packets exchanged between Qb agents is linear to the number of end-hosts.

Figure 11 shows the additional load due to Qb packets that was carried through the network. The results oscillates but always stays bellow 0.05%. For the simulation, in order to avoid the effects of queueing on performance we have used packets of constant sizes both for application packets and for Qb packets (in reality neither of them is) but the order of magnitude is indicative of cost effectiveness.

E. Current implementation limitations

We concentrated our study at the behavior of the part of the network reaching from end-hosts up to the backbone. Specifically, in the experiments, we have set the traffic sink for all connections at the end of the backbone. Therefore we cannot claim that we have achieved end-to-end performance, but we argue that we have made a step towards end-to-end soft QoS

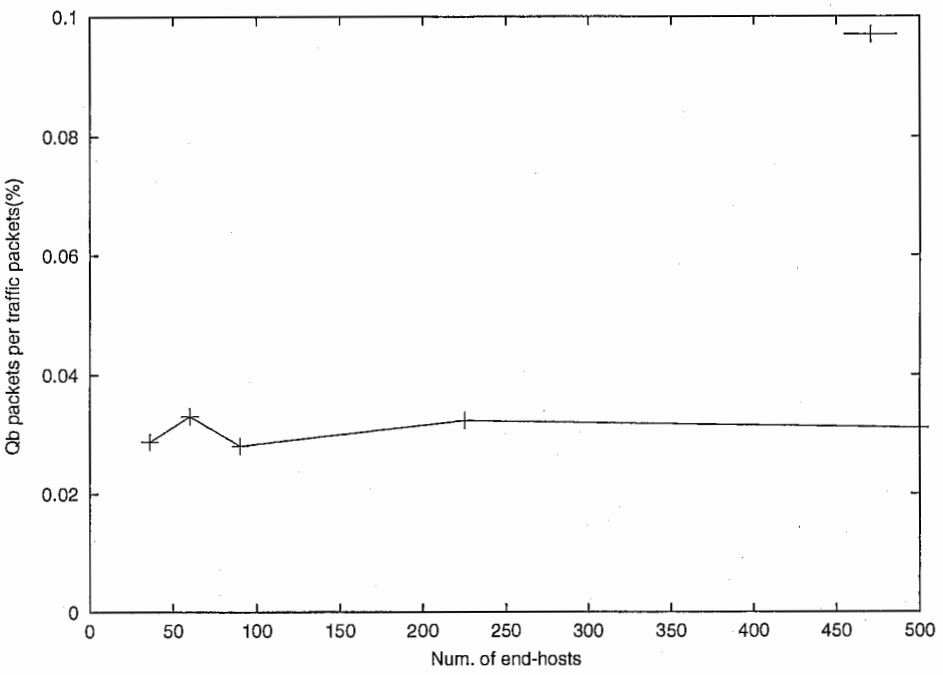

Fig. 11. Additional load due to Qb packets vs. number of end-hosts.

assurances. This argument is associated with the assumption that if network performs well up to the point of maximum concentration, it will *statistically* perform well all the way to the terminating end of sessions. Some explanation is due here; from the designing principle of networks, resources are getting scarce when moving uplink towards the backbone, because more and more connections attempt to share them. On the other hand, when going from the backbone down to the end-hosts, resources are getting abundant, because connections are scattered in the various subtrees. So apart from cases of termination points that do not have enough capacity to support their "popularity" (which is a design problem anyway) we expect that performance degradation is mainly caused because of connections competing when entering the network than connections conflicting close to terminating ends. Moreover we expect this probability to decline with the scale of network.

Up to what extend is the above assumption plausible for real-world scenarios is work currently in progress. However, preliminary results suggest that an extension of the scheme will be needed to cope with non-homogeneous distributions of traffic termination points.

V. Concluding remarks & future directions

In this paper we have described a proposed scheme for providing soft QoS assurances to applications with QoS requirements. The proposed scheme operates over a DiffServ-capable internetwork that exhibits hierarchical structure typical of the Internet.

The proposed scheme reflects the diversity of network infrastructure, thus having all the merits of a fully distributed system. Moving up the hierarchy towards the inter-domain backbone, state is concentrated at every layer, thus exhibiting scalability and avoiding the proliferation of state typical of the connection-style approach. Feedback is kept at a local basis, thus being cost efficient while adaptation to meet demand yields good resource utilization. Resource allocations keep incoming-outgoing traffic balance at every layer, avoiding conflicting allocations at a global level.

We have presented extensive arguments about how the above principles are derived and why such a scheme is applicable in the case of a large-scale network with the peculiarities of Internet. Furthermore we have conducted simulation experiments to evaluate the performance of the proposed scheme and studied network behavior under different experimental scenarios. Analyzing the results we testified the efficiency of the proposed system, while at the same time identified it's current limitations.

Implementation limitations as well as multitude of input/output parameters ask for further experimentation in various aspects of the scheme. More specifically, what are the limitations of soft QoS assurance? Is it possible to give hard guarranties and under which conditions? How can the proposed scheme be combined with other schemes for superior results? We consider these questions as our future work.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services," IETF, RFC2475, Dec. 1998.

[2] E. Zegura, K. Calvert and S.Bhattacharjee, "How to Model an Internetwork," in Proceedings of IEEE INFOCOM, April 2000.

[3] V. Jacobson, "Quality of Service for the Next Generation Internet," NGI whitepaper, March 1997.

[4] T. Aimoto and S. Miyake, "Overview of DiffServ Technology: Its Mechanism and Implementation," IEICE Trans. Inf. & Syst., vol.E83-D, no.5, pp.957–964, 2000.

[5] V. Jackobson and M. J. Karels, "Congestion Avoidance and Control," Proceedings of SigCOMM 1988, pp. 314-329.

[6] V. Jacobson, K. Nichols, K. Poduri "An Expedited Forwarding PHB," IETF, RFC2598, June 1999.

[7] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski "Assured Forwarding PHB Group," IETF, RFC2597, June 1999.

[8] S. McCanne and S.Floyd, "ns Network Simulator," http://www.isi.edu/nsnam/ns/ .

[9] D. Hands and M. Wilkins, "A Study of the Impact of Network Loss and Burst Size on Video Streaming Quality and Acceptability," Proccedings of IOMS'99, 1999.