Publication P1

Jani Lakkakorpi, Ove Strandberg, and Jukka Salonen. 2005. Adaptive connection admission control for differentiated services access networks. IEEE Journal on Selected Areas in Communications, volume 23, number 10, pages 1963-1972.

Adaptive Connection Admission Control for Differentiated Services Access Networks

Jani Lakkakorpi, Ove Strandberg, and Jukka Salonen

***Abstract*—In our earlier work, we have proposed some modifications for the bandwidth broker framework. With our modifications, it is possible to use measurement-based admission control in addition to the more traditional parameter-based admission control. Moreover, we have presented a new flexible admission control scheme that has proven to be very efficient in terms of bottleneck link utilization. Two problems, however, have arisen: the use of scheduling weights in admission control and bursty connection arrivals. In this paper, we present that the former one can be dealt with the use of adaptive scheduling weights, while the latter one can be fought with adaptive reservation limits. The proposed new algorithms are validated through simulations and their performance is compared against the nonadaptive basic scheme.**

***Index Terms*—Connection admission control, differentiated services (DiffServ), Internet protocol (IP) quality-of-service (QoS).**

I. Introduction

As MORE AND more IP-based applications with quality-of-service (QoS) requirements keep on emerging, it becomes more and more evident that there is a need for connection admission control (CAC) in Internet protocol (IP) networks, too. For a transmission control protocol (TCP)-based file transfer, there is nothing dramatic in a situation, where the bandwidth of the connection is suddenly halved—we just have to wait a little bit longer. For a voice-over-IP (VoIP) conversation, however, this would not be acceptable due to excessive packet delay and loss.

Thus far, one of the designing principles in the IP has been that datagrams are delivered as best effort (BE), that is, all packets are treated equally in the routers and no guarantees about packet delivery are given. However, nowadays, the formerly more or less unused type-of-service (TOS) field in the IP packet header can be used for differentiating more important or urgent packets from the less important or urgent ones. Additionally, if we want to support different types of reservations (e.g., "hard" versus "soft" reservations) for different types of connections, our routers need to have the proper built-in mechanisms, such as priority queueing or deficit round-robin (DRR) [1] for packet scheduling and random early detection (RED) [2] for congestion management, in order to allow this differentiation.

Manuscript received February 15, 2005; revised March 26, 2005. This paper was presented in part at the IEEE GLOBECOM 2004 Conference, Dallas, TX.

J. Lakkakorpi and O. Strandberg are with the Nokia Research Center, 00180 Helsinki, Finland (e-mail: jani.lakkakorpi@iki.fi; ove.strandberg@nokia.com).

J. Salonen is with Nokia Networks, 33100 Tampere, Finland (e-mail: jukka.v.salonen@nokia.com).

Digital Object Identifier 10.1109/JSAC.2005.854121

At this stage, the concept of reservation needs to be clarified. A reservation is a set of connection admission control, packet scheduling, and dropping rules that have to cooperate in order to realize the desired behavior. There can be admission-controlled users and nonadmission controlled users. The former group should consist of such users that can actually benefit from reservations (e.g., VoIP or video streaming users), while the latter group, using, for example, file transfer protocol (FTP), would not necessarily benefit from reservations.

A. Differentiated Services (DiffServ)

After the failure of integrated services (IntServ) [3], differentiated services (DiffServ) [4] has been the biggest effort for adding QoS in the Internet. This effort eventually resulted in a scalable architecture, where no state information is needed in the core routers but they can concentrate on packet forwarding using appropriate scheduling and dropping mechanisms. Edge routers take care of traffic classification, marking, and policing. Packet remarking, dropping and shaping can be applied to nonconforming flows. The standardized per-hop behaviors (PHB) are expedited forwarding (EF) [5], and assured forwarding (AF) [6]. Naturally, BE is supported as well.

According to its original definition, EF is a "virtual leased line" treatment that can be used to build a low loss, low latency, low jitter, assured bandwidth end-to-end service through DiffServ domains. EF is usually implemented with priority queueing, which requires the use of rate limiter (with adjustable rate and burst size) protecting other traffic. Naturally, the edge policing has to be strict for this kind of traffic.

AF, on the other hand, can be used for different purposes. According to the official definition, the AF PHB group provides delivery of IP packets in four independently forwarded AF classes. Within each class, an IP packet can be assigned one of three different levels of drop precedence. Moreover, reordering of packets belonging to the same microflow is not permitted if the packets belong to the same AF class. Weighted scheduling using, e.g., DRR is probably the only reasonable way to implement AF, since RFC 2597 states the following: "where each AF class is in each DS node allocated a certain amount of forwarding resources (buffer space and bandwidth)." Differentiated packet dropping can be implemented by applying RED for each AF class and drop precedence level separately.

One practical way to use AF is to implement the so-called "Olympic service," which consists of three service classes: gold, silver, and bronze corresponding to AF classes 3, 2, and 1. Packets are assigned to these classes so that gold class packets experience lighter load than silver class packets, which in turn experience lighter load than bronze class packets. AF class

selection can be based, for example, on application requirements. Moreover, packets within each AF class can be further separated by assigning them either high, medium, or low drop precedence corresponding to AF drop precedence levels 3, 2, or 1. Packets with low drop precedence level are the most important ones. The drop precedence level of a packet can be assigned, for example, by using a leaky bucket traffic policer at the network edge.

A major problem with AF is the management of weights—it is very hard to build the aforementioned "Olympic service" unless the weights are configured in strict priority fashion. Otherwise, we would have to know the traffic mix accurately and adjust the weights accordingly. Of course, "Olympic service" is not the only service model that can be implemented with AF.

During the lifetime of the IETF DiffServ working group, several other proposals emerged as well, most notably simple integrated media access (SIMA) [7], which decouples packet urgency (delay priority) and importance (loss priority) from each other. This is something that is not possible with AF, since precedence matters only within an AF class. In SIMA, packet importance is dynamically calculated based on the ratio of measured bit rate and nominal bit rate (that the customer has bought) at the edge router. Urgency, however, depends on the application needs.

Probably the biggest handicap of DiffServ is that the management issues are not properly dealt with. It is crucial that the volume of EF traffic on a bottleneck link stays below the corresponding limit as otherwise the rate limiter will start dropping EF packets. Likewise, in the case of "Olympic service," the traffic volumes of different AF classes should be linked to the corresponding AF weights in order to implement the delay differentiation goal. Admission control is something that could fill this gap. Actually, it is even mentioned in RFC 2597 that the AF PHB group can be used to implement a low loss and low latency service using an overprovisioned AF class—if the maximum arrival rate to that class is known *a priori* in each DS node. It is also mentioned that the specification of the required admission control services is beyond the scope of the RFC.

B. Access Network is the Bottleneck

The last mile in many access networks consists of narrow-bandwidth links, e.g., leased lines. DiffServ can help to utilize these links in the most effective manner. DiffServ is managed trough service level agreements (SLAs), whose enforcement should preferably include dynamic admission control [8]. Otherwise, the narrow-bandwidth access networks could become heavily congested or underutilized. It is possible to do dynamic admission control in IP networks, e.g., by probing the path with active measurements or use an agent called bandwidth broker [9] to assist in the decision whether a connection is admitted to the network or not. We have selected the latter approach.

The rest of this paper is organized as follows. Section II presents our modifie.er framework and the basic admission control algorithms [10], [11], while Sections III and IV are the core content of this paper—they present our adaptive scheduling weight and reservation limit tuning algorithms, correspondingly. Section V validates the proposed scheme.

II. Modified Bandwidth Broker Framework

There are many alternative ways to do admission control in IP networks—although none of them is widely used at this moment. These schemes can be first divided into parameter-based admission control (PBAC) and measurement-based admission control (MBAC). MBAC can be further divided into schemes that involve active measurements (i.e., sending probe packets) and schemes that do not involve active measurements.

This paper focuses on the bandwidth broker approach. Bandwidth brokers have traditionally been designed to support PBAC only. However, this paper introduces modifications to the traditional bandwidth broker framework that will enable both the use of link measurements and reservation information in admission control decisions. The use of adaptive packet scheduling weights and adaptive reservation limits in the admission control algorithm shall be introduced, too.

Our access network is equipped with DiffServ routers—only standardized PHBs (EF, AF, and BE) have to be supported. DiffServ is needed in realizing the bandwidth reservations. In addition to that an improved bandwidth broker, that knows all link loads and existing bandwidth reservations per DiffServ class, is needed. Different admission control rules are applied for different types of connections and the same traffic class separation is performed in the routers as well. Packet scheduling has an essential role in protecting admission-controlled traffic from nonadmission controlled traffic—as well as in protecting the admission-controlled traffic classes from each other. During congestion periods it is usually the nonadmission controlled traffic that has to adapt, while sufficient QoS is guaranteed for the admission-controlled traffic classes. However, some resources are reserved for the nonadmission controlled traffic, too. Naturally, the admission control function in the bandwidth broker is fully aware of the packet scheduling parameters.

In order to realize flexible bandwidth sharing, link capacity is divided in a hierarchical fashion so that there is a configurable total limit for all admission-controlled (non-BE) traffic, configurable limits for real-time and nonreal-time traffic aggregates, and configurable limits for each admission-controlled traffic class. Moreover, for each stage, there is a reservation limit and a load limit—it is possible to combine MBAC and PBAC. By carefully constructing the admission control hierarchy and setting the corresponding limits, we can fulfill the QoS requirements of our traffic without compromising the goal of high bottleneck link utilization. However, fulfilling the application requirements is not enough. We must also think what the operator gets out of all this—if the answer is nothing, the deployment of QoS is not going to happen. When reservations are requested, the peak rate of the connection is multiplied by a price factor before comparison to available bandwidth. This provides the means for the operators to maximize their revenue.

Since our admission control framework supports multiple admission-controlled classes, we need to pay special attention to scheduling issues. If there is a single admission-controlled class, say EF, we only have to make sure that the reserved EF bandwidth and the measured EF load are below their corresponding limits on each link. Packet scheduling will protect EF traffic from nonadmission controlled traffic—but only if the relationship between the applied limits and the scheduling

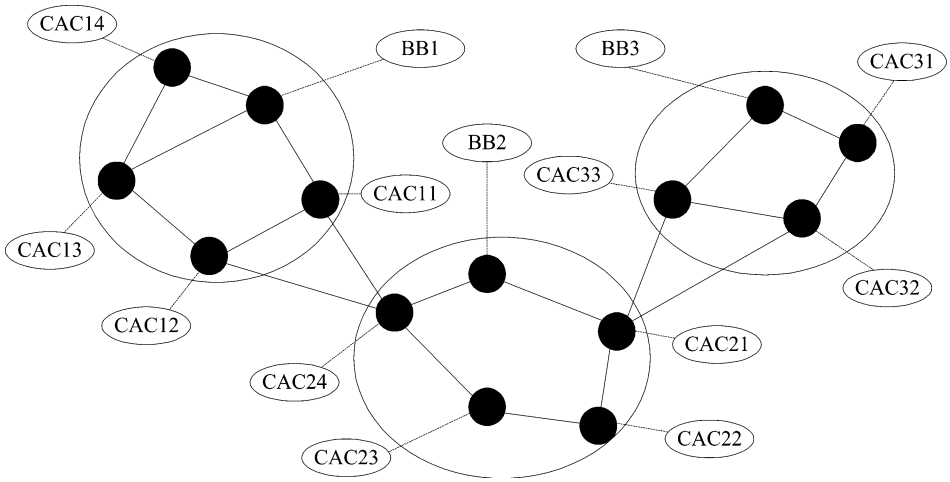

Fig. 1. Bandwidth brokers, CAC agents, and their routing domains.

parameters is appropriate. However, if there are multiple admission-controlled classes, we have to take their scheduling weights into account in admission control or configure the weights in strict priority fashion, which would result in the aforementioned "Olympic service" model—and, thus, in delay differentiation between the AF classes. If the AF scheduling weights are not configured in strict priority fashion, we have to somehow figure out what are the most suitable weights. In this paper, a novel mechanism where the AF scheduling weights partly depend on the volume of reservation requests per AF class is proposed.

The other adaptive feature that shall be presented in this paper relates to reservation limits—if they are set high enough, admission decisions will be made based on link measurements only. This approach, "pure MBAC," has its risks. If we experience a sudden "burst" of connection arrivals, there is nothing that MBAC can do—it will take some time until the updated link loads arrive at bandwidth broker. The solution is to make the reservation limits adaptive—they would depend on the current link loads. This will provide us with "safety margins."

Due to the fact that average bit rates can be substantially lower than the corresponding requested peak rates, the use of PBAC can leave the network underutilized. Link load measurements are needed for more efficient network utilization. EF and BE loads have already been suggested for the QBone architecture [12]. In theory, it is possible that all admitted traffic sources start sending data at their peak rates at the same time. However, the probability for this is extremely low—especially if the number of traffic sources is very high. Moreover, it is possible to protect oneself against such an event by carefully combining MBAC and PBAC.

We present a flexible admission control mechanism for Diff-Serv access networks by extending and modifying the existing bandwidth broker framework proposed by Nichols *et al.* [9] and later implemented by Schelén [13]. The extra information needed for measurement-based admission decisions—link loads—is retrieved from router statistics and it is periodically sent to the bandwidth broker agent of a routing domain. As a second enhancement, we allow connection admission control for multiple traffic classes, e.g., EF, AF1, and AF2. The motivation for doing CAC for selected AF traffic is that there are

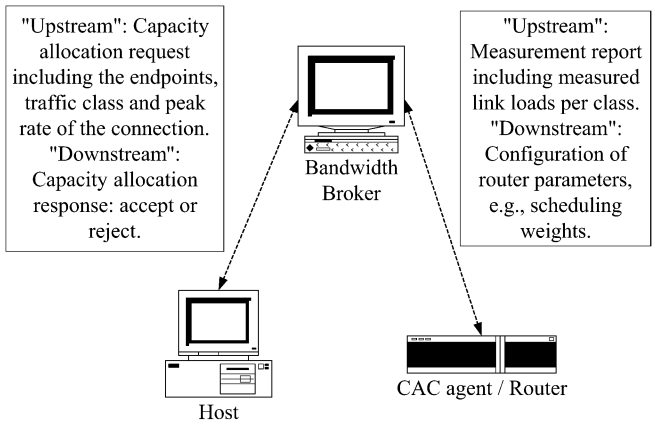

Fig. 2. Required signaling traffic in the proposed bandwidth broker framework. Periodic measurement reports provide the bandwidth broker with link loads and, thus, allow the use of MBAC.

real-time applications with relaxed QoS requirements. These traffic sources (e.g., video or audio streaming) do not need the "virtual wire" (EF) treatment. Some statistical guarantees, however, have to be provided.

In our modified bandwidth broker framework, we have added a CAC agent in all routing domain nodes (see Fig. 1). One of these CAC agents will act as the bandwidth broker by storing the information on reservations and measured link loads within the routing domain. Just like in [13], the bandwidth broker knows the routing topology by listening to OSPF [14] messages. Link bandwidths within the routing domain are obtained through the simple neywork management protocol (SNMP) [15].

In addition to reserved link capacities for different traffic classes, the admission decision is based on measured link loads on the path between the endpoints (see Fig. 2). If there is not enough both unoccupied and unreserved bandwidth on the path, the connection is blocked. Maximum reservable bandwidth on a link can exceed link capacity. Thus, when the maximum reservable bandwidth is high enough, it is the unoccupied bandwidth only that matters. The relationship between the maximum reservable bandwidth and link bandwidth is configurable for each traffic class.

A. Available Bandwidth Calculation

A CAC agent monitors and updates the loads of those links that are attached to its local router, while the bandwidth broker applies exponential averaging on the loads received from all CAC agents. The CAC agents send their current link loads every p s to the bandwidth broker. During a single measurement period, the link loads are sampled p/s times, and at the end of each measurement period the maximum value is selected to represent the current load. Whenever a measurement report arrives to bandwidth broker, the link database is updated by recalculating the applicable link loads and unoccupied link bandwidths for each traffic class and superclass

$$load_{class} := (1-w) * load_{class} + w * currentLoad_{class}, \quad (1)$$

$$unoccBw_{class} = bw * (loadLim_{class} - load_{class}). \quad (2)$$

Exponential averaging weight (w), measurement period (p), and sampling period (s) should be carefully selected, and their optimization is for further study. The optimal values for w and p depend on traffic patterns and how fast we want to adapt to changes in link loads. Unreserved bandwidths are updated whenever a reservation is setup, modified, or torn down while available bandwidths are calculated only when there is a resource request for a specific path

$$unresvBw_{class} = bw * (resvLim_{class} - resv_{class}), \quad (3)$$

$$avBw_{class,path} = \min_{link \in path}(\min(unoccBw_{class,link}, unresvBw_{class,link})). \quad (4)$$

With bw we denote the link bandwidth, $load_{class}$ denotes the measured link load for a given class, while $resv_{class}$ denotes the reserved link capacity for a given class. For AF classes, the calculation of available bandwidth can be more complex. This is due to weighted scheduling between the AF queues. We can either configure the weights for all AF queues in strict priority fashion and apply (2)–(4), or we can take the AF weights $(weight_{AFi})$ into account when calculating the unoccupied bandwidth values for each link

$$unoccBw_{AFi} = bw * \min\left(loadLim_{AFi} - load_{AFi}, 1 - load_{EF} - \frac{load_{AFi}}{weight_{AFi}}\right). \quad (5)$$

B. Flexible Connection Admission Control

If we concentrate too hard on real-time application requirements, it may be impossible to use business [16] or any other objectives in CAC decisions. In flexible CAC, real-time connections cannot claim all the bandwidth since link bandwidth between RT and NRT traffic is shared dynamically. Instead of a constant value, the load limit for RT traffic will be the minimum of total load limit less NRT traffic load and maximum RT load limit. Similarly, the load limit for NRT traffic will be the minimum of total load limit less RT traffic load and maximum NRT load limit:

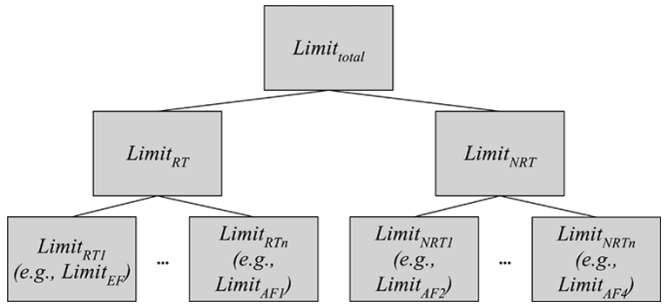

Fig. 3. Load/reservation limit hierarchy.

$$loadLim_{RT} = \min(loadLim_{total} - load_{NRT}, loadLim_{RT\_MAX}) \quad (6)$$

$$loadLim_{NRT} = \min(loadLim_{total} - load_{RT}, loadLim_{NRT\_MAX}). \quad (7)$$

The total load limit is there in order to protect nonadmission controlled traffic—if one wants to protect it. Moreover, we can take the reserved link capacities into account in our admission decisions—reservation limits for RT and NRT traffic are calculated just like the load limits:

$$resvLim_{RT} = \min(resvLim_{total} - resv_{NRT}, resvLim_{RT\_MAX}) \quad (8)$$

$$resvLim_{NRT} = \min(resvLim_{total} - resv_{RT}, resvLim_{NRT\_MAX}). \quad (9)$$

We can prioritize either PBAC or MBAC by tuning the maximum capacity that can be reserved for a given traffic class on a link $(resvLim_{class})$. If the reservation limit is low enough, it will be the PBAC that will rule. Fig. 3 illustrates the load/reservation limit hierarchy. Three limits can affect each admission decision: total limit, RT/NRT limit and own class limit. However, each level in the hierarchy does not have to affect, i.e., we can, for example, set the NRT limit to equal the total limit. Note that a limit cannot exceed its parent class limit.

Probably the most practical way to apply flexible CAC is to configure all AF scheduling weights in strict priority fashion so that AF1 has the biggest weight—this results in delay differentiation between different AF classes. However, it is also possible to apply (5) for calculating the unoccupied bandwidths for AF classes. The latter method will most probably result in lower admission ratios and resource utilization, but it may be useful when the goal of using AF is not delay differentiation but something else—like bandwidth sharing.

In addition to dynamic RT and NRT limits, we have introduced a coefficient that is a function of the price the user is paying for a given service. The requested bandwidth (peak rate) is multiplied by this coefficient, and the result is compared with available bandwidth. If, for example, $f(price) = 1.0$, we favor connections with the lowest peak rates.

```
Bandwidth Broker:
for each admission request:
  classify connection (class = EF/AF1/AF2)
  admit = true
  if (class != AF2)
    calculate avBw_class, path and avBw_RT, path
    if ((avBw_class, path < f(price)*requestedRate) OR (avBw_RT, path < f(price)*requestedRate))
      admit = false
  else
    calculate avBw_NRT, path
    if (avBw_NRT, path < f(price)*requestedRate)
      admit = false
  if (admit == true)
    for all links on the path:
      resv_class =+ requestedRate
      re-calculate unresvBw_class, unresvBw_RT, unresvBw_NRT

for each connection tear-down:
  classify connection (class = EF/AF1/AF2)
  for all links on the path:
    resv_class =- requestedRate
    re-calculate unresvBw_class, unresvBw_RT, unresvBw_NRT

for each measurement report arrival:
  update link database: re-calculate unoccBw:s

All CAC agents (including Bandwidth Broker):
timer expires:
  update link loads
  send update to Bandwidth Broker
  set timer to expire after p seconds
```

Fig. 4. Flexible CAC algorithm instance: admission decisions for RT (EF, AF1) and NRT (AF2) connections.

In flexible CAC, RT could denote, for example, the aggregate of EF and AF1 traffic classes. However, the scope of RT can be extended to cover more traffic classes. Similarly, NRT could include just AF2 traffic, but its scope can be extended to cover more traffic classes (see Fig. 3). Adjustable parameters are the following: $loadLim_{total}$, $loadLim_{RT\_MAX}$, $loadLim_{NRT\_MAX}$, $resvLim_{total}$, $resvLim_{RT\_MAX}$, $resvLim_{NRT\_MAX}$, and the load and reservation limits of individual traffic classes (e.g., EF, AF1, and AF2).

Fig. 4 illustrates how admission decisions are made in an example flexible CAC instance with three traffic classes. New connections request resources (peak rate from source to destination) from the bandwidth broker of their own routing domain. Other bandwidth brokers may have to be consulted as well if the destination is not in the same domain. If there are enough resources, the requested bandwidth for the admitted connection is added to reserved values for all links along the path. Otherwise, the connection is rejected. Policing is needed for all admitted flows to keep their peak bit rates below the agreed ones.

III. Adaptive AF Weight Tuning

Flexible CAC offers two operating modes for calculating the available bandwidth for AF classes: we can either have strict priority like AF weights and omit them in the calculation, or we can take the normal AF weights into account when calculating the available bandwidths. If we want to protect our nonadmission controlled traffic (also, in shorter time scale), the latter mode is preferable.

A. Related Work

Wang *et al.* [17] present an adaptive scheduling scheme to support premium service, i.e., EF in the DiffServ architecture. The scheme is designed for weighted packet scheduling, e.g., weighted round-robin (WRR). The core idea is to tune the scheduling weights of different traffic classes adaptively, according to the dynamics of the average queue sizes. The goal is naturally to achieve low loss, delay, and jitter for the premium service. The authors claim that neither strict admission control nor accurate traffic conditioning are needed. We disagree with this claim—in our opinion, admission control becomes necessary immediately when the connection arrival intensity is high enough. Moreover, we feel that priority queue, equipped with a rate limiter, is a better way to implement EF.

Kawahara and Komatsu [18] introduce a scheme, called dynamically weighted queueing, for allocating bandwidth fairly in the DiffServ architecture. The proposed method estimates the number of active users in each class by simple traffic measurements. This estimate is then used in tuning the weights assigned to different class queues. Shimonishi *et al.* [19] propose a similar technique, where the sum of committed information rates (CIRs) of active flows in each class is estimated, and the link bandwidth is allocated according to the sum of CIRs. CIR-proportional allocation is combined with equal-share allocation in order to guarantee some resources for the best effort class connections with zero CIRs.

None of the aforementioned schemes, however, involves admission control. At least, these schemes do not couple admission control and adaptive scheduling weights.

B. Motivation for AF Weight Tuning

In case we have two AF classes only, there is no need to tune the scheduling weights due to the fact that the other one, AF2, is our BE. Thus, fixed weight allocations should be enough. With a more complex instance of flexible CAC, however, we might want to tune the AF1 and AF2 weights. If we give our "best

```
Bandwidth Broker:
for each measurement report arrival:
  for each link mentioned in the message:
    update = false
    for each AF class under CAC:
      if (unoccBw_AFi < minUnoccBw_AFi)
        minUnoccBw_AFi = unoccBw_AFi
        weight_AFi := load_AFi/(1 - load_EF - unocc)
      if (unoccBw_AFi < 0)
        update = true
    if update
      for each AF class under CAC:
        weight_AFi := weight_AFi/sum(weight_AFj, j=1...N) * (1 - weight_BE)
        minUnoccBw_AFi = bw
        enforce the minimum and maximum AF weights

timer expires:
  for each link:
    update = false
    for each AF class under CAC:
      if ((minUnoccBw_AFi/bw < lowThreshold) OR (minUnoccBw_AFi/bw > highThreshold))
        update = true
      minUnoccBw_AFi = bw
    if update
      for each AF class under CAC:
        weight_AFi := weight_AFi/sum(weight_AFj, j=1...N) * (1 - weight_BE)
        enforce the minimum and maximum AF weights
  set timer to expire after T_w seconds
```

Fig. 5. AF scheduling weight tuning algorithm.

effort" class, AF3, a fair share of forwarding resources, say 10%, it is no longer possible to have strict priority like weights (e.g., 90:9:1) for the three AF classes. Moreover, static normal AF weights could result in low bottleneck link utilization.

C. AF Weight Tuning Algorithm

The AF weights are tuned individually for each link. The tuning process receives periodic input about the unoccupied AF bandwidths for every link within the bandwidth broker area. If certain thresholds are reached, new AF scheduling weights for the involved links and the CAC algorithm are calculated and taken into use. Fig. 5 illustrates how the proposed algorithm can be implemented.

The bandwidth broker monitors continuously the $unoccBw_{AFi}$ values. The minimum values from each link are stored into link database. After each periodical check, every T_W s, these values are reset. If certain thresholds were reached, new AF weights are applied for the involved links. If the $minUnoccBw_{AFi}/bw$ value is smaller than *lowThreshold* or larger than *highThreshold*, we shall update $weight_{AFi}$ for the link at issue. After each measurement report arrival, it is checked whether $unoccBw_{AFi}$ is smaller than $minUnoccBw_{AFi}$. If that is the case, a new $weight_{AFi}$ is computed and stored

$$weight_{AFi} = \frac{load_{AFi}}{(1 - load_{EF} - unocc)} \tag{10}$$

where $unocc$ denotes the amount of unoccupied capacity that we would like to be always available. In general, *lowThreshold* should be less than $unocc$, which should be less than *highThreshold*. A negative $unoccBw_{AFi}$ value will immediately (after measurement report arrival) trigger AF weight tuning. Naturally, the final AF weights depend on the number of AF classes (N), excluding the "best effort" class

$$weight_{AFi} := \frac{weight_{AFi}}{\left(\sum_{i=1}^{N} weight_{AFj}\right)} * (1 - weight_{BE}). \tag{11}$$

IV. Adaptive EF and RT Reservation Limit Tuning

Since we are not aware of any related work combining parameter-based and MBAC, there are hardly any existing research results on tuning the reservation limits based on link measurements.

A. Motivation for Reservation Limit Tuning

One weakness of our admission control framework is that there is no protection against a sudden burst of connection arrivals. Of course, we could solve this problem with strict reservation limits. However, that would lead to low bottleneck link utilization [11]. A better solution could be the use of adaptive reservation limits for real-time traffic. The goal of EF and RT reservation limit tuning is to achieve more stable link utilization, even in the presence of bursty connection arrivals and, thus, higher admission ratios.

B. Reservation Limit Tuning Algorithm

The EF and RT reservation limits are tuned individually for each link. The tuning process receives periodic input about the EF and RT loads for every link within the bandwidth broker area. If certain thresholds are reached, new reservation limits are calculated and taken into use.

The bandwidth broker monitors periodically the $load_{EF}$ and $load_{RT}$ values of each link. If $load_{class}$ does not fall into the desired interval, we shall update $resvLim_{class}$ for the link at

```
Bandwidth Broker:
timer expires:
  for each link:
    if (loadEF < (loadLimEF - increment))
      resvLimEF = resvEF + increment
    if (loadEF > (loadLimEF + increment))
      resvLimEF = resvEF - increment
    if (loadRT < (loadLimRT - increment))
      resvLimRT = resvRT + increment
    if (loadRT > (loadLimRT + increment))
      resvLimRT = resvRT - increment
  set timer to expire after TR seconds
```

Fig. 6. EF and RT reservation limit tuning algorithm.

issue. Here, *increment* denotes the amount of capacity that we can increment to or decrement from the reservation limit. If the reservation limit is too low compared with the actual link usage, we will increase the reservation limit. Similarly, if the reservation limit is too high compared with the actual link usage, we will decrease the reservation limit. It should be noted that we are by no means disabling the measurement-based part of our admission control scheme—connections can be blocked because of exceeded load threshold already before it would be time to adjust the reservation limit. Fig. 6 illustrates how the proposed algorithm can be implemented.

V. Performance Evaluation

We use a modified version of the ns-2 simulator [20]. Six simulations with different seed values are run in each simulated case (95% confidence intervals are used). Simulation time is always 1200 s of which the first 600 s are discarded as warming period.

A. Simulation Cases and Network Topology

All cases are simulated with eight connection arrival intensities. We use a flexible CAC instance with three classes: EF, AF1, and AF2 (EF and AF1 belong to RT superclass). Admission control parameters are listed in Table I, while the simulation topology is illustrated in Fig. 7.

B. Network Equipment

All routers implement the standard PHBs; EF is realized as a priority queue and AF with a DRR [1] system consisting of three queues. The EF queue is equipped with a token bucket rate limiter having a rate of $0.8*$ link bandwidth and a bucket size 4500 bytes. Default, strict priority like, quanta for AF1, AF2, and AF3 queues are the following: 1800, 180, and 20 bytes. Queue sizes ($queueSize_{class}$) are also given in bytes: 5000 for EF, 15 000 for AF1, 20 000 for AF2, and 25 000 for AF3. Weighted RED (WRED) [2] is applied for all AF queues—with the following parameters: $minThresh_{class\_DP1} = maxThresh_{class\_DP1} = 1.0 * queueSize_{class}$,$minThresh_{class\_DP2} = maxThresh_{class\_DP2} = 0.883 * queueSize_{class}$, $minThresh_{class\_DP3} = maxThresh_{class\_DP3} = 0.767 * queueSize_{class}$, $maxDropPr = 1.0$, and $AQS\ Weight = 1.0$. These parameters will result in simplified WRED without queue size averaging.

TABLE I
Admission Control Parameters

Parameters	SP like AF weights	Normal AF weights	Adaptive AF weights
$weight_{AF1}$	0.9	0.45	adaptive
$weight_{AF2}$	0.09	0.45	adaptive
$weight_{AF3/BE}$	0.01	0.1	0.1
T_W	N/A		10.0 s
lowThreshold	N/A		0.05
highThreshold	N/A		0.15
unocc	N/A		0.1
T_R	N/A or 10.0 s (w. EF/RT res. limit tuning)		
increment	N/A or 0.05 (w. EF/RT res. limit tuning)		
$resvLim_{EF}$	10.0 or adaptive (w. EF/RT res. limit tuning)		
$resvLim_{RT\_MAX}$	10.0 or adaptive (w. EF/RT res. limit tuning)		
$resvLim_{AF1}$	10.0		
$resvLim_{AF2}$	10.0		
$resvLim_{NRT\_MAX}$	10.0		
$resvLim_{total}$	10.0		
$loadLim_{EF}$	0.5		
$loadLim_{AF1}$	0.5		
$loadLim_{AF2}$	0.9		
$loadLim_{RT\_MAX}$	0.9		
$loadLim_{NRT\_MAX}$	0.9		
$loadLim_{total}$	0.9		
$f(price)_{all}$	1.0		
s	500 ms		
p	1.0 s		
w	0.5		

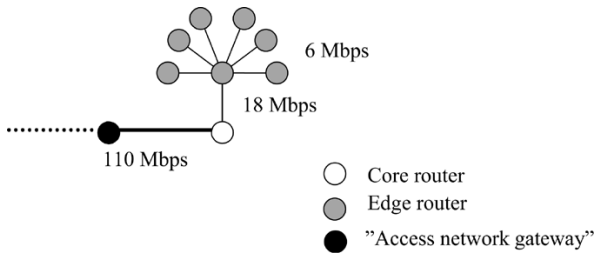

Fig. 7. Example access network topology.

C. Traffic Characteristics

Connections are set up between the access network gateway and edge routers. Bursty connection arrivals are created with a two-state Markov chain, where the transition probabilities from normal state to burst state and *vice versa* are both 0.1. In the normal state, new connections arrive at each edge router with exponentially distributed interarrival times. However, in the burst state, the interarrival time is always zero. Holding times are exponentially distributed with a mean of 100 s for RT (EF and AF1) connections and 250 s for other connections. Our traffic mix consists of VoIP calls, videotelephony, video streaming [21], web browsing [22], and e-mail downloading [23]. There are three different service levels within each AF class—their selection is based on subscription information. However, they do not have any effect on admission decisions. Signaling traffic between the bandwidth broker and the CAC agents is modeled in semi-realistic fashion. CAC agents do send real-router load reports to bandwidth broker but resource requests and replies are modeled in a statistical fashion. Bandwidth broker is located at the gateway that connects the access network to service provider's core network. Service mapping

TABLE II
TRAFFIC MIX AND SERVICE MAPPING

Service	Service level	PHB	Share of offered connections	Requested bandwidth (peak rate)
VoIP calls	N/A	EF	20.0%	36 kbps
Videotelephony	N/A	EF	20.0%	84 kbps
Video streaming	Gold	AF11	4.0%	250 kbps
	Silver	AF12	4.0%	250 kbps
	Bronze	AF13	4.0%	250 kbps
Guaranteed browsing	Gold	AF21	8.0%	250 kbps
	Silver	AF22	8.0%	250 kbps
	Bronze	AF23	8.0%	250 kbps
Normal browsing and e-mail downloading	Gold	AF31	8.0%	N/A
	Silver	AF32	8.0%	N/A
	Bronze	AF33	8.0%	N/A

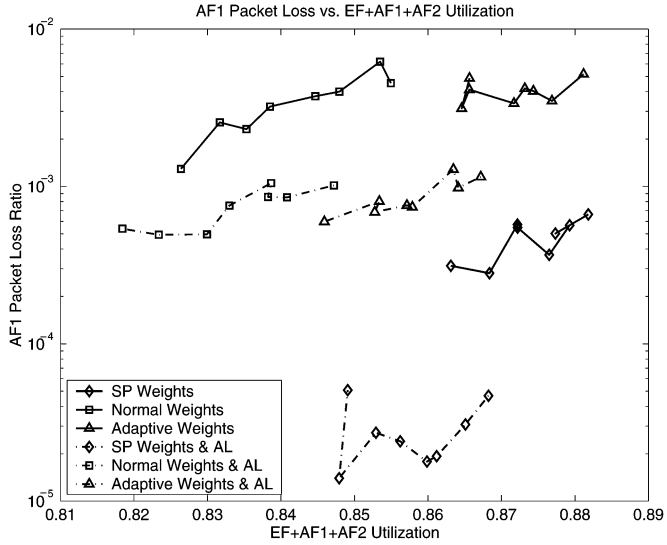

Fig. 8. AF1 loss versus $\mathrm{EF} + \mathrm{AF1} + \mathrm{AF2}$ load.

is done according to Table II. Simple token bucket policers are used to limit the sending rates of admitted sources.

D. Simulation Results

The loss-load graphs of Figs. 8–10 illustrate the main results of our simulations. Bottleneck link utilization is maximized either with adaptive or strict priority like AF weights. Without adaptive reservation limits maximum[1] AF1 packet loss can be prohibitive (even as high as 8% in the case of normal AF weights and the highest connection arrival intensity) due to bursty connection arrivals. AF packet loss is minimized when reservation limit tuning is used together with strict priority like AF weights. With normal AF weights, AF packet loss is somewhat higher (maximum AF1 packet loss is less than 2% with the highest connection arrival intensity). Moreover, AF packet loss is decreased also in the case where AF weights are tuned in conjunction with the reservation limits. This is a nice result, since it proves (although informally) that the two tuning processes are not disturbing each other.

Fig. 11 illustrates how the admission-controlled load develops as a function of connection arrival intensity. The dynamic weights for AF1 and AF2, as well as reservation limits for EF and RT are illustrated Figs. 12 and 13, correspondingly. The purpose of these two graphs is just to illustrate how AF weights and reservation limits are tuned.

[1]Maximum AF1 packet losses (sample length is 20 s) are not illustrated here.

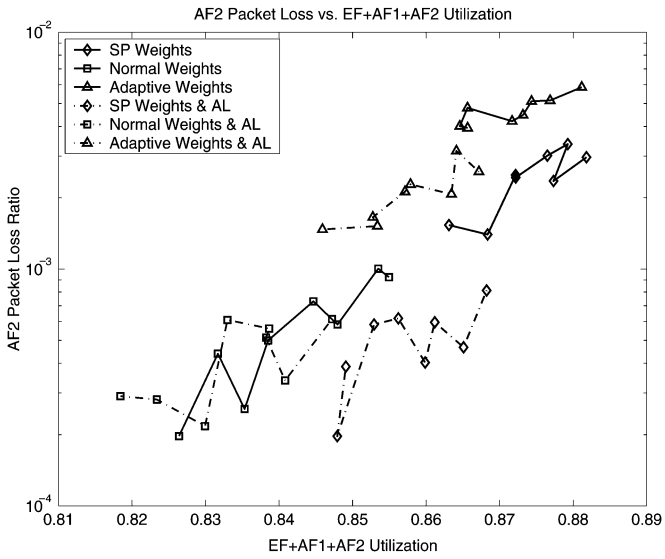

Fig. 9. AF2 loss versus $\mathrm{EF} + \mathrm{AF1} + \mathrm{AF2}$ load.

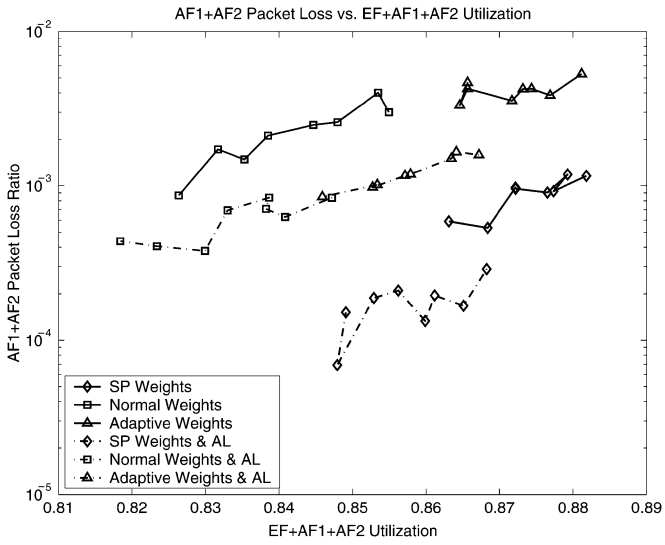

Fig. 10. $\mathrm{AF1} + \mathrm{AF2}$ loss versus $\mathrm{EF} + \mathrm{AF1} + \mathrm{AF2}$ load.

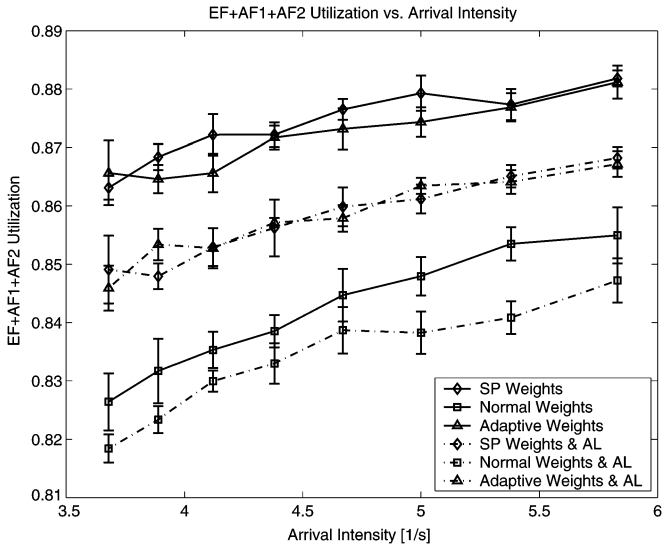

Fig. 11. $\mathrm{EF} + \mathrm{AF1} + \mathrm{AF2}$ load versus connection arrival intensity.

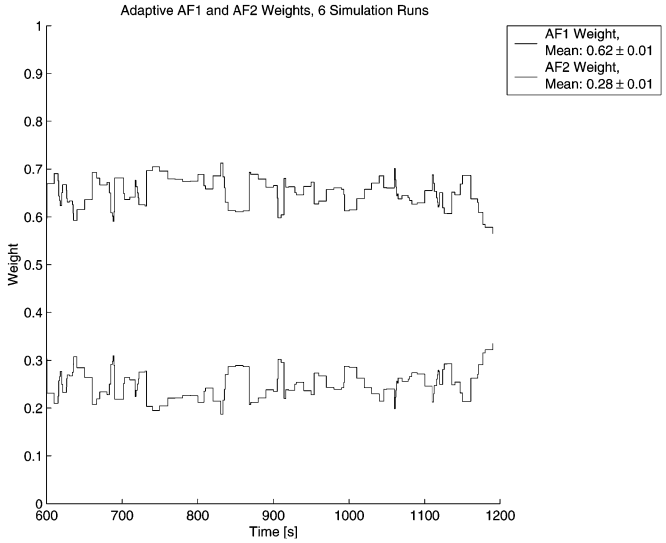

Fig. 12. Adaptive AF1 and AF2 weights.

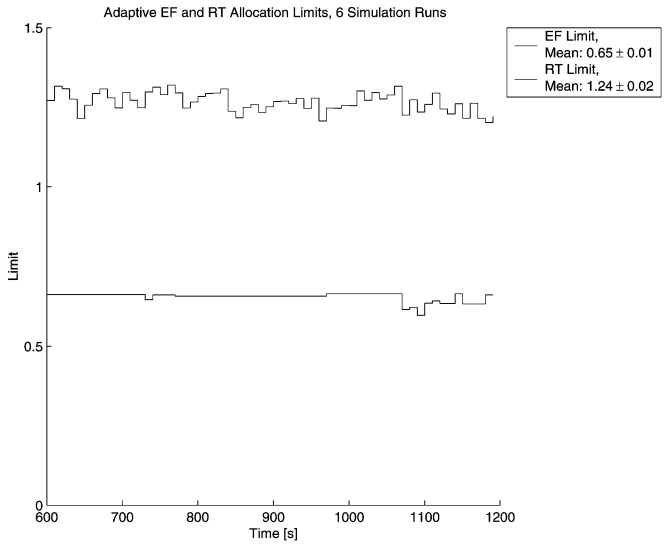

Fig. 13. Adaptive EF and RT allocation limits.

VI. Conclusion

There is a need for nonstrict priority like AF weights—the main motivation arises from the desire to protect nonadmission controlled traffic. Thus, AF weights should be taken into account in admission decisions. Simulations show that static, nonstrict priority like AF weights result in lower bottleneck link utilization than adaptive AF weights. Naturally, the experienced poor performance with static AF weights means that the weights were inappropriate considering the traffic mix. However, the ideal scheduling weights cannot be known beforehand.

Adaptive reservation limits are an effective way to protect oneself against bursty connection arrivals and still maintain high bottleneck link utilization. The tuning of EF and RT reservation limits seems to lower the bottleneck utilization a little. However, that is the price we have to pay for our "safety margins."

Signaling overhead is very low, both in the basic framework and in the adaptive enhancements. Thus, there is no real tradeoff between the bottleneck link utilization level and signaling traffic.

Comparison to related work is not really possible due to lack of similar proposals. There are a lot of proposals that involve adaptive scheduling weights—none of them, however, couples scheduling and admission control like we do.

Acknowledgment

The authors would like to thank K. Kilkki, P. Karlstedt, and the anonymous reviewers for their invaluable comments.

References

[1] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, pp. 375–385, Jun. 1996.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 1, pp. 397–413, Aug. 1993.

[3] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," *Request for Comments 1633*, Jun. 1994.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Dec. 1998.

[5] B. Davie, A. Charny, J. C. R. Bennett, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB," RFC 3246, Mar. 2002.

[6] J. Heinänen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," RFC 2597, Jun. 1999.

[7] K. Kilkki, *Differentiated Services for the Internet*. Indianapolis, IN: Macmillan, 1999.

[8] L. Breslau, S. Jamin, and S. Shenker, "Comments on the performance of measurement-based admission control algorithms," in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, pp. 1233–1242.

[9] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," RFC 2638, Jul. 1999.

[10] J. Lakkakorpi, "Simple measurement-based admission control for DiffServ access networks," in *Proc. SPIE ITCom 2002, Internet Perform. Control Netw. Syst. III*, Boston, MA, Jul. 2002, pp. 108–119.

[11] ——, "Flexible admission control for DiffServ access networks," in *Proc. SPIE ITCom 2003, Perform. Control Next Generation Commun. Netw.*, Orlando, FL, Sep. 2003, pp. 41–52.

[12] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, V. Narayan, and F. Reichmeyer, "Internet2 QBone: Building a testbed for differentiated services," *IEEE Netw.*, vol. 13, no. 5, pp. 8–16, Sep.–Oct. 1999.

[13] O. Schelén, "Quality of service agents in the Internet," Ph.D. dissertation, Luleå University of Technology, Luleå, Sweden, 1998.

[14] J. T. Moy, *OSPF: Anatomy of an Internet Routing Protocol*. Reading, MA: Addison-Wesley, 1998.

[15] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," RFC 1157, May 1990.

[16] K. Kilkki and J. Ruutu, "Selection of QoS mechanisms based on operator's business objective," in *Proc. 16th Nordic Teletraffic Seminar*, Espoo, Finland, Aug. 2002, pp. 313–324.

[17] H. Wang, C. Shen, and K. G. Shin, "Adaptive-weighted packet scheduling for premium service," in *Proc. IEEE ICC*, Helsinki, Finland, Jun. 2001, pp. 1846–1850.

[18] R. Kawahara and N. Komatsu, "Dynamically weighted queueing for fair bandwidth allocation and its performance analysis," in *Proc. IEEE ICC 2002*, New York, Apr.–May 2002, pp. 2379–2383.

[19] H. Shimonishi, I. Maki, T. Murase, and M. Murata, "Dynamic fair bandwidth allocation for DiffServ classes," in *Proc. IEEE ICC*, New York, Apr.—May 2002, pp. 2348–2352.

[20] UCB/LBNL/VINT. (2003) Network Simulator—ns (Version 2). [Online]. Available: http://www.isi.edu/nsnam/ns/index.html

[21] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. Robbins, "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Commun.*, vol. 36, no. 7, pp. 834–844, Jul. 1988.

[22] M. Molina, P. Castelli, and G. Foddis, "Web traffic modeling exploiting TCP connections' temporal clustering through HTML-REDUCE," *IEEE Netw.*, vol. 14, no. 3, pp. 46–55, May-Jun. 2000.

[23] V. Bolotin, "Characterizing data connection and messages by mixtures of distributions on logarithmic scale," in *Proc. 16th Int. Teletraffic Congr.*, Edinburgh, U.K., Jun. 1999, pp. 887–894.

Jani Lakkakorpi received the M.Sc. and Lic.Sc. degrees in electrical engineering (majoring in teletraffic theory) from the Helsinki University of Technology, Helsinki, Finland, in 1999 and 2004, respectively. He is currently working towards the Ph.D. degree at the Helsinki University of Technology.

In January 2000, he joined the Nokia Research Center, Nokia, Helsinki, Finland, where he currently works as a Research Engineer. His research interests include quality-of-service (especially, differentiated services), admission control, TCP optimization, and traffic engineering.

Ove Strandberg received the M.Sc. degree in electrical engineering from the Helsinki University of Technology, Helsinki, Finland, in 1992.

He has been working for the Nokia Research Center for over 15 years, working in areas of transmission technology to IP technology. His current interests are radio access features and, especially, QoS issues in IP networks.

Jukka Salonen received the M.Sc. degree in electrical engineering from the Tampere University of Technology, Tampere, Finland, in 1996.

He has been working for Nokia Networks since 2001.