Admission of QoS Aware Users in a Smart Network

EROL GELENBE and GEORGIA SAKELLARI
Imperial College London
and
MAURIZIO D'ARIENZO
Seconda Università degli studi di Napoli

Smart networks have grown out of the need for stable, reliable, and predictable networks that will guarantee packet delivery under Quality of Service (QoS) constraints. In this article we present a measurement-based admission control algorithm that helps control traffic congestion and guarantee QoS throughout the lifetime of a connection. When a new user requests to enter the network, probe packets are sent from the source to the destination to estimate the impact that the new connection will have on the QoS of both the new and the existing users. The algorithm uses a novel algebra of QoS metrics, inspired by Warshall's algorithm, to look for a path with acceptable QoS values to accommodate the new flow. We describe the underlying mathematical principles and present experimental results obtained by evaluating the method in a large laboratory test-bed operating the Cognitive Packet Network (CPN) protocol.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms: Algorithms, Measurement, Performance

Additional Key Words and Phrases: Measurement-based admission control, quality of service, self-aware, cognitive packet network

ACM Reference Format:

Gelenbe, E., Sakellari, G., and D'Arienzo, M. 2008. Admission of QoS aware users in a smart network. ACM Trans. Autonom. Adapt. Syst. 3, 1, Article 4 (March 2008), 28 pages. DOI 10.1145/1342171.1342175 http://doi.acm.org/ 10.1145/1342171.13421745

This article is an extended version of the paper "Controlling Access to Preserve QoS in a Self-Aware Network" that appeared in the IEEE 1st International Conference on Self-Adaptive and Self-Organizing Systems (SASO'07). © IEEE 2007.
Research supported by EPSRC (UK) under grant number GR/552360/01 and by Project HYPERION of the UK MoD Defence Technology Centre on Data and Information Fusion.
Author's address: E. Gelenbe, Intelligent Systems and Networks Group, Electrical and Electronic Engineering Department, Imperial College, London SW7 2BT; email: e.gelenbe@imperial.ac.uk; q.sakellari@imperial.ac.uk; maudarie@unina.it.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2008 ACM 1556-4665/2008/03-ART4 $5.00 DOI 10.1145/1342171.1342175 http://doi.acm.org/10.1145/1342171.1342175

1. INTRODUCTION

A need for stable, reliable and predictable networks that provide Quality-of-Service (QoS) to their users, has emerged from the growth in multimedia traffic and the use of real-time applications, such as Internet telephony (voice-over-IP), video on demand, remote teaching, remote medical diagnosis and treatment, and online trading systems. The current best effort Internet architecture does not secure that the real-time applications will receive the QoS required in order to function properly. New architectures and smarter networks have appeared that try to solve this problem. By smart network we mean a self-aware network (SAN) that probes itself through special packets, collects data at different distributed points, and adaptively takes corrective action based on the QoS objective that it is pursuing. The SANs use adaptive packet routing protocols, such as Cognitive Packet Network (CPN) [Gelenbe et al. 1999], which is designed to perform self-improvement by using random neural networks (RNN) with reinforcement learning (RL), and genetic algorithms.

In all types of networks, with SAN not constituting an exception, congestion is an issue to be carefully examined. A way to control traffic congestion and guarantee QoS throughout the lifetime of a connection is admission control (AC). A proposal for such an AC, which will improve the performance of a smart network is the purpose of the work presented in this article. Our mechanism decides whether a new connection should be accepted in the network by estimating both the resources it will need and the impact it will have on the ongoing connections.

The remainder of this article is organized as follows: the next section reviews the main AC proposals; Section 3 describes the characteristics and the mechanisms of self-aware networks; Section 4 presents our proposed multiple criterion CAC algorithm which consists of three stages, the identification, the probing, and the decision stage; Section 5 reports experimental results of the AC algorithm achieved on a real CPN testbed, and finally we conclude the article in Section 6. An earlier version of this article, excluding Sections 4.1, 5.1, 5.3 and part of Section 6, was referenced in a conference [Gelenbe et al. 2007].

2. AC ALGORITHMS—TYPES AND HISTORY

Admission control algorithms can be classified according to whether the traffic parameters are specified a priori (parameter-based) or whether the admission decisions rely on measurements of the actual traffic load (measurement-based). The parameter-based AC algorithms can be analyzed by formal methods, while the measurement-based ones can only be analyzed through experimentation on real networks, or with simulation, or emulation.

2.1 Parameter-Based AC

These algorithms compute the amount of network resources required to support new flow by given a priori flow characteristics. They can be further classified as nonstatistical and statistical allocation algorithms. The nonstatistical allocation or deterministic (also called peak bandwidth) allocation is the simplest form among all admission control algorithms. The algorithm ensures that the sum of requested resources and the existing connections is bounded by the physical link capacity. The most significant disadvantage of this type is that it assumes that connections transmit at their peak rates all the time, thus it allocates more bandwidth than is required to provide QoS guarantees for the existing connections, and the network resources are underutilized. An other drawback is that there is no multiplexing gain among the sessions admitted into the network, since it works on a flow by flow basis and does not consider sharing bandwidth resources among connections. Statistical allocation is a group of much more complicated admission control algorithms. The bandwidth for a connection is assigned at less than the peak bandwidth of the connection, depending on the statistical distribution of the arriving cells [Perros and Elsayed 1996]. Statistical allocation results in statistical multiplexing gain, when dealing with sources that arrive in bursts, since it assumes sharing bandwidth resources with other connections; thus the sum of all peak rates may be greater than the capacity of the output link. This type is difficult to implement effectively because of the uncertainty in the distribution of the incoming traffic and the inaccurate and difficult-to-calculate statistical information of the traffic arrival process.

Following are the most well-known parameter based algorithms:

Rate or Simple Sum is the simplest parameter-based algorithm, and the most widely implemented by switch and router vendors [Jamin et al. 1997b]. It ensures that the sum of requested resources does not exceed the link capacity. Therefore, a new call is admitted if: $V + r^{new} < \mu$, where V is the total amount of the existing reserved rates, μ the link bandwidth, and r^{new} is the rate demanded by the call requesting admission. The Simple Sum algorithm does not take into account QoS metrics other than bandwidth, and it assumes that every user will use all of its reserved bandwidth.

Acceptance Region schemes decide whether to admit a new flow according to the current state of the system and whether the state lies within the acceptance or rejection region. The acceptance region is calculated in order to maximize the line utilization for a nominal packet loss, given a set of flows with a given declaration of peak and mean rates. The calculation in both Gibbens et al. [1995] and Tse and Grossglauser [1997] assumes that calls arrive according to a Poisson process, the calls admitted are independent and stay in the network for exponentially distributed time, all calls have identical bandwidth requirement statistics, and they are described by a Markov fluid process. The model in Gibbens et al. [1995] is an attempt to deal with high offered load situations by not considering for acceptance any call, every time an arriving call is rejected, until after a call currently in progress has ended.

Though the acceptance region algorithms are quite simple, this simplicity comes as a result of simplifications of the network model, which result in limitations in such algorithms. For instance, they perform quite poorly at low link capacity. Another limitation is related to the often made assumption of homogeneous on-off sources. Thus it may not be clear whether such algorithms are still applicable when the traffic sources do not fit this model.

Equivalent Bandwidth (also known as equivalent capacity or efficient bandwidth) is the minimum bandwidth that is needed to carry the traffic that is generated by a source in isolation without violating the QoS requirements. In this approach, each source is assigned an equivalent bandwidth and a new call is accepted if the sum of these equivalent bandwidths is less than the capacity of the links.

The schemes of Guerin et al. [1991] and Guerin and Gun [1992] use a fluid-flow model for the source and a bandwidth allocation process to calculate the equivalent bandwidth by taking into account the impact of source characteristics (the duration of the burst period) either when the impact of individual connection characteristics is critical or when the effect of statistical multiplexing is of significance.

Floyd [1996] proposes an admission control algorithm, based on the Guerin et al. [1991] scheme, which estimates the equivalent capacity of a class using the Hoeffding Bound, a looser bound of the sum of N independent, random variables. The Hoeffding Bound does not assume normal distribution of the aggregate traffic, so this model does not assume a normal approximation of the arrival rate distribution, and thus is preferable for classes with either a moderate number of admitted connections or for traffic from heterogeneous sources with a wide range in the peak rates.

Equivalent bandwidth schemes are characterized by their simplicity, since determining whether a given set of traffic sources can be accommodated without any QoS violation comes down to comparing the sum of the equivalent bandwidths of individual sources to the link capacity. This approach does not consider the effect of buffering, which increases the effective capacity of a system. Of course, since Equivalent Capacity AC algorithms are parameter-based, they reserve the resources that are specified by the source traffic description, which could lead to low network utilization, since users could request more resources than they require.

The diffusion based statistical AC uses statistical bandwidth based on closed-form expressions that use diffusion approximation models. It assumes that traffic follows an on-off behavior and exploits information about buffer sizes and the multiplexed traffic that shares a common link, to obtain a diffusion approximation based on an estimate of cell loss.

The scheme in Gelenbe et al. [1996] uses two diffusion models: one for a finite buffer (FB) ATM multiplexer, and another for an infinite buffer (IB) ATM multiplexer. The cell loss probability is estimated by the overflow probability, which is the overall probability of exceeding the actual buffer capacity, B.

A new connection is admitted if the statistical bandwidth on every intermediate link along the selected path is less than the link capacity.

The use of diffusion-based techniques has been shown to be conservative with respect to cell loss, but more economical in bandwidth allocation, and has the disadvantages of the parameter-based algorithms.

2.2 Measurement-Based AC (MBAC)

This approach relies on measurements of actual traffic load in making admission decisions. It uses these network measurements to estimate the current load of existing traffic, instead of computing the traffic characteristics out of the user-specified connection's parameters. The measurement-based schemes alleviate the burden on the users to accurately specify the parameters for their traffic flow and provide a mechanism that adapts according to the network conditions. Thus measurement-based AC is a more practical approach for achieving statistical multiplexing gain with variable-rate traffic.

Following are the most well-known MBAC mechanisms that have been proposed for conventional networks:

Measured Sum [Jamin et al. 1997b] is the measurement-based version of the Simple Sum algorithm. It tries to increase the network utilization by measuring the actual network load and substituting the reserved rates of the existing users for the measured load. A new call is admitted if: $\hat{V} + r^{new} < \upsilon\mu$, where $\hat{V}$ is the measured load of the existing traffic, μ the link bandwidth, and υ a user-defined utilization target. Again, the admission decision is based only on the availability of bandwidth and does not consider any other QoS metric.

Measurement Based Admission Control with Delay and Bandwidth Constraint schemes do both delay and bandwidth checking and are used with predictive service for tolerant applications that allow a certain degree of QoS violations. When a new flow requests service, the network must characterize its traffic. The algorithm described in Jamin et al. [1997a], is designed for predictive service that approximates the maximum delay of predictive flows by replacing the worst-case parameters in the analytical models with measured quantities. The computation of worst-case queueing delay is different for guaranteed and predictive services. Such algorithms cannot be used in networks with strict QoS requirements and are only applicable in networks with predictive service. Also the schemes tend to exceed the needed bandwidth reservation, since they use worst-case delays, which is rarely the case since multiple sources will rarely simultaneously transmit packets at peak rate.

Endpoint Admission Control [Breslau et al. 2000; Bianchi et al. 2000; Elek et al. 2000; Ganesh et al. 2005; Gibbens and Kelly 1999; Cetinkaya and Knightly 2000] is a measurement-based scheme in which the end host (endpoint) probes the network by sending probe packets at the data rate it would like to reserve, and records the resulting level of packet losses, specially marked packets, or other QoS criteria. The host then admits the flow only if the loss or marking percentage is below some threshold value. Endpoint admission control requires no explicit support from the routers, which do not need to keep a per-flow state or process reservation requests. This approach simply uses the fact that routers may drop or mark packets in a normal manner. In some cases the probe packets are treated equally with the data packets and in others they are sent at a different priority level.

In Elek et al. [2000], the probe packets are sent in a separate (lower) priority class and use packet drops to indicate congestion. The scheme assumes that the traffic consists of a number of identical on-off sources and N is the number of the established sessions with mean rate m and peak rate p. The loss probability of the controlled-load service (CLS) traffic is estimated by the fluid-flow approximation. When a new user requests entrance to the network, a probe is sent from the source to the destination. The destination counts the received packets until the probe time period expires and sends a measurement report, under high priority, to the source, with the number of probe packets received. Based on that report, if the calculated probe loss probability is less than a threshold, the source decides to accept the new call. If the probe does not succeed in reaching the destination due to temporal network overload or opposing probe processes, the session establishment fails and the source must wait for a random time (back-off time) before issuing a new probe. The back-off time is calculated from a uniform distribution of some width, which is doubled for each consecutively blocked attempt to reach the same receiver.

End-point admission schemes suffer from the shortcomings of any measurement-based scheme, where estimates may not be in line with what will be observed when the real traffic is sent instead of the probe traffic.

Measurement-based AC algorithms are shown to achieve much higher utilization than parameter-based [Jamin et al. 1997a], and they are more adaptive to network changes. Of course, the more accurate and up-to-date the measurements are, the better the algorithm. Also since most of the real-time applications are quite adaptive to occasional service deterioration, MBAC schemes are more tolerant to the lack of precision that a measurement-based admission control scheme might have. It is therefore of no surprise that the last few years the research is focused on accurate network monitoring tools and, as far as admission control is concerned, it is turned towards measurement-based approaches.

3. SELF AWARE NETWORKS

Self Aware Networks (SAN) is a proposal of QoS-enabled networks with enhanced monitoring and self improvement capabilities that use adaptive packet routing protocols, such as Cognitive Packet Network (CPN) [Gelenbe et al. 1999, 2001, 2002, 2004] and address QoS by using adaptive techniques based on online measurements. CPN is a distributed protocol that provides QoS-driven routing, in which users, or the network itself, declare their QoS requirements (QoS Goals) such as minimum delay, maximum bandwidth, minimum cost, and so on. It is designed to perform self-improvement by learning from the experience of smart packets, using random neural networks (RNN) [Gelenbe 1993] with reinforcement learning (RL), and genetic algorithms.

CPN makes use of three types of packets:

—smart packets (SP) for discovery;
—source routed dumb packets (DP) to carry the payload;
—and acknowledgement (ACK) packets to bring back information that has been discovered by SPs, and is used in nodes to train neural networks.

SPs are generated either by a user request to create a path to some CPN node, or by a user request to discover parts of the network state, including location of certain fixed or mobile nodes, power levels at nodes, topology, paths, and their QoS metrics. To avoid overburdening the system with unsuccessful requests or packets that are in effect lost, all packets have a life-time constraint based on the number of nodes visited.

Each node in the CPN acts as a storage area for packets and mailboxes (MBs) and also stores and executes the code used to route smart packets. Therefore for each successive smart packet, each router executes the code, updates its parameters, and determines the appropriate outgoing link based on the outcome of this computation. RL is carried out using a QoS Goal, such as Packet Delay, Loss, Hop Count, Jitter, and so on. The decisional weights of an RNN are increased or decreased based on the observed success or failure of subsequent SPs to achieve the Goal. Thus RL will tend to prefer better routing schemes, more reliable access paths to data objects, and better QoS.

More analytically, when a Smart Packet arrives at its destination, an ACK is generated and heads back to the source of the request, following the reversed path of the SP. In each CPN node of the reversed path that the ACK packet visits, it updates the mailbox with the information it has discovered, and finally provides the source node with the successful path to the destination node. That route is used as a source route by subsequent DPs of the same QoS class having the same destination, until a newer and/or better route is brought back by another ACK. ACK messages also contain timestamp information gathered at each node back to the source, which, together with the one gathered by the smart packets on the same nodes, can be used to monitor the QoS metrics on a single link and/or partial or complete paths.

As far as the decision process is concerned, each node stores a specific RNN for each QoS class, and for each active source-destination pair. Each RNN node, which represents the decision to choose a given output link for a smart packet, has as many neurons as the possible outgoing links. Decisions are taken by selecting the output link j for which the corresponding neuron is the most excited: $q_i \leq q_j$ for all $i = 1, \ldots, n$ where n is the number of neurons (possible outgoing links). The state q_i, of the ith neuron in the network, represents the probability that the ith neuron is excited and therefore the probability that the ith outgoing link will be selected for the smart packet's routing. For $1 < i < n$ the state of the i-th neuron satisfies the following system of nonlinear equations:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)},$$

where

$$\lambda^+(i) = \sum_j q_j w_{ji}^+ + \Lambda_i \;\; and \;\; \lambda_-(i) = \sum_j q_j w_{ji}^+ + \lambda_i,$$

while w_{ji}^+ is the rate at which neuron j sends excitation spikes to neuron i when j is excited; w_{ji}^- is the rate at which neuron j sends inhibition spikes to neuron i when j is excited; and $r(i)$ is the total firing rate from the neuron i. For an n neuron network, the network parameters are those n by n weight matrices $W^+ = \{w^+(i, j)\}$ and $W^- = \{w^-(i, j)\}$ that need to be learned from input data.

As far as the learning process is concerned, CPN reinforcement learning changes neuron weights to reward or punish a neuron according to the level of goal satisfaction measured on the corresponding output. Each QoS class for each source-destination pair has a QoS goal G, which expresses a function to be minimized for example, Transit Delay, or Probability of Loss or Jitter, or a weighted combination and so on. The level of goal satisfaction is expressed by a reward. Given some goal G that a packet has to minimize, the reward R is formulated simply as $R = 1/G$.

The RNN weights are updated based on a threshold T:

$$T_k = \alpha T_{k-1} + (1 - \alpha) R_k,$$

where R_k, k = 1, 2, ... are successive measured values of reward R, and α is some constant ($0 < \alpha < 1$) that is used to tune the responsiveness of the algorithm: for instance $\alpha = 0.8$ means that on average, five past values of R are being taken into account. Neurons are rewarded or punished based on the difference between the current reward R_k and the last threshold T_{k-1}. So, if the most recent value of the reward, R_k, is larger than the previous value of the threshold T_{k-1}, then we very significantly increase the excitatory weights going into the neuron that was the previous winner (in order to reward it for its new success), and make a small increase of the inhibitory weights leading to other neurons. If the new reward is not greater than the previous threshold, we moderately increase all excitatory weights leading to all neurons, except for the previous winner, and significantly increase the inhibitory weights leading to the previous winning neuron, in order to punish it for not being very successful this time.

So, if r_i is the firing rate before the update takes place for every neuron i:

$$r_i = \sum_{1}^{n} [w^+(i, m) + w^-(i, m)],$$

we first compute T_{k-1} and then update the network weights as follows for all neurons $i \neq j$:

—If $T_{k-1} \leq R_k$
 —$w^+(i, j) \leftarrow w^+(i, j) + R_k$,
 —$w^-(i, j) \leftarrow w^-(i, j) + \frac{R_k}{n-2}$, if $k \neq j$.
—Else
 —$w^+(i, j) \leftarrow w^+(i, j) + \frac{R_k}{n-2}$, if $k \neq j$,
 —$w^-(i, j) \leftarrow w^-(i, j) + R_k$.

Since the relative size of the weights of the RNN, rather than the actual values, determine the state of the neural network, we renormalize all the weights by carrying out the following operations. First for each i we compute:

$$r_i^* = \sum_{1}^{n} [w^+(i, m) + w^-(i, m)],$$

and then renormalize the weights with:

$$w^+(i, j) \leftarrow w^+(i, j) * \frac{r_i}{r_i^*}$$

$$w^-(i, j) \leftarrow w^-(i, j) * \frac{r_i}{r_i^*}.$$

Finally, the probabilities, q_i, are computed using the nonlinear iterations described before. The largest of the q_is is again chosen to select the new output link used to send the smart packet forward. This procedure is repeated for each smart packet, for each QoS class, and each source-destination pair.

The SAN can specify its own overall criteria, and in a certain sense the admission control does exactly that, since users are only admitted if their needs can be met, so that the SAN has an overriding goal of satisfying users as best as it can. On the other hand, individual users can also specify their own criteria, and then the SAN monitors the users and the network resources so as to satisfy the users as well as possible.

4. A MULTIPLE CRITERION AUTOMATIC ALGORITHM FOR AC AND MONITORING

The measurement-based AC algorithm we propose, which was first introduced in Sakellari et al. [2006] and Gelenbe et al. [2007], and is extended and explained more analytically in this article, is based on measurements of the QoS metrics on each link of the network. This does not require any special mechanism since, as stated in the previous Section, the SAN already collects QoS information on all links and paths that the SPs have explored and on all paths that any user is using in the network. Furthermore, since different QoS metrics are specified for different users according to their needs, SAN can collect data for the different QoS metrics that are relevant to the users themselves.

The AC decision of our proposed scheme consists of three stages. In the first, the identification stage, the network identifies the quality criteria that a new user has and translates them to QoS metrics, if that user is in no position to specify them himself. In the second, the probing stage, the AC estimates the impact of the new flow by probing the network. Finally, in the third, the decision stage, the AC searches whether there is a feasible path that can accommodate the new call by considering the impact of that new flow on the network without affecting the quality of formerly accepted flows.

4.1 Identification Stage—Identification of the Users' QoS

Being able to specify the QoS metric a user needs is in some cases, extremely useful and desirable. For instance, in networks for battle space communication and information services, or in mass media networks, most of the users know exactly the bounds of QoS that they need in order to have a good service. But in everyday networks, there might be users who may not know what kind of service they need and do not know how their needs translate in terms of QoS values. In those cases the SAN can specify its own overall criteria so that it will satisfy users as best it can. This ability was further extended by our algorithm so that the SAN can also provide the users with the appropriate QoS values.

So, in the first stage of our self-adaptive algorithm, when a user requests to enter the network, and has not specified any QoS requests, the network estimates its needs, by looking at the user's identity (the type of the application, the type of user, or the purpose that the user wants to use the network for), and provides the necessary QoS values required to achieve the required functionality of the user's application based on minimal QoS needs that are well known (e.g. voice over IP, or real-time video streams). For the numerical values of the QoS metrics according to the medium used (data, audio, or video) and the specific user application, we resort to the values of the ITU-T International Telecommunication Union standardization, shown in the tables of ITU [2001], where the minimal QoS needs of delay, variance of delay, packet loss, and data rates, or data amounts, are specified in order for an application to work efficiently. So, for example, if a user wants to make an ATM transaction he/she will need less than a two-second one-way delay, at least 10KB bandwidth and no loss, while for a voice conversation over the network the delay must be less than 150ms, jitter less than 1ms, packet loss less than 3% and the bandwidth, if not defined otherwise, should be between 4–64Kbps.

Of course, this matter should be further investigated so that all parameters that affect a connection could be identified, and the restrictions of the quality metrics could be appointed to each user. For instance, QoS metrics such as security or monetary cost could be considered. For the time being we believe that these four metrics (delay, jitter, packet loss, and bandwidth) can provide good bounds that can guarantee service quality, especially in multimedia traffic networks.

In the case where not all users are equal, and the traffic in the network must be prioritized, each new user has to answer a series of questions that clarify a user's identity in order for them to be categorized in a priority category. For example apart from "what type of application" or "what medium will be used," questions like: "is the user an enterprise or individual," "is the user a regular user of the network," "what is the user's rank," "how long will the user need to use the network for," and so forth, could lead to categorizing the user according to the priority their traffic should have in the network. So when a new user is identified as the highest priority he/she should immediately be accepted into the network. Of course the QoS levels of all the existing users must be fulfilled. Therefore, the entrance of a high priority user could lead to the breach of the current QoS contracts between the network and the existing users. In order to deal with this issue, many algorithms reserve resources for high priority users, running the risk of having unused and wasted resources. An alternative solution could be that when we are in need of resources in order to serve a high priority user, one or some of the lowest priority users already in the network, could be rejected so that all other users will meet their QoS goals; but this is something that needs to be further investigated.

4.2 Probing Stage—Self-Observation of the Network and Estimation of the Impact of a New Flow

Let us consider the network graph $G(N, E)$ with nodes N, $n = |N|$, and the set E of directional links (i, j), where $i, j \in N$. The CPN algorithm explores $G(N, E)$ and collects QoS data about the parts of the network that are currently being used, or which have been explored by SPs. We assume that this data is available in one or more locations in the form of $n \times n$ *link QoS matrices* Q_v,

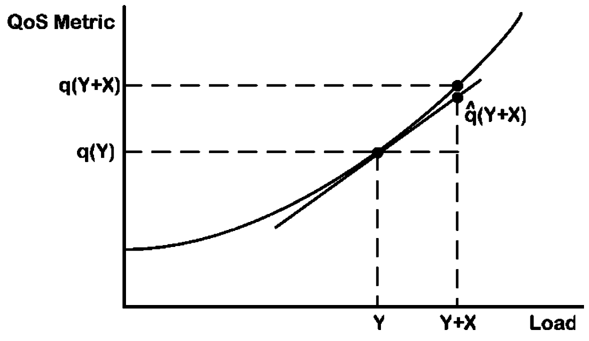

Fig. 1. QoS metric vs load.

for every distinct QoS metric v = 1, . . . m, where m is the total number of QoS metrics that may concern a user. The elements of Q_v are:

—$Q_v(i, j) = r$ where $r \geq 0$ is a real number representing the QoS of link (i, j) that has been measured at some recent enough time, and

—$Q_v(i, j) =$ *unknown* if i and j are not directly connected, or if either an SP has not explored the link for the QoS metric v, or if this happened so long ago that the value could be inaccurate.

Every time a new user requests to enter the network from a source s to a destination d, with total traffic rate X, and specific QoS constraints, probe traffic of traffic rate equal to a small percentage of X is send from s to d for a small time interval t. Then, new QoS data are collected and new QoS matrices, $\hat{q}'_w(i, j)$, are created for each QoS metric of interest, including the new users', and for all links (i, j). Some links may not be concerned with the probe traffic so for that link we take $\hat{q}'_w(i, j) = 0$. The path that the probe packets will follow, will be the one that the SPs have chosen as more appropriate in order to satisfy the QoS needs of the new flow. It is very likely to also be the path that will be followed after the new user's full traffic is inserted.

Finally an estimation of the link QoS values of all metrics is calculated and stored in the gathering point in the form of link QoS matrices:

$$\hat{Q}_w(i, j) = Q_w(i, j) + X\hat{q}'_w(i, j), \quad \text{for all concerned links,}$$

or

$$\hat{Q}_w(i, j) = Q_w(i, j), \quad \text{for unconcerned links.}$$

This estimation is based on the fact that every QoS metric can be considered as a value that increases as the traffic load increases. The addition of a new connection will increase the load of the paths it may be using, and therefore it is assumed that the value taken by the QoS metrics will increase. For example, delay increases as the network traffic load increases.

Let us consider some link (i, j). A small increase, x, in the load that is obtained in a controlled manner, for example, by sending probe packets at rate x, generates an estimate of the manner in which the QoS metric, q, varies around the current load point, Y:

$$\hat{q}' = \frac{q(Y + x) - q(Y)}{x}. \tag{1}$$

The impact of a new flow with total traffic rate X can then be evaluated by using the estimate and the measured derivative from (1):

$$\hat{q}(Y + X) = q(Y) + \hat{q}'X, \tag{2}$$

without having to know the initial load Y. Here we are using the simple Taylor rule for continuous and differentiable functions that are close to a point x. The function $f(x)$ can be approximated as $f(x + y) \sim f(x) + yf'(x)$. We recognize that this is an approximation, and an estimate of the error is: $error \sim y^2 f''(x)/2$. We have included this error estimate in some of the measurements that are reported.

This estimate may be optimistic or pessimistic. However it is likely that the path that CPN will select for the probe traffic, because it provides the most favorable impact on current flows, and because it satisfies the QoS needs of the new flow, is also likely to also be the best path in terms of actual observed QoS after the new user's full traffic is inserted.

A major advantage of this computation is that, contrary to the existing measurement-based AC schemes that use probing, it is not required to send the probe packets at the same rate as the new call's requested rate. We can have an accurate estimation by sending at much lower rates. This way the probing process has no significant impact on the network's congestion.

Obviously, the more probe packets we send, the more accurate the information that the source gathers will be. However, a large number of probe packets may contribute to congestion in addition to the congestion caused by data traffic. Also, the longer the probing procedure lasts, the more accurate the measurements will be, but then a user might be required to wait for unacceptably long time before the admission decision. The optimal values for probing rates and times are something that depends on the overhead due to probing, and should be further investigated.

4.3 Decision Stage

Let us assume that the users may be concerned with m distinct QoS metrics $q_v \in R, v = 1, \dots m$, that are specified in terms of QoS constraints [$q_v \in C_v(u)$ for each user u], where $C_v(u) \subset R$ is typically an interval of acceptable values of the QoS metric v for user u. We will detail the AC algorithm in terms of forwarding packets from some source s to a destination d. However this approach can be generalized to the case where u is requesting some service S.

From the link matrices Q_v of the previous stage we can compute:

—The set of known (explored) paths $P(s, d)$ from s to d, and

—The *path QoS matrices* K_v, where $K_v(s, d)$ is the known best value of the QoS metric v for any path going from s to d if such a path exists and if the links on the path have known entries in the link QoS matrices. Other entries in K_v are set to the value "unknown."

By "best value" we mean that several paths may exist for the source-destination pair (s, d), but $K_v(s, d)$ will store, for instance, the smallest known delay for all paths going from s to d if q_v is the delay metric. We will discuss later, how the path QoS matrices are computed from the link matrices.

4.4 The AC Algorithm

Let u be a new user requesting admission for a connection from source s to destination d, carrying a traffic rate X, and with QoS constraint $q_v(u)$. Suppose also that the network is currently carrying other users z, any one of whom will be generically represented by some QoS constraint $q_w(z)$. The proposed AC algorithm proceeds as follows:

—Find the set $P(s, d)$. If it is empty, send SPs to discover paths. If unsuccessful, reject the request. Otherwise monitor the current network, create the $Q_w(i, j)$ matrices for all discovered links, and all QoS metrics (including $w = v$), and then send probe traffic at rate x along the network.

—Use the probe traffic to obtain $\hat{q}'_w(i, j)$ for each QoS metric w of interest, including $w = v$, and for all links (i, j). Note that some links may not be concerned by the probe traffic so for those links we take $\hat{q}'_w(i, j) = 0$. The path that the probe packets will follow, will be the one that the SPs have chosen as more appropriate so that it satisfies the QoS needs of the new flow. It is therefore very likely to also be the path that will be followed after the new user's full traffic is inserted.

—Compute the estimation

$$\hat{Q}_w(i, j) = Q_w(i, j) + X\hat{q}'_w(i, j) \tag{3}$$

for all concerned links and all QoS metrics. For unconcerned links we take $\hat{Q}_w(i, j) = Q_w(i, j)$.

—Compute $\hat{K}_w$ from $\hat{Q}_w$ (to be detailed in the following) for all the QoS metrics of interest, including v.

—Finally, if $\hat{K}_v(s, d) \in C_v(u)$ AND $\hat{K}_w(s', d') \in C_w(z)$ for all other current users z with source-destination pair (s', d') and QoS metric $q_w \in C_w(z)$, then accept u; else reject the request.

4.5 Computing the QoS Matrices

The well known Warshall's algorithm [Warshall 1962] determines for each $i, j \in N$, whether there is a path from node i to node j by computing the Boolean matrix K, the transitive closure of the graph's adjacency matrix Q, in less than n^3 Boolean operations.

$$K = \bigcup_{k=1}^{n} Q^k \tag{4}$$

or

$$K^n[i, j] = K^{n-1}[i, j] \bigvee \left(K^{n-1}[i, n] \bigwedge K^{n-1}[n, j]\right), \tag{5}$$

where $K^1[i, j] = Q[i, j]$ and the matrix elements are treated as Boolean values with $\bigvee$ being the logical OR and $\bigwedge$ the logical AND.

Floyd's algorithm [Floyd 1962] extends Warshall's algorithm to obtain the cost of the "smallest cost path" between any pair of vertices in the form of a real-valued matrix.

$$K^n[i, j] = min\{K^{n-1}[i, j], (K^{n-1}[i, n] + K^{n-1}[n, j])\}. \tag{6}$$

Thus, connecting this to our algorithm, Floyd-Warshall's technique can be used to construct K_v from Q_v, and hence $\hat{K}_v$ from $\hat{Q}_v$, if the QoS metric q_v is additive, so that $K_v(i, j)$ is the smallest value of the QoS metric among all known paths from i to j. Delay and the variance of delay, are both additive metrics. Although loss rate is not additive (it is sub-additive in the sense that the path loss rate is smaller than the loss rate of individual links in the path), and the number of lost packets is an additive metric. Note that $K_v(i, j)$ are all non-negative quantities.

For non-additive metrics we have developed a generalization of the Floyd-Warshall, which is described next.

4.6 Generalization of the Floyd-Warshall Algorithm to Non-Additive QoS Metrics

Now consider the matrix Q_v mentioned previously, whose entries are the measured QoS values $r \geq 0$ over links (i, j) whenever such a link exists, or otherwise have the value "unknown."

The matrix K_v, which is calculated as shown in the following, provides us with the "best QoS value" for every path between every pair of vertices (i, j).

$$K_v = \bigoplus_{k=1}^{n} [Q_v]^k \tag{7}$$

or

$$K_v^n[i, j] = K_v^{n-1}[i, j] \otimes \left(K_v^{n-1}[i, n] \oplus K_v^{n-1}[n, j]\right), \tag{8}$$

where in (8) $K_v^1 = Q_v$, and in (7) the operator $\bigoplus$ between two real valued matrices B, C ($D_v = B_v \bigoplus C_v$) is defined as $D_v(i, j) = \otimes_{t=1}^{n}[B_v(i, t) \oplus C_v(t, j)]$. The operator $\oplus$ between two QoS parameters depends on the QoS metric that is being considered and can be the addition (+) for delay and variance, the minimum (min) for bandwidth and so on. The $\otimes$ is also an operator that depends on the specific QoS metric q, and selects the best value among the elements on which it operates. For example, in the case of the delay, loss, or variance metrics it will obtain the minimum value, while for bandwidth or security it will select the maximum value, for all paths going from i to j.

5. EXPERIMENTAL RESULTS

Two types of experiments were conducted. The first refers to the probing stage and the accuracy of the estimation, and the second to the decision stage and the efficiency of our AC algorithm.

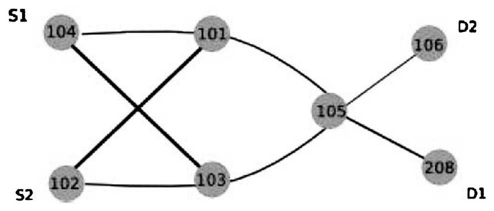

Fig. 2. The 7-node CPN testbed.

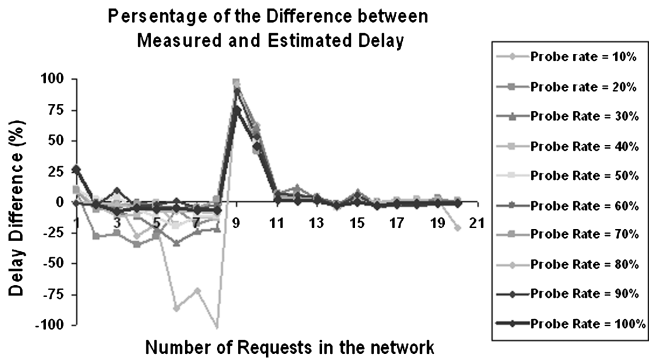

Fig. 3. Percentage of the average difference between estimated and measured delay for probe rate = 10% to 100% of the requested rate.

5.1 Probing Stage—Accuracy of the Estimation

In this section we present experimental results on the accuracy of the estimation of the impact that the new flow will have on the network. The experiments were conducted in a seven node network, with link capacity of 10Mbps.

In total, 2000 runs were conducted, with a duration of 50s each. For the sake of comparison, half of those experiments where conducted with no background traffic and half with one-flow background traffic, of 2Mbps, from $S2$ to $D2$ (Figure 2). The experiments with background traffic are very similar to those with no background traffic, and therefore are not included.

$S1$ generates a request to start sending packets to $D1$ every 30s. This is always accepted and it stays at the network until the end of the experiment. The rate of each new flow is 1Mbps. The probe packets were sent with rates raging from 10% to 100% of the requested rate for a time interval of 4s. Every combination was repeated five times. The values presented in the graphs are the average values of those repeats for each case.

The admission control mechanism is disabled so that the network will enter congestion state. This would not normally be the case, since the purpose of the Admission Control Algorithm is to make sure the network will not accept flows that will lead to congestion, but we allow it to investigate the accuracy of our estimation in all network load conditions.

In Figure 3, we compare the average percentage of the difference between the value of the estimated delay according to our algorithm and the measured delay after having actually sent the requested flow, for all probing rates. Figure 5 is

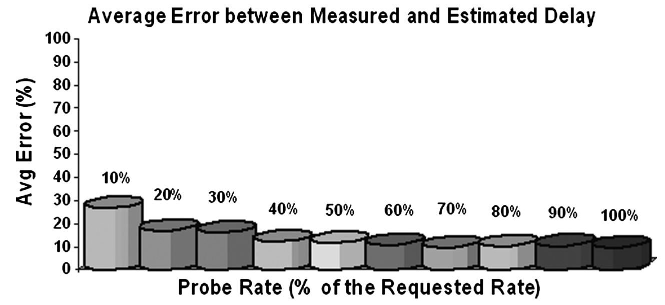

Fig. 4. Percentage of the average error between estimated and measured delay for probe rate = 10% to 100% of the requested rate.

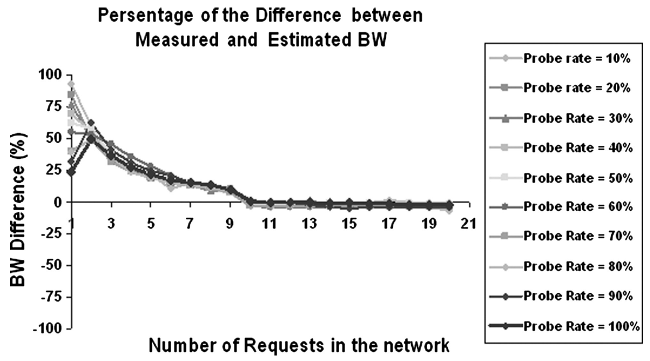

Fig. 5. Percentage of the average difference between estimated and measured bandwidth for probe rate = 10% to 100% of the requested rate.

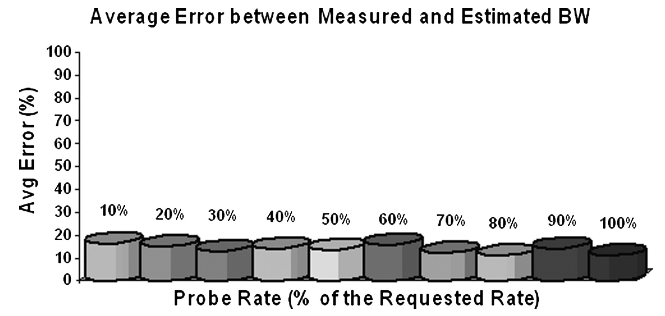

Fig. 6. Percentage of the average error between estimated and measured bandwidth for probe rate = 10% to 100% of the requested rate.

the same for bandwidth. Figures 4 and 6 show the average error percentage, for all probing rate cases, for delay and bandwidth respectively. We observe that our algorithm can provide accurate estimations (less than 15% average error) even if we send probe packets at a small percentage, for example, 40%, of the actual rate. Also, the error of estimation is low in all cases, but the biggest deflection is when the network enters the congestion stage and the slope of delay towards load is very steep. A way to improve the accuracy of the estimation would be to use the second derivative, but this is something that we do not cover in this article.

5.2 Decision Stage

Configuration of the experiments:

—The experiments were conducted in a 44-node test-bed representing the Switch-LAN network topology.[1]

—All links have the same capacity (10Mbps).

—All users have the same priority and the same QoS requirements: delay $\leq$ 50ms, jitter $\leq$ 5ms, bandwidth $\geq$ 3Mbps and packetloss $\leq 5\,\%$. These values are quite fastidious (for high-quality video the delay need only be less than 150ms) and are chosen in order to evaluate our algorithm for very demanding requests.

—There are 7 Source-Destination (S-D) pairs ($S_A - D_{A1}, S_A - D_{A2}, S_A - D_{A3}, S_B - D_{B1}, S_B - D_{B2}, S_C - D_{C1}, S_C - D_{C2}$), that correspond to 7 users ($A1, A2, A3, B1, B2, C1, C2$).

—In order to avoid having more than one user requesting entrance to the network at the same time, which would lead to multiple flows of probe traffic and misleading measurements, each user enters a queue ("request queue") at the data gathering point.

—The users who are denied access, instead of being considered rejected, enter another queue ("reject queue") and request to enter again. This process continues until they are finally accepted, or a specific period of time ("user lifetime") has passed. In our experiments the user lifetime of each user is 150s, meaning that every user will wait to be served for at most 150s and if that time expires the user will leave the network and will be considered rejected. The "reject queue" has bigger priority than the "request queue" and is served first. Of course if there were different users with different priorities there would be as many request and reject queues as the number of priority classes and each successive queue will start serving its user only if the previous, and higher in priority, queue is empty.

—After making a request, if the request is satisfied, the user will wait for a constant time W and then make another request. The same is true if its request is not satisfied.

—Our experiments covered three cases:

  —The Admission Control is disabled (NO AC).

  —The AC is enabled (WITH AC).

  —The AC is enabled but the length of the feasible paths is restricted (WITH AC & MPL).

The third approach tries to take into consideration the fact that the algorithm may accept a very long feasible path, which CPN would possibly not use for

[1]The Swiss Education & Research Network, http://www.switch.ch/network/

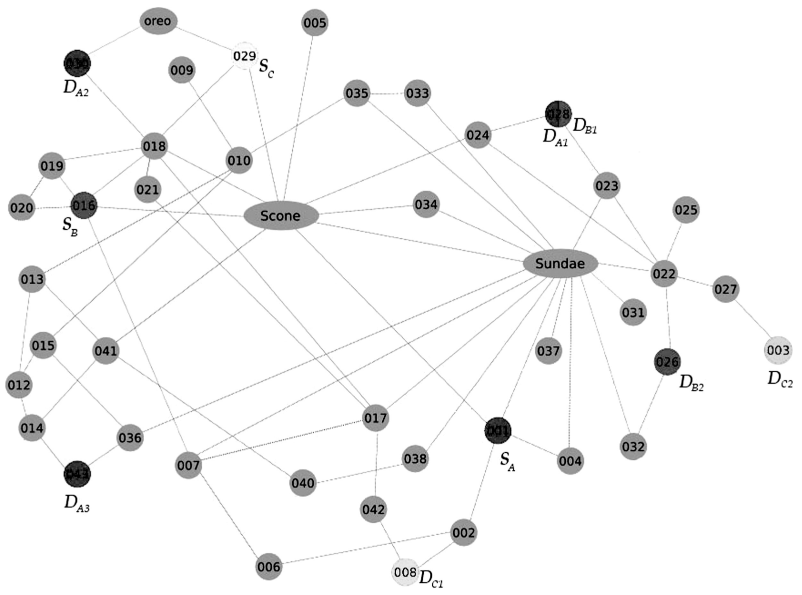

Fig. 7. The CPN testbed used in our experiments.

the new traffic. To avoid this conflict between the two intelligent mechanisms, we set a maximum path length (MPL) limit for the length of the feasible path. In our experiments the limit of the feasible path length was set to 6.

—In total 99 experiments were conducted (33 for each case), lasting 15 minutes each.

—In all cases each experiment had a time W, in the $(150, 120, 90, 80, 70, 60, 50, 40, 30, 20, 15)$ s range, which corresponds to a total arrival rate λ, for all three users of $(2.8, 3.5, 4.67, 5.25, 6, 7, 8.4, 10.5, 14, 21, 28)$ rqsts/min respectively.
Thus the load on the system was increased in each of the successive experiments. Each experiment was conducted 3 times and the results presented in this article are the average value of those three runs.

Figures 8, 9, and 10 summarize the experimental results. In Figure 8 we compare the total rejection rate for the connection requests in all three cases, while Figure 10 reports the satisfaction rate of a user (here user $A1$) in all three cases. By satisfaction of a user, we mean that all four QoS requirements of that user are fulfilled. Of course when the AC is disabled the rejection rate is always zero. Figure 9 shows the average time a user has to wait until they are served or the user lifetime expires, when the AC is enabled. In the case where the AC is disabled, all users are served the moment it is possible.

We observe that in both cases where the AC algorithm is enabled, the satisfaction of user $A1$ is much higher than when there is no AC. By restricting the length of the feasible path that is used to make the decision, we make the algorithm more strict, so the rejection rate should increase, as Figure 8 confirms. But since the feasible path's length is limited, the percentage of the user traffic

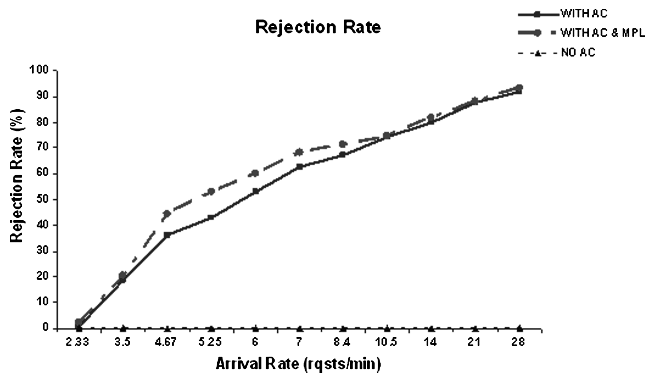

Fig. 8. Average rejection rate.

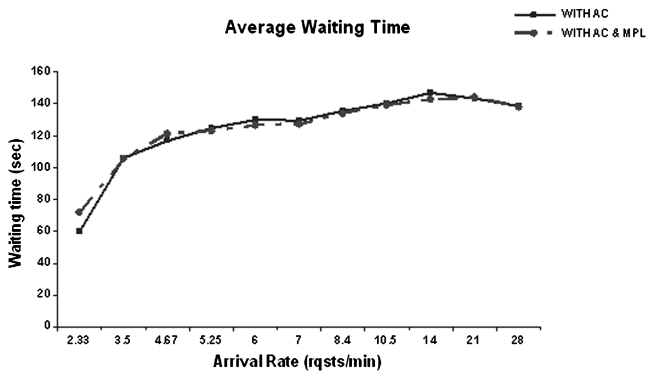

Fig. 9. Average time a user waits before he is served.

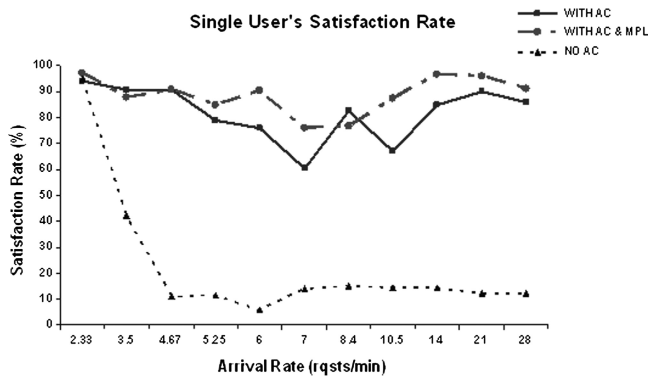

Fig. 10. Average user satisfaction.

that might not follow the feasible path is decreased. This leads to more accurate results and better satisfaction (Figure 10). We believe that finding the optimal limit of the feasible path length is something that will improve our algorithm and should therefore be further investigated.

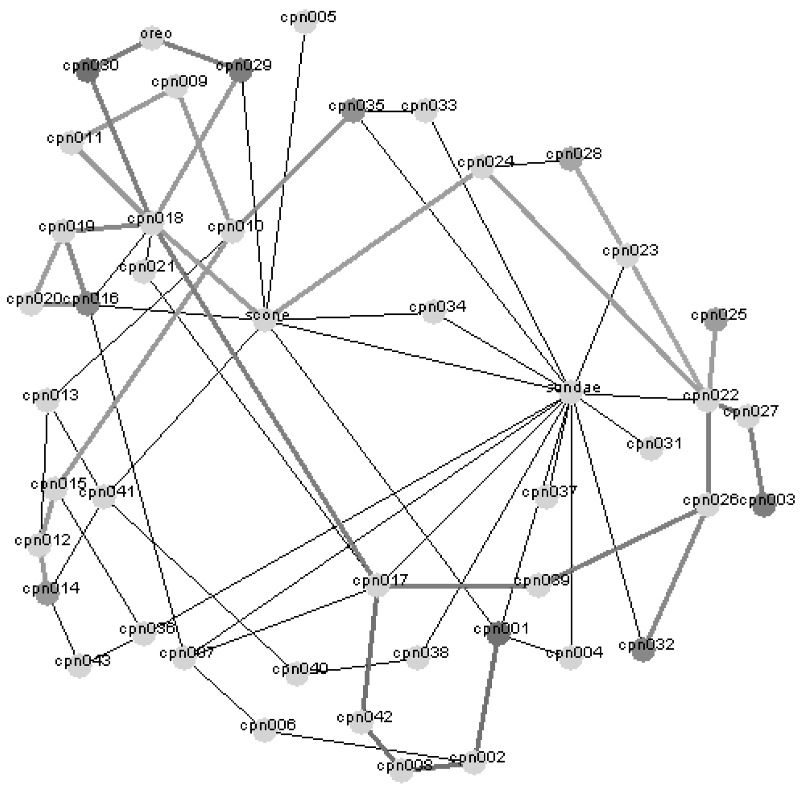

Fig. 11. Snapshot of the network before the failures.

5.3 Robustness of the Algorithm During Network Failures

The purpose of the experiments conducted in this section is to evaluate the robustness of our algorithm in the case of failures in the network. Simulating failures in a controlled and reproducible way is important in order to evaluate the robustness and the resilience of a network. A node can be considered under failure if traffic cannot go through it. So if we manage to disable all the Ethernet interfaces of a node, and exclude it from its neighbors, packets will not be able to reach it, just like in the case of a real breakdown. The failure can then be restored by re-enabling all the Ethernet interfaces.

The testbed used is again the one shown in Figure 7. The configuration of the testbed is similar to the one in Section 5.2 (7 source-destination pairs, 10Mbps links, rejection queues, probing rate 40% of the requested bandwidth, probing time 4s, flow lifetime 150s, etc.). The time W for which each user waits until he/she makes another request is 60s, which corresponds to total arrival rate $\lambda = 7$requests/min, for all three users of the network. Finally, all users have the same three QoS requirements: delay less than 150s, jitter less than 1ms, and bandwidth more than 3Mbps.

Two types of failures were taken under consideration: major failures and failure propagations. In the first major failures occur in critical parts of the

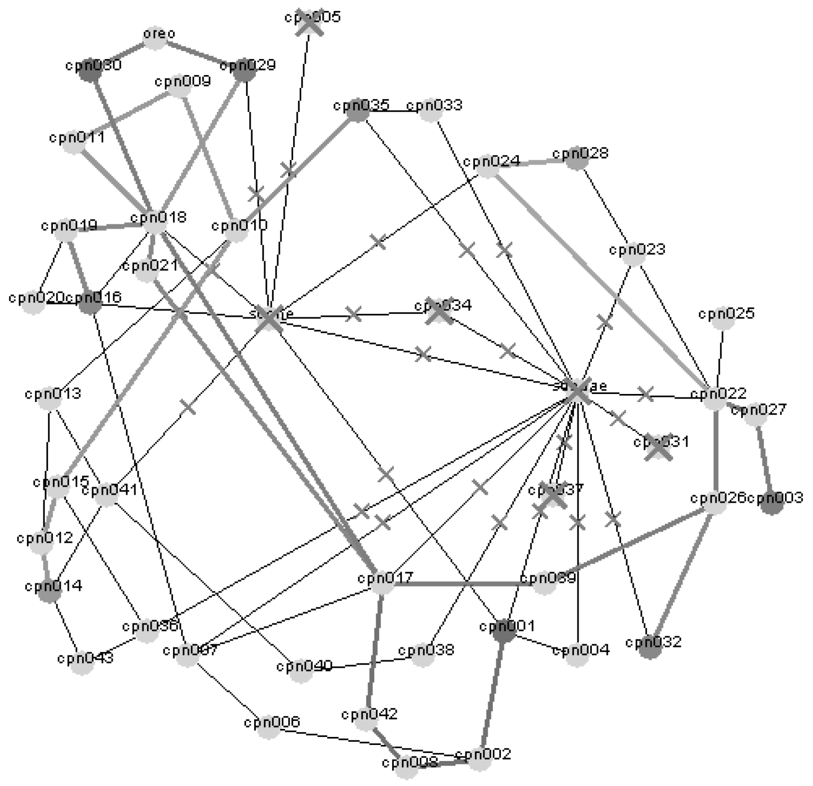

Fig. 12. Snapshot of the network when the two biggest nodal points are under failure.

network, and in the second the failures are spread all over the network just like a computer worm.

5.3.1 *Major Failures.* In this section we investigate the robustness of the network when failures occur in two of the biggest nodal points in the network, nodes Sundae and Scone (Figures 11 and 12).

Two types of experiments were conducted. One when we have AC, and one without. Each experiment lasted about 900s, where in the fist 300s there are no failures, in the next 300s, the two busiest nodes are under failure, and in the final 300s the failures are restored. Each experiment was conducted 3 times; the values shown in the Figures are the average values of all experiments.

In Figures 13 and 14, we compare the average delay and the average variance of delay, of one user in the network, when we have AC, and when not, with respect to the requested value. Figure 15 reports the total (since the beginning of the experiment) rejection rate of the user when the AC algorithm is enabled. We observe that when the AC algorithm is enabled, the QoS values of the user are almost always satisfied, even when the network is under failure. Again, by optimizing the probing rate and time, and thus minimizing the error of the estimation, the algorithm can be further improved.

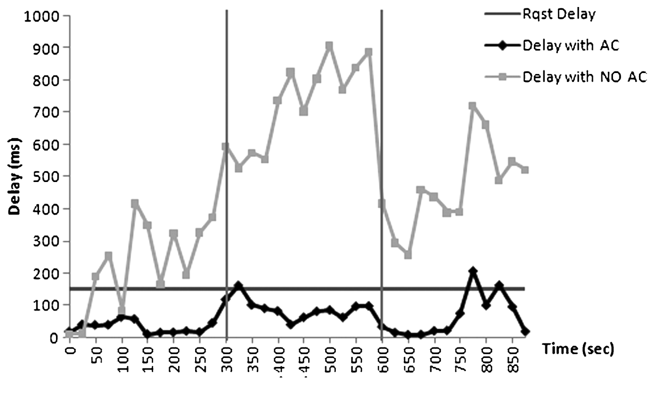

Fig. 13. Average delay when we have AC and when not, in comparison with the requested value, during experiments with major network failures.

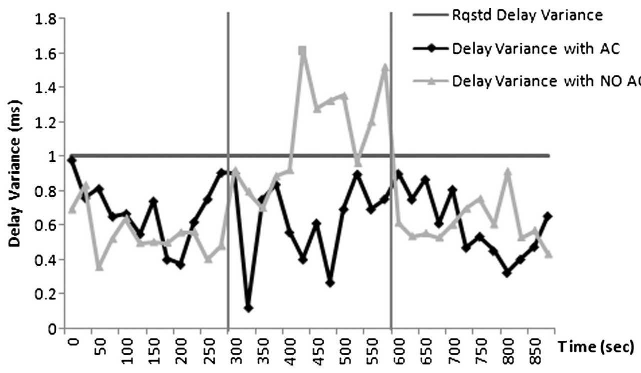

Fig. 14. Average variance of delay when we have AC and when not, in comparison with the requested value.

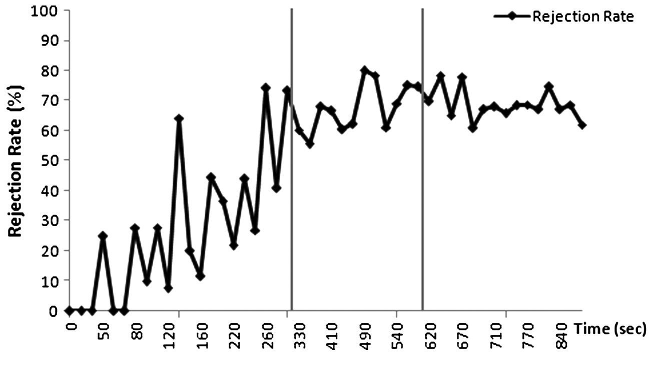

Fig. 15. Percentage of the AC rejection rate of the requests.

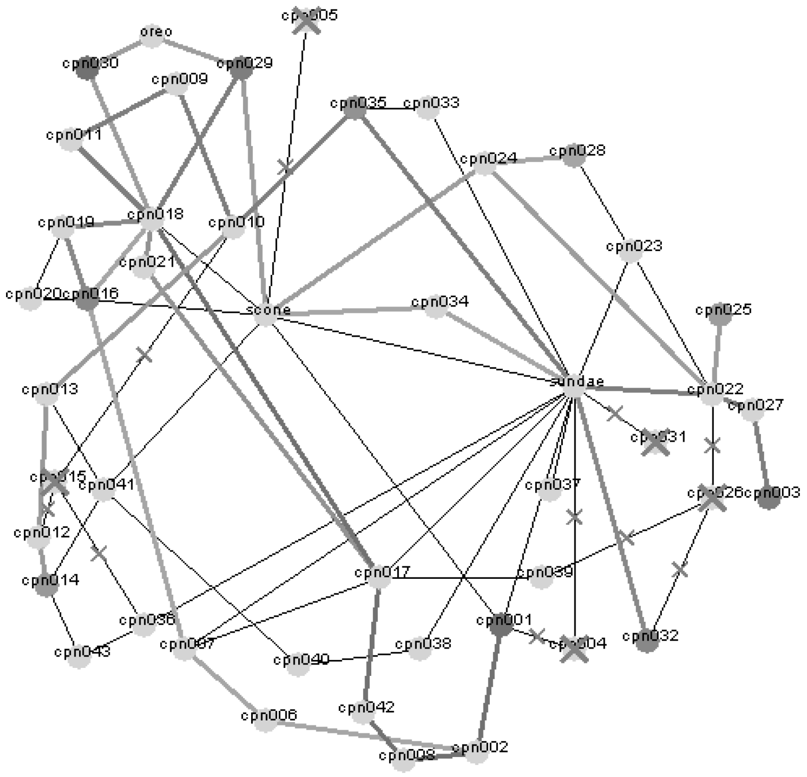

Fig. 16. Snapshot of the network at the beginning of the failure spread.

5.3.2 *Failure Propagation.* Next we conducted some experiments in which the failures are considered to be propagated all over the network, just like a computer worm (Figures 16, 17, 18).

In our experiments, the failure spread is modelled according to the AAWP (analytical active worm propagation) model. This is a discrete-time and continuous state deterministic approximation model, proposed in Chen et al. [2003] to model the spread of active worms that employ random scanning. In the AAWP model, a computer cannot infect other hosts before it is completely infected. That is expressed by the infection delay time, which is the time the worm needs to infect a machine. In our implementation this is currently a random value between two constant values (15 and 30 seconds) but could in the future depend on the distance between the infected node and the node it tries to infect, the degree of network congestion, or other such parameters.

In our present application we have assumed that all hosts can reach (infect, immunize) each other directly (no topology issue), and when a node is infected it is considered under failure (traffic cannot go though it), but it can infect others. More analytically, the failure starts to propagate after the first 300s of the experiment. We assume that at the beginning of the spread one random node is infected (*hitlist* = 1) and starts to infect the rest of the nodes. In order to capture the patching impact on the worm propagation, after 100s since the beginning of

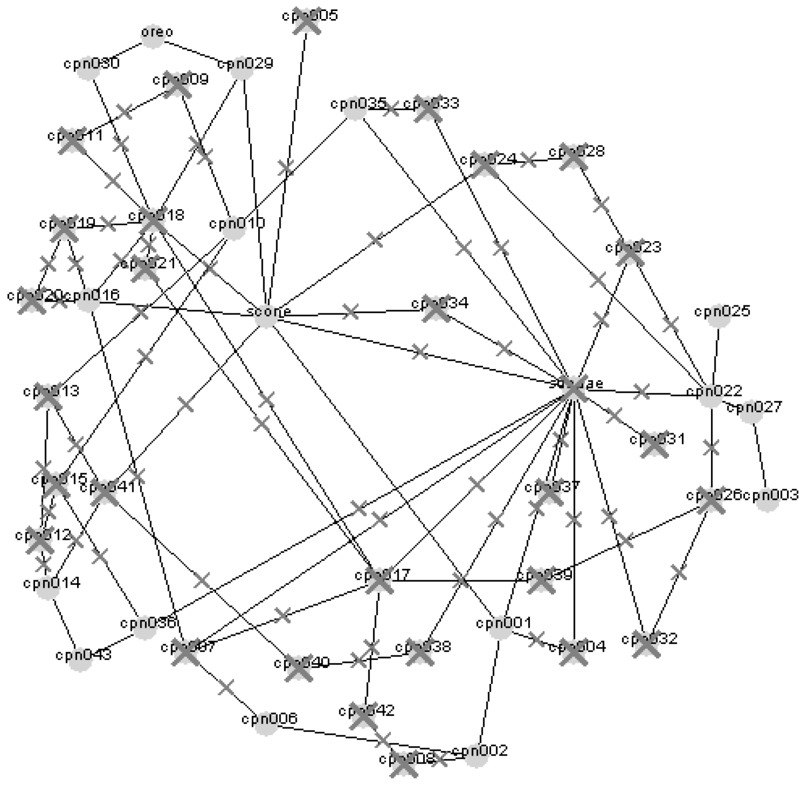

Fig. 17. Snapshot of the network after the failure has been propagated through most of the network.

the spread (400th second of the experiment), we dynamically immunize some hosts by randomly choosing some nonimmunized hosts (infected or simply vulnerable) to become immune. Again, the time a node needs to immunize another node is a random value between 15 and 30. The source-destination pairs are considered invulnerable (immune) from the beginning, so that they will not suffer any failures. We used the random scanning mechanism but others can be used in the future, such as local subnet, or topological scanning.

As in the previous failure scenario, two types of experiments were conducted: one with AC enabled and one with it not. Each experiment lasted about 900s, where in the first 300s there are no failures, then the spread and the immunization start, and last for a total of an average 310s, and finally in the last 300s all the failures have been restored and the network is operating normally. Each experiment was conducted three times. The values shown in Figures 19, 20, and 21 below are the average values of all experiments.

In Figures 19 and 20, we compare the average delay and the average variance of delay, of one user in the network, with and without AC, with respect to the requested value. Figure 21 reports the total rejection percentage of the user's requests when the AC algorithm is enabled. Once more, we observe that in this failure scenario, when the AC algorithm is enabled, the QoS values of the

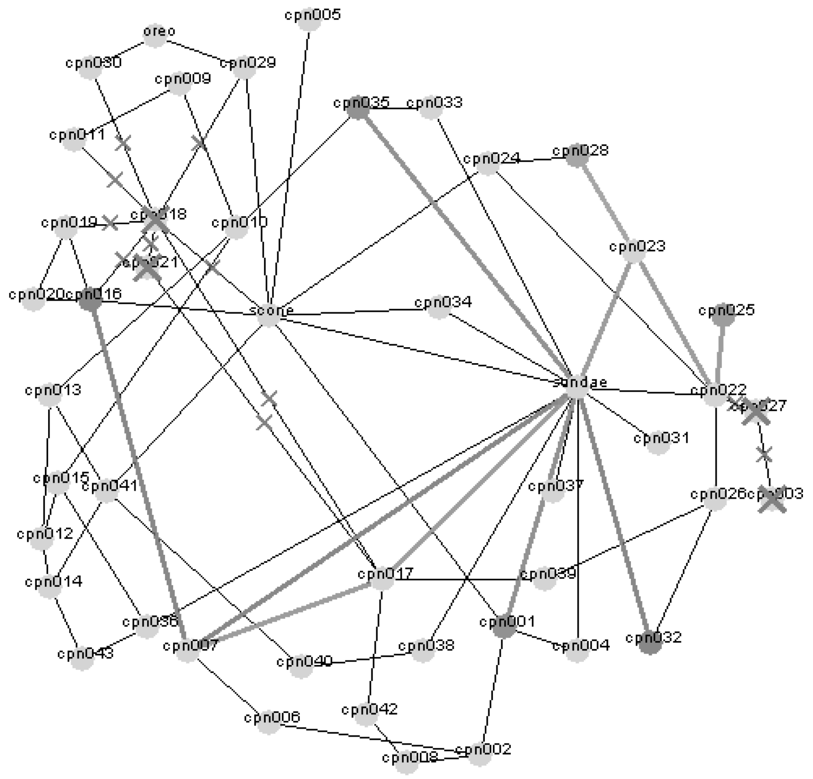

Fig. 18. Snapshot of the network after the failures are almost restored.

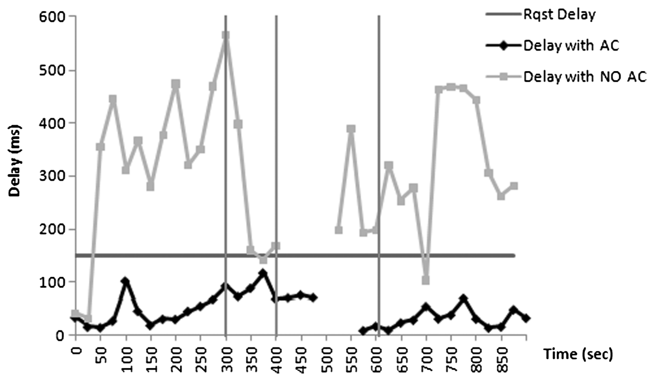

Fig. 19. Average delay with and without AC, with respect to the requested value, during a "worm" spread failure scenario.

user are almost always satisfied, regardless of the fact that network nodes are constantly and randomly under failure. The values missing in the graphs are due to the fact that at some point, most of the network nodes were under failure and there wasn't any route connecting the user's source and the destination.

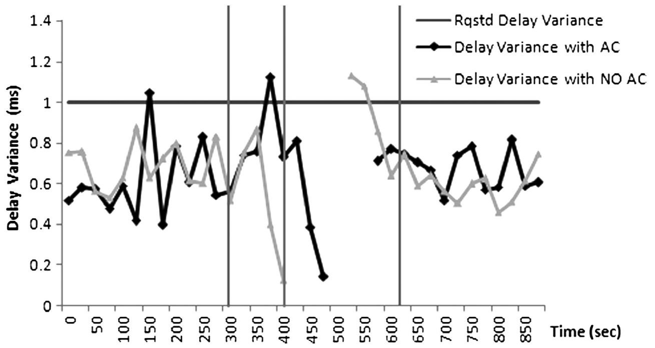

Fig. 20. Average variance of delay with and without AC, with respect to the requested value.

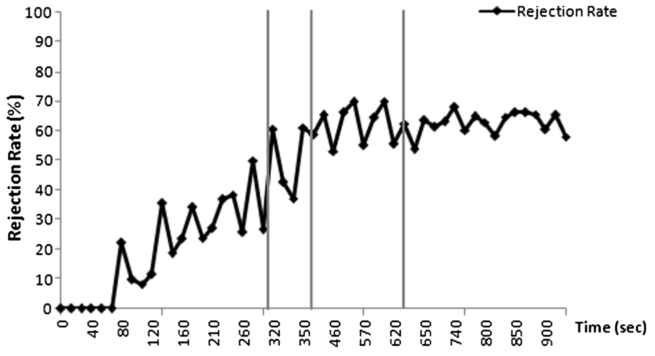

Fig. 21. Percentage of the AC rejection rate of the requests.

6. CONCLUSIONS

In this article we have proposed a measurement-based AC algorithm that uses measurements gathered by the CPN Self-Aware Network architecture. A novelty of the proposed AC algorithm is that the users can be the ones who specify the QoS constraints they need in order to obtain the network service they require for a successful connection. If this is not possible, the network adapts to the user's needs and provides him/her with the appropriate QoS constraints. The AC algorithm then investigates whether a request for a connection should be accepted, based on the incoming connection's QoS request and the impact it will have on ongoing connections.

We describe the algorithm and provide experimental results conducted in a laboratory test-bed showing the accuracy of the estimation of the impact that the new connection will have on the network, the effectiveness of our algorithm with respect to the user's satisfaction, and the robustness of the algorithm in failure scenarios. Contrary to existing methods, our algorithm offers a way to estimate the new flow's impact by probing at a small rate, so that probe packets will not contribute noticeably to the network's congestion. It uses that estimation in order to investigate whether there is a feasible path that can accommodate the new user without affecting the existing users' QoS constraints. The optimization of the probing rate and time, and of the accuracy of the probing estimation, would most probably lead to the improvement of our algorithm.

Much further work can be done in this area. In a forthcoming paper we will provide experimental results showing the effectiveness of the algorithm when users have different QoS criteria, including nonadditive QoS metrics, and compare our algorithm with existing ones. The prioritization and self-organization of users during network failures will also be presented in forthcoming papers.

REFERENCES

Bianchi, G., Capone, A., and Petrioli, C. 2000. Throughput analysis of end-to-end measurement-based admission control in IP. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*. Tel Aviv, Israel, 1461–1470.

Breslau, L., Knightly, E. W., Shenker, S., Stoica, I., and Zhang, H. 2000. Endpoint admission control: architectural issues and performance. In *Proceedings of ACM Special Interest Group on Data Communications (SIGCOMM 2000)*. Stockholm, Sweden, 57–70.

Cetinkaya, C. and Knightly, E. W. 2000. Egress admission control. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*. Tel Aviv, Israel, 1471–1480.

Chen, Z., Gao, L., and Kwiat, K. 2003. Modeling the spread of active worms. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*. San Francisco, CA, USA.

Elek, V., Karlsson, G., and Ronngren, R. 2000. Admission control based on end-to-end measurements. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*. Vol. 2. Tel Aviv, Israel, 623–630.

Floyd, R. W. 1962. Algorithm 97: Shortest path. *Comm. ACM 5*, 6 (June), 345.

Floyd, S. 1996. Comments on measurement-based admissions control for controlled-load services. Tech. rep., Lawrence Berkeley Laboratory. (July).

Ganesh, A. J., Key, P., Polis, D., and R.Srikant. 2005. Congestion notification and probing mechanisms for endpoint admission control. *IEEE/ACM Trans. Netw. 14*, 3 (June), 568–578.

Gelenbe, E. 1993. Learning in the recurrent random neural network. *Neural Computation 5*, 1 (Jan.), 154–164.

Gelenbe, E., Gellman, M., Lent, R., Liu, P., and Su, P. 2004. Autonomous smart routing for network qos. In *Proceedings of the First International Conference on Autonomic Computing (ICAC)*. New York, NY, 232–239.

Gelenbe, E., Lent, R., Montuori, A., and Xu, Z. 2002. Cognitive packet networks: QoS and performance. In *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*. IEEE Computer Society, Fort Worth, TX, 3–12. Opening Keynote Paper.

Gelenbe, E., Lent, R., and Xu, Z. 2001. Design and performance of cognitive packet networks. *Performance Evaluation 46*, 2-3 (Oct.), 155–176.

Gelenbe, E., Mang, X., and Oenvural, R. 1996. Diffusion based statistical call admission control in ATM. *Performance Evaluation 27/28*, (Oct.), 411–436.

Gelenbe, E., Sakellari, G., and D' Arienzo, M. 2007. Controlling access to preserve QoS in a self-aware network. In *Proceedings of the First IEEE International Conference on Self-Adaptive and Self-Orgnizing Systems (SASO 2007)*. Boston, MA.

Gelenbe, E., Xu, Z., and Seref, E. 1999. Cognitive packet networks. In *Proceedings of the 11th International Conference on Tools with Artificial Intelligence (ICTAI '99)*. IEEE Computer Society Press, Chicago, IL, 47–54.

Gibbens, R. J. and Kelly, F. 1999. Distributed connection acceptance control for a connectionless network. In *Proceedings of the 16th International Teletraffic Congress (ITC 99)*. Vol. 2. Edinburgh, UK, 941–52.

Gibbens, R. J., Kelly, F. P., and Key, P. B. 1995. A decision-theoretic approach to call admission control in atm networks. *IEEE J. Sel. Areas Comm. 13*, 6 (Feb.), 1101–1114.

GUERIN, R., AHMADI, H., AND NAGHSHINEH, M. 1991. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE J. Sel. Areas Comm. 9*, 7 (Sep.), 968–981.

GUERIN, R. AND GUN, L. 1992. A unified approach to bandwidth allocation and access control in fast packet-switched networks. In *Proceeding of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'92)*. Vol. 1. Florence, Italy, 1–12.

ITU. 2001. End-user multimedia QoS categories. Tech. rep., ITU-T Recommendation G.1010. (Nov.).

JAMIN, S., DANZIG, P., SHENKER, S., AND ZHANG, L. 1997a. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Trans. Netw. 5*, 1 (Aug./Sep.), 56–70.

JAMIN, S., SHENKER, S. J., AND DANZIG, P. B. 1997b. Comparison of measurement-based admission control algorithms for controlled-load service. In *Proceedings of the Conference on Computer Communications (IEEE INFOCOM '97)*. Vol. 3. IEEE Computer Society Press, Kobe, Japan, 973–980.

PERROS, H. G. AND ELSAYED, K. M. 1996. Call admission control schemes: A review. *IEEE Comm. Mag. 34*, 11 (Nov.), 82–91.

SAKELLARI, G., D' ARIENZO, M., AND GELENBE, E. 2006. Admission control in self aware networks. In *Proceedings of the 49th Annual IEEE Global Telecommunications Conference (GLOBECOM 2006)*. San Francisco, CA.

TSE, D. AND GROSSGLAUSER, M. 1997. Measurement-based call admission control: Analysis and simulation. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '97)*. Vol. 3. IEEE Computer Society Press, Kobe, Japan, 981–989.

WARSHALL, S. 1962. A theorem on Boolean matrices. *J. ACM 9*, 1 (Jan.), 11–12.

Received January 2007; revised September 2007; accepted November 2007