A Multiple Criteria, Measurement-Based Admission Control Mechanism for Self-Aware Networks

Georgia Sakellari
Imperial College London
g.sakellari@imperial.ac.uk

Erol Gelenbe
Imperial College London
e.gelenbe@imperial.ac.uk

***Abstract*—High demand and network congestion, can prevent multimedia applications and users from obtaining the network service they require for a successful operation. A way to control traffic congestion and satisfy all users' QoS requirements, without overprovisioning, is Admission Control (AC). This work presents a measurement-based AC mechanism, which improves the performance of a Self-Aware Network (SAN) [1] and provides QoS throughout the lifetime of all connections. Our algorithm is a multiple criteria AC algorithm, where each user can specify the QoS metrics that interest him/her. Our scheme which decides whether a new call should be allowed to enter the network based on measurements of the QoS metrics on each link of the network before and after the transmission of probe packets. The decision is based on a novel algebra of QoS metrics, inspired by Warshall's algorithm, that searches whether there is a feasible path to accommodate the new flow with out affecting the existing users. Our algorithm and the underlying mathematical principles will be briefly described and we will present experimental results, conducted in a large laboratory test-bed, under highly congested circumstances.**

I. Introduction

Providing Quality of Service (QoS) to the users of a network has led to the growth of new architectures and "smarter" networks which try to provide a more stable and reliable environment and guarantee packet delivery under specific QoS constraints. Such networks are the Self-Aware Networks (SAN) [1]. These are packet networks that collect data at different distributed points, by probing the network with special ("smart") packets. According to the QoS objective that those packets are pursuing a SAN adaptively takes corrective action in order address QoS and provide reliable service to its users. The SANs use adaptive packet routing protocols, such as the Cognitive Packet Network (CPN) [2] which is designed to perform Self-Improvement by using Random Neural Networks (RNN) [3] 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 of such an AC, which will improve the performance of a Smart Network is the purpose of the work presented in this paper. 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 decision is based on a novel algebra of QoS metrics, inspired by Warshall's algorithm, which looks for a path with acceptable QoS values to accommodate the new flow.

The reason we chose our algorithm to be measurement-based, is because 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. Also, this kind of AC algorithms are shown to achieve much higher utilisation than parameter-based [4], and they are more adaptive to network changes. 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.

II. 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) ([2], [5], [6]) and address QoS by using adaptive techniques based on on-line 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, etc. It is designed to perform Self-Improvement by learning from the experience of smart packets, using random neural networks (RNN) [3] 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.

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. When a Smart Packet arrives

to 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 which 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, it can be used to monitor the QoS metrics on a single link and/or partial or complete paths.

Each node also 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 for which the corresponding neuron is the most excited

The RL is carried out using a QoS Goal G, such as Packet Delay, Loss, Hop Count, Jitter, etc. 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 decisional weights of a 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.

The RNN weights are updated based on a threshold T. 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 increase very significantly 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, then we simply increase moderately all excitatory weights leading to all neurons, except for the previous winner, and increase significantly the inhibitory weights leading to the previous winning neuron, in order to punish it for not being very successful this time.

III. A MULTIPLE CRITERION AUTOMATIC ALGORITHM FOR AC AND MONITORING

The measurement-based AC algorithm we propose [7], [8], [9] is based on measurements of the QoS metrics on each link of the network. This does not require any special mechanism since, 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 one (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 stage (probing stage) the AC estimates the impact of the new flow by probing the network. Finally, in the third stage (decision stage) the AC searches whether there is a feasible path which can accommodate the new call by considering the impact of that new flow on the network without affecting the quality of formerly accepted flows.

A. 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 every day networks, there might be users that 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 as 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, 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 the 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). More specifically, 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 [10], 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.

B. 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 being currently 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, 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) which 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 a SP has not explored the link for

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 by the probe traffic so for that 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 satisfy the QoS needs of the new flow, so, 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

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

This estimation is based on the fact that every QoS metric can be considered as a value which 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.

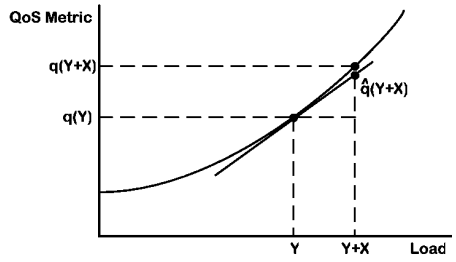

Fig. 1. QoS metric vs Load

Let as consider some link (i,j). A small increase x in the load that is obtained in a controlled manner, e.g. 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:

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

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

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

without having to know the initial load Y.

The above 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.

C. Decision Stage

Let us assume that the users may be concerned with m distinct QoS metrics $q_v \in R$, $v = 1, ...m$ 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 generalised 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 on how the path QoS matrices are computed from the link matrices.

D. 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 which 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 that 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 satisfy the QoS needs of the new flow, so, it is very likely to also be the path that will be followed after the new user's full traffic is inserted.
- Afterwards compute the estimation

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

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 below) 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.

E. Computing the QoS matrices

The well known "Warshall's algorithm" [11] 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.

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

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 [12] 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.

$$K^n[i,j] = min\Big\{ K^{n-1}[i,j], \left(K^{n-1}[i,n] + K^{n-1}[n,j] \right) \Big\}$$

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 all of the $K_v(i,j)$ are non-negative quantities.

For non-additive metrics we have developed a generalisation of the Floyd-Warshall. Due to space limitations we refer the reader to [9] for more details.

IV. Experimental results

The experiments where conducted in a 46-node test-bed representing the SwitchLAN network topology[1]. All links have the same capacity ($10\,Mbits/s$). All users have the same QoS requirements: $delay \leq 150\,ms$, $jitter \leq 1\,ms$, and $packetloss \leq 5\,\%$. There are 7 Source-Destination (S-D) pairs $(S_A - D_{A1}, S_A - D_{A2}, S_A - D_{A3}, S_B - D_{B1}, S_B - D_{B2}, S_C - D_C, S_D - D_D)$, that correspond to 7 users $(A1, A2, A3, B1, B2, C, and D)$. In order to avoid having more than one users requesting to enter 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. After making a request, if the request is satisfied then the user will wait for a constant time $W = 20s$ and then make another request. The same is true if its request is not satisfied. $W = 20s$ corresponds to total arrival rate $\lambda = 21\,rqsts/min$. In order for the algorithm to estimate the impact of the new user we probe the network at probe rate equal to 40% of the user's rate and for probing time of $2s$. When a call is accepted the source will generate a constant bit rate UDP traffic at $1Mbps$ that last for $600s$. Thus the load on the system is constantly increasing until the $600th$ second, and since the capacity of each link is $10Mbps$ this means that the network will be highly congested very quickly. Each experiment, which lasts for $15min$ ($900s$), was conducted 5 times and the results presented in this paper are the average value of those runs. Our experiments covered three cases:

- The Admission Control is disabled (No AC).
- The AC is enabled but all users are accepted (AC-All Accepted).
- The AC is enabled (With AC).

The second approach is considered in order to study the efficiency of our algorithm under the same conditions, since it is a centralised algorithm, and not allowing all users to enter the system at the same time, by itself, contributes to reducing the congestion.

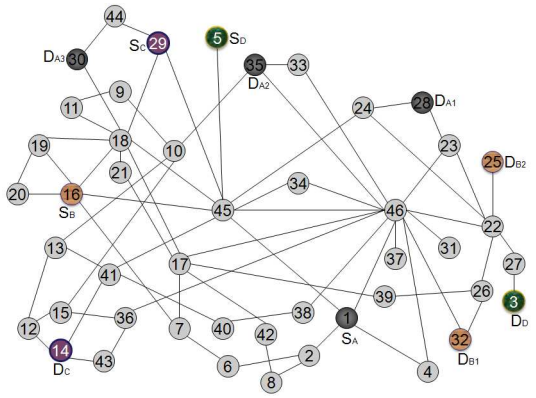

Fig. 2. The 46-node CPN testbed used in our experiments

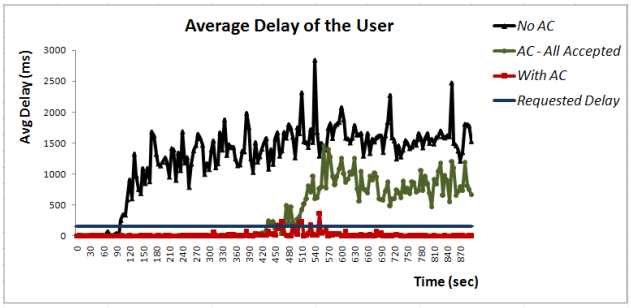

Fig. 3. Average Delay of the User

Figures 3, 4, 5, 6, and 7 summarise the experimental results. In figures 3, 4 and 5 we compare the average delay, jitter and packet loss of a user (here user $D1$) in the network in all three cases, while figure 7 reports the rejection rate when the AC is enabled. Of course when the AC is disabled, or when

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

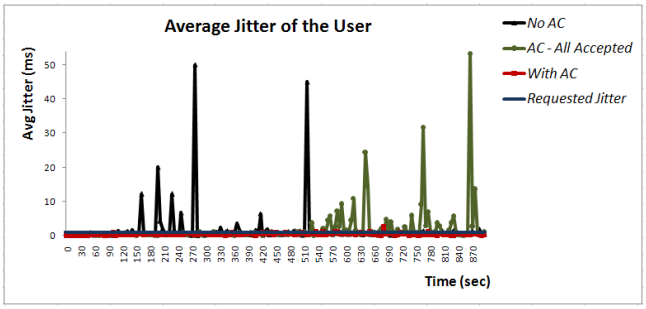

Fig. 4. Average Jitter of the User

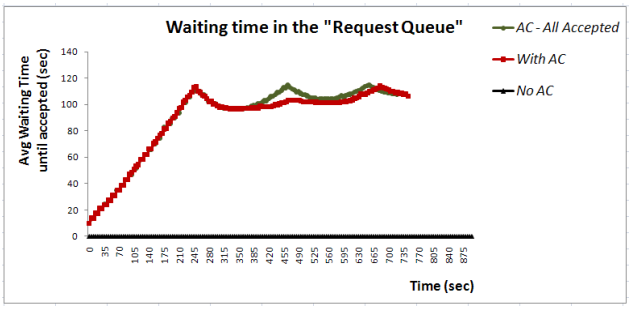

Fig. 6. Average time a user waits in the "request queue" before being served

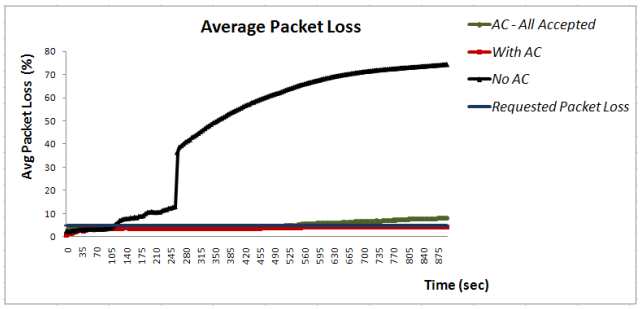

Fig. 5. Average Packet Loss of the User

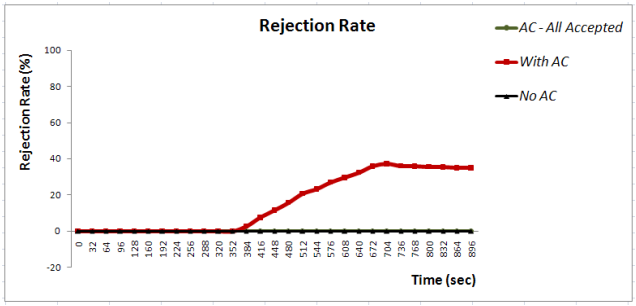

Fig. 7. Average Rejection Rate

it is enabled but all users are accepted, the rejection rate is always zero. Figure 6 shows the average time a user has to wait until it is finally accepted into the network, when the AC is enabled. In the case where the AC is disabled users do not wait in the "request queue" but are served the moment it is possible.

We observe that in both cases where the AC algorithm is enabled the satisfaction of the user is much better than when there is no AC. By satisfaction we mean that all the QoS criteria that a user has specified, when he/she entered the network, are satisfied. In the case where the AC is working fully the user under examination is satisfied **93.89%** of the time, contrary to when AC mechanism is enabled but all users are accepted, where the user is satisfied **49.44%** of the time. When the AC is disabled this percentage drops even further to **11.67%**. We also believe that finding the optimal probe rate and probing time values could further improve our algorithm.

Another important observation is that when there is no congestion (first $100s$ of the experiment) and all the users can be accepted the algorithm delays the users from being served without a reason. A solution to this problem could be to monitor the network and according to the load conditions to decide whether the AC mechanism should be enabled or not.

V. Conclusions

In this paper we have presented experimental results showing the efficiency of our algorithm during a period of time where the network progressively congests. The results showed that even under high congestion the algorithm performs very well but when the network is not congested the centralised character of the algorithm decelerates the service of the customers. Our future work thus will concentrate on which are those network characteristics which can proclaim when the AC mechanism is needed in a network.

References

[1] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and qos," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1478–1489, Sep. 2004.

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

[3] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, Jan. 1993.

[4] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 56–70, Aug./Sep. 1997.

[5] E. Gelenbe, R. Lent, and Z. Xu, "Design and performance of cognitive packet networks," *Performance Evaluation*, vol. 46, no. 2-3, pp. 155–176, Oct. 2001.

[6] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "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)*. Fort Worth, TX: IEEE Computer Society, Oct. 2002, pp. 3–12, opening Keynote Paper.

[7] G. Sakellari, M. D' Arienzo, and E. Gelenbe, "Admission control in self aware networks," in *Proceedings of the 49th annual IEEE Global Telecommunications Conference, GLOBECOM 2006*, San Francisco, CA, USA, Nov./Dec. 2006.

[8] E. Gelenbe, G. Sakellari, and M. D' Arienzo, "Controlling access to preserve qos in a self-aware network," in *Proceedings of the First IEEE International Conference on Self-Adaptive and Self-Orginizing Systems (SASO 2007)*, Boston, MA, USA, Jul. 2007.

[9] ——, "Admission of qos aware users in a smart network," *ACM Transactions on Autonomous and Adaptive Systems*, Mar. 2008.

[10] ITU, "End-user multimedia qos categories," ITU-T Recommendation G.1010, Tech. Rep., Nov. 2001.

[11] S. Warshall, "A theorem on boolean matrices," *Journal of the ACM*, vol. 9, no. 1, pp. 11–12, Jan. 1962.

[12] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of ACM*, vol. 5, no. 6, p. 345, June 1962.