A Distributed Admission Control Mechanism for Multi-Criteria QoS

Georgia Sakellari and Erol Gelenbe
Imperial College London, Intelligent Systems and Networks Group,
Electrical & Electronic Engineering Dept., SW7 2BT London, UK
Email: {g.sakellari, e.gelenbe}@imperial.ac.uk

***Abstract*—Admission control based on multiple criteria has always been a desirable feature for computer networks. In the past, we presented such a system, but it was based on a centralised decision mechanism, which by itself constitutes a significant limitation. Here, we describe a decentralised version of the initial admission control algorithm, while still keeping the multiple-criteria aspect, with each user being able to specify their Quality of Service (QoS) metrics at the level of the individual. Our scheme 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 without affecting the existing users. The decision is made at each source node individually, based on either only personal information or also on information exchange among the nodes that are involved. The performance of the algorithm is evaluated in terms of QoS throughout the lifetime of all connections. The experiments presented in this paper were conducted in an actual laboratory test-bed of realistic topology and under highly congested circumstances. The results are particularly encouraging.**

I. Introduction

High demand and network congestion can prevent multimedia applications and users from obtaining the network service they require for a successful operation. Admission control (AC) is the process that determines whether an incoming request should be accepted or rejected. This usually requires estimation of the level of QoS that a new user session will need and investigation of whether there are enough resources available to service that session without affecting the QoS of the existing users of the network. So, when a flow requests real-time service, the network needs to be able to characterise the requirements of the new flow and make an admission decision based on an estimation of its current and projected state.

In this paper we present a distributed admission control (DAC) system based on our earlier centralised algorithm proposed in [1], [2], [3]. The novelty of the algorithm lays on the fact that, contrary to existing schemes (e.g. [4], [5], [6], [7], [8]), the users can specify their own QoS criteria tailored to their individual needs. We developed this algorithm as an added feature of the Cognitive Packet Network (CPN) [9], [10], [11] for two reasons. In CPN it is the users rather than the nodes that are given control of the routing. Each user can specify the QoS criteria based on which data will be routed in the network, since the routing algorithm is directly related to the QoS desired by the end user. Different users can have different QoS goals depending on the application and the service they want to use, and therefore we leave the choice of the paths chosen to CPN. Also, CPN constantly collects real-time QoS data. Extracting real-time QoS information from a network is not an easy task, especially if that involves not only end-to-end values but also link values. It usually takes special monitoring mechanisms working on top of the network layer, capturing packets and creating log files at specific time intervals. The shorter those intervals are the closer to real-time values we get and the more demanding the monitoring system is in terms of CPU consumption. The CPN protocol gathers real-time QoS information at each node. Each packet or acknowledgement arriving at a node copies the network information that is relevant to it and keeps it in its header. Therefore, each node can read this information and keep statistical information about the current traffic.

II. Our proposed algorithm

Our AC algorithm is based on the centralised scheme described in [1], which consists of three stages. In the first one, the identification stage, the network identifies the quality criteria of users that have not specified their requirements and translates them into QoS metrics according to existing ITU-standard tables.

The second is the probing stage, where the algorithm estimates the impact that a new flow will have on the network, based on link QoS information acquired before and after sending probe packets to the desired destination. This information is in the form of link QoS matrices for every distinct QoS metric that may concern a user. The main difference with the centralised mechanism is that instead of collecting that QoS information at a central data centre where the decision is made, each input node collects its personal information and decides independently. Therefore, when a new request arrives at an input node, it is the node itself that probes the network and decides whether to accept the new flow or not.

Finally, in the decision stage, the AC searches for a feasible path that can accommodate the new call by considering its impact on the network. The admission control system bases the decision on the limited information that it has from the links that are affected by the probe traffic and from the existing flows of that input node. More specifically, the estimated

link QoS matrices of the probing stage are the input of the generalised Floyd-Warshall's algorithm described in [1]. The output of this algorithm is path QoS matrices that provide us with the "best QoS value" for every path between every pair of vertices, using any intermediate vertices. For example in case of the delay, loss or variance metric by "best value" we mean the minimum value, while for bandwidth or security it is the maximum value, etc. From those path matrices the algorithm checks whether the best QoS values of the new users and all existing users of the node correspond to the required ones. If all of them are satisfied then the new user is accepted into the network.

In order to gather link QoS information at each source node we make use of the ability of CPN to provide end-to-end QoS information. In CPN, each source contains a table, called DPRR (Dumb Packet Route Repository), which keeps the route that was followed by each data packet and was reported back by the corresponding acknowledgment (ACK) packet. We have modified this table to also keep QoS information, such as average end-to-end delay and jitter of the data packets. Since the header of each ACK packet contains QoS information from each hop of the path, we have also created another table, LINK_DPRR, to store average link QoS information about the links visited by the packets originating from each source. Both tables are updated every time a data packet's acknowledgment returns to the source. We chose to collect QoS values in an exponential average manner, over a given time window, so as to limit the effect of short-term fluctuations.

A. Configuration of the experiments

In order to evaluate our mechanisms we conducted our experiments on the real testbed located at Imperial College London.

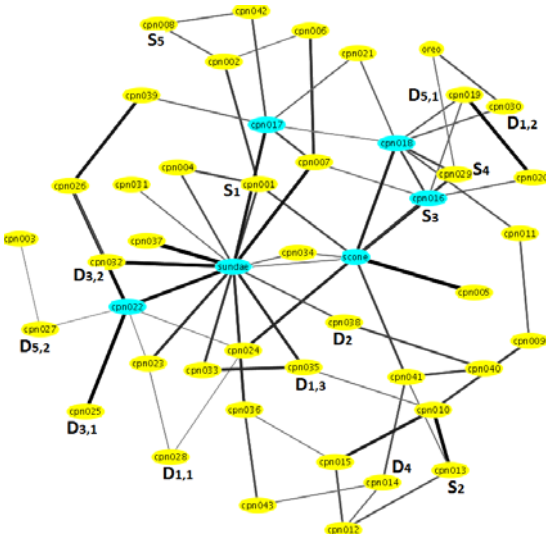

Fig. 1. Realistic topology with artificial delays. The thickness of the links represents their delay. The grey and thinner lines are low-delay links, while the darker (blacker) and thicker ones denote higher delays

The experiments are conducted on a real, 46-node testbed, the topology of which is based on the SwitchLAN network topology[1], which is used by universities other organisations in Switzerland. In order to make the environment more realistic we used actual details of the 46-router backbone, with OSPF costs, and link-level delays which were given by the administrators of the SwitchLAN network to the authors of [12]. At the time the experiments presented here were conducted one node (cpn034) was down due to hardware problems. In our experiments all links have $10\,Mbits/s$ capacity and all users have the same QoS requirements: round-trip $delay \leq 50\,ms$ and $jitter \leq 2.5\,ms$. There are 9 Source-Destination (S-D) pairs. At a specific input node, in order to avoid having more than one users requesting to enter the network at the same time each user enters a queue ("request queue") at the data gathering point. In the centralised version there is one such queue and it is located in the data collection point, so that all users from all input nodes will queue there in order to enter the network. After making a request, each user waits for a random time W and then makes a request again. We set the random waiting time W among requests, so as to have different rate for the arrivals. W is chosen to be uniformly distributed in the range of $[0, 15]$ seconds. We set the probing rate at 40% of the user's rate and the probing duration at $2s$. When a call is accepted, the source generates UDP traffic of $300kbps$ constant bit rate that lasts for a duration which is uniformly distributed in the range of values $[10, 200]$ seconds. Also, apart from the artificial delays which each link has we have introduced constant background traffic of $3.2Mbps$. Each experiment lasts for $5min$ $(300s)$ and is conducted 5 times. The results presented here are the average values of those runs.

Our experiments covered three cases: (i) the Admission Control does not probe and simply allows everybody to enter the network (DAC-All), (ii) the centralised AC, proposed in [1], is enabled (CAC) and (iii) our distributed AC is enabled (DAC).

B. Experimental results

In order to evaluate the performance of our algorithms our experimental results cover three areas. First we measure the QoS metrics of the users, the end-to-end delay and the jitter, and we compare them over the requested values. Secondly, we measure the time a user has to queue before given a decision on whether to enter the network or not. Finally we measure the number of the requests that were served during the duration of the experiments and how many of them were actually accepted.

Figures 2 and 3 present the average roundtrip delay and average jitter of all users in the network throughout the duration of the experiment compared with the requested values.

Let us consider as satisfaction of a user the percentage of time, over the whole lifetime of the user, where all the QoS requirements of that user are fulfilled. Here, on average, when there is no AC and all users are accepted in the network, the users' satisfaction is 23.08%. When we apply the centralised

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

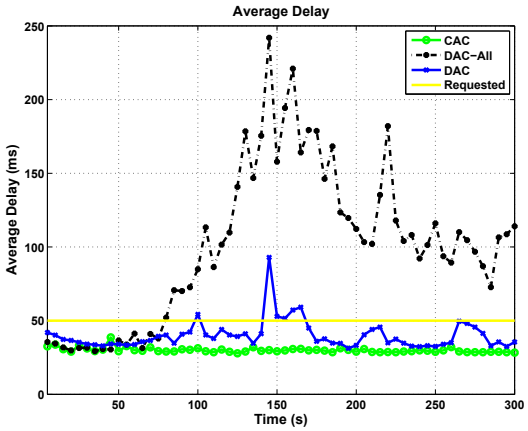

Fig. 2. Average Delay of the Users

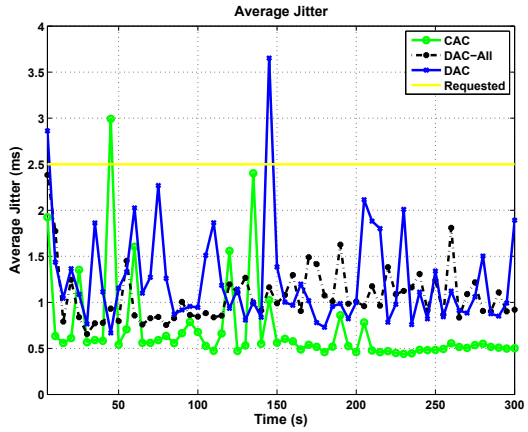

Fig. 3. Average Jitter of the Users

AC the satisfaction is 98.46% and when the distributed AC is active the users are satisfied on average 89.23% of the time.

Waiting time is the time a user has to stay in the "request queue" before it is decided whether he/she is accepted into the network or not. Figure 4 shows that the average time the users have to wait until being admitted when we don't have AC is $0.92s$. This waiting time is because if more than one users request to enter the network at the same input node they enter a queue. When we apply our DAC the time a user waits, until being admitted or rejected, increases to $3.01s$ mainly due to the probing which lasts $2s$. Finally, when the centralised AC is applied the average waiting time increases sharply to $49.69s$ since all users queue at the central point.

Waiting in order to be served affects the number of users that are actually accepted into the network. Figures 5 and 6 show that centrally controlled admission of users results in small number of users served and accepted into the network which was expected based on the waiting times we show before, while in the other two cases the number of requests served is more or less the same. The fact that more requests where made in DAC than when all users were accepted is due to the

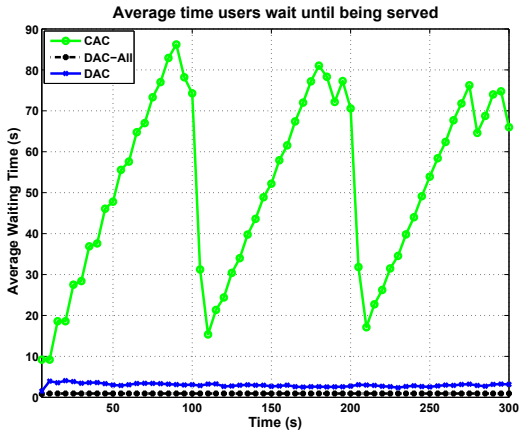

Fig. 4. Average time a user waits in the "request queue" before being served

randomness in the generation of the requests. Figure 6 shows that by distributing the AC mechanism more users are accepted into the network while keeping their QoS values close to the requested ones.

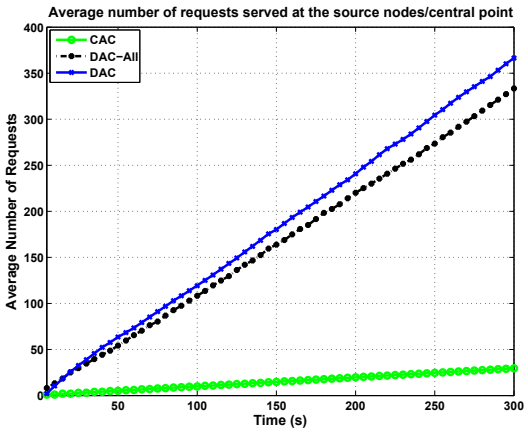

Fig. 5. Average Number of Requests

In practice, the centralised algorithm offers greater user satisfaction but fewer users. The satisfaction of the DAC is lower probably because multiple probes are in the network and the estimation of the algorithm is not accurate. Also, the fact that each input node has limited information and does not know the QoS values of all the links in the network could make the algorithm stricter. In order to provide a solution to the latter we introduced two simple mechanisms for coordination between the input nodes.

III. Node Coordination

In order to further improve the performance of our distributed AC, we tried to provide each source node with global network information by applying two simple coordination mechanisms between the input nodes.

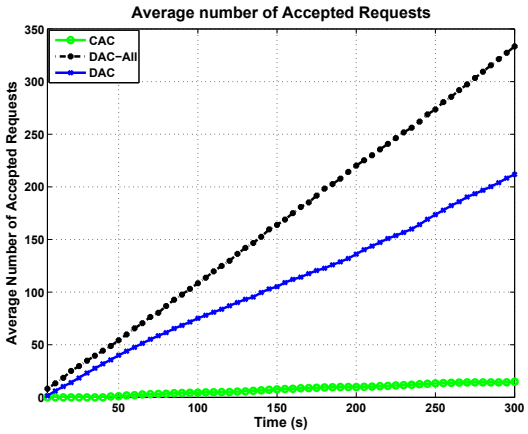

Fig. 6. Average Number of Accepted Requests

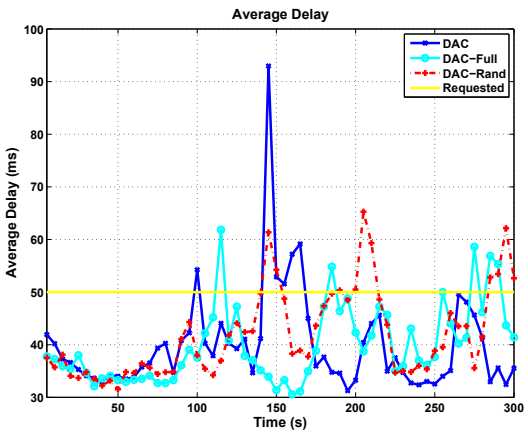

Fig. 7. Average Delay of the Users

A. *Fully coordinated nodes*

Firstly we implemented a simple coordinating mechanism in order for all the input nodes to have more or less the same information about the links of the network that are being used. This was done by exchanging messages between the input nodes. Every time a node collects link information it sends those values, and the time it took the measurements, to all of the other input nodes. Thus, when a node wants to make a decision it bases it on the most recent link values taken from all the nodes.

B. *Random coordination between nodes*

Having nodes to exchange messages every time they measure a different link QoS value introduces additional overhead in the network. Therefore we also implemented a lighter coordination mechanism. In this mechanism, every time an input node has new QoS measurements instead of sending them to every source node in the network it randomly chooses one and only sends it to it.

C. *Experimental results*

The experiments have the same configuration as before and cover three cases: (i) the DAC with no coordination between the input nodes (DAC), (ii) the DAC with full coordination between the input nodes (DAC-Full) and (iii) the DAC with random coordination between the input nodes (DAC-Rand).

As we can see from figures 7, 8 when the coordination is applied the QoS values of the users are closer to the requested ones. As far as satisfaction is concerned though, when full coordination is applied the average user satisfaction is 86.15% and when random coordination is active the satisfaction drops to 83.08%. This could be due to the messages exchanged between the input nodes. Also the coordination should have more obvious benefits in a much bigger testbed.

Figure 9 shows that coordination increases slightly the waiting time due to the message exchanges. As expected the waiting time of the full coordination is bigger than the random coordination.

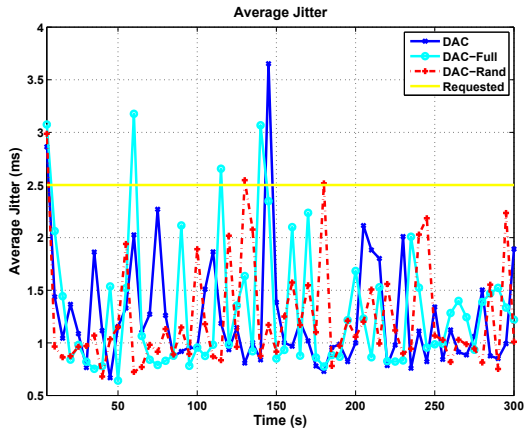

Fig. 8. Average Jitter of the Users

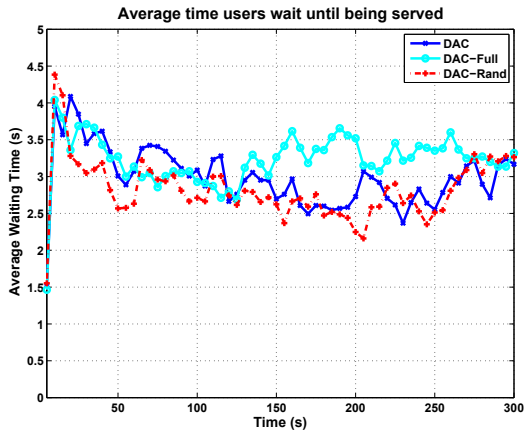

Fig. 9. Average time a user waits in the "request queue" before being served

Figure 10 shows that the same number of requests are being served in all three cases while figure 11 shows that with the random coordination more users are accepted since fewer messages are exchanged between the input nodes.

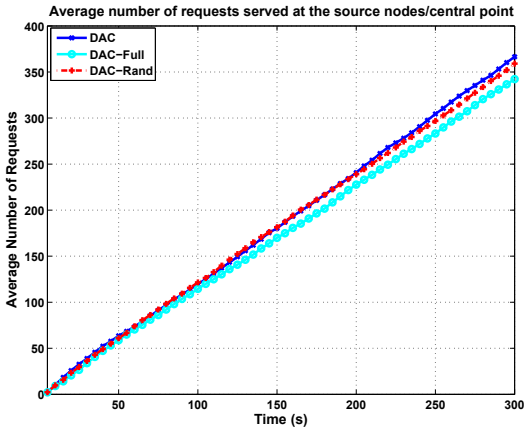

Fig. 10. Average Number of Requests

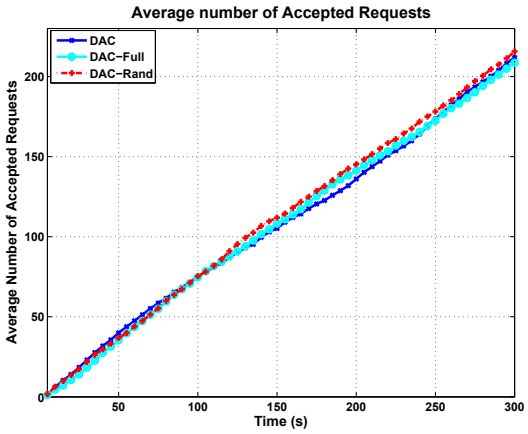

Fig. 11. Average Number of Accepted Requests

Regardless of the fact that the QoS values experienced by the users are very close to the requested ones, the satisfaction of the users is lower. This is because the sources probe the network at the same time making the estimations inaccurate. A way to overcome this could be to use a token passing mechanism for the probing stage. Another idea is to use a mechanism which will supervise the decision stage. For example, when the network is close to congestion the admission could be serialised by a mechanism, such as an auction process, which will choose the less demanding of the requests.

IV. Conclusions

In this paper we have presented a distributed version of the AC algorithm proposed in [1]. It is obvious that having a centralised AC mechanism raises security and robustness issues, as a result of the existence of a single point of failure. Also centralisation causes long queuing delays. On the other hand, the distributed version each input node bases its decisions on limited and often less accurate information. Also, each source node probes the network independently causing false estimations. The experimental results showed that by decentralising our AC algorithm the users do get QoS values that are kept closer to the required ones. Our simple coordination mechanisms do not improve the users' satisfaction, although the QoS values are kept close to the requested ones. A direction towards improving our distributed AC could be to look into other coordination mechanisms between the decision (input) nodes, such as the use of auctions. Our future work will concentrate on the investigation of these ideas.

Acknowledgment

The author would like to thank the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project which is jointly funded by a BAE (British Aerospace) Systems and EPSRC (Engineering and Physical Sciences Research Council) strategic partnership (EP/C548051/1) and the SATURN (Self-organizing Adaptive Technology underlying Resilient Networks) project which is sponsored by the UK Technology Strategy Board as part of the Saturn Consortium.

References

[1] E. Gelenbe, G. Sakellari, and M. D' Arienzo, "Admission of QoS Aware Users in a Smart Network," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 1, pp. 4:1–4:28, Mar. 2008.

[2] 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, pp. 1–5.

[3] 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, pp. 205–213.

[4] G. Bianchi, F. Borgonovo, A. Capone, L. Fratta, and C. Petrioli, "Endpoint Admission Control with Delay Variation Measurements for QoS in IP Networks," *Computer Communication Review*, vol. 32, no. 2, pp. 61–69, Apr. 2002.

[5] C. Cetinkaya and E. W. Knightly, "Egress Admission Control," in *Proceedings of IEEE INFOCOM 2000*, vol. 1, Tel Aviv, Israel, Mar. 2000, pp. 1471–1480.

[6] 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.

[7] S. Georgoulas, P. Trimintzios, G. Pavlou, and K. Ho, "An integrated bandwidth allocation and admission control framework for the support of heterogeneous real-time traffic in class-based ip networks," *Computer Communications*, vol. 31, no. 1, pp. 129–152, Jan. 2008.

[8] J. Milbrandt, M. Menth, and J. Junker, "Experience-Based Admission Control in the Presence of Traffic Changes," *Journal of Communications*, vol. 2, no. 1, pp. 10–21, Jan. 2007.

[9] 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, USA: IEEE Computer Society Press, Nov. 1999, pp. 47–54.

[10] 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.

[11] 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, Texas, USA, Oct. 2002, pp. 3–9, opening Keynote Paper.

[12] M. Gellman and P. Liu, "Random Neural Networks for the Adaptive Control of Packet Networks," in *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN 2006)*, Athens, Greece, Sep. 2006, pp. 313–320.