A Decentralised, Measurement-based Admission Control Mechanism for Self-Aware Networks

Georgia Sakellari

Imperial College London
Intelligent Systems and Networks Group
Electrical & Electronic Engineering Dept.
SW7 2BT London, UK
Email: g.sakellari@imperial.ac.uk

Abstract. This paper presents a decentralised Admission Control (AC) algorithm, based on the centralised proposed in [1–4]. Our algorithm is a multiple criteria AC algorithm, where each user can specify the QoS metrics that interest him/her, and 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. Our algorithm will be briefly described and we will present experimental results, conducted in a large laboratory test-bed, under highly congested circumstances.

1 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. Here we describe a decentralised AC algorithm which is based on the centralised, multiple-criteria, measurement-based AC algorithm presented in [1–4]. Our algorithm is targeted for self-aware networks (SAN) [5] and more specifically for the Cognitive Packet Network (CPN) [6] used by the SANs for two reasons. Firstly because the CPN protocol is directly related to the QoS desired by the end user and each user can specify different QoS goals based on which the routing decisions are being made. Secondly, CPN constantly collects real-time QoS data, so their is no need for special monitoring mechanisms which would work on top of the network layer, capturing packets and creating log files at specific time intervals.

2 Our proposed algorithm

Our AC algorithm consists of three stages. In the identification stage the network identifies the quality criteria of a new user request and translates them into QoS metrics. In the probing stage each input node estimates the impact that its new flow will have to the network, based on personal link QoS information acquired by sending probe packets to the desired destination and by receiving information from the rest of the nodes about their finding from similar probings. Finally, in the decision stage each node searches for a feasible path that can accommodate the new call by considering the impact of its new flow on the network. All of the stages are similar to those of the centralised AC described in [1, 4], the only difference is that now, instead of collecting QoS information about all links to a central data centre where the decision is being made, each input node collects its personal information, about specific links, and decides independently.

2.1 Configuration of the experiments

In order to evaluate our mechanisms we conducted experiments in a real, 46-node testbed located at Imperial College London (same as in [4]. 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 that correspond to 7 users. After making a request, the user will wait for a random time W and then make a request again. We set the random waiting time W among requests in order to have different rate for the arrivals. W is chosen to be uniformly distributed in the range of values $[0, 15]$ 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 $1Mbps$ constant bit rate that lasts for $600s$. Thus, the load on the system is constantly increasing at least until the $600th$ second. Since the capacity of each link is $10Mbps$, this means that the network becomes highly congested very quickly. Each experiment, lasts for $15min$ ($900s$) and was conducted 5 times. The results presented here are the average values of those runs.

Our experiments covered three cases: (i) The Admission Control is disabled (NoAC), (ii) i) the centralised AC, proposed in [1], is enabled (CAC) and (iii) the decentralised AC is enabled (DAC).

In figures 1, 2 and 3 we compare the average packet loss, delay and jitter of a user in the network in all three cases. We observe that in both cases where the AC algorithm is enabled, the satisfaction of the user is much higher than when there is no AC. By satisfaction we mean the percentage of time throughout the experiment duration that all three QoS criteria, that user $D1$ has specified, are met. In the case of the centralised AC user $D1$ is satisfied 81.09% of the time, contrary to the decentralised AC mechanism where the user is satisfied 18.92% of the time. When the AC is disabled, this percentage drops even further to 8.11%. It needs to be noted that even if the percentage of the decentralised AC is low the user's QoS values are much closer to the requested ones than when we don't have AC.

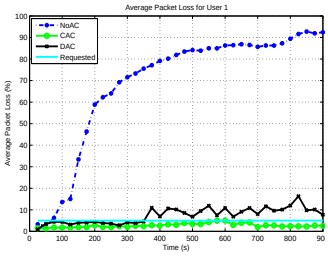

Fig. 1. Average Packet Loss

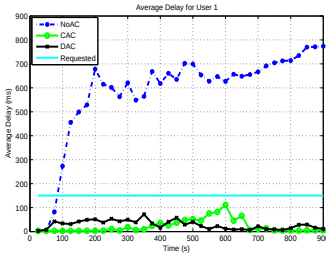

Fig. 2. Average Delay

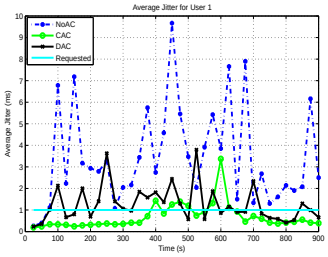

Fig. 3. Average Jitter

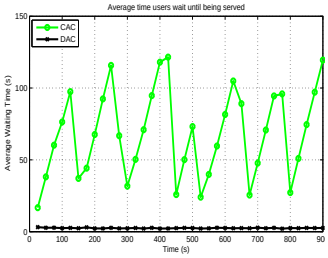

Fig. 4. Average time a user waits in the "request queue" before being served

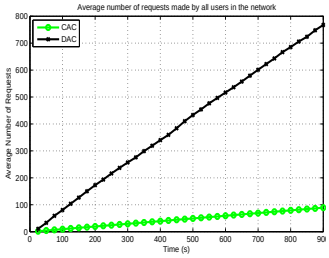

Fig. 5. Average Number of Requests

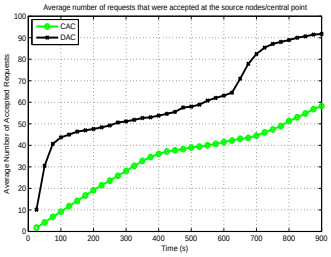

Fig. 6. Average Number of Accepted Requests

Figure 4 shows the average time a user has to wait until it is accepted into the network, when the AC is enabled. In the case of the centralised AC the users queue at the central point while in the decentralised version there are individual "request queues" at each input node. Here we present the average waiting time over all these queues. When the AC is disabled, users do not wait in a queue but are served as quickly as possible. In our experiments, a user has to wait on average $68.11s$ when the AC procedure is centralised and only $2.49s$ when the AC decision is taken independently at each input node.

Figures 5, 6 report the number of requests made in the whole network and the number of accepted requests respectively, when the AC schemes are enabled. We observe that with the decentralised algorithm the number of requests served and accepted into the network is much higher. This is due to the fact that the users's do not need to wait in a single queue at the central point and are therefore served much faster.

3 Node coordination

In order to further improve the performance of our decentralised AC, we tried two simple coordination mechanisms between the input nodes. The admission

decision of our decentralised algorithm is based on the limited personal QoS information that each input node has from the links that are affected by the probe traffic and from the existing flows initiated by that node. In the experiments presented at the previous section we observed that the satisfaction of the accepted users in the decentralised version is worse than in the centralised one. This is mainly because each input node has limited information and does not know the QoS values of all the links like in the centralised version. Also, multiple probes are in the network and the estimation of the algorithm is not accurate.

Here we test two simple coordinating mechanisms in order for all the input nodes to have more "global" information about the links of the network. First we exchanging messages between all the input nodes. Every time a node measures link information it sends those values along with the time it measured them to all of the other input nodes. When a node wants to make a decision it bases it on the most recent link values taken from all the 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.

3.1 Experimental results

The experiments have the same configuration as in section 2.1 and cover three cases: (i) the decentralised AC with no coordination between the input nodes (DAC), (ii) the decentralised AC with full coordination between the input nodes (DAC-Full) and (iii) the decentralised AC with random coordination between the input nodes (DAC-Rand).

From figures 7, 8 and 9 we observe that the satisfaction of the user improves when coordination is used. Same as before, in the case of the decentralised AC user $D1$ is satisfied 18.92% of the time. When we apply full coordination the user satisfaction increases to 27.03% and when we have random coordination the satisfaction surprisingly increases further to 40.54%. Additionally, the percentage of the satisfaction is still low mainly because of the jitter restriction. This is maybe because we use CPN with only delay as QoS goal and therefore CPN chooses the smallest delay paths while our AC algorithm looks at delay jitter and loss. We believe that if you use a combinatory QoS goal the results will improve.

Figure 10 shows that when we have coordination, the waiting time is slightly longer due to the message exchanges. More specifically, for DAC the average waiting time is $2.49s$, while for the fully coordinated DAC it is $2.69s$ and for the randomly coordinated is $2.56s$.

Figure 11 shows that almost the same number of requests are being made in all three cases while figure 12 shows that by having coordination more users are accepted into the network. So, when coordination is used not only the satisfaction is improved but also the number of users accepted into the network increases. This is because with the coordination the input nodes have more information

about the network status. Additionally, with the random coordination even more users are accepted since fewer messages are exchanged between the input nodes.

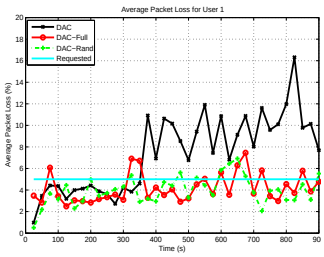

Fig. 7. Average Packet Loss

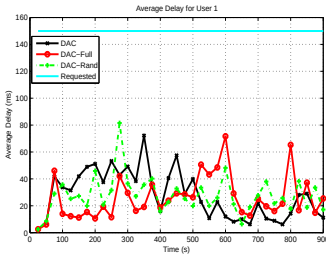

Fig. 8. Average Delay

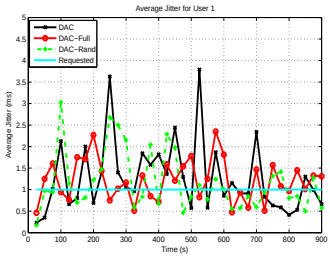

Fig. 9. Average Jitter

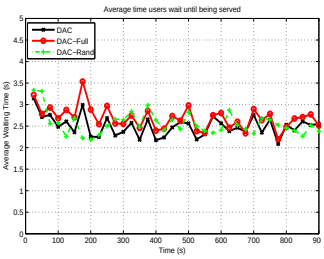

Fig. 10. Average time a user waits in the "request queue" before being served

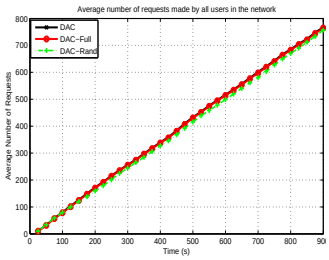

Fig. 11. Average Number of Requests

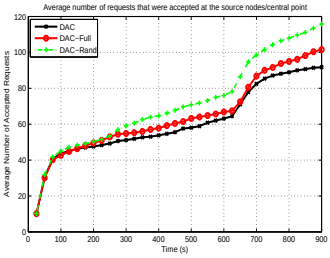

Fig. 12. Average Number of Accepted Requests

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 still quite low. This is because the sources probe the network at the same time making the estimations inaccurate. A way to overcome this could be by making the algorithm stricter, for example by decreasing the requested QoS values or by putting a restriction on the feasible path's size. Of course making the algorithm too strict could lead to poor use of the network resources and low utilisation. Another way would be to use a token passing mechanism for the probing stage, but this would increase the waiting times. Another idea is to use the distributed algorithm when the demand is small and then serialise the admission process by an auction process when the demand gets high. We are currently investigating the latter.

4 Conclusions

It is obvious that having a centralised AC mechanism raises security issues, since there is a single point and if this fails the system collapses. Also in the centralised version users have to wait for a relatively long time in order to be served. On the other hand, in the decentralised version each input node bases its decisions on restricted information. Also, each source node probes the network independently causing false estimations and additional traffic to the network. The experimental results showed that by decentralising our AC algorithm the network does not get over-congested and the QoS values are kept close to the required ones, but, as far as the satisfaction of the users is concerned, it is less likely that the user-specified QoS requirements will be met. A direction towards improving the decentralised AC could be to look into other coordination mechanisms between the decision (input) nodes. An idea would be to use an auctioning mechanism to supervise the decision stage.

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. Gelenbe, E., Sakellari, G., D' Arienzo, M.: Admission of QoS Aware Users in a Smart Network. ACM Transactions on Autonomous and Adaptive Systems **3**(1) (Mar. 2008) 4:1–4:28
2. Sakellari, G., D' Arienzo, M., Gelenbe, E.: 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) 1–5
3. Gelenbe, E., Sakellari, G., D' Arienzo, M.: 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) 205–213
4. Sakellari, G., Gelenbe, E.: A Multiple Criteria, Measurement-Based Admission Control for Self-Aware Networks. In: Proceedings of the third International Conference on Communications and Networking in China (CHINACOM'08), Hangzhou, China (Aug. 2008)
5. Gelenbe, E., Lent, R., Nunez, A.: Self-Aware Networks and QoS. Proceedings of the IEEE **92**(9) (Sep. 2004) 1478–1489
6. Gelenbe, E., Xu, Z., Seref, E.: 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) 47–54