An Enforced Inter-Admission Delay Performance-Driven Connection Admission Control Algorithm

Stanislav Belenki
Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, Sweden
tel. +46 31 772 1710
belenki@ce.chalmers.se

ABSTRACT

Connection Admission Control (CAC) is an important function in a computer network that supports Quality of Service (QoS). The function of CAC is to decide whether a new connection can be admitted on the network or a part of the network in such a way that the QoS of the new connection and the already established connections will remain within the requested limits. CAC must also ensure that network resources are used efficiently avoiding unnecessary rejections of candidate connections. Some CAC algorithms attempt to estimate or derive available resources in each hop between the source and the destination. Then they use parameters of the candidate flow and the knowledge about the available resources to make the admission decision. Very often such algorithms use measurements to find out availability of the resources. Hence the name of such algorithms, Measurement-Based CAC (MBCAC). Another class of CAC algorithms assumes that it is the end nodes and not the network that should perform the connection admission. These algorithms use probing packets to probe the path between the end nodes for the requested level of QoS. In this paper a heuristic-based per-hop CAC algorithm is introduced that adapts the average rate of connection admission in response to the measured system performance. In particular, the algorithm decreases the rate of the connection admission when the system is overloaded and increases it when the system is underutilized. The paper uses simulations to show that the algorithm is free from the shortcomings of the current MBCAC algorithms.

1. INTRODUCTION

The connection admission control (CAC) is one of the most important functions in a computer network that provides a guaranteed or at least predictive quality of service (QoS). The goal of the CAC is to ensure that a newly established connection will receive the requested QoS while already established flows will retain the QoS they requested. Thus, a CAC algorithm running on a network node must know the amount of resources (link bandwidth, buffer space) that is unused by the already established connections. If these resources are greater or equal to those required by the candidate connection, it is allowed to be established on that node.

One of the most challenging tasks an efficient CAC algorithm faces in today's networking assumptions is to provide high resource utilization while maintaining the QoS. A number of CAC algorithms have been developed to achieve these goals [1, 2, 3, 4]. The common feature among these algorithms is the identification of the available bandwidth. The identification is done on the basis of either measurements of the network (node) resource utilization alone or of the measurements and some sort of a statistical model. The model is used to predict the overload probability given the utilization measurements and traffic parameters of the new flow. CAC algorithms that use measurements of the network traffic activity to estimate or predict the available bandwidth are called measurement-based CAC (MBCAC). [1] presents an MBCAC that uses a time window estimator to find the aggregate rate of the established flows. If the estimate plus the token rate of the candidate flow are less than the node's capacity the flow is admitted. [2], on the other hand, assumes that the aggregate traffic behavior can be described as a normally distributed random process and that the number of flows currently served by the system (node) is known. The MBCAC then uses values of the measured mean, variance of the aggregate traffic, and the number of present flows to derive the probability of the aggregate traffic exceeding the link capacity. The authors of [3, 4] present a family of MBCAC algorithms that use tangents to the effective bandwidth curve to identify the region in which QoS commitments are not violated. The effective bandwidth for a given QoS violation probability is found via the Chernoff bound.

A comparative study of most types of MBCAC algorithms was recently presented in [5]. The authors established that performance frontiers of MBCAC algorithms of various types are nearly the same. The study also showed that most of the algorithms do not have one explicit parameter that is responsible for the loss rate performance and that those algorithms that do have such a parameter miss the loss target by several orders of magnitude.

An important aspect of a practical CAC algorithm is that it must rely on the information that is available within the QoS framework of the computer network in general and the Internet in particular. It should be noted that this information is limited. The limitation is especially valid assuming that the Differential Services (DiffServ) framework [6] will most likely dominate the Internet in the near future. The framework does not provide the network with traffic descriptors of the flows or with the loss and delay specification of the QoS that the flows demand. Instead, the framework allows individual flows to inform the network about the type of QoS they wish to receive. The QoS type is defined as Per Hop Behavior (PHB), which denotes the externally observed QoS that the particular flow receives. Thus, a CAC algorithm operating in a DiffServ network does not know the traffic parameters of the candidate flows. Neither does the algorithm know the exact QoS parameters the traffic wishes to observe. And, just like any other CAC algorithm, this one must have an explicit tuning parameter that regulates its performance in terms of the loss rate.

Elek et al. recently presented in [7] an end-to-end measurement-based approach to provide a predictable loss rate on the entire path for individual flows. The sender node probes the path by sending a sequence of packets at the peak rate to the destination node. The receiver measures the loss among the probe packets and reports the loss to the sender. If the loss is below a certain target level, the sender starts to send the data packets to the receiver. This end-to-end, probing-based CAC has an advantage over per-hop MBCAC algorithms in that it does not require the network nodes to make the admission decisions. This agrees well with the DiffServ architecture. The probing requires awareness in the end node network applications, however, which can be an obstacle to wide-scale deployment. Furthermore, the question of the unresponsive end-node behavior, similar to that in the TCP and UDP flows, has not yet been fully addressed.

The goal of this paper is to give a demonstration of a performance-driven MBCAC algorithm that is free from the drawbacks of the current MBCAC schemes. The CAC algorithm differs from the other algorithms in the ways it estimates resource availability in the system. Instead of estimating the bandwidth available for the new flows, the algorithm attempts to control the number of flows in the system given the target loss rate of the aggregate traffic. This principle is based on the assumption that there is an upper bound on the number of flows a system can serve given the target performance. Thus, by controlling the number of flows in the system in response to the measured performance, it should be possible to achieve the target level of performance. The algorithm presented here does not require signaling of traffic parameters of individual flows, unlike the other MBCAC methods. The only signaling required is that of new flow arrivals. The algorithm attempts to control the number of flows in the system by enforcing an average flow inter-admission delay. The value of the delay is adjusted as the system's performance deviates from the target.

The paper is organized as follows. Section 2 presents the general structure of the algorithm. Section 3 gives the results of the simulations of the algorithm and a discussion. The control plane considerations of the algorithm are described in section 4, and section 5 contains conclusions and future work.

2. THE GENERAL STRUCTURE OF THE ALGORITHM

The CAC algorithm proposed here is assumed to control the performance of a finite queue system by adjusting the enforced average flow inter-admission delay given the target level of the packet loss rate. In this paper the loss rate is assumed to be the measure of the QoS. The enforcement of the delay in turn allows controlling the number of flows in the system. The adjustment of the delay is based on the measured performance of the system. Thus, the algorithm is performance-driven. In particular, when the real performance of the system is poorer than the target, the number of flows is considered to be too high and the enforced delay value is adjusted to reduce their population in the system. Similarly, if the system is under-utilized while flows are being rejected, the enforced delay value is adjusted to allow a greater number of flows in the system.

The algorithm has three major blocks: an admission block, a performance measurement block, and an enforced average inter-admission delay adjustment block. Figure 1 shows the relations among the blocks of the algorithm and the system. The solid lines represent the aggregate traffic flow and the dashed lines indicate the measurement and the control actions explained below. Following are detailed descriptions of each of the blocks.

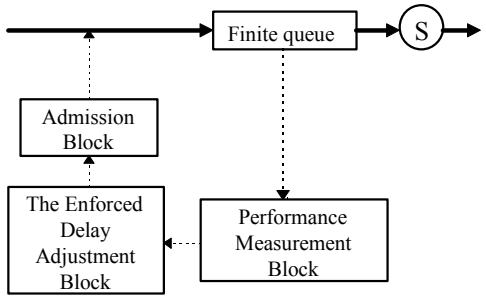

Figure 1. Relations among the blocks of the algorithm and the system

2.1 Admission block

The admission block is responsible for admitting new flows on the basis of the enforced average flow inter-admission delay (also denoted Del). To implement this enforcement the

block uses a counter of admissible flows, denoted C, which is incremented by one every Del seconds. When the enforcement is in use the operations on the counter are as follows: when a new flow arrives and the counter value is non-zero, the flow is admitted and the counter is reduced by one. If the counter is zero, the flow is rejected (Figure 2).

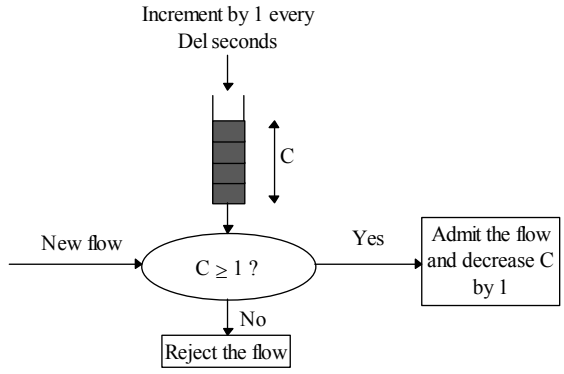

Figure 2. Actions of the admission block

```
Initialization (done once)
LossQuota = LinkRate*TL*W_time/PacketSize;
ThisLossQuota = 0;
Time = 0;
Initialization complete

if(packet loss){
  ThisLossQuota ++;
    if(ThisLossQuota > LossQuota){
      ThisLossQuota = 0;
      W_work = 0;
      Time = 0;
      C = 0;
      indication("violation") ;
    }
}

if(Time == W_time){
  WLossQuota = W_work*TL;
  if(ThisLossQuota <= WLossQuota)
    indication("underutilization");
  else{
    C = 0;
    indication("violation");
  }
  ThisLossQuota = 0;
  W_work = 0;
  Time = 0;
}
```

Figure 3. Pseudo-code of the performance measurement block.

2.2 Performance measurement block

The performance measurement block monitors the actual loss rate caused by the current value of Del. The monitoring is done using a measurement window of duration W_time seconds. Given the total number of packets arrived over a single measurement window, W_work, the number of packets that can be lost without violating the target loss rate, TL, is WLossQuota = W_work*TL. This quota of packets that can be lost can be found only in the end of the measurement interval. But a heavy overload can exceed this quota much ahead of the completion of the measurement window. To account for such cases the algorithm uses another quota of lost packets, LossQuota = LinkRate*TLR*W_time/PacketSize. This quota assumes that the incoming traffic rate is equal to or greater than the link rate, which is reasonable in case of an overload. Figure 3 contains a pseudo-code for the measurement block.

In the code ThisLossQuota is an accumulator of lost packets in the current measurement window. The algorithm uses LossQuota to identify a violation of the QoS whenever a packet is lost, and WLossQuota to identify a QoS violation in the end of a measurement window. Maintenance of W_work is not shown for simplicity. Time is a timer used to check boundaries of the measurement window. indication is an instruction that issues a "violation" or "underutilization" indication to the delay adjustment block. If any of the quotas is exceeded the block issues a "violation" indication to the delay adjustment block. In this case the measurement process is restarted and the counter of admissible flows C is set to zero. This is done under the assumption that the previous history of the system performance is irrelevant to the new delay value and the high loss that occurred in the past can trigger a false violation alarm.

2.3 Delay adjustment block

The delay adjustment block acts on indications received from the performance measurement block. A "violation" indication means that flows are admitted too often and the value of Del must be increased. An "underutilization" indication means that no loss rate violation has been experienced within the last measurement window and it is allowed to reduce the value of Del to admit more flows. The way in which the delay value is changed upon the indications from the measurement block defines performance of the system. Figure 4 shows a pseudo-code for the delay adjustment block. Let us describe handling of the "violation" indication first.

If it is the very first "violation" indication, the flow inter-admission delay is initialized to AD, the measured inter-arrival delay of candidate flows. The lowest margin of Del, Del_{MIN}, is set to twice the value of AD. In case where this "violation" indication is not the first one and the previous indication was "underutilization", Del_{MIN} is set to the current value of Del. This is because the current delay value is too low to prevent target loss rate violations, and when Del is adjusted later on, it must not be reduced lower than this value again. The next step is to increase the value of Del. The increase of Del is AD/GF, where AD is as before and GF is the so-called growth factor of Del. Thus, the measured inter-arrival delay of candidate flows obtained before the very first violation is also used to increase the inter-admission delay.

After Del and Del_{MIN} have been set, the algorithm calculates the step of reduction of Del, DelReduction, which is used when the adjustment block receives an "underutilization" indication. The reduction step is calculated as a fraction of the difference between the updated value of Del and the lowest margin Del_{MIN}. The reduction factor RF defines the value of the fraction.

When an "underutilization" indication is received, the algorithm lowers the value of Del by the previously calculated value of the reduction step. The max statement ensures that the delay is not reduced below the lowest margin. If the constant ExpReduction is true, the reduction step is recalculated as the RF fraction of the difference between Del and Del_{MIN}. In this case, subsequent "underutilization" indications result into exponential reduction of the value of the inter-admission delay since the last violation of QoS. If ExpReduction is false, the reduction of Del is linear.

Let us come back to the point where the lowest margin of the delay is set. As it was mentioned earlier, Del_{MIN} is set to Del if the previous indication was "underutilization". A heavy overload may require that the number of flows in progress at the moment of the resulting violation be reduced so that the overload is eliminated. It is likely that a single increase in the value of Del is not enough to eliminate the overload. Thus, the performance measurement block may issue a number of subsequent "violation" indications. The resulting increase in the value of Del serves not as an estimate of the optimal inter-admission delay, but as a drastic measure to quickly drop the number of flows in the system. Therefore, setting Del_{MIN} to any of the values of Del in this sequence would lead to an over-conservative lowest margin of the delay.

```
if(indication == "violation"){
  if(first "violation"){
    DelMIN = AD;
    Del = AD*2;
  }
  else{
    if(previous indication == "underutilization")
      DelMIN=Del;

    Del = Del + AD/GF;
  }
  DelReduction = (Del - DelMIN)/RF
}

if(indication == "underutilization"){
  Del = max(DelMIN, Del - DelReduction)
  if(ExpReduction)
    DelReduction = (Del - DelMIN)/RF;
}
```

Figure 4. Pseudo-code of the delay adjustment block

3. SIMULATION SETUP, RESULTS, AND DISCUSSION

3.1 Simulation Setup

The goal of the simulation is twofold. First, it is necessary to identify what impact different values of the algorithm's parameters have on the performance of the algorithm. Second, the performance of the algorithm must be compared with the performance of other connection admission control algorithms.

Some traffic models used in the simulations are chosen because they are similar to those used in other work on admission control algorithms. EXP1 model is an ON-OFF traffic source with exponentially distributed ON and OFF periods. The average duration of the ON periods is 312 msec and is 325 msec for the OFF periods. This model was used in [5] to compare various measurement-based CAC algorithms. The source generates 64 packets per second during each ON period. All the packets are 1 Kb long. The POO1 traffic model is the same as EXP1, but the ON and OFF periods are Pareto distributed. A traffic model similar to POO1 represents the toughest challenge for measurement-based CAC algorithms in terms of matching the target and actual performance [5]. For heterogeneous traffic scenarios, the POO1 model was modified to generate data at 128 Kbit/sec and 1 Mbit/sec rates. In the latter case, the ON intervals were 50 msec on average. All the flows generated have exponentially distributed lifetimes with the average being 300 seconds. The flow inter-arrival delay is exponentially distributed with 400 msec average.

All the simulations were carried out in a single hop environment where the network node represents a 10 Mbit/sec server and a 160-packet long, tail-drop queue. Each set of simulations was repeated with ten random seeds.

3.2 Performance of the algorithm versus the values of its parameters

The algorithm has the following parameters:

Length of the measurement window, W_time;
Target loss rate, TL;
Del growth factor, GF;
Del reduction factor, RF.

In addition, the algorithm can employ different ways of updating the value of Del. For example, the delay can be increased and/or decreased in an exponential or linear fashion.

Before presenting particular simulation results, we will discuss what can be an appropriate value of W_time. The measurement window is used to detect violations of the target performance and underutilization of the system. Therefore, the window must be long enough to capture changes caused by the growth or decrease in the number of flows. However, a

value of W_time that is too long forces the algorithm to underutilize the system when Del is too high, and a too short measurement window may take burstiness of the flows for a performance trend presumably produced by the changing number of flows.

To summarize, the length of W_time must be chosen as following: W_time = max(cell-level dynamics, min(Del, flow inter-departure delay)). The minimum is needed to avoid underutilization when no flows are admitted. Only one out of three components that must be used to find W_time is known.

W_time (seconds)	GF	RF	Loss target
10	1	2	10^{-6}
30	2	10	10^{-4}
100	4	20	10^{-2}
300	5	50	

Table 1. Values of some parameters in the algorithm

While finding the flow inter-departure delay can be facilitated by SIP or H.323 signaling (see the section on the control plane of the algorithm), discovery of the shortest interval that captures burstiness of the flows is not an easy task.

Target loss rate	Actual loss rate	Actual utilization
10^{-2}	$0.48*10^{-2}$; $0.65*10^{-2}$	0.910; 0.930
10^{-4}	$0.92*10^{-4}$; $1.2*10^{-4}$	0.853; 0.863
10^{-6}	$1.5*10^{-5}$; $2.4*10^{-5}$	0.841; 0.852

Table 2. Actual system performance versus target loss rate

This question is left for further studies. The values of W_time used in the simulations are shown in Table 1 together with values of other parameters of the algorithm. The effect of the target loss rate on the performance of the algorithm is first studied. In this set of simulations, the measurement window is 300 seconds, GF is 1 and RF is 2. The exponential reduction of Del is used.

The performance metrics are the actual utilization and loss rate. Table 2 shows results of the simulations. Each value is shown as a 95% confidence interval. The results in the table indicate that the target loss rate value has a direct impact on the actual loss rate. This parameter defines the number of packets that can be lost within a measurement window. As a consequence, the algorithm signals a violation at different population levels in the system. Figures 5 and 6 show this effect on examples of some of the simulation runs for target loss rates set to 10^{-2} (Figure 5) and 10^{-6} (Figure 6). In these and the other examples of the simulation runs, the top line is the number of flows in the system, the step line is Del in 10 msec units, and the spikes are the number of lost packets.

The impact of different values of W_time, GF, and RF is explored next. In this case, both linear and exponential reduction methods were used for the delay. The target loss level was set to 10^{-4}. In the exponential reduction GF and RF pairs chosen were (1, 2), and (4, 10).

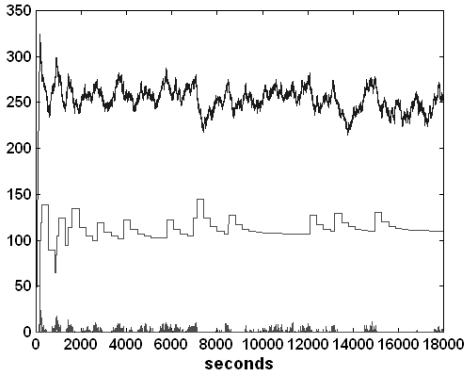

Figure 5. Target loss rate = 10^{-2}

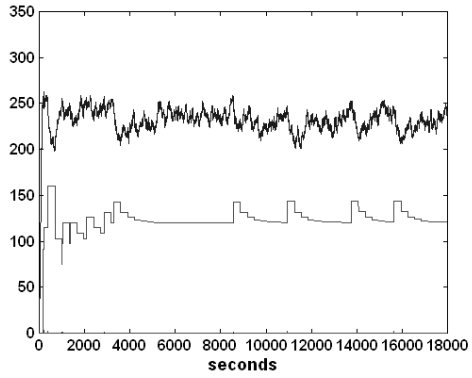

Figure 6. Target loss rate = 10^{-6}

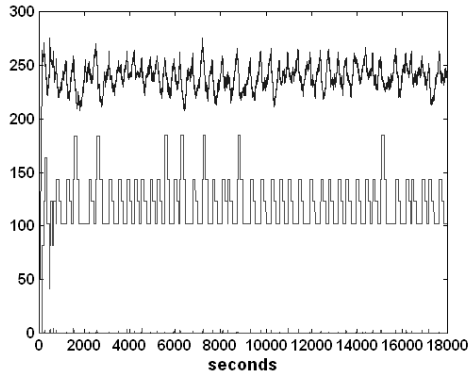

Figure 7. Linear reduction, GF = 1, RF = 2, W_time = 100 seconds

The pairs in the linear reduction were set to (1, 2) and (4, 20). The highest utilization (0.88) and loss rate ($1.4*10^{-4}$) were achieved with the linear reduction where GF was set to 1, RF was set to 2, and W_time was set to 100 seconds. The lowest utilization and loss rate (0.83 and $0.6*10^{-4}$, respectively) were observed in the case of the exponential reduction with GF = 4, RF = 20, and W_time = 300 seconds.

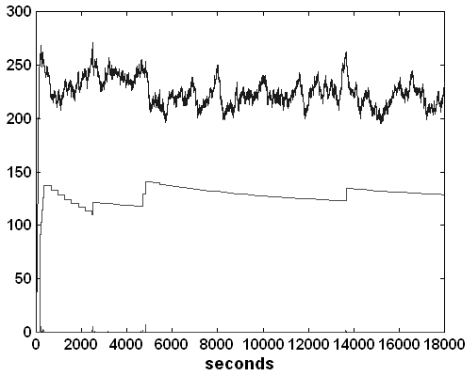

Figure 8. Exponential reduction, GF = 4, RF = 20, W_time = 300 seconds

Figures 7 and 8 give examples of simulation runs for each of the two extremes.

In general, the following patterns can be identified through this set of simulations. With GF = 1 and RF = 2 utilization and loss are lowest with W_time equal to 10 and 300 seconds and highest with W_time set to 30 and 100. This is true both for linear and exponential reductions. The difference between the utilization values is about 1%. This behavior can be explained by the following factors. When W_time is 10 seconds, the number of packets that can be lost before the violation is triggered is the smallest as compared with the other cases. Thus, the algorithm starts to react at a smaller flow population size. With W_time = 300 seconds, it takes longer to move from an excessive value of Del to a lower one that allows more flows to be admitted. This leads to long periods of underutilization, as is seen in Figure 8.

In simulations with higher GF and RF values, the difference between the utilization and loss rate performance is less significant. There is a slight increase in loss and utilization as the measurement window becomes longer. As before, this is caused by the increasing number of packets that can be lost before the violation is signaled.

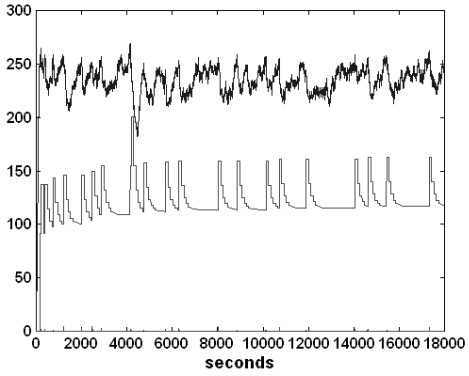

Figure 9. Exponential reduction, GF = 1, RF = 2, W_time = 100 seconds

The way in which the Del value is reduced also has its impact on the algorithm's performance. Figure 9 shows the same case as is shown in Figure 7 but with the exponential reduction. Since Del is reduced here at a finer granularity than in the linear reduction case, the algorithm can better match the estimate of the inter-admission delay against the optimal value. This reduces the thrashing of Del, which can be seen in Figure 7. A reduction in thrashing in turn reduces the number of violations and consequently the loss rate. So, the loss rate of the simulation run depicted in Figure 9 is $0.53*10^{-4}$ (as opposed to $1.4*10^{-4}$ in the linear reduction case).

Tuning performance of the algorithm in term of its parameters

The results given in Table 2 were achieved by varying the target loss rate. They indicate that this algorithm parameter has the most pronounced effect on the actual performance. It can be argued that the range of values used for the target loss rate is the widest in comparison with the values of the other parameters. Hence, the influence on the performance. While this is generally true, such values would not reflect the possible real life values of these parameters.

The requirement in choosing GF and RF values is that RF must be several times larger than GF, while GF should be more than 1. The latter recommendation helps avoid overshooting of the delay value when it is near the optimal level. A gradual reduction in Del provided by relatively large RF sets the violation instances further apart. This is especially valid when Del has reached a near-optimal level while the granularity of steps at which Del is changed does not allow a closer match with the optimal value. The simulations have shown that small growth steps in Del cope well with reaching a proximity to the optimal Del value. Special care must be taken when the exponential reduction is used. In this case, too high RF means a reduction with long tail, which may lead to underutilization.

The choice of W_time affects the speed of Del convergence to the optimal value and also affects the actual loss rate. It should be noted that the convergence of Del takes place only once after the flow aggregate controlled by the algorithm has been initiated. If Del has succeeded in converging to a near optimal value during, say, the first busy hour period, the next busy hour will be served without the convergence phase.

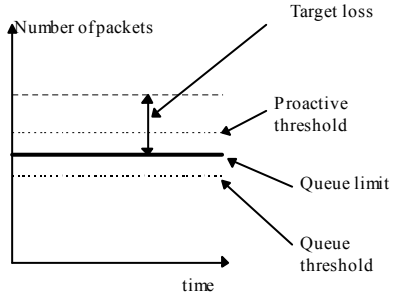

Figure 10. Queue threshold concept

Achieving tight loss rate targets requires a special care. In this case, it takes fewer packet losses to cross the violation threshold, and thus less time. To avoid the problem of the tight loss rate targets, it may be reasonable to introduce a

proactive threshold for the queue occupancy. Combined with the detection of the lost packets, the threshold would then trigger the adjustment block actions. This idea is illustrated in Figure 10.

Comparison with the measured-sum measurement-based connection admission control algorithm

Results reported in [5] demonstrate that there is a rather insignificant difference between various types of measurement-based CAC algorithms. Hence, the performance of the algorithm presented here is compared with only one of these algorithms.

The measured-sum algorithm was chosen for this purpose [1]. The algorithm has two measurement windows of T and S packet transmission times. The S window is used to measure the link load. S is smaller than T. At the end of each T window, the algorithm selects the maximum load value (ν) and the maximum observed delay (D). These values are used in admission decisions in the next T window. If a higher delay or utilization value is observed during the next T window, this value replaces the one set at the beginning of the window. The admission decision is based on the target utilization (νt) and delay (Dt) values. If the sum of $\nu + r$, where r is the peak rate of the candidate flow, is more than νt or $D + d > Dt$ (d is the delay incurred by the candidate flow), the candidate flow is rejected. This paper does not use leaky buckets to shape the flows, and thus d is set to $1024/(\mu - \nu)$. μ stands for the link capacity. If a candidate flow is admitted, ν and D are updated with the values of the sums in the admission decision inequalities.

The MBCAC algorithm is applied to the same node model as the enforced flow inter-admission delay algorithm. The size of the T window is varied from 500 to 3000 packet slots while the S window is kept 100 packet slots. Dt is set to 16 msec and νt is 1.

The enforced inter-admission delay algorithm is simulated using the following parameters: GF = 5, RF = 50, W_time = 100 seconds, linear reduction.

Loss versus utilization performance

In this set of simulations the EXP1 traffic model was used for both algorithms in order to make a comparison similar to that made in [5]. Figure 11 contains the loss versus utilization curves for the measured-sum MBCAC algorithm (dashed line) and the enforced inter-admission delay algorithm (solid line).

It can easily be seen that the measured sum algorithm has about a 6 % higher utilization at the same loss rate than the other algorithm. This is because of the following two factors. The first factor is the back-off done by the enforced inter-admission delay algorithm when the value of Del is increased. The second factor is the difference between the average inter-admission delay value and its instantaneous optimal value, which creates additional variations in the number of flows, seen in Figure 8.

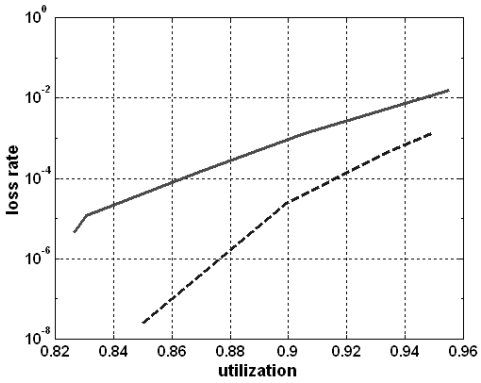

Figure 11. Loss rate versus utilization comparison

Figures 12 and 13 give examples of the actual loss rate on a one-minute time scale for the two algorithms. The loss rate over the entire simulation run is about 10^{-3} for both cases. It can be seen that the measured-sum algorithm provides a more uniform loss rate pattern than the enforced flow inter-admission algorithm.

It should be noted that the measured-sum MBCAC algorithm requires that the network and the end nodes support RSVP and thus has much more information than the inter-admission delay algorithm that requires neither RSVP nor any other QoS protocol or framework. Therefore, the somewhat lower performance of the inter-admission delay algorithm is traded for independence from the QoS architecture of the network.

Performance in heterogeneous flow scenarios

Three flow models were used in the simulations presented here. One is the POO1 model and the two others are also ON-OFF sources with Pareto-distributed ON and OFF periods but with 128 Kbit/sec and 1 Mbit/sec peak rates, respectively. The average duration of the ON periods in the 1 Mbit/sec model is set to 50 msec to reduce the average rate of these flows and allow a reasonable statistical gain on the 10 Mbit/sec link.

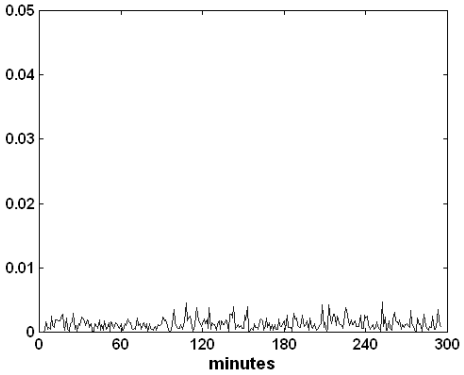

Figure 12. One-minute loss rate samples for the measured sum algorithm

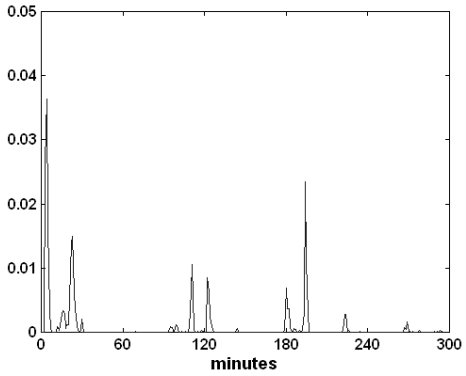

Figure 13. One-minute loss rate samples for the enforced inter-admission delay algorithm

The simulations show that the enforced inter-admission delay algorithm does not favor smaller flows over flows with a higher peak rate, as does the measured-sum MBCAC algorithm.

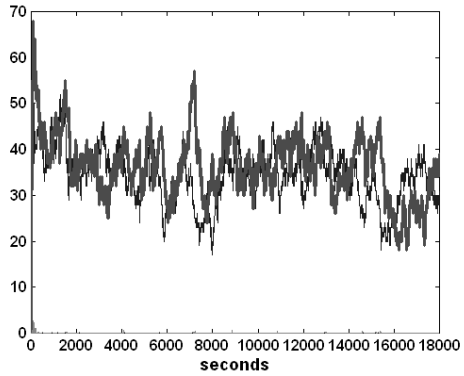

Figure 14. Enforced inter-admission delay algorithm, POO1 and 1 Mbit/sec flows, 50/50 mix

Figures 14 and 15 show some simulation runs with POO1 and 1 Mbit/sec flow mixes for each of the algorithms. Admission requests for each of the flow types are generated with equal probability. The number of 1 Mbit/sec flows is shown as a bold line. The measured-sum algorithm shows similar biased behavior even when the difference between the peak rates of the candidate flows is not very significant. Figure 16 shows a simulation run for the measured-sum algorithm with POO1 and the 128 Kbit/sec flow mix. Again, the algorithm heavily discriminates against the faster flows in favor of the slower ones. This is because the algorithm, like many other MBCAC algorithms, bases the admission decision on the traffic descriptor of the candidate flow. It is more likely that a smaller bandwidth is available when a candidate flow arrives, smaller flows are thus more frequently admitted because they are less demanding. On the other hand, the enforced inter-admission delay algorithm admits flows with various patterns as often as they appear among the candidate flows. As additional evidence of the unbiased behavior of the algorithm, Figure 17 shows the number of admitted flows when the 1 Mbit/sec flow requests are generated with probability 0.7. The average number of POO1 flows in this case is 17, and the average number of the 1 Mbit/sec flows is 40. This fair treatment of flows with different peak rates is observed in all the simulations of the heterogeneous scenarios of the algorithm.

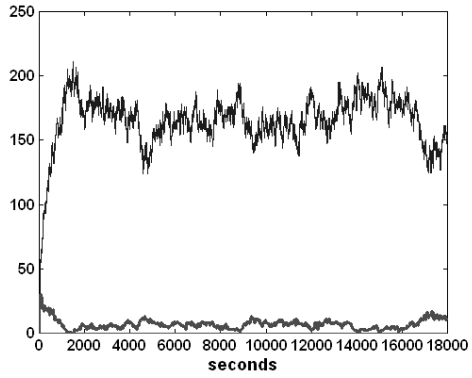

Figure 15. Measured sum MBCAC algorithm, POO1 and 1 Mbit/sec flows, 50/50 mix

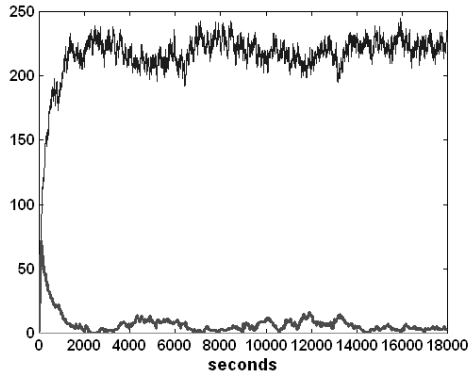

Figure 16. Measured sum MBCAC algorithm, POO1 and 128 Kbit/sec flows, 50/50 mix

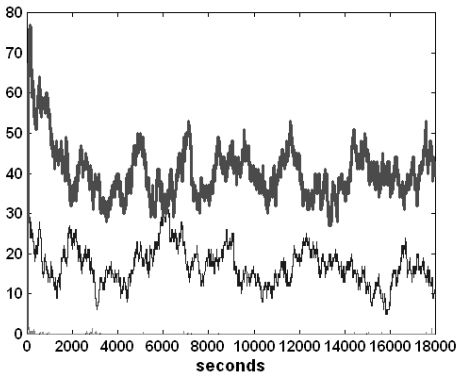

Figure 17. Enforced inter-admission delay algorithm, POO1 and 1 Mbit/sec flows, 30/70 mix

4. CONTROL PLANE OF THE ALGORITHM

In general, from the moment a candidate flow is admitted on a particular link or traffic aggregate, the network node must distinguish this flow from those that did not gain admission. The differentiation can be done in several ways depending on the protocol environment assumed.

Frameworks such as IntServ and ATM assume that the end nodes comply with control messages from the network. For example, if a call setup has failed, the network returns a "tear-down" or a "setup failure" message to the source end node. In this case the end node refrains from sending the flow or attempts to re-establish the connection. In this way the network nodes explicitly know which flows or calls are admitted. If a network node observes a successful exchange of flow setup messages, the node allows the flow through. Otherwise, if the flow setup fails, the node deletes any partial state associated with the flow.

The algorithm described in this paper does not require presence of ATM or RSVP signaling messages and can operate without these or any other QoS frameworks. To enforce the admission decision, a network node would need to filter on header fields of arriving packets to identify whether they belong to a new or an already established flow. The filtering rule can be based, for example, on the source-destination IP address pair, IPv6 flow identifier, or MPLS label. The latter is possible only if the labels of individual flows are not merged upstream. Thus, the network node keeps a table of flow IDs that contains identifiers of admitted flows. Packets with header field values that match any of the table entries are forwarded, while the other packets are dropped. Figure 18 shows the processing required to identify the new flows and connection of the processing with the admission decision. Besides being added to the table, flows that are admitted must be removed from it when they terminate. Since no tear-down messages are assumed, the only possibility to remove a flow from the table is based on the activity of the flow. In other words, a timer is used that is updated whenever a packet of the flow is observed. If the timer expires, the flow is considered terminated and is removed from the table.

Another way to maintain the table of admitted flows is to use H.323 suite [9] or SIP [10] messages. SIP is an end-to-end signaling protocol used for establishing, maintaining, and tearing down end-to-end flows. These functions are also available in the H.323 protocol suite. In this case there is no need to detect new flows by comparing the header field values of each arriving packet with the table entries. Furthermore, the identification of flow termination does not require a timer on each table element. Figure 19 shows a diagram of actions for flow admission in this case.

We will now briefly discuss what can go wrong in the control plane. First, assume the case in which a downstream router rejects a flow. This means that the flow is admitted on only a part of the path to the destination. Ideally, the router that rejected the flow would send an upstream tear-down control message. The message would instruct the upstream routers to remove the flow ID from the tables. It is easy to identify two problems here, the stemming from the possibility that the upstream and downstream routes may not be symmetric. The second problem is related to the message getting lost. Together this means that the control message does not reach some or all of the network nodes. There are two possible outcomes of this situation. The one assumes that the rejection of the flow means dropping the flow setup message. Consequently, the end-to-end flow setup fails and the compliant end nodes do not initiate the flow itself. Alternatively, in the case of the absence of end-to-end control messages, the end nodes observe no connectivity. In both cases, no further packets appear in this flow and the upstream nodes remove the flow ID after the inactivity time-out. On the other hand, an end node that starts sending data without waiting for the flow to be admitted on the entire path to the destination may still generate the flow despite an incomplete setup or the absence of the receiver response.

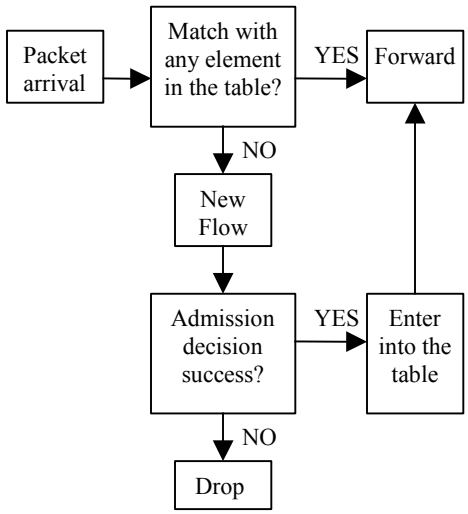

Figure 18. Actions taken per packet in absence of flow set-up signaling

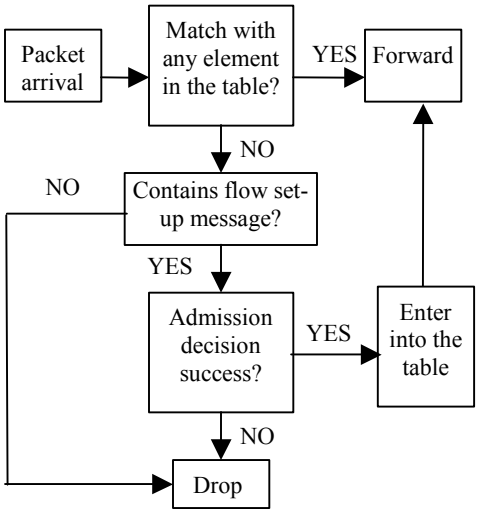

Figure 19. Actions taken per packet in the presence of flow set-up signaling

The flow ID will not be timed out in this case and the flow may eventually be admitted on the entire path to the destination. The down side of a persistent end node behavior of this kind is the wasted work that the upstream nodes spend on forwarding the flow while it is being discarded downstream.

5. CONCLUSIONS AND FURTHER WORK

To summarize, this paper showed what kind of QoS guarantees in terms of packet loss rate can be achieved in a network that does not implement any QoS framework by using a per-hop MBCAC algorithm. The algorithm bases admission decisions on the performance-adjusted average flow inter-admission delay. This feature makes the algorithm different from the Measurement-Based CAC (MBCAC) algorithms, which make their admission decision on the basis of the estimated or predicted resources available and the traffic parameters of the candidate flows. This difference makes the algorithm presented here independent of the QoS architecture of the Internet because it neither requires signaling of the traffic parameters of the individual flows nor it demands any additional signaling over the one that exists today. The only signaling the algorithm needs is the arrival of the new flows, which is basic for any connection admission control. The simulation results reported demonstrate that the algorithm performs well in the heavy-tailed traffic environment. The simulations showed that the actual performance of the algorithm in terms of the loss rate is easily tuned by changing the target loss rate parameter. A comparative study was made of the enforced inter-admission delay algorithm and the measured-sum MBCAC algorithm. This showed that the enforced inter-admission algorithm does not achieve as high a utilization as the simulated MBCAC algorithm. At the same time, the inter-admission algorithm does not require as much information as is needed in the MBCAC algorithm. The inter-admission delay algorithm also showed an unbiased behavior in the heterogeneous flow scenarios, because it does not use the traffic parameters of the candidate flows in its admission decisions. Still, unawareness of the traffic parameters of the candidate flows has its shortcomings. Consider a case where the aggregate flow mix is heterogeneous and proportions of flows with different rates are not constant. In this case, the algorithm will adjust the value of Del to the traffic mix with the lowest number of flows. At the same time, the lowest margin of the delay, Del_{MIN}, will not let the algorithm adapt to a mix with smaller flows. Similarly, if the system is highly utilized and the algorithm admits a flow with rate that exceeds the spare bandwidth, a loss rate violation will follow, forcing the algorithm to re-adjust the value of Del. Despite the fact that the algorithm maintains a per-flow state, the traffic parameter unawareness prohibits the use of traffic shapers and packet schedulers together with the algorithm. For example, in case of a heterogeneous traffic mix, a packet scheduler would distribute the link capacity evenly among the flows, which is wrong, since different flows have different rates. However, the question of traffic policing is not the same as the question of the admission control. Considering a network domain, the policing could be done at the edge of the domain while the core would be unaware of the traffic parameters of the individual flows. Combining the admission algorithm with a traffic policing algorithm as well as extending the algorithm to support traffic descriptors is a subject of future research.

It was also found that the algorithm did not succeed in exactly matching the 10^{-6} target loss rate. Future work will include an implementation of the mechanism described in the discussion of the simulation results to achieve a better match between the actual and the target loss rates in the cases of tight target loss rate values. Another question is an integration of the buffer and the link utilization measurements with those of the actual loss rate to enable a faster discovery of the target loss violation and violation elimination instants. In addition, it would be interesting to investigate whether the algorithm can be tuned with respect to the time scale of the target loss rate definition.

In its present state the algorithm requires manual configuration. Making the algorithm more viable in a real life situation requires self-tuning of such parameters as the measurement window size and the steps of adjustment of the delay.

Attention should also be given to the various effects of the control plane on the performance of the algorithm, particularly to how different flow ID timeout intervals affect the overall treatment of the flows.

6. ACKNOWLEDGEMENTS

The author thanks his colleagues Vishaka Nanayakkara, Doctor Olov Schelén at Luleå University of Technology, and Professor Gunnar Karlsson at The Royal Institute of Technology, Stockholm, for their helpful discussions and comments on the paper. The author would like to thank the anonymous reviewers whose helpful comments significantly improved readability of this paper.

7. REFERENCES

[1] Jamin S., Danzig P. B., Shenker S. J., Zhang L. A Measurement-Based Admission Control Algorithm for Integrated Service Packet Networks. IEEE/ACM Transactions on Networking, vol. 5, no. 1, February 1997.

[2] Grossglausser M., Tse D. N. C. A Framework For Robust Measurement Based Admission Control. IEEE/ACM Transactions on Networking, vol. 7, no. 3, June 1999.

[3] Gibbens R. J., Kelly F. Measurement-Based Connection Admission Control. 15th International Teletraffic Congress, June 1997.

[4] Gibbens, R. J., Kelly, F. P., Key, P. B. A Decision-Theoretic Approach to Call Admission Control in ATM Networks. IEEE/ACM Transactions on Networking, vol. 13, no. 6, August 1995.

[5] Breslau, L., Jamin, S., Shenker, S. Comments on the Performance of Measurement-Based Admission Control Algorithms. In Proceedings of INFOCOM 2000, vol. 3.

[6] Differential Services (DiffServ) Workgroup. http://www.ietf.org/html.charters/diffserv-charter.html

[7] Elek, V., Karlsson, G., Ronngren, R. Admission Control Based On End-To-End Measurements. In Proceedings of INFOCOM 2000, vol. 2.

[8] Multiprotocol Label Switching (MPLS) Workgroup. http://www.ietf.org/html.charters/mpls-charter.html

[9] H.323 - Framework and wire-protocol for multiplexed call signalling transport. http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-H.323

[10] Session Initiation Protocol (sip) Charter. http://www.ietf.org/html.charters/sip-charter.html