Admission Control for TCP Flows Using Packet Classes and Edge-to-edge Measurements of Aggregates

Lluís Fàbrega, Teodor Jové, Pere Vilà, José Marzo, Liliana Carrillo
Institute of Informatics and Applications (IIiA), University of Girona
Campus Montilivi, 17071 Girona, Spain

Abstract—**TCP flows from applications such as the web or ftp are well supported by a Guaranteed Minimum Throughput Service (GMTS), which provides a minimum network throughput to the flow and, if possible, an extra throughput. We propose a scheme for a GMTS using Admission Control (AC) that is able to provide different minimum throughput to different users and that is suitable for "standard" TCP flows. Moreover, we consider a multidomain scenario where the scheme is used in one of the domains, and we propose some mechanisms for the interconnection with neighbor domains. The whole scheme uses a small set of packet classes in a core-stateless network where each class has a different discarding priority in queues assigned to it. The AC method involves only edge nodes and uses a special probing packet flow (marked as the highest discarding priority class) that is sent continuously from ingress to egress through a path. The available throughput in the path is obtained at the egress using measurements of flow aggregates, and then it is sent back to the ingress. At the ingress each flow is detected using an implicit way and then it is admission controlled. If it is accepted, it receives the GMTS and its packets are marked as the lowest discarding priority classes; otherwise, it receives a best-effort service. The scheme is evaluated through simulation in a simple "bottleneck" topology using different traffic loads consisting of "standard" TCP flows that carry files of varying sizes. The results prove that the scheme guarantees the requested throughput to accepted flows and achieves a high utilization of resources.**

Keywords-guaranteed throughput; TCP; admission control

I. Introduction

Internet traffic is dominated by TCP connections carrying files generated by applications such as the web, ftp, or other [1]. The users of these applications expect no errors in the file transfer and the best possible response time. The source breaks up the file and sends a flow of packets at a certain sending rate, which the network delivers to the destination with some variable delays and some losses. Losses are recovered by TCP through retransmission, which adds more delay (increasing the transfer time) and may also cause the reception of duplicated packets (which are discarded by the destination). From the point of view of the network, the decisive Quality of Service parameter is the average receiving rate or network throughput (including the duplicates).

A basic feature of TCP flows is their elastic nature. Sources vary the sending rate (up to the capacity of the network input link) to match the maximum available throughput in the network path. Since the available throughput changes over time, TCP uses rate-adaptive algorithms that increase and decrease the sending rate in order to match these variations and minimize packet loss. TCP increases the sending rate if packets are correctly delivered and decreases it if packets are lost [2]. On the other hand, another important feature of TCP flows is the heavy tail behavior of the file size distribution observed in traffic measurements [1], that is, most elastic flows carry short files and a few flows carry very long files.

Although the traditional view is that elastic flows do not require a minimum throughput, unsatisfied users or high layer protocols impose a limit on the file transfer time. Aborted transfers imply a waste of network resources, which can be even worse if the user tries the transfer again [3]. Moreover, in commercial Internet, users will pay extra for a desired performance. Hence, there is a minimum throughput required or desired by users.

Elastic flows are well supported by a Guaranteed Minimum Throughput Service (GMTS), which provides a minimum throughput and, if possible, an extra throughput. There is an input traffic profile, based on some sending traffic parameters (average rate, peak rate, burst size, etc.), which defines the desired minimum throughput. If the average sending rate is within the profile, ("in") packets have a guaranteed (minimum) delivery, and otherwise ("out") packets have a possible (extra) delivery only if there are available resources.

The service delivery between the user and the provider is ruled by an agreement (Service Level Agreement or SLA). Since the Internet is made of multiple autonomous domains, the end-to-end service is the concatenation of the service provided by each of the domains in the path. Each domain uses its own way to provide the service and has a user-provider relationship with its neighbor domains according to an SLA.

We propose a scheme for a GMTS using Admission Control (AC) that is able to provide different minimum throughput to different users and that is suitable for "standard" TCP flows. The paper is organized as follows. In section II we review the related work. In section III we describe our scheme, including the AC and the interdomain operation. In section IV we evaluate the performance of the scheme through simulation using "standard" TCP flows in a "bottleneck" topology. Finally, we summarize the paper in section V.

II. Related Work

The traditional network service in the Internet is the best-effort service, which together with TCP rate-adaptive algorithms [2], aim to provide a fair throughput service. Another proposal is the Assured Service [4], defined in the Differentiated Services (Diffserv) architecture [5]. It is a proportional throughput service able to provide different throughputs to different users. However, in congestion (when users' demands exceed resources), both services cannot provide a desired

Work supported by Spanish Science &Technology Ministry (TIC2003-05567) and Generalitat of Catalonia's research support program (SGR 00296).

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

minimum throughput to any flow. Congestion could be avoided using resource overprovisioning, but this is a highly wasteful solution. If more efficient provisioning is used, congestion can be dealt with by AC. Its goal is to provide a desired minimum throughput to the maximum possible number of flows in congestion.

A classical distributed AC method with hop-by-hop decision, based on per-flow state and signaling in the core routers, is not appropriate for elastic flows because of the low scalability and the high overhead of explicit flow signaling messages. Therefore other kinds of AC methods have been proposed [6][7][8][9]. They have the following in common: they avoid the use of per-flow signaling, per-flow state is reduced to the edge or is not needed, and they are based on measurements.

In the schemes proposed in [6][7] the provided throughput is the same for all accepted flows, where a flow is defined as a TCP connection. A single link (or a logical path with a long-term resource reservation) is considered, and its current load is estimated from measurements of the queue occupancy. The scheme does not require per-connection state.

In the scheme proposed in [8] the provided throughput is the same for all accepted flows, where a flow is defined as a sequence of related packets (from a single file transfer) within a TCP connection. A network path is considered, and two measurement methods are proposed to estimate the available throughput in a path: one measures the throughput of a phantom TCP flow, and the other measures the current loss rate of the path to convert it into a throughput estimate. Per-flow state is kept only at the edge, in a list of active flows updated using an implicit way, that is, the start of a flow is detected when the first packet is received, and its end is detected when no packet is received within some defined timeout interval.

Our alternative scheme for a GMTS in [9] is similar to [8] in that a flow is a sequence of related packets (from a single file transfer) within a TCP connection, a network path is considered and per-flow state is kept at the edge and updated using an implicit way. However, it is able to provide different throughput (to flows from different users) since traffic conditioning at the ingress differentiates between the "in" and "out" traffic of the flow. Moreover, it uses a small set of packet classes, specifically, four classes plus the best-effort class, where each one has assigned a different discarding priority. The AC is based on edge-to-edge per-flow throughput measurements. The first packets of the flow, before the AC decision is made, are marked at the ingress as one of the R classes (R_{IN}, R_{OUT}). From these packets, the throughput of the flow is measured at the egress. The measurement is sent to the ingress, where by comparing it with the requested minimum throughput, the AC decision is made. From now on, the flow's packets are marked as one of the A classes (A_{IN}, A_{OUT}) if it is accepted, or as the best-effort class if it is rejected. The discarding priorities are chosen so that accepted flows are protected against flows in the AC phase, which roughly means discarding packets from R classes before packets from A classes. However, the best performance of the scheme is achieved using a special modification in the sending rate algorithms of TCP, since the short-term rate fluctuations of a "standard" TCP source makes the performance get worse.

III. Our Proposed Scheme

The new scheme we propose here and the scheme in [9] have some common features: a flow is defined as a sequence of related packets (from a single file transfer) within a TCP connection; a network path is considered; per-flow state is only kept at the edge and updated using an implicit way; the ability to provide different throughput (to flows from different users) differentiating between the "in" and "out" traffic; the use of packet classes (each one with a different discarding priority); and the use of edge-to-edge measurements.

The main goal of the new scheme in comparison with [9] is to achieve a better performance for "standard" TCP flows. In [9] the throughput of each single flow is measured at the egress during a time period. For a "standard" TCP flow, the measurement duration must be long in order to average its short-term rate fluctuations and obtain a correct measurement. However, a long measurement duration implies that a lot of flows meet simultaneously during the AC phase and compete for the resources. The per-flow measurement reflects an equal sharing of the resources, and therefore, during congestion, the AC tends to reject too many flows and the resource utilization decreases.

In our new scheme we use per-aggregates measurements instead of per-flow measurements. Throughput measurements of flow aggregates will reduce the impact of the short-term fluctuations of TCP flows. The aggregated measurement at the egress is sent to the ingress where it is used to obtain an estimation of the available throughput in the path. Moreover, a desired consequence is that we eliminate the duration of the AC phase used in [9], that is, the AC decision is made at once when a new flow arrives at the ingress since the available throughput in a path is known in advance. Another difference between both schemes is that the number of classes here is three instead of four (plus the best-effort class).

A. The Assumptions of the Scheme

This is a list of the assumptions considered by the scheme:

- There is a multidomain scenario, where neighbor domains have a user-provider relationship, and our scheme is used in one of the domains.
- The delivery of services between a provider and a user is defined in a Service Level Agreement (SLA).
- A user may ask for the GMTS for each flow of an aggregation of flows to any destination and for any flow's throughput, as long as a contracted value is not exceeded.
- A flow is a sequence of related packets (from a single file transfer) in a TCP connection (not a complete TCP connection, which usually has periods of inactivity).
- The user indicates he is asking for the GMTS for a flow marking the flow packets with an agreed contracted value.
- In the agreement the user specifies the minimum throughput of flows, e.g., the same value for all flows, according to the application (ftp, web, etc.) or other.
- The network uses packet classes, such as the classes in Diffserv [5].

- The network uses pre-established logical paths from ingress points to egress points, such as the Label Switched Paths on MPLS networks [10], when they are used without a long-term resource reservation. Instead, once a flow is accepted by our AC, resources in the path are reserved during the flow's lifetime.
- A list of active flows is maintained at the edge using an implicit mechanism to detect the start and end of flows [8].

B. The Architecture of the Scheme

The scheme with the AC (Fig. 1) uses four packet classes: two A (Acceptance) classes, A_{IN} and A_{OUT}, the PR (Probing) class and the BE (Best-Effort) class. In the output links of all routers there is a FIFO queue with discarding priorities per class in the order (from low to high) A_{IN}, A_{OUT}, BE and PR.

When the first packet of a new flow arrives at the network, the flow is assigned to a logical path, *LP*, the list of active flows at the ingress is updated, and the desired minimum throughput, $R_{NEW,LP}$, is read from the user-provider agreement. The AC evaluates whether its minimum throughput requirement can be provided without losing the minimum throughput guaranteed to the accepted flows. The AC decision is made at once, since the available throughput in the path *LP*, $TH_{AV,LP}$, is known in advance at the ingress. The flow is accepted if

$$R_{NEW,LP} \leq TH_{AV,LP}\,, \qquad (1)$$

and otherwise is rejected. If it is accepted, it receives the GMTS, and its packets are marked as A, specifically the "in" packets as A_{IN} and the "out" packets as A_{OUT} depending on the comparison between the average sending rate and the desired minimum throughput. If the flow is rejected, it receives a best-effort service, and its packets are marked as BE.

In order to discover $TH_{AV,LP}$ we use a special probing flow that is sent continuously through the path *LP*. It is a Constant Bit Rate flow with packets marked as PR. The egress node measures the aggregated throughput of A_{OUT}, BE and PR classes (i.e., all classes except A_{IN}) of flows assigned to the path. The obtained measurement, M_{LP}, is an estimation of the available throughput in the logical path. This information is sent to the ingress where the AC decision is made. In subsection C we give details of how $TH_{AV,LP}$ is calculated from M_{LP}.

Note that the available (unreserved) throughput in a link is the portion of the link's bandwidth that is not used by the traffic A_{IN} of all the logical paths passing through this link. This unreserved bandwidth is used by the rest of traffic according to the discarding priorities of classes A_{OUT}, BE and PR (the lowest discarding priority class achieves the resources firstly, the next class the remaining ones, etc.), and proportionally to the input traffic load of each class and each path. This means that the available throughput is distributed among the paths that coincide in the link. The distribution is made in each link of the logical path until reaching the egress. Therefore the obtained measurement is an estimation of the available throughput for the new arriving flows assigned to the logical path. Note that if the aggregated load of classes A_{OUT}, BE and PR does not entirely fill the available throughput in the link, then the measurement would be pessimistic. However, this can be solved increasing the rate of the probing flow.

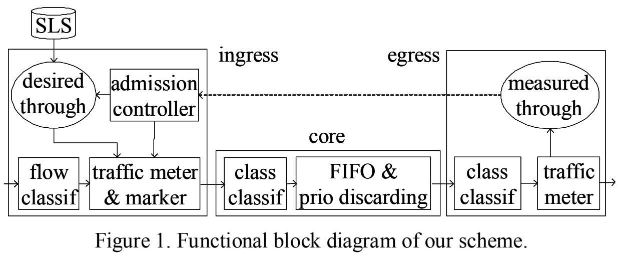

Figure 1. Functional block diagram of our scheme.

C. The Available Throughput in a Path

At the egress a class classifier selects the packets marked as A_{OUT}, BE and PR from a logical path *LP*. The measurements are made in time periods of duration T_{LP}. The measured aggregated throughput M_{LP} is simply the ratio between the total bytes of the received packets during the measurement period divided by T_{LP}. At the end of each measurement period a signaling packet is sent to the ingress with M_{LP}.

The measurement M_{LP} is not directly the value of $TH_{AV,LP}$. Once a flow is accepted, the measurement does not immediately take into account the resulting decrease in the available throughput. Some time is needed to obtain an updated measurement at the egress, and to carry the updated measurement to the ingress. Specifically, in a given time *t*, where M_{LP} is the last measurement of a path *LP* received at the ingress at time t_I ($t_I \leq t < t_I+T_{LP}$), we consider that the following accepted flows are not taken into account in M_{LP} (Fig. 2):

- Accepted flows in *LP* during the time interval $[t_I\text{-}rtt_{IE}\text{-}T_{LP},\ t_I\text{-}rtt_{IE}]$, where rtt_{IE} is the ingress-egress round trip-time, because they are measured in M_{LP} during too short a time.
- Accepted flows in *LP* during $[t_I\text{-}rtt_{IE},\ t]$, since they are not measured in M_{LP}.

Note that we just need to take into account the accepted flows in *LP* and not the ones accepted in other coincident logical paths, since M_{LP} is an estimation of the available throughput only for *LP*. Our approach is simply subtracting from M_{LP} the desired minimum throughput of the accepted flows in *LP* not taken into account in M_{LP}, and using the resulting value as the new $TH_{AV,LP}$ for future AC decisions. On the contrary, when the accepted flows end, we do not modify M_{LP} explicitly, but we allow the measurement to adapt to the traffic changes (a conservative approach). Summing up, the available throughput in the path *LP* in a time *t* is

$$TH_{AV,LP}(t) = M_{LP} - \sum_{i} R_{ACC,LP}(i)\,, \qquad (2)$$

where M_{LP} is the last received measurement, and $R_{ACC,LP}$ is the minimum throughput of accepted flows in $[t_I\text{-}rtt_{IE}\text{-}T_{LP},\ t]$.

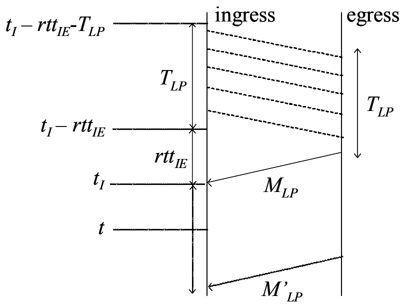

Figure 2. The measurement M_{LP} and the accepted flows.

D. The Interdomain Operation

In a multidomain scenario, the end-to-end service is provided by the concatenation of the service provided by each of the domains of the followed path. Each domain uses its own way to provide the GMTS to the flow (e.g., our scheme) and has a user-provider relationship with its neighbor domains according to an agreement. The interdomain operation refers to questions such as the indication of the required service, the identification of the flow that requests the service, the indication of service acceptance or rejection, or other. Fig. 3 shows a basic scenario with an upstream domain U, our domain O (the one using our scheme) and the downstream domain D. The question here is to determine the interconnection needs of our scheme when acting as a user or as a provider.

The interconnection aspects of our domain O with the upstream domain U (U acts as a user and O as a provider) have been already dealt with in subsection III.A. No per-flow signaling is needed: the start and end of flows is detected using an implicit way, a mark on the packets indicates the desired service, and the minimum throughput is specified in the agreement.

To study the interconnection needs of our domain O with the downstream domain D we consider the following situation (O acts as a user and D as a provider). Suppose that a flow is accepted in domain O but it is rejected in domain D, and O ignores it. The consequences would be two: firstly, domain O would keep a reservation for the flow that would be useless, wasting resources that might be used by other flows; secondly, domain O would wrongly consider that the flow does receive the service and that the agreement with the upstream domain U is being satisfied. For these reasons we propose the use of signaling packets to notify whether domain D can provide the GMTS to the flow or not. This information is used in domain O in the following way: if the answer is "yes", nothing else is done, but if the answer is "no", the egress forwards this signaling packet to the ingress, where the AC will reject this previously accepted flow. This last point requires the egress to have a list of active and accepted flows (updated the same way as the list at the ingress) together with its assigned logical path and corresponding ingress.

IV. Evaluation Of The Scheme Through Simulation

We have added the functional blocks of our scheme (Fig. 1) to the Diffserv module of the ns simulator [11]: at the ingress routers and for each arriving flow, a sending rate meter suitable for "standard" TCP flows" (based on the Time Sliding Window algorithm defined in [4]), a marker (with the four marks) and an admission controller; at the egress routers and for each path, a throughput meter for classes A_{OUT}, BE and PR, which counts the total received bytes during the measurement duration, and notifies the measurement to the ingress router (with the corresponding delay). In the output links of all routers there is a FIFO queue with priority discarding for the four packet classes, based on the drop-tail algorithm.

A. Description of the Simulations

In the simulations we use a "standard" TCP source, specifically TCP New-Reno. We generate TCP flows that carry a single file from an ingress point to an egress point through a logical path. Each flow is characterized by the file size and the starting time. File sizes are obtained from a Pareto distribution that approximates reasonably well the heavy-tail behavior of the file size distribution observed in measurements [1]. In all simulations the tail parameter is 1.1 and the minimum file size is 10 packets (the packet length is 1,000 bytes). With these parameters the distribution has an infinite variance. After generating the values (about 10,000 flows in each simulation) more than 50% are below 20 packets, the mean σ is about 74 packets and the maximum one reaches 19637 packets. On the other hand, the starting times are obtained from a Poisson arrival process, characterized by the average number of arrivals per second, λ flow/s. Therefore the average offered traffic load to a logical path is equal to $\lambda\sigma$ bps.

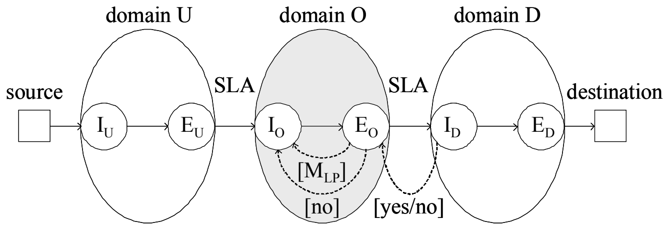

Figure 3. Interdomain operation of our domain with neighbor domains.

The simulations are performed in a "bottleneck" topology (Fig. 4), with two logical paths, e0-e3 and e2-e3. We generate flows for each path at an arrival rate of λ flow/s, the value of λ is the same for both paths. In order to study underloading and overloading situations in the "bottleneck" link of 2 Mbps, we vary the average offered traffic load of each path in the range of 0 to 3 Mbps (the average offered traffic load to the "bottleneck" link, $2\lambda\sigma$ bps, varies from 0 to 6 Mbps).

To evaluate the performance of the scheme we use three parameters that measure the utilization of resources as well as the throughput obtained by flows. We consider the satisfied flows, defined as the ones that complete the transfer and get at least 95% of the desired throughput (the throughput is calculated as the ratio between the total received packets divided by the flow's lifetime). Then we obtain, for each logical path, the following performance parameters: 1) the average "total" satisfied traffic load, which is the aggregated throughput of all satisfied flows, taking into account the minimum and the extra throughput; 2) the average "minimum" satisfied traffic load, which takes into account the minimum throughput; and 3) the average throughput of satisfied flows (note that it will always be a value above 95% of the desired throughput). The first parameter evaluates the total use of resources, the second one its reserved use, and the third one indicates how much extra throughput the satisfied flows get.

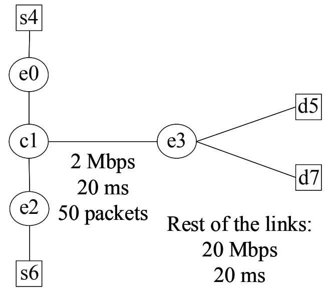

Figure 4. Network topology (showing the link's bandwidth and delay).

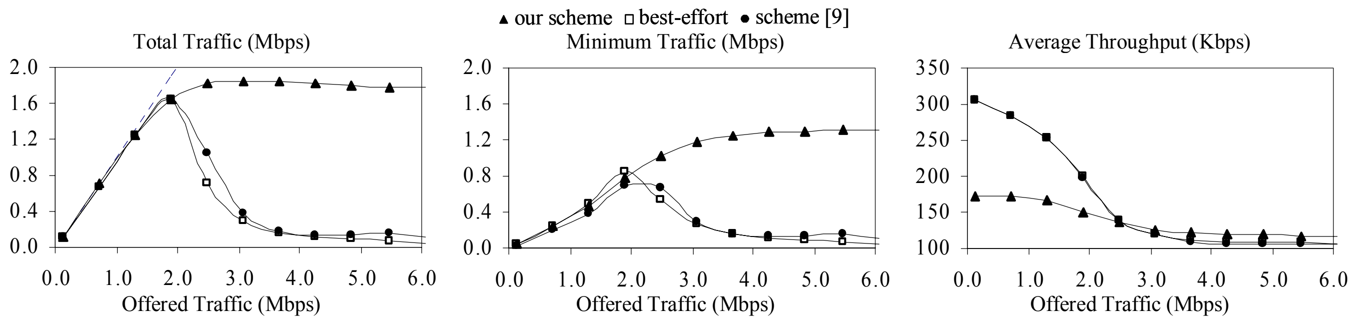

Figure 5. Comparison of our scheme with the scheme in [9] (in both $T_{03} = T_{23} = 0.5$ s) and with the best-effort service. The desired throughput is 90 Kbps.

The average values of satisfied traffic load are obtained by averaging over the simulation time, but without considering the initial period of the simulation transient phase. In all simulations a minimum of 10,000 flows are generated after this initial period. We make 10 independent replications of each simulation, and we estimate the mean value by computing the sample mean and the 95% confidence interval [12].

B. Simulation Results

In the first place, we compare our scheme with the scheme in [9] (in both, the measurement duration for the logical paths is the same, $T_{03} = T_{23} = 0.5$ s) and with the best-effort service. The desired throughput for all TCP flows is 90 Kbps. In Fig. 5 we show the results for the traffic of the "bottleneck" link (the sum of the total and the minimum satisfied traffic for both paths, and the average throughput of satisfied flows in the two paths, versus the average offered traffic load to the link). Ideally, if the maximum utilization was achieved, the minimum and the total satisfied traffic would be similar and around 2 Mbps and the average throughput of satisfied flows would be around the desired throughput of 90 Kbps.

The results in Fig. 5 prove that our scheme is much better than the best-effort service, and also better than the scheme in [9] when "standard" TCP flows are used. As expected, in underloading, when resources are enough to satisfy all flows, all schemes achieve the same value for the total satisfied traffic, which at the same time is equal to the offered traffic (the dashed line indicates the equality). In overloading, the utilization of the best-effort service tends to zero, similar to the scheme in [9]. The utilization of our scheme stays almost constant for the entire range of offered traffic loads (the total satisfied traffic is about 1.8 Mbps and the minimum satisfied traffic is about 1.5 Mbps). On the other hand, as expected, the average throughput of satisfied flows decreases for high values of the offered traffic because the unreserved resources decrease (the final value is about 117 Kbps).

In the second place, we study the influence of the measurement duration T_{LP} on the performance (Fig. 6). We use different values of T_{LP} from 0.1 s to 2.0 s (with $T_{03} = T_{23}$). Again the desired throughput for all flows is 90 Kbps, and the results shown are for the traffic of the "bottleneck" link. The measurement duration of 0.5 s achieves the best performance, and when the measurement duration is shorter (0.2 s and 0.1 s) or longer (1.1 s and 2.0 s) the performance gets worse since the utilization decreases. This is because if T_{LP} is too short, the measurement is sensitive to bursts, but if it is too long, the measurement does not reflect the changes in the traffic load.

A very important issue is to study in detail if our scheme guarantees the minimum requested throughput to the accepted flows. We have analyzed the simulation results to obtain the percentage of accepted flows that complete the file transfer, the percentage of accepted flows that are satisfied (the ones that complete the file transfer and get at least the threshold throughput –95% of the desired minimum throughput), and the frequency distribution of the throughput of the accepted flows to know how far the accepted and non-satisfied flows are from the threshold throughput. The results can be summarized as follows: 1) all accepted flows complete the file transfer, 2) a high percentage of accepted flows are satisfied, and 3) a high percentage of the accepted and non-satisfied flows achieve a throughput close to the threshold. In Fig. 7 we show some of these results, when the measurement duration is $T_{03} = T_{23} = 0.2$ s and the desired throughput for all TCP flows is 90 Kbps (the threshold is 85.5 Kbps). The graph on the left shows percentages of accepted flows versus the offered traffic load (to the "bottleneck" link). The percentage of accepted flows that are satisfied is 100% for a great range of offered traffic load and then it decreases slowly for high values. In the graph on the right we show the frequency distribution of throughput of accepted flows (the dashed line indicates the threshold

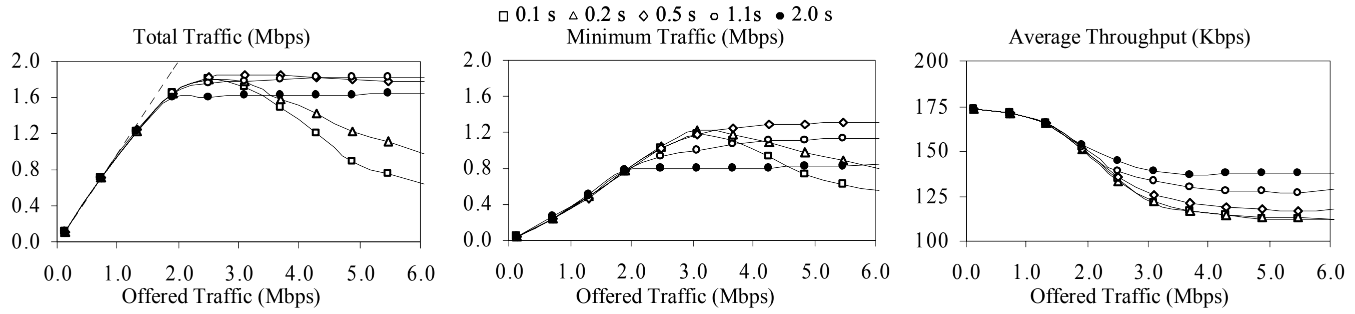

Figure 6. The influence of the measurement duration ($T_{03} = T_{23}$) in the performance of our scheme. The desired throughput is 90 Kbps.

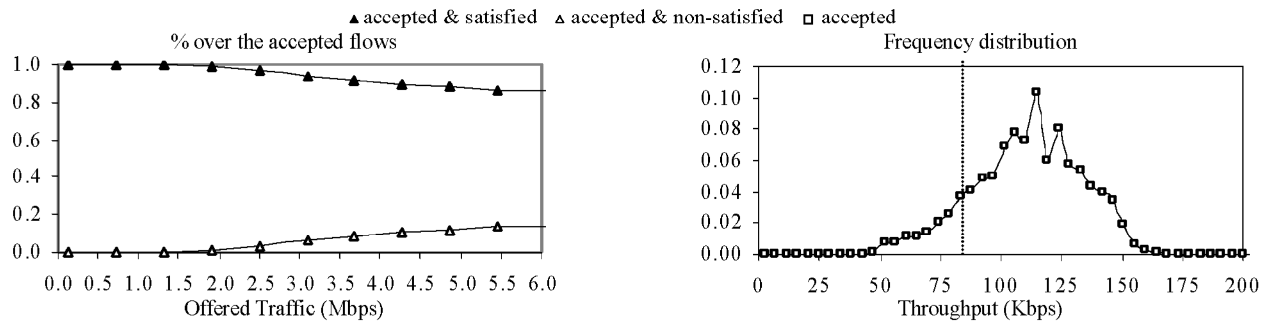

Figure 7. On the left, percentage of satisfied flows and non-satisfied flows over the total accepted flows, versus the offered traffic load; on the right, frequency distribution of throughput of accepted flows when the offered traffic load is 6 Mbps. The desired throughput is 90 Kbps and $T_{03} = T_{23} = 0.2$ s.

throughput) when the offered traffic load is 6 Mbps (the worst case). Although about 13% of flows are not satisfied, a throughput smaller than 63 Kbps is just achieved by the 3% of flows, which is a very small value.

Finally, we study the ability of our scheme to provide different minimum throughput to flows from different users. We consider two users, A and B, with desired minimum throughput of 90 and 135 Kbps respectively. User A sends flows through the logical path e0-e3 and user B through e2-e3. For each user the average offered traffic load vary from 0 to 3 Mbps. The measurement duration for the logical paths is $T_{03} = T_{23} = 0.5$ s. In Fig. 8 we show the results for the traffic of each user (and also the aggregation of both users). We achieve different average throughputs (the final values are about 181 and 114 Kbps), but clearly there is an unfair sharing of the link utilization. This means that our AC accepts more flows demanding a small minimum throughput than flows demanding a large one. However, this is a behavior similar to the one observed in any AC scheme where flows demanding more resources experience a higher blocking probability [13].

V. Conclusions

We have proposed a scheme with AC to guarantee a minimum throughput to "standard" TCP flows, using a small set of packet classes (with different discarding priorities). The AC is implicit, it is based on edge-to-edge measurements of flow aggregates, and it uses a special probing packet flow that is sent continuously from ingress to egress to discover the available throughput in a path. We have evaluated the scheme through simulation in a simple "bottleneck" topology using different traffic loads consisting of "standard" TCP flows that carry files of varying sizes. The simulation results confirm that the scheme guarantees the minimum throughput to the accepted flows and achieves a high utilization of resources.

References

[1] N. Brownlee, KC Claffy, "Understanding Internet traffic streams: dragonflies and tortoises", IEEE Communications Magazine, 2002.

[2] V. Jacobson, "Congestion avoidance and control", ACM Computer Communication Review, 1998.

[3] J. W. Roberts, L. Massoulié, "Arguments in favour of admission control for TCP flows", Proceedings of the 16th ITC, 1999.

[4] D. D. Clark, W. Fang, "Explicit allocation of best-effort packet delivery service", IEEE/ACM Transactions on Networking, 1998.

[5] S. Blake, D. Black, M. Carlson, E. Davies, W. Weiss, Z. Wang, "An architecture for Differentiated Services", RFC 2475, 1998.

[6] R. Mortier, I. Pratt, C. Clark, S. Crosby, "Implicit admission control", IEEE Journal of Selected Areas in Communications, 2000.

[7] A. Kumar et al., "Nonintrusive TCP connection admission control for bandwidth management of an Internet access link", IEEE Communications Magazine, 2000.

[8] S. Ben Fredj, S. Oueslati-Boulahia, J.W. Roberts, "Measurement-based admission control for elastic traffic", Proceedings of the 17th ITC, 2001.

[9] Ll. Fàbrega, T. Jové, P. Vilà, J. Marzo, "A packet class-based scheme for providing throughput guarantees to TCP flows", Proceedings of the 5th IEEE IPOM, 2005.

[10] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, 2001.

[11] UCB/LBL/VINT Network Simulator – ns (version 2)", http://www.isi.edu/nsnam/ns/.

[12] A. M. Law, W. D. Kelton, "Simulation modelling and analysis", MacGraw-Hill, 2000.

[13] J. W. Roberts, U. Mocci, J. Virtamo (Eds.), "Broadband Network Teletraffic. Performance Evaluation and Design of Broadband Multiservice Networks. Final Report of Action COST242", LNCS vol. 1155 Springer-Verlag, 1996.

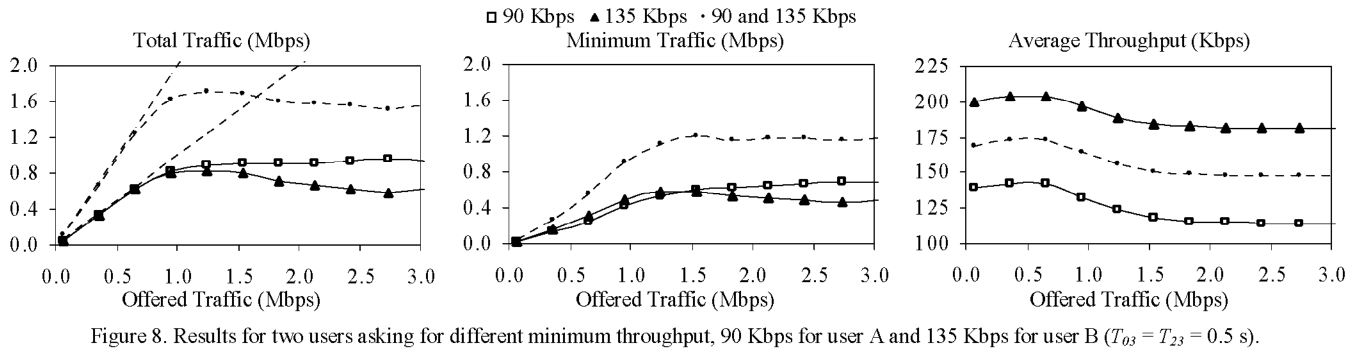

Figure 8. Results for two users asking for different minimum throughput, 90 Kbps for user A and 135 Kbps for user B ($T_{03} = T_{23} = 0.5$ s).

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: UNIVERSITAT DE GIRONA. Downloaded on April 26,2010 at 10:25:44 UTC from IEEE Xplore. Restrictions apply.