Application of Admission Control and Traffic Shaping for providing TCP Throughput Guarantees

Halina Tarasiuk, Robert Janowski and Wojciech Burakowski

Institute of Telecommunications, Warsaw University of Technology,
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
{halina, robert, wojtek}@tele.pw.edu.pl

Abstract. The paper presents an approach for providing requested throughput guarantees, say R_{req}, for greedy TCP (*Transmission Control Protocol*) connections (e.g. ftp). For this purpose, we recommend at the network entry point to shape, with rate R, traffic emitted by each of TCP connections as well as to perform CAC (*Connection Admission Control*) function to limit number of running connections. By using shaper the TCP traffic submitted to the network tends to the form of constant bit rate (CBR). This feature permits us to apply a well known CAC algorithm relevant for CBR traffic. In the case of CAC limit we observe excellent equal bandwidth sharing between competing TCP connections when $R=R_{req}$. For the case when the system is lightly loaded we discuss some strategies allowing for TCP connections to increase their throughput by setting $R>R_{req}$.

1 Introduction

The traffic carried by the Internet is rather of complicate nature and is a mixture of traffic generated by a variety of applications. When, in addition, we use only one service class, named best effort, the only way to satisfy the users about the packet transfer quality is to provision the network on an over-dimensioned state. However, even if it is possible for the IP core it is rather difficult to apply this strategy in the access networks as WiFi, UMTS, xDSL or LAN. Therefore, to improve quality of service (QoS) in the Internet other solutions are currently extensively studied and tested. One of them is to maintain at the network layer a number of different service classes instead of best effort only. Each service class is aimed to provide strictly determined QoS guarantees about packet transfer characteristics (delay, losses) for selected types of applications. For example, we can define a separate service class for handling exclusively VoIP connections. The set of currently considered service classes is presented in [1]. In this paper we focus on one of these classes, named the High Throughput Data (HTD), which is dedicated for handling traffic generated by greedy TCP connections (e.g. ftp). The definition and implementation of this class is currently under investigation by the EuQoS[1] project [7], which is focused on end-to-

[1] EuQoS - End-to-end Quality of Service Support over Heterogeneous Networks, 6 Framework European project, 2.3.5 Research Networking Test Beds, Integrated Project

end quality of service support over heterogeneous networks as IP, WiFi, UMTS, LAN/Ethernet, xDSL, and Satellite.

We assume that for HTD class we have assigned an amount of network resources (buffer size, link rate) in each part of the network. Consequently, the traffic submitted to other classes will not disturb the service of the traffic submitted to the class in question. This can be achieved by applying appropriate packet scheduling mechanism e.g. WFQ (*Weighted Fair Queuing*) and CAC (*Connection Admission Control*) function. Furthermore, we assume that for HTD class we submit only traffic generated by the greedy TCP flows and we focus on providing throughput guarantees for each connection. However, we do not forget about aggressive capabilities of TCP for getting more bandwidth if available.

The approach we propose assumes that a new TCP connection requests some throughput guarantees, named R_{req}. This request is evaluated by appropriate CAC function. For admitted connection, at the network entry point the TCP traffic is submitted to the shaper with the rate R and, in general case, $R \geq R_{req}$. To set the $R > R_{req}$ we consider some strategies when the system is lightly loaded.

The investigated in this paper solution can be regarded as a continuation of the method described in [3] and related to the minimum throughput guarantees for greedy TCP connections in the case of Premium Multi-Media (PMM) service class (similar as the considered HTD class), as defined in AQUILA[2] project [4]. For PMM service we assumed that a given TCP connection may get minimum requested throughput but not more even available bandwidth. So, in the approach presented in this paper the idea is to get an advantage of the TCP behaviour that allows to increase throughput if possible. Summary of all approaches tested in the AQUILA network for the PMM service are presented in [4] and [6].

In the literature we can also find the investigated by many authors CAC schemes for TCP connections, e.g. in [9] and [10] authors assume to reject new TCP connections simply by dropping SYN and ACK SYN segments in the case of the network congestion. For instance, the congestion can be identified when number of waiting packets in the queue exceeds predefined threshold. However, this approach could guarantee a fair share of link capacity between running TCP connections but without possibility for providing them throughput guarantees or throughput differentiation. Another interesting result, which can be adopted, but not in a straightforward way, for the purpose of CAC, is reported in [11]. The presented approach is based on using token bucket marking mechanism. The list of proposed CAC algorithms for elastic traffic and based on some declarations is also presented in [12], but none of them is explicitly targeted for guaranteeing the requested rate for TCP.

The paper is organised as follows. In section 2 we discuss the strategy for using TCP traffic shaping vs. the bandwidth sharing. Section 3 presents the admission control rules and section 4 contains the proposed schemes for assigning shaping rates. Finally, section 5 concludes the paper.

---

[2] AQUILA – Adaptive Resource Control for QoS using an IP-based Layered Architecture, 5 Framework European project, IST 1999-10077

2 Shaping vs. bandwidth sharing

In the literature the bandwidth sharing topic has received a lot of attention e.g. in [5]. In this section we point out on some advantages of shaping the traffic emitted by TCP flow. For this purpose, we compare two systems, without and with shapers, as depicted on Fig. 1.

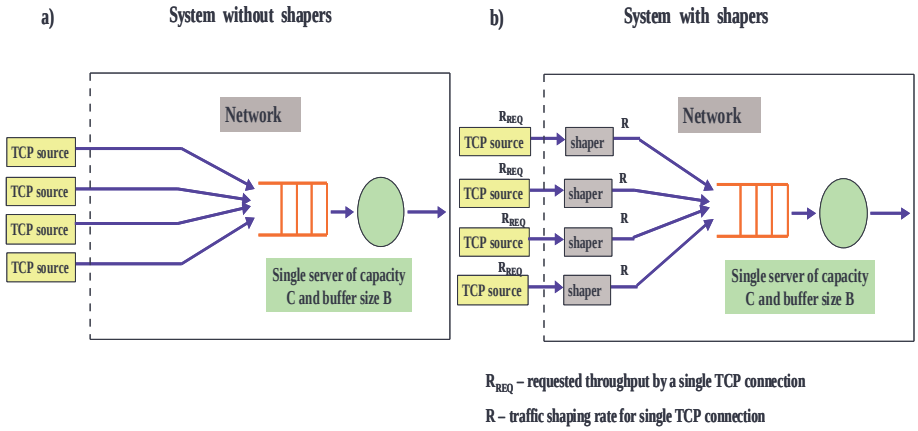

Fig. 1. Systems without and with shapers

For both cases, we model the network by a single bottleneck link with capacity C and buffer size B. Fig. 1a shows a typical scenario of system without shapers, when a number of TCP sources submit traffic to the system. However, for this scheme it has been recognised that even the TCP connections have similar RTT the throughput received by each of them may differ essentially. This is due to the bursty nature of TCP traffic. On the contrary, when we use the shapers, as depicted on the Fig. 1b, one may expect that TCP traffic is smoothed and, in this way, more equal bandwidth sharing and less packet losses can be achieved. Furthermore, by decreasing shaping rate the TCP traffic becomes more smoothed while the additional packet delay introduced by the shaper increases. In the limit case the TCP traffic tends to constant bit rate traffic. In Table 1 we show exemplary numerical results for the system with 100 homogeneous TCP connections submitting traffic to the link of capacity C=100 Mbps and buffer size B=20 packets. For each of TCP sources we assumed TCP Reno with segment size=1500 bytes, advertised window size=64 KB. The reported results correspond to the systems with no shapers and with shapers of rates R=1, 1.1, 1.2, 1.5 or 2 Mbps. Furthermore, the minimum delay (min RTT) introduced by the network part was set to 0, 30 or 60 ms. Finally, the effectiveness of the system was evaluated by simulation (in ns-2) from the point of view of the following parameters: the value of average RTT (including additional delay introduced by the shaper), packet loss rate per TCP connection (with 95% confidence interval) and throughput received by each of the TCP connection (with 95% confidence interval).

Table 1. Exemplary numerical results showing the impact of the TCP traffic shaping for the case with n=100, B=20, C=100 Mbps
Min RTT = 0 msec

Shaping rate R [Mbps]	Average RTT [ms]	Packet loss ratio per TCP connection	Throughput per TCP connection [Mbps]
1	528	0	0.999±0.0001
1.1	193	$1.5\ 10^{-2} \pm 2.0\ 10^{-3}$	0.992±0.015
1.2	140	$4.1\ 10^{-2} \pm .3.0\ 10^{-3}$	0.95±0.03
1.5	74	$3.910^{-2} \pm 1.310^{-3}$	1.0±0.18
2	25	$5.0\ 10^{-2} \pm 6.6\ 10^{-3}$	1.0 ±0.227
No shaping	-	Almost all TCP connections were stopped	

Min RTT = 30 msec

Shaping rate R [Mbps]	Average RTT [ms]	Packet loss ratio per TCP connection	Throughput per TCP connection [Mbps]
1	528	0	0.999±0.0001
1.1	207	$1.4\ 10^{-2} \pm 2.0\ 10^{-3}$	0.999±0.02
1.2	120	$1.7\ 10^{-2} \pm .1.0\ 10^{-3}$	0.999±0.02
1.5	60	$2.410^{-2} \pm 2.010^{-3}$	0.964±0.03
2	40	$3.2\ 10^{-2} \pm 3.2\ 10^{-3}$	0.940±0.06
No shaping	31	$2.6\ 10^{-2} \pm 3.0\ 10^{-3}$	0.933±0.123

Min RTT = 60 msec

Shaping rate R [Mbps]	Average RTT [ms]	Packet loss ratio per TCP connection	Throughput per TCP connection [Mbps]
1	528	0	0.999±0.0001
1.1	209	$0.85\ 10^{-2} \pm 1.5\ 10^{-3}$	0.976±0.017
1.2	133	$1.1\ 10^{-2} \pm 1.6\ 10^{-3}$	0.992±0.025
1.5	88	$1.5\ 10^{-2} \pm 1.2\ 10^{-3}$	0.973±0.038
2	67	$1.6\ 10^{-2} \pm 1.7\ 10^{-3}$	0.948±0.05
No shaping	61	$1.6\ 10^{-2} \pm 2.0\ 10^{-3}$	0.982±0.07

The presented results say that the impact of the shaper, as it was expected, in the considered case is significant. First of all, by using the shaper and fixing the rate at the value of equal bandwidth sharing (in the considered system, shaping rate R=1 Mbps) we receive very desirable results that throughput values for each of running connections are similar. In addition, no packet losses were observed. On the contrary, for the system without traffic shaping, the received throughput by the TCP connections differ essentially and for the case of min RTT=0 almost all TCP connections were stopped. So, we conclude that the best solution for getting the equal share between TCP connections is to use the traffic shapers with the rate = C/n, where n is the number of connections. It is worth to mention that for this case the TCP traffic

outgoing from the shaper and submitted to the link tends to constant bit rate traffic when the number of running connections is large. This observation will help us in proposing CAC algorithm (see section 3).

Other conclusions coming from the Table 1 are the following:

- Additional delay introduced by the shaper has no significant impact on the received TCP throughput.
- Increasing the shaping rate above 1 Mbps makes larger variation of the throughput received by the TCP connection. This effect is not surprising since now the offered TCP traffic can temporary exceeds the link capacity and no sufficient mechanisms for getting equal sharing bandwidth.
- By increasing the shaping rate above 1.1 Mbps we decrease the packet loss ratio but not in an essential way.
- Impact of using shaper is more visible for the cases with smaller min RTT.

Another observation is that when traffic load counted as:

$$traffic\ \ load = \sum_{i=1}^{i=n} R_i \le C\ , \tag{1}$$

where n denotes the number of running connections and R_i (i=1,…,n) is the shaping rate of the i-th connection, is not greater than the link capacity C then by increasing buffer size B we may only minimize packet losses, if they are. For example, in the case of R=1 Mbps and B=20, the measured packet loss ratio was 0 and increasing buffer size, e.g. up to 100, does not change anymore the system performances.

3 Connection Admission Control

The motivation for applying CAC function in the case of greedy TCP connections is to provide throughput guarantees for admitted connections. In this section we present a method for CAC for the case of the system from Fig. 1b as described in section 2. So, we assume that TCP traffic is shaped at the network entry point. For the clarity of the text presentation, we focus on the case when the TCP connections are homogenous, it means they have the same values of min RTT and the advertised window size, as well as the throughput requested by each of the connections is the same.

For such a system we state that if we apply CAC the throughput received by the connection i (i=1,2,..n), when n is the number of running TCP connections, is equal to the shaping rate R_i. Anyway, this throughput is possible to be got only if:

$$\sum_{i=1}^{i=n} R_i \le \rho\, C\ , \tag{2}$$

where ρ is the utilization factor (ρ<1). As one can realize, the formula (2) is the same as formula for CAC function [8] in the case when the system serves CBR connections. In fact, when we use shaper for TCP traffic, the traffic outgoing from the shaper takes a form of CBR traffic, with rate equal to the shaping rate. Note that a greedy TCP source will tend to use whole capacity allocated to it. The value of parameter ρ we calculate from the analysis of the M/D/1 queuing model [8] taking into account the values of the target packet loss ratio (*IPLR - IP Packet Loss Ratio*) and buffer size B. The formula is the following:

$$\rho = \frac{2\,B}{2\,B - \ln(IPLR)}\,. \tag{3}$$

So, for the case of the link capacity C and the requested throughput for each connection equal to R_{req} the maximum number of admitted connections, say N_{max}, we calculate from:

$$N_{\max}\ R_{req} \le \rho\, C\,, \tag{4}$$

where value of ρ we calculate assuming low value of IPLR, e.g. IPLR=10^{-3} as typical for the Internet network with QoS [2]. In fact, for TCP Reno setting IPLR on that level does not degrade the TCP throughput comparing with setting IPLR on the value close to 0.

Table 2 and Table 3 contain the exemplary numerical results corresponding to the verification of the CAC formula. Table 2 corresponds to CAC calculation while Table 3 corresponds to the verification by simulation. For CAC calculation, as the input data are: buffer size B, packet loss ratio IPLR and the rate requested by each of the TCP connections. The results are the values of utilization factor ρ and maximum number of the TCP connections we can admit. To verify the CAC function, we measure the values of IPLR and received throughput of each connection assuming the values of B, R_{req} and N_{max} from CAC calculation.

Table 2. Exemplary numerical results corresponding to CAC calculation, assumed link capacity C=100 Mbps

Input data for CAC			CAC calculation	
Buffer size B [packets]	Packet loss ratio IPLR	Requested throughput R_{req} = R [Mbps]	ρ - utilization factor	N_{max} – max. number of connections
10	10^{-3}	1	0.74	74
20	10^{-3}	5	0.85	17
50	10^{-3}	10	0.935	9

Table 3. Exemplary numerical results corresponding to CAC verification by simulation, assumed link capacity C=100 Mbps

Input data for simulation			Simulation results	
Buffer size B [packets]	Requested throughput R_{req} = R [Mbps]	N_{max} – max. number of connections	Packet loss ratio IPLR	Received rate per connection [Mbps]
10	1	74	0	1±0.001
20	5	17	0	4.998±0.003
50	10	9	0	9.999±0.002

The results presented in Table 3 confirm our expectation about applied CAC function. For all considered exemplary cases, the received TCP throughput was exactly the same as the requested one.

4 Assigning shaping rates

The exemplary numerical results referring to the verification of the assumed CAC function and presented in Table 3 were obtained for the case when the shaping rates are the same for each of running TCP connections under CAC limit. The simplest way, say scheme #1, for achieving the above is that for every new connection we set the shaping rate at the value corresponding to the CAC limit conditions, e.g. according to the requested throughput.

In this section we focus on the problem of assigning the shaping rates for achieving the following goals:

- when the system gets the CAC limit conditions, the shaping rates should be assigned according to the requested throughput;
- when the system is below CAC limit conditions, the shaping rates should be assigned to share the whole system capacity in a fair way. In this case, the received TCP throughput for running connections is above the requested rates.

The solution, say scheme #2, for getting the above goals can be possible only when we update the shaping rates for each of running connections every time when new connection starts or one of the running connections terminates. For example, if C=100 Mbps and n=10 then each connection may get throughput equal to10 Mbps, if n=20 then each connection may get 5 Mbps. Unfortunately, this can be achieved only if the shaping rates in the considered cases will be equal to 10 Mbps and 5 Mbps, respectively. However, it is obvious that such approach can be rather difficult to implement since it requires too many actions when number of running connections is large.

Fig. 2 shows the characteristics of the usage capacity as a function of admitted TCP connections for the schemes #1 and #2. These schemes can be regarded as limit cases. We can say that scheme#1 is the most pessimistic while scheme#2 is the most

optimistic. Anyway, we are searching for a scheme that can be relatively easy for implementing and gives characteristics in the potential working area as depicted on

Fig. **2**. Of course, it would be desirable that the characteristics of such a scheme will tend to be closer to scheme#2.

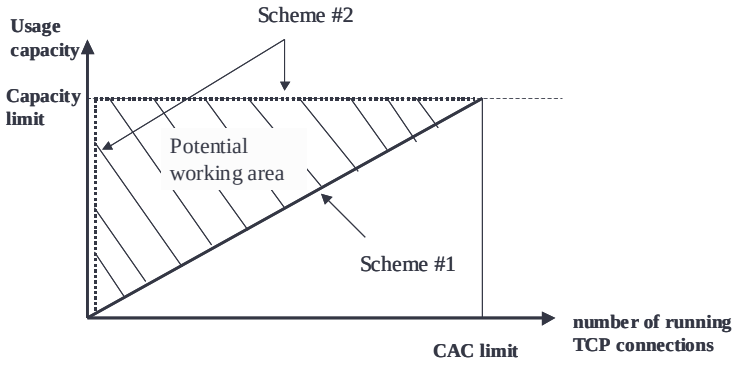

Fig. 2. Usage capacity vs. number of running TCP connections for schemes#1 and 2

Taking into account the drawbacks of scheme#1 and #2, we propose an intermediate scheme for assigning the shaping rates. It is evident that one can find a number of such schemes. In this paper, we present a scheme, say scheme#3, which assumes that the shaping rate assigned for a new connection is set at the beginning and is not changed during the connection. For scheme#3, the values of the shaping rates for a new connection we assign as follows. The value of the shaping rate, say R_{new}, we calculate on the basis of the equation:

$$R_{new} = \max\{R_{req}, (\rho\, C - \sum_{i=1}^{i=n} R_i)/k\}\ , \tag{5}$$

where k is a constant factor, k > 1. By using expression (5), we assign the higher shaping rate when the system is lightly loaded, as it is done in scheme#2. If the system load increases then the assigned shaping rates for new connections decrease but never below R_{req}, as in scheme#1. As a consequence, for scheme#3 the CAC rule according to (4) is not valid. Instead of (4), now the CAC rule is as follows. New flow can be admitted only if:

$$R_{new} + \sum_{i=1}^{i=n} R_i \le \rho\, C\,, \tag{6}$$

where ρ is calculated from (3). Note that in scheme#3 for each connection we guarantee throughput resulting from the assigned shaping rates.

Table 4 shows the received simulation results for the systems with scheme#1, #2 and #3, each with C=100 Mbps, B= 10 packets and R_{req}=1 Mbps. For each TCP

connection we want to transfer file of constant size equal to 10Mbits. In addition for scheme#3, we set k=3. The traffic conditions for all compared systems were assumed as follows. IPLR is equal to 10^{-3}. Then, from (3) we receive value of ρ=0.74. So, the maximum number of admitted connections in each system is 74. Next, the requests for new connections arrive to the systems according to Poisson process with the intensity λ. Assuming the call blocking probability for scheme#1 at the level of 1%, the value of λ (λ=5.98 calls/sec) we calculate from Erlang formula with n=74 and connection service time=10Mbits / 1Mbps=10 sec.

Table 4. Comparison of schemes #1, #2 and #3

Scheme	Mean file transfer time [sec]	Mean system state when call blocking	Call blocking probability
Scheme#1	10	74	1%
Scheme#2	0.319	no blocking	no blocking
Scheme#3	3.1	31	0.6%

The presented exemplary results in Table 4 say that by appropriately assigning the shaping rates, e.g. by using scheme#3, for TCP connections we can get better resource utilisation of the system and lower file transfer times. The ideal scheme is the scheme#2, for which we can get mean transfer file 0.319 sec and no call blocking was observed. Anyway, by using scheme#3 we have much better system behaviour than for scheme#1. Comparing these two schemes, scheme#3 significantly overcomes scheme#1 since we observe 3 times shorter transfer times and 2 times lower call blocking. Summarising, by using shaping rates for greedy TCP connections with a possibility for setting these rates depending on the current system load we can guarantee the requested throughput for each of the connections and we can increase this throughput for the connections arriving during the periods when the system is lightly loaded.

5 Conclusions

In this paper we have presented an approach for handling greedy TCP connections requiring throughput guarantees. In this approach we apply traffic shapers as well as CAC function. By shaping TCP traffic we do smoothing of this traffic even to the form of CBR traffic as well as we observe excellent fairness between competing TCP connections in access to the resources. Furthermore, as we have proved, the CAC function for handling TCP connections can be the same as for handling CBR streams. Another presented result corresponds to the strategy for assigning the shaping rate. It appears that by using the strategy that takes into account the current system load we can provide higher throughput than the requested one but only for the connections that start at the moments when system is not heavy loaded. This result is very desirable since limits the impact of the shapers when the system is essentially below the CAC limit.

Further work is focus on implementation of the discussed scheme in the testbeds.

References

1. Babiarz, J., Chan, K., Baker, F.: “Configuration guidelines for DiffServ service classes”, IETF draft-ietf-tsvwg-diffserv-service-classes-00, work in progress, February 2005.
2. ITU-T Recommendation Y.1541, “Network performance objectives for IP-based services”, ITU, May 2002.
3. Burakowski, W., Tarasiuk, H.: "Admission Control for TCP Connections in QoS IP Network", In Proc. of HSI’2003 Conference, June 2003, Seoul, Korea, LNCS 2713, Springer Verlag 2003, pp. 383-393.
4. Brandauer, C., Burakowski, W., Dabrowski, M., Koch, B., Tarasiuk, H.: "AC algorithms in Aquila QoS IP network", European Transaction on Telecommunications, John Wiley & Sons, Inc., Vol. 16, No. 3, May-June 2005, pp. 225-232.
5. Roberts, J.: “A survey on statistical bandwidth sharing” , Computer networks, Vol. 45, pp. 319-332, 2004.
6. Dabrowski, M., et al.: „Evaluation of the AQUILA Architecture: Trial Results for Signalling Performance, Network Services and User Acceptance”, Proc. of the Art-QoS 2003 Workshop, Warsaw, Poland, LNCS 2698, Springer Verlag 2003, pp. 218-233.
7. Dugeon, O., Morris, D., Monteiro, E., Burakowski, W., Diaz, M.: "End to End Quality of Service over Heterogeneous Networks (EuQoS)", In Proc. of NetCon'05, IFIP TC6 Conference, Lannion, France, 14-18 November 2005.
8. Roberts, J., Mocci, U., Virtamo J. (eds.): “Broadband network teletraffic: Performance evaluation and design of broadband multiservice networks”, Final report COST 242, LNCS 1155, Springer 1996.
9. Benameur, M., Ben Fredj, S., Delcoigne, F., Oueslati-Boulahia, S., and Roberts, J. W.: “Integrated Admission Control for Streaming and Elastic Traffic”, Quality of Future Internet Services, LNCS 2156, Springer 2001.
10. Mortier, R., Pratt, I., Clark, C. and Crosby, S.: “Implicit Admission Control”, IEEE Journal on Selected Areas in Communications, Vol. 18, No. 12, December 2000.
11. Sahu, S., Nain, P., Towsley, D., Diot, C., Firoiu, V.: “On Achievable Service Differentiation with Token Bucket Marking for TCP”, Proc. ACM SIGMETRICS’00, Santa Clara, CA, June 2000.
12. Brichet, F., Mandjes, M., Sanchez-Canabate, M. F.: “Admission control in multiservice networks”, Proceeding of the Mid-Term Seminar of COST257, 1999, Villamora, Portugal.