Multiparameter TCP Congestion Control

Michal Olšovský*
Institute of Computer Systems and Networks
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
olsovsky@fiit.stuba.sk

Abstract

Network throughput increase is usually associated with replacement of communication links and appropriate network devices. However it is necessary to bear in mind that effective and less intrusive increase of network throughput can be achieved via the improvement of existing protocol stack, mostly at network and transport layer. In this paper we propose an advanced notification system for TCP congestion control called ACNS. This new approach allows TCP flows prioritization based on the flow age and carried priority. The aim of this approach is to penalize old greedy TCP flows with a low priority in order to provide more bandwidth for young and prioritized TCP flows while providing more accurate details for loss type classification which is especially useful in wireless environment. By means of penalizing specific TCP flows significant improvement of network throughput can be achieved.

Categories and Subject Descriptors

C.2 [**Computer-communication Networks**]: Network Protocols

Keywords

congestion control, delay, flow age, network, notification system, performance, prioritization, priority, RTT, queue, TCP, throughput

1. Introduction

The very first version of the most common transport protocol TCP was introduced in RFC793 [5]. To match the increasing traffic requests (bandwidth, delay, etc.), it was necessary to improve not only the hardware part of the communication networks, but the software part as well. Improvements of the TCP, mostly called TCP variants or extensions, are mainly focused on the best and most effective usage of available communication lines [4, 1].

First TCP improvements focused on higher performance were published in [10]. Since 1992 [6], there have been many new TCP variants which can be divided into 2 main groups. Based on the end network type we can recognize wired group and wireless group. These groups can be further divided into smaller groups whose key element is the congestion detection. Based on this hierarchy, we can recognize variants like CUBIC, Compound TCP, Sync-TCP for wired networks [14, 3, 12] and JTCP, TCP CERL and Compound TCP+ for wireless networks [11, 13, 2]. All these variants have one thing in common: they do not make any steps for congestion avoidance unless the congestion is detected by TCP end nodes [9]. Slightly different approach can be identified while using Explicit Congestion Notification (ECN) system when TCP end nodes allow nodes in the network to inform them about the situation in the network and give them some kind of feedback [7].

However this feedback is sent within all existing TCP flows which support ECN apart from any flow characteristic. The only sent feedback stands for the order to decrease the congestion window. While keeping in mind existing ECN system we have identified two limitations. Firstly all TCP flows in the network from the TCP end nodes point of view are treated equally. Secondly there is only one command - to decrease the size of the congestion window which is sent to all TCP end nodes at the same time. Mechanisms and further concepts introduced in the following chapters are aimed to solve highlighted issues.

2. Concept

The idea of our approach ACNS (Advanced Congestion Notification System) is to allow the TCP to inform only specific TCP end nodes about the congestion in the network and instruct them to change the congestion window. Such functionality will provide more bandwidth to younger and prioritized TCP flows by freezing and possibly decreasing the congestion window of older TCP flows. We propose a set of weights assigned to each TCP flow for further calculations which in turn will result in a specific command which will be sent within particular flows. These weights are based on the following three parame-

*Doctoral degree study programme in field Applied Informatics. Supervisor: Assoc. Prof. Margaréta Kotočová, Institute of Computer Systems and Networks, Faculty of Informatics and Information Technologies, STU in Bratislava.

Work described in this paper was presented at the 8th Student Research Conference in Informatics and Information Technologies IIT.SRC 2013.

Olšovský, M. Multiparameter TCP Congestion Control. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies, Vol. 5, No. 2 (2013) 39-43

ters:

1. TCP flow age.
2. TCP flow priority.
3. Queue length.

ACNS may result into two situations. First situation is typical situation when the command is calculated by and received from the nodes in the network. TCP end nodes receive the packet, check the header and modify the congestion window according to received command. The mechanism of TCP flow weight calculation and command determination is described in chapter 2.1. Second situation represents typical loss in the network. At this point the end node has to determine which command will be used based on the commands received in the past (chapter 2.2).

2.1 Flow weight and command calculation

While the TCP communication passes nodes in the network, it is necessary to calculate the weight of the TCP flow in order to send commands to the end nodes. As we mentioned earlier the calculation has 3 input parameters - TCP flow age, TCP flow priority and queue length.

TCP flow age is unique per flow and is changing in the time. As the age is theoretically unlimited, this parameter would bring some indeterminism to the final weight calculation. To solve this issue we have introduced age normalization (1): age of the flow f_i is represented as a part of the oldest flow age f_{max}. Using normalization age values can vary from 0 to 1 (including).

$$\forall\ i \in (1; |F|): \quad T(f_i) = \frac{f_i}{f_{max}} \tag{1}$$

Similar normalization is used for the second parameter priority p. Priority normalization is done within the function $F(p)$ using maximal priority p_{max}. The last input parameter, actual queue length $A(q)$, is changing in time and is shared across all flows. It represents the actual usage of the queue and can vary from 0 up to 1.

Final weight W for flow f_i used for command determination can be calculated using (2) where $F(p)$ stands for priority part and $T(f)$ represents age part. Both subparts can be calculated using (3). It is possible to put more weight on a specific part or eliminate the other part by the weight factors (v_p,v_a) but the sum of these factors must be equal to 1.

Calculated weight W needs to be transformed into command C which will be sent to the TCP end nodes. The command can be 1, 2 or 3. As the calculated weight W is a real number, we need to convert it to one of available commands using comparison with 2 thresholds th_{A1} and th_{A2}.

$$W = \frac{A(q)}{F(p) + T(f)} \tag{2}$$

$$F(p) = \frac{v_p * p_{max}}{p_i}; T(f) = v_a * age(f_i) \tag{3}$$

2.2 Determining command upon loss

Commands received within acknowledgements can be useful when loss occurs as they represent the situation in the network right before the loss. Using these commands we are able to determine trend command C_{trend} directly in the TCP end nodes.

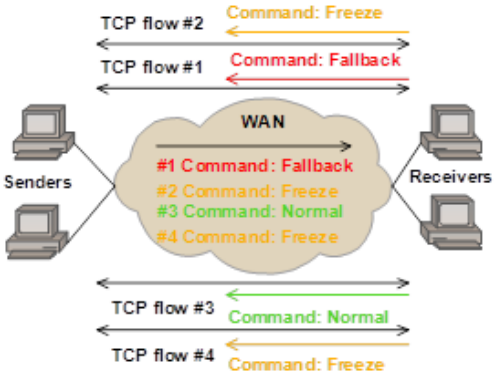

Figure 1: ACNS overview.

At first end nodes calculate trend weight W_{trend} using w_{count} of the latest received commands. Even if we use only few commands we have to distinguish between their ages. This is achieved by assigning metric to every used command using the exponential decrease with additional step parameter [8]. Calculated metric for every received command is normalized using sum of metrics of all used commands X. Setting P the array of all received commands the trend weight W_{trend} for specific TCP flow can be calculated using (4).

Later on calculated trend weight W_{trend} needs to be transformed into trend command C_{trend} which will be used by end node itself. Calculation is similar to the calculation for standard command, the only difference is in used thresholds th_{L1} and th_{L2}. These thresholds can be set according to standard mathematical rounding or can use custom values.

$$W_{trend} = \sum_{i=1}^{\omega_{count}} \frac{e^{-\frac{i}{\sigma}} \cdot P[i]}{X} \tag{4}$$

2.3 Commands

TCP end node can modify the size of congestion window according to one of six commands. As we stated before half of these commands can be received within acknowledgements and half needs to be calculated. Commands usage overview is shown in Figure 1.

Received commands:

1. $C = 1$: "*normal*". There is no congestion in the network. Congestion window can be modified following the used TCP variant.
2. $C = 2$: "*freeze*". There are signs of incoming congestion. As this command receive only specific TCP end nodes, not all TCP flows are penalized. After receiving, TCP end nodes freeze their congestion window cwnd.
3. $C = 3$: "*fallback*". There is a congestion in the network. After receiving, congestion window will not be decreased by multiplicative factor however it will be decreased to the last known smaller value.

Calculated commands:

1. $C_{trend} = 1$: "*freeze*". Loss occurred without any signs of congestion (probably communication channel interference). Command of 2 is put in the list of received commands P (different treatment during another loss).
2. $C_{trend} = 2$: "*fallback*". Loss occurred within indicated incoming congestion. Congestion window will be decreased to the last known smaller value. Command of 3 is put in the list of received commands P.
3. $C_{trend} = 3$: "*decrease*". Loss occurred within ongoing congestion. Congestion window will be reduced following the used TCP variant. Command of 3 is put in the list of received commands P.

3. Cooperation with existing TCP variants

Introduced approach can be used in combination with any existing and future TCP variant. Detailed cooperation with used TCP variant is explained in the following chapters.

3.1 Change upon recipient of acknowledgement and upon loss

End nodes can receive three different commands within the acknowledgement. Together with these three commands available congestion window (cwnd) changes are defined in (5) where $cwnd_{last}$ is defined as *W(t-1)* and $cwnd_{last\_x}$ is defined as *W(t-x)* Function W stands for congestion window size changing function in time. After receiving command of 1, end node will be allowed to use its own TCP implementation to calculate new congestion window.

$$cwnd = \begin{cases} cwnd_{TCP} & C = 1 \\ cwnd_{last} & C = 2 \\ cwnd_{last\_x} & C = 3 \end{cases} \quad (5)$$

Using self-calculated trend commands end nodes are able to modify congestion window as defined in (6). After receiving command of 3 end node will decrease the congestion window according to used TCP variant.

$$cwnd = \begin{cases} cwnd_{last} & C_{trend} = 1 \\ cwnd_{last\_x} & C_{trend} = 2 \\ cwnd_{TCP} & C_{trend} = 3 \end{cases} \quad (6)$$

3.2 Integration into IPv4 and TCP

Our approach keeps full compatibility with existing IPv4 and TCP, even with ECN system. Backward compatibility means that the end nodes will use system which both of them support and new ACNS commands will appear as ECN commands for non-compatible host. Integration in IPv4 header lays in reusing ECN bits with one additional unused bit from field *Flags* called *CMI*. Routers are willing to encode more important commands into IPv4 header (Table 1 - NS stands for *nonce sum* flag) and overwrite existing ones. TCP sender uses messages 2/3 within one data window. When sending last packet from specific data window, sender uses messages 4/5 in order to ask routers to encode command in the IPv4 header (saves routers system resources). Messages 6 and 7 are created only by routers. Similar reuse appears in the TCP header. Combination of existing ECN bits and new bit from the *reserved* field called *CMT* allows us to (en)code all necessary ACNS messages (Table 2 - NS stands for *nonce sum* flag). TCP receiver decodes command from IPv4 header and encodes the command in the TCP acknowledgement header sent to the TCP sender. TCP receiver can use messages 1/2 in order to signalize normal congestion window processing. In case of upcoming congestion, TCP receiver can inform TCP sender with message 5 in order to freeze congestion window or with message 6 to apply command fallback. All messages from Table 2 are used only by TCP end nodes.

Table 1: ACNS messages encoded in IPv4 header.

#	ECN		CMI	Message
1	*0*	*0*	*0*	ACNS not supported
2	*1*	*0*	*0*	ACNS in ECN mode (set by end node), ACNS message: command normal (left by routers), NS = 0
3	*0*	*1*	*0*	ACNS in ECN mode (set by end node), ACNS message: command normal (left by routers), NS = 1
4	*1*	*0*	*1*	ACNS supported (set by end node), ACNS message: routers to set command, NS=0
5	*0*	*1*	*1*	ACNS supported (set by end node), ACNS message: routers to set command, NS=1
6	*1*	*1*	*1*	ACNS message: command freeze (set by routers)
7	*1*	*1*	*0*	ACNS message: command fallback (set by routers)

Table 2: ACNS messages encoded in TCP header.

#	CWR	ECE	CMT	Message
1	0	*0*	*0*	command normal (R), NS=0
2	*0*	*0*	*0*	command normal (R), NS=1
3	*1*	*0*	*0*	congestion window reduced (S), NS=0
4	*1*	*0*	*0*	congestion window reduced (S), NS=1
5	*0*	*1*	*1*	command freeze (R)
6	*0*	*1*	*0*	command fallback (R)

4. Simulation results

System ACNS was implemented in the network simulator ns-2 where the simulations were performed as well. Within the simulation topology (Figure 2) we used 3 concurrent TCP flows (detailed characteristic in Table 3) and 3 concurrent UDP flows (detailed characteristic in Table 4). ACNS system parameters were set according to Table 5. Simulation results are shown in Table 6 and Table 7. Monitored network parameters were throughput, RTT, amount of sent data and packet loss.

According to the simulations results, using ACNS it is possible to increase TCP flows throughput (Figure 3 -

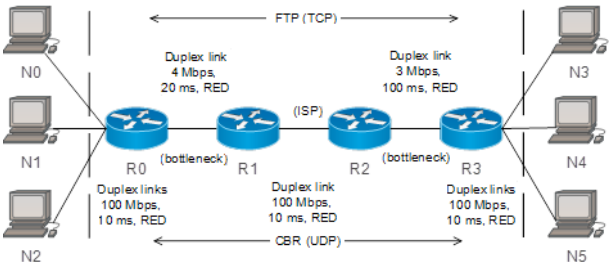

Figure 2: Simulation topology.

Table 3: TCP flows parameters.

Flow	TCP	Prio.	Start	End	From	To
#1	CUBIC	0	0.1	118	N0	N5
#2	CUBIC	26	15.1	118	N0	N4
#3	CUBIC	46	30.1	118	N1	N3

Table 4: UDP flows parameters.

Flow	Rate	Prio.	Start	End	From	To
#1	0.5	0	0.1	118	N0	N3
#2	0.6	0	20.1	118	N1	N4
#3	0.5	0	40.1	118	N2	N5

Table 5: ACNS system parameters.

th_{L1}	th_{L2}	ω_{count}	σ	v_p	v_a	th_{A1}	th_{A2}
1,8413	2,1587	10	3	0,8	0,2	0,95	1,0

Table 6: Simulation results I.

#	System	Throughput [Mb/s]					
		Average			Maximal		
		#1	#2	#3	#1	#2	#3
1	-	0.8	0.4	0.2	2.7	2.1	1.8
2	ACNS	0.3	0.7	1	3	3	3
#	System	RTT [s]					
		Average			Maximal		
		#1	#2	#3	#1	#2	#3
1	-	377	377	316	973	1230	335
2	ACNS	327	329	332	468	484	406

Table 7: Simulation results II.

#	System	Sent data [MB]			
		#1	#2	#3	Total
1	-	11.78	6.15	2.89	20.82
2	ACNS	4.4	11.26	14.33	29.99
#	System	Loss [packets]			
		#1	#2	#3	Total
1	-	110	86	40	236
2	ACNS	0	2	0	2

Table 8: Network performance improvements.

Network parameter	Improvement
Total average throughput	+ 44,5 %
Total average RTT	- 7,1 %
Total data sent	+ 44,0 %

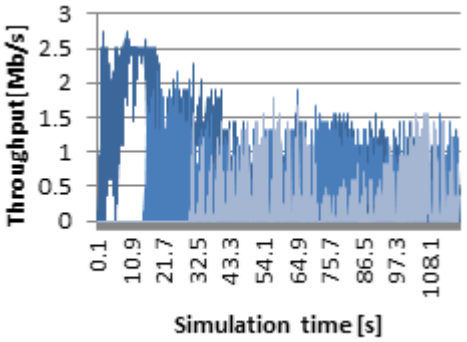

Figure 3: TCP flows throughput (without ACNS).

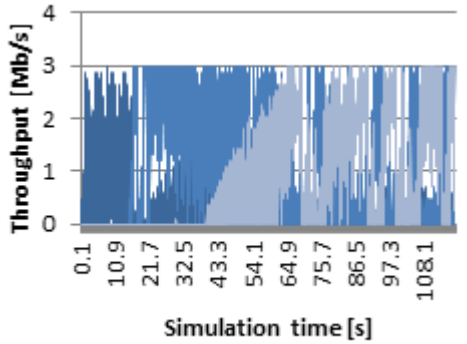

Figure 4: TCP flows throughput (with ACNS).

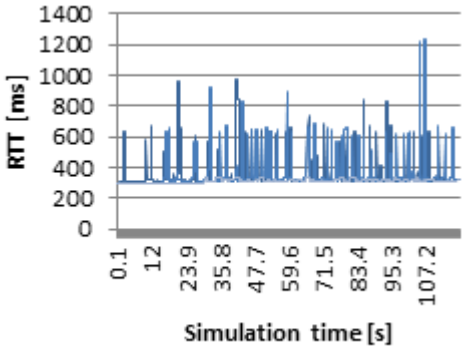

Figure 5: TCP flows RTT (without ACNS).

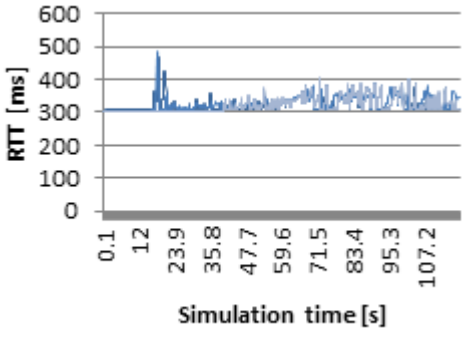

Figure 6: TCP flows RTT (with ACNS).

without ACNS, Figure 4 - with ACNS) by 44% which lead to increased amount of sent data (44% increase). Using our new approach TCP flows RTT can be decreased (Figure 5 - without ACNS, Figure 6 - with ACNS) by 7%. Network performance improvements are summarized in Table 8. All these improvements were achieved with nearly none losses.

5. Conclusion

In this paper we have introduced an advanced notification system for TCP congestion control called ACNS. In comparison with existing approaches, our approach ACNS can be used in combination with any existing of future TCP variant. One can compare this approach with existing ECN system however ECN system does not distinguish between TCP flows and between certain phases of congestion. Our approach enables prioritization of TCP flows using their age and carried priority. As a result, only specific TCP flows are penalized and not in the same way.

The goal of ACNS is to avoid congestion by means of providing more bandwidth to new flows while penalizing old flows and later on if congestion occurs it uses TCP variant mechanism to eliminate the congestion. Using ACNS significant improvement of network throughput can be achieved. Depending on the TCP flows prioritization it is possible to achieve up to 44 % increase of throughput and the amount of transferred data and around 7 % RTT decrease with nearly none losses. To sum it up, ACNS allows TCP performance increase without the need to increase capacity of the communication links.

Acknowledgements. The support by Slovak Science Grant Agency (VEGA 1/0676/12 "Network architectures for multimedia services delivery with QoS guarantee") is gratefully acknowledged.

References

[1] M. Bateman and et al. A comparison of tcp behaviour at high speeds using ns-2 and linux. *Proceedings of 11th ACM CNS '08*, 2008.

[2] A. Botta, A. Dainotti, and A. Pescape. Multi-protocol and multi-platform traffic generation and measurement. *Proceedings of INFOCOM 2007 DEMO Session*, 2007.

[3] H. Chao and X. Guo. Quality of service control in high-speed networks. *John Wiley & Sons, Ltd.*, 2005.

[4] S. Ha and et al. Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating System Review*, 42, 2008.

[5] I. S. Institute. Transmission control protocol. *RFC 793*, 1981.

[6] A. Karnik and A. Kumar. Performance of tcp congestion control with explicit rate feedback. *IEEE/ACM Transactions on Networking*, 13:108–120, 2005.

[7] M. Kwon and S. Fahmy. Tcp increase/decrease behaviour with explicit congestion notification (ecn). *Proceedings of IEEE International Conference on Communications*, 4:2335– 2340, 2002.

[8] L. Malago and M. Matteucci. Towards the geometry of estimation of distribution algorithms based on the exponential family. *Proceedings of the 11th workshop on Foundations of genetic algorithms*, pages 230–242, 2011.

[9] J. Martin, A. Nilsson, and I. Rhee. Delay-based congestion avoidance for tcp. *IEEE/ACM Transactions on Networking*, 2003.

[10] M. Mirza, J. Sommers, and P. Barford. A machine learning approach to tcp throughput prediction. *IEEE/ACM Transactions on Networking*, 18:1026–1039, August 2010.

[11] M. Todorovic and N. Lopez-Benitez. Efficiency study of tcp protocols in infrastructured wireless networks. *Proceedings of International conference on Networking and Services*, 2006.

[12] S. Tsao, Y. Lai, and Y. Lin. Taxonomy and evaluation of tcp-friendly congestion-control schemes on fairness, aggressiveness, and responsiveness. *IEEE Network*, 2007.

[13] M. Welzl. Network congestion control - managing internet traffic. *John Wiley & Sons, Ltd.*, 2005.

[14] W. Xiuchao, C. Mun, A. Ananda, and C. Ganjihal. Sync-tcp: A new approach to high speed congestion control. *Proceedings of 17th IEEE International Conference on Network Protocols*, 2009.