A Novel Technique for TCP based Congestion Control

Martin Hruby, Michal Olsovsky and Margareta Kotocova
Institute of Computer Systems and Networks, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovicova 3, 842 16 Bratislava 4, Slovakia

Keywords: ACNS, Congestion Control, Flow Age, Network, Notification System, Performance, Prioritization, Priority, RTT, Queue, TCP, Throughput.

Abstract: Network performance increase does not have to be associated only with the upgrade of existing infrastructure and devices. More effective and less expensive increase of network performance can be achieved via sophisticated improvement of existing protocols, mainly at network and transport layer. The aim of this paper is to introduce our new approach for solving network performance issues which occur mostly due to congestion. New approach called Advanced Notification Congestion System (ACNS) allows TCP flows prioritization based on the TCP flow age and carried priority in the header of the network layer protocol. The main aim is to provide more bandwidth for young and high prioritized TCP flows by means of penalizing old greedy flows with a low priority. Using ACNS, substantial network performance increase can be achieved.

1 INTRODUCTION

The very first version of the most common transport protocol TCP was introduced in RFC793 (Information Sciences Institute, 1981). 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 (Ha, 2008), (Bateman, 2008).

First TCP improvements focused on higher performance were published in (Mirza, 2010). Since 1992 (Karnik, 2005), 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 (Xiuchao, 2009), (Chao, 2005), (Tsao, 2007) and JTCP, TCP CERL and Compound TCP+ for wireless networks (Todorovic, 2006), (Welzl, 2005), (Botta, 2007). All these variants have one thing in common – they don’t make any steps for congestion avoidance unless the congestion is detected by TCP end nodes (Martin, 2003). 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 (Kwon, 2002).

However this feedback is sent within all existing TCP flows which support ECN apart from any flow characteristic (Kuzmanovic, 2003). 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:

1. All TCP flows in the network from the TCP end nodes point of view are treated equally. It means that TCP end nodes do not observe any kind of prioritization or any kind of penalization with other TCP flows.
2. There is only one command (decrease 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.

36

Hruby M., Olsovsky M. and Kotocova M..
A Novel Technique for TCP based Congestion Control.
DOI: 10.5220/0004504100360043
In *Proceedings of the 4th International Conference on Data Communication Networking, 10th International Conference on e-Business and 4th International Conference on Optical Communication Systems* (DCNET-2013), pages 36-43
ISBN: 978-989-8565-72-3
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

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 parameters:

1. TCP flow age - TCP flows can exist in network for various time; old TCP flows are probably greedy data flows while young flows can be with certain probability classified as flows which need to exchange just small amount of data (e.g. short HTTP communication).
2. TCP flow priority - while bearing in mind the age of TCP flows sometimes it's necessary not to penalize old TCP flows. Therefore the priority carried in the network layer protocol header is considered as second parameter which influences the final weight of appropriate TCP flow.
3. Queue length – as the penalization is worthy only when there is a congestion in the network, the last flow weight calculation input parameter is the actual 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 section 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 (Section 2.2).

2.1 Flow Weight Calculation

While the TCP communication passes nodes in the network, it's 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} (2). Using normalization age values can vary from 0 to 1 (including). Comparison of standard and normalized age is depicted on the Figure 1 (normalized age values are shown 100 times larger).

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

$$f_{\max} = \max(F) \tag{2}$$

Similar normalization is used for the second parameter – priority p. Priority normalization is done within the function $\mathcal{F}(p)$ using maximal priority p_{max}. The last input parameter, actual queue length $\mathcal{A}(q)$, 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 ∂ for flow f_i used for command determination can be calculated using (3) where $\mathcal{F}(p)$ stands for priority part and $\mathcal{T}(f)$ represents age part. Both subparts can be calculated using (4). 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.

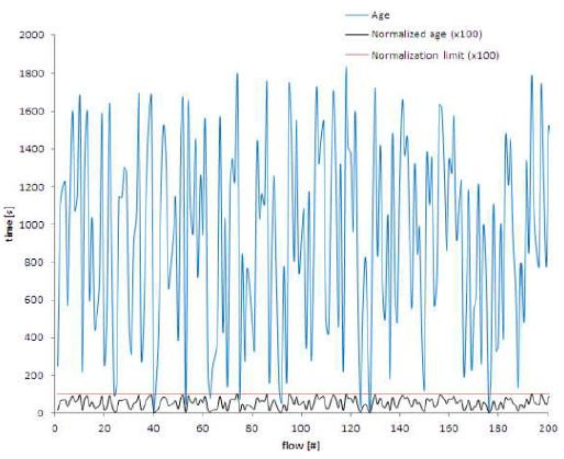

Figure 1: Age and normalized age.

Calculated weight ∂ needs to be transformed into command ω which will be sent to the TCP end nodes. The command can be 1, 2 or 3. As the calculated weight ∂ is a real number, we have to convert it to one of available commands using comparison with 2 thresholds th_{A1} and th_{A2} (5).

$$\partial = \frac{\mathcal{A}(q)}{\mathcal{F}(p) + \mathcal{T}(f)} \tag{3}$$

$$\mathcal{F}(p) = \frac{v_p * p}{p_{max}};\ \mathcal{T}(f) = \frac{v_a}{age(f)} \tag{4}$$

$$\omega = \begin{cases} 1 & \partial < th_{A1} \\ 2 & th_{A1} \leq \partial < th_{A2} \\ 3 & th_{A2} \leq \partial \end{cases} \tag{5}$$

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 ω_{trend} directly in the TCP end nodes.

At first end nodes calculate trend weight ∂_{trend} using ω_{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 (Cheng, 2004) (6) with additional step parameter σ (Malagò, 2011). Calculated metric for every received command is normalized using sum of metrics of all used commands χ (7). Setting P the array of all received commands the trend weight ∂_{trend} for specific TCP flow can be calculated using (8).

Later on calculated trend weight ∂_{trend} needs to be transformed into trend command ω_{trend} which will be used by end node itself. Calculation is similar to the calculation for standard command as in (5), 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.

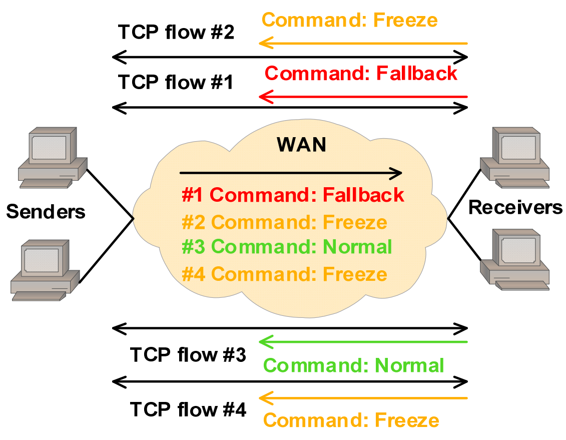

Figure 2: ACNS overview.

$$x = e^{-\frac{\lambda}{\sigma}} \tag{6}$$

$$\chi = \sum_{\lambda=1}^{\omega_{count}} e^{-\frac{\lambda}{\sigma}} \tag{7}$$

$$\partial_{trend} = \sum_{\lambda=1}^{\omega_{count}} \frac{e^{-\frac{\lambda}{\sigma}}.P[\lambda]}{\chi} \tag{8}$$

2.3 Commands

TCP end node can modify congestion window according to one of six commands. Half of these commands can be received within TCP acknowledgements and half needs to be calculated. Commands usage overview is shown in Figure 2.

Received commands:

1. $\omega = 1$ – "*normal*" - There is no congestion in the network. Congestion window can be modified according to the used TCP variant.
2. $\omega = 2$ – "*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 $cwnd$ (10).
3. $\omega = 3$ – "*fallback*" - There is a congestion in the network. After receiving, congestion window won't be decreased by multiplicative factor however it will be decreased to the latest known smaller value (11).

Calculated commands:

1. $\omega_{trend} = 1$ – „*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. $\omega_{trend} = 2$ – „*fallback*" – Loss occurred within indicated incoming congestion. Congestion window will be decreased to the latest known smaller value. Command of 3 is put in the list of received commands P.
3. $\omega_{trend} = 3$ – „*decrease*" - Loss occurred within ongoing congestion. Congestion window will be reduced according to 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 sections.

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 (9) where $cwnd_{last}$ is defined in (10) and $cwnd_{last\_x}$ in (11). 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.

$$cwnd = \begin{cases} cwnd_{TCP\_variant} & \omega = 1 \\ cwnd_{last} & \omega = 2 \\ cwnd - cwnd_{last\_x} & \omega = 3 \end{cases} \quad (9)$$

$$cwnd_{last} = W(t\text{-}1). \quad (10)$$

$$cwnd_{last\_x} = W(t - x) \quad (11)$$

$$cwnd = \begin{cases} cwnd_{last} & \omega_{trend} = 1 \\ cwnd_{last\_x} & \omega_{trend} = 2 \\ cwnd_{TCP_{var.}} & \omega_{trend} = 3 \end{cases} \quad (12)$$

Using self-calculated trend commands end nodes are able to modify congestion window as defined in (12). After receiving command of 3 end node will decrease the congestion window according to used TCP variant.

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 agree on using system which both of them support. New ACNS commands will appear as ECN commands for non-compatible nodes.

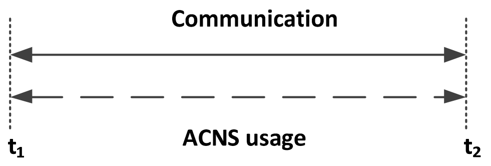

Figure 3: Network layer – ACNS usage.

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) 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. From the network layer point of view the whole communication is considered as one phase because it is not divided into 3 phases as TCP does. While keeping in mind network layer, ACNS can be used during the whole communication (Figure 3).

Similar reuse appears in the TCP header. Combination of existing ECN bits and new bit from the *reserved* field called *CMT* allows us to encode and decode all necessary ACNS messages (Table 2). From the transport layer point of view the whole communication is divided into 3 phases – connection establishment, data exchange and connection termination (Figure 4). Usage of ACNS system will be agreed during the connection establishment (three-way handshake). TCP sender will offer ACNS system within *ACNS-setup SYN* packet (flags SYN=1, ACK=0, ECE=1, CWR=1, CMT=1). If TCP receiver supports ACNS, it will reply with *ACNS-setup SYN-ACK* packet (flags SYN=1, ACK=0, ECE=1, CWR=0, CMT=1) (Figure 5). TCP end nodes agree on using ACNS during data exchange however they won't use ACNS during the initial phase because ACNS does not apply to control packets. While using ACNS during data exchange, TCP end nodes set appropriate bits in IPv4 and TCP header according to the used message. ACNS is used during the whole data exchange until the connection termination phase (Figure 6).

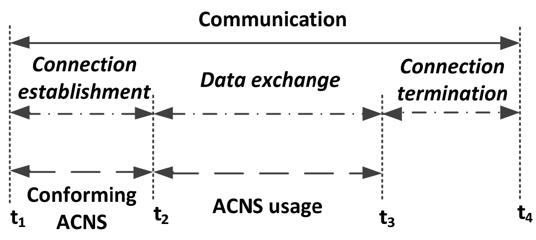

Figure 4: Transport layer – ACNS usage.

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.

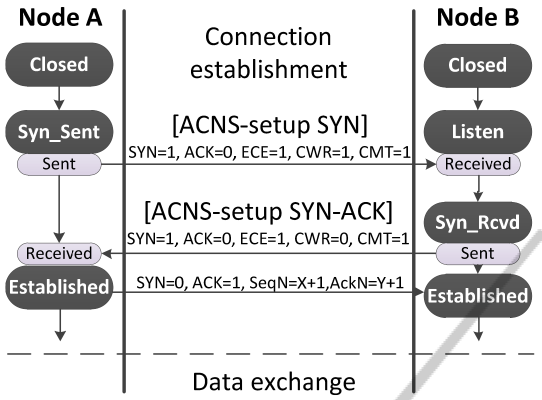

Figure 5: Connection establishment - Conforming ACNS.

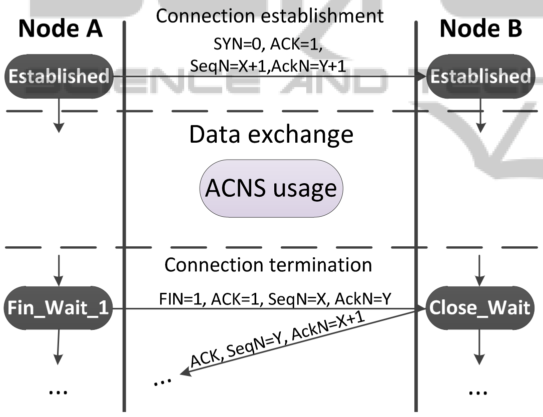

Figure 6: Data exchange - ACNS usage.

From the nodes in the network point of view ACNS resides in the TCP end nodes and in the routers as well. To sum it up, TCP end nodes use ACNS for:

- Messages encoding.
- Messages decoding.
- Modifying TCP congestion window
- Command self-calculation upon packet loss

Nodes in the network (routers) use ACNS for:

- TCP flows classification
- Messages encoding.
- Messages decoding.
- Commands calculation

ACNS messages used within IPv4 and TCP headers are summarized in the following tables (Table 1 and Table 2).

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*	ACNS message: command normal (receiver), NS = 0
2	*0*	*0*	*0*	ACNS message: command normal (receiver), NS = 1
3	*1*	*0*	*0*	ACNS message: congestion window reduced (sender), NS = 0
4	*1*	*0*	*0*	ACNS message: congestion window reduced (sender), NS = 1
5	*0*	*1*	*1*	ACNS message: command freeze (receiver)
6	*0*	*1*	*0*	ACNS message: command fallback (receiver)

4 SIMULATION RESULTS

System ACNS was implemented in the network simulator ns-2.35 where the simulations were performed as well. One of the most important implementation parts was the implementation of flow classification as this part required its own data structure for storing all necessary flow details as following (structure in Table 3):

- Hash (hash)
- Source IP address (sIP)
- Destination IP address (dIP)
- Source port (sPo)
- Destination port (dPo)
- Time of flow add (addTime)
- Time of last flow update (lastTime)
- Age (comAge)

- Priority (priority)
- ACNS compatibility (system)

Simulation topology used for simulations represents connections between 2 remote sites which are connected via Internet Service Provider (ISP). We assume that the bottlenecks do not exist in the ISP network (over provisioned links) however the "last-mile" links (used for connection to the ISP network) are willing to become bottlenecks during the communications.

High level overview of the simulation topology is shown in Figure 7. Detailed simulation topology is shown in Figure 8. The simulation consisted of 3 concurrent TCP flows and 3 concurrent UDP flows (detailed characteristic in Table 5 and Table 6). All TCP and UDP flows ended at simulation time of 118 seconds when the whole simulation ended. All UDP flows had priority of 0 (best-effort).

ACNS system parameters introduced in section 2 were set according to Table 4. Network parameters which were monitored during the simulations are throughput (maximal, average), RTT (maximal, average), amount of sent data and packet loss.

Table 3: TCP flows parameters.

#	hash	sIP	dIP	sPo	dPo
	addTime	lastTime	comAge	priority	system

Table 4: ACNS system parameters.

th_{L1}	th_{L2}	ω_{count}	σ	v_p	v_a	th_{A1}	th_{A2}
1,84	2,15	10	3	0,8	0,2	0,95	1,0

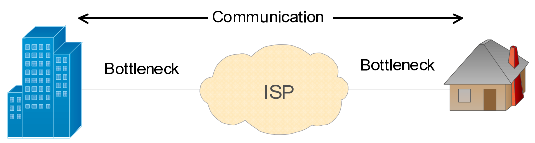

Figure 7: High level simulation topology overview.

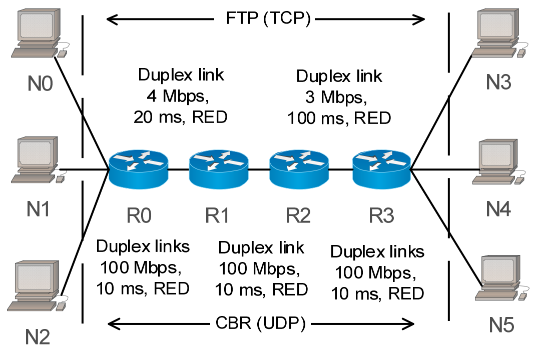

Figure 8: Detailed simulation topology.

Table 5: TCP flows parameters.

Flow	Variant	Priority	Time [s]	From	To
#1	CUBIC	0	0.1 - 118	N0	N5
#2	CUBIC	26	15.1	N0	N4
#3	CUBIC	46	30.1	N1	N3

Table 6: UDP flows parameters.

Flow	Bit rate [Mb/s]	Start [s]	From	To
#1	0.5	0.1	N0	N3
#2	0.6	20.1	N1	N4
#3	0.5	40.1	N2	N5

Comparison of achieved simulation results is shown in the following tables:

- Table 7 – average and maximal throughput
- Table 8 – average and maximal RTT
- Table 9 – amount of sent data
- Table 10 – packet loss.

Table 7: Simulation results - throughput.

#	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

Table 8: Simulation results - RTT.

#	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 9: Simulation results – sent data.

#	System	Amount of 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

Table 10: Simulation results – packet loss.

#	System	Loss [packets]			
		#1	#2	#3	Total
1	-	110	86	40	236
2	ACNS	0	2	0	2

For better illustration comparison of actual throughput and RTT changing in time is shown in the following figures. Figure 9 shows the changing actual throughput of all 3 TCP flows without ACNS system. Significant throughput changes at around 15th and 30th second are related to the start of new TCP flows. The same reason is responsible for significant throughput changes at around the 20th and 40th second where the next UDP flows started. Figure 10 shows the actual throughput of all 3 TCP flows with ACNS system active. Throughput changes around 15th, 20th, 30th and 40th second are related to the start of the new TCP and UDP flows.

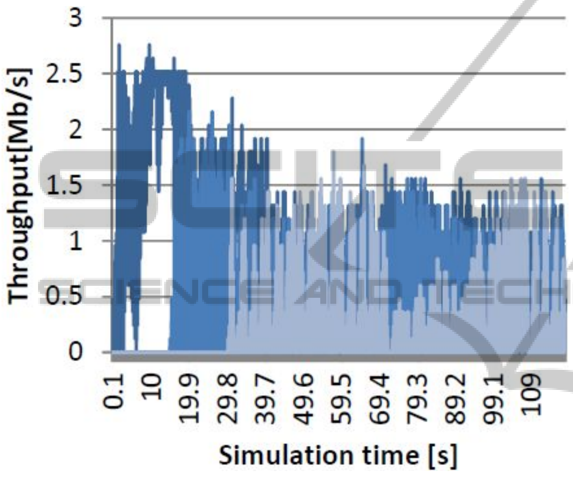

Figure 9: TCP flows throughput (without ACNS).

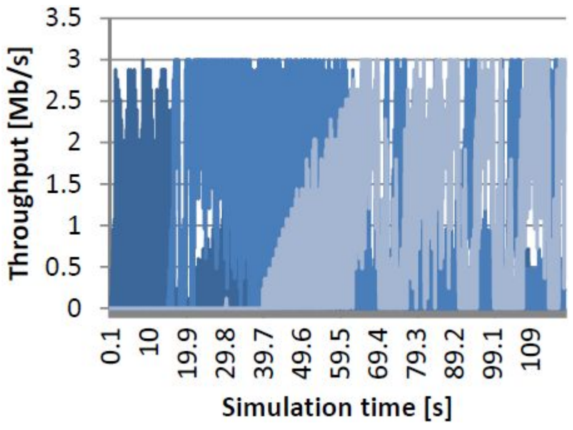

Figure 10: TCP flows throughput (with ACNS).

Figure 11 shows how the actual RTT of all 3 TCP flows was changing during the simulation without ACNS system. Using this picture we can conclude that RTT was changing due to the actual queue length. On the other hand Figure 12 shows the change of RTT while the ACNS system was in use. Once the ACNS was stabilized, the RTT decreased significantly and for the rest of the simulation RTT achieved lower values in comparison with simulation where ACNS system wasn't used.

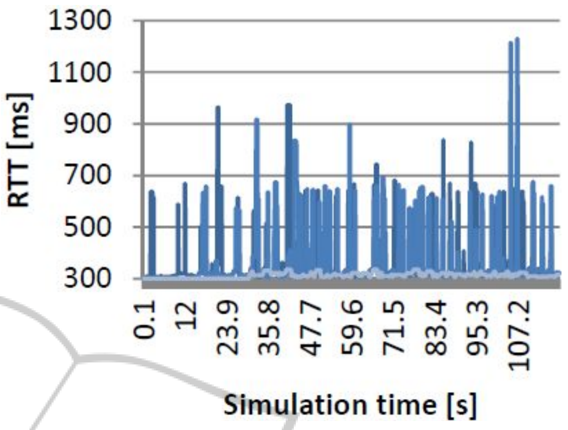

Figure 11: TCP flows RTT (without ACNS).

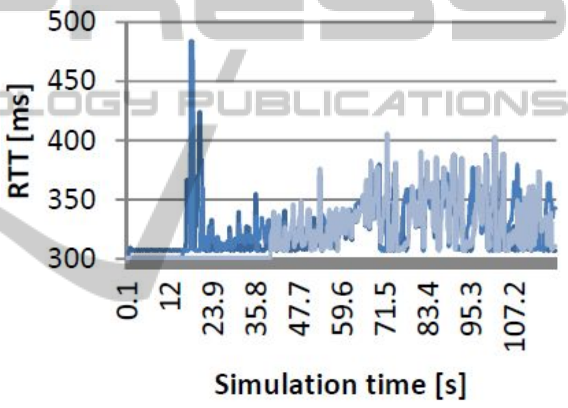

Figure 12: TCP flows RTT (with ACNS).

Table 11: Network performance improvements.

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

According to the simulations results, using ACNS it's possible to increase TCP flows throughput (Figure 9 - without ACNS, Figure 10 – 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 11 - without ACNS, Figure 12 – with ACNS) by 7%. Network performance improvements are summarized in Table 11. All these improvements were achieved with nearly none losses.

5 CONCLUSIONS

In this paper we have introduced an advanced notification system for TCP congestion control called ACNS. 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. 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

Information Sciences Institute, “Transmission Control Protocol”, *RFC 793*, 1981.

Ha, Sangtae, et al. “CUBIC: A new TCP-Friendly High-Speed TCP Variant”, *ACM SIGOPS Operating System Review*, V. 42, 5, 2008.

Bateman, M., et al., “A comparison of TCP behaviour at high speeds using ns-2 and Linux”. In *Proceedings of 11th ACM CNS '08*, 2008.

Mirza, M.; Sommers, J.; Barford, P.; "A Machine Learning Approach to TCP Throughput Prediction," Networking, *IEEE/ACM Transactions on*, vol.18, pp.1026-1039, Aug. 2010.

Karnik, A.; Kumar, A.; "Performance of TCP congestion control with explicit rate feedback," *Networking, IEEE/ACM Transactions on*, vol.13, pp. 108- 120, 2005.

Xiuchao, W., Mun, C.C., Ananda, A.L. and Ganjihal, C., “Sync-TCP: A new approach to high speed congestion control”, In *17th IEEE International Conference on Network Protocols*, 2009. ICNP 2009.

Todorovic, M. and Lopez-Benitez, N., “Efficiency Study of TCP Protocols in Infrastructured Wireless Networks”, *In proceedings of International conference on Networking and Services*, ICNS '06, 2006.

Welzl, M. “Network Congestion Control - Managing Internet Traffic“, John Wiley & Sons, Ltd. 2005. ISBN 978-0-470-02528-4.

Chao, H.J. and Guo, X. “Quality of Service Control in High-Speed Networks“, John Wiley & Sons, Ltd. 2005. ISBN 0-471-00397-2.

Botta, A., Dainotti, A. and Pescape, A., “Multi-protocol and multi-platform traffic generation and measurement”, *INFOCOM 2007 DEMO Session*, May 2007.

Tsao, S., Lai, Y. and Lin, Y., “Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness”, In *journal IEEE Network*, 2007.

Martin, J., Nilsson, A. and Rhee, I., “Delay-based Congestion Avoidance for TCP”, *IEEE/ACM Transactions on Networking*, June 2003.

Malagò, L.; Matteucci, M.; “Towards the geometry of estimation of distribution algorithms based on the exponential family”. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*. New York, pp. 230-242, 2011.

Kwon, M.; Fahmy, S.; "TCP increase/decrease behaviour with explicit congestion notification (ECN)," *IEEE International Conference on Communications*, ICC 2002, vol.4, pp. 2335- 2340, 2002.

Cheng Jin, David X. Wei, and Steven H. Low, "FAST TCP: Motivation, Architecture, Algorithm, Performance", *INFOCOM - Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, pp.2490-2501, 2004.

Kuzmanovic, A. and Knightly, E. W., “TCP-LP: Low-Priority Service via End-Point Congestion Control”, *IEEE/ACM Transactions on Networking*, 2003.