Advanced notification system for TCP performance increase

Michal Olšovský*
Institute of Computer Systems and Networks
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, 842 16 Bratislava, Slovakia
olsovsky@fiit.stuba.sk

Abstract

Network throughput increase is usually associated with replacement of communication links and appropriate network devices. This approach is usually intrusive, expensive and nevertheless time consuming. On the other hand 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 - especially network and transport layer protocols.

In this paper we propose an advanced notification system for TCP congestion control called ACNS (Advanced Congestion Notification System). This new approach allows TCP flows prioritization based on the flow age and carried priority. The aim of this approach is to penalize old and 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. Using ACNS specific TCP end nodes can be informed how to modify the size of their congestion window. This allows us to notify TCP end nodes sooner than the congestion results in packet loss. By means of penalizing and prioritizing specific TCP flows significant improvement of network throughput can be achieved.

Categories and Subject Descriptors

C.2 [**Computer-communication Networks**]: Network Protocols; C.2.1 [**Computer-communication Networks**]: Net work Architecture and Design

*Recommended by thesis supervisor: Assoc. Prof. Margaréta Kotočová. Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on November 25, 2013.

Olšovský, M. Advanced Notification System for TCP Performance Increase. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 6, No. 2 (2014) 1-10

Keywords

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

1. Introduction

Nowadays it is well established that TCP and UDP are the dominant transport protocols. TCP apart from UDP offers reliable communication between two end nodes over unreliable IP communication channel. As a result TCP is mainly used for reliable data transfers and is called as one of the core protocols of the Internet protocol suite [23]. Main TCP features are reliability, full duplex, flow control, multiplexing, connection oriented and variable transfer speed [16, 22].

The very first version of the most common transport protocol TCP was introduced in RFC793 [8]. First well established TCP variant was introduced in 1988 and was called TCP Tahoe. TCP Tahoe includes three key techniques: slow start, congestion control and fast retransmission. 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 (protocol stack) as well. From the TCP point of view these improvements were related to new key techniques as well as to changes in congestion control. Improvements of the TCP, mostly called TCP variants or extensions, are mainly focused on the best and most effective usage of available communication lines [3, 6].

2. Related work

First TCP improvements focused on higher performance were published in [16]. Since 1992 [25], there have been many new TCP variants which were aimed to increase TCP performance. In general we can recognize two ways of dealing with congestion and congestion control. First way uses implicit feedback gathered by the TCP end nodes that do not use any additional system but use only information gathered during their own observation of the network and congestion in the network.

2.1 Wired networks

TCP variants using implicit feedback can be divided into two main groups based on the end network type. Thus we can recognize TCP variants suitable for wired access networks and TCP variants for wireless access networks. Moreover each of these two groups can be further divided

based on the congestion detection [14]. One subgroup of TCP variants uses packet loss as a sign of existing and ongoing congestion in the network. This detection is reliable however on the other hand TCP variants using this detection are characteristic with worse fairness, especially in competition with TCP variants using increased delay as a sign of congestion. The reason is that few moments before the congestion queues are getting filled up, packets spent wait more time in the queues and thus the whole round-trip time (RTT) is increased. At this point second group of TCP variants using increased latency in the networks (result of full queues) is detecting congestion and slows down the transmission (decreases the size of the congestion window) However TCP variant from the first group have not detected any packet loss so far so they are not aware of any upcoming congestion. As a result they will not slow down but moreover they will increase the size of the congestion window.

Main packet loss based TCP variants are TCP Reno, TCP NewReno, BIC, CUBIC and SQUARE [6, 1]. Important delay based TCP variants are TCP Vegas, TCP Illinois, SyncTCP and NF-TCP [24, 15]. Apart from above clearly defined groups there are some TCP variants which used combine approach - Fast TCP, Hamilton TCP, Compound TCP and RADIC-TCP [9].

2.2 Wireless networks

Original TCP was developed for wired networks therefore the use of this protocol in wireless environment often leads to performance degradation. The reason behind this low performance lays in the original TCP design where it was supposed that every packet loss would be a result of congestion and not a result of errors in communication channel. However as it well known wireless environment is especially susceptible to signal interference which leads to higher bit error rate. As a result damaged packets are dropped by lower OSI model layers and TCP reacts to these drops of faulty packets as to standard drops due to congestion. TCP variants for wireless networks are trying to minimize this issue. We can recognize the same division into packet loss based, delay based and combined TCP variants.

Main packet loss based TCP variants are TCP Westwood, TCP Hybla and TCP Jersey[7]. Important delay based TCP variants are JTCP and EJTCP [13]. Apart from above clearly defined groups there are some TCP variants which used combine approach and thus need separate category - TCP Veno, TCP Cerl, Compound TCP+ [20, 17].

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 [5].

2.3 Explicit feedback

More accurate reaction to upcoming congestion can be achieved using some kind of notification system which informs TCP end nodes about the situation in the network using the information gathered from the nodes in the network. Similar approach can be achieved while using Explicit Congestion Notification (ECN) system. TCP end nodes allow nodes in the network to inform them about the situation in the network and give them some kind of feedback.

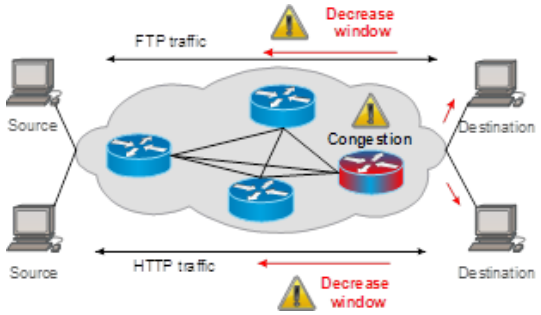

Figure 1: ACNS overview.

Feedback is transferred within IP/TCP headers and is encoded into these headers instead of dropping such a capable packets. As a results TCP sender will decrease the congestion window. 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 size of the congestion window as it is shown in Figure 1[10, 12].

2.4 Problem identification

While keeping in mind existing ECN system and the behavior of TCP variants we have identified two limitations.

Firstly all TCP flows in the network from the TCP end nodes point of view are treated equally. It means that in case of congestion all TCP flows are penalized by means of receiving order to decrease the congestion window. Their characteristics like their age or priority are not considered which means that even short TCP flows or flow with higher priority are treated equally with old and greedy flows.

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. We believe that additional command will fill the gap between decreasing and increasing congestion window and will help to improve the TCP performance in general. Mechanisms and further concepts introduced in the following chapters are aimed to solve highlighted issues.

3. Advanced notification system model

The idea of our new approach ACNS (Advanced Congestion Notification System) is to allow the TCP and the networks itself to inform only specific TCP end nodes about the congestion in the network and instruct them to change the congestion window by means of decreasing or increasing its size. Such functionality will provide more bandwidth to younger and prioritized TCP flows by freezing and possibly decreasing the congestion window of older and greedy TCP flows (more details about short and long age TCP flows like their avarege age, number of packets exchanged, amount of data sent can be found in [2, 21]). We propose a set of weights assigned to each TCP flow for further calculations which will result into a specific command that will be sent within particular flows.

Weight of the TCP flow is based on the following three parameters:

1. TCP flow age.

2. TCP flow priority.
3. Queue length.

TCP flow age reflects to the time how long the TCP flow has been observed in the network. TCP flow priority stands for the priority carried in the header of network layer protocol. As the congestion in the network is usually associated with the fulfilling queues therefore we have chosen queue length as our last key parameter.

ACNS can work in two modes. First mode is a 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 for encoded command and modify the congestion window according to the decoded command. The mechanism of TCP flow weight calculation and command determination is described in the next chapter 3.1. Second situation represents packet loss in the network. From the TCP end node point of view it doesn't matter where the packet loss happened because the result will be always the same - end node will not receive the command from the network. Therefore the end node has to determine which command needs to be used and this will be done using the commands received in the recent past (chapter 3.2).

3.1 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 back to the TCP 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). Result of flow age normalization is shown in Figure 2.

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

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) and (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.

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}.

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

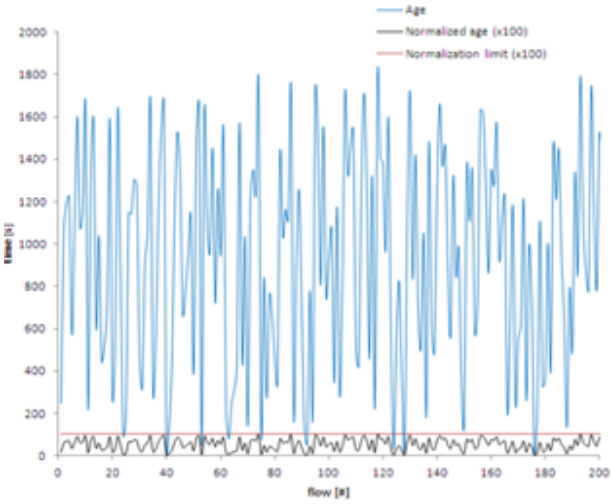

Figure 2: Flow age normalization.

$$T(f) = v_a * age(f_i) \tag{3}$$

$$F(p) = \frac{v_p * p_{max}}{p_i} \tag{4}$$

3.2 Determining command upon packet 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.

At first TCP end node calculates 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 [4, 13].

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 (5).

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.

$$W_{trend} = \sum_{i=1}^{\omega_{count}} \frac{e^{-\frac{i}{\sigma}} \cdot P[i]}{X} \tag{5}$$

3.3 Commands overview

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 3.

Received commands (commands calculated by nodes in the network):

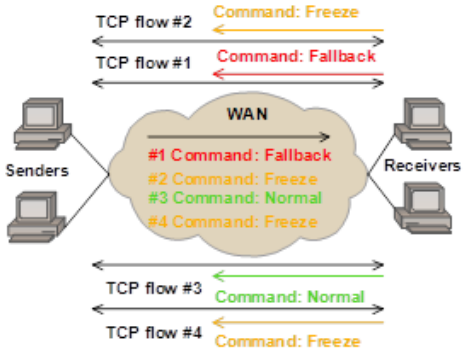

Figure 3: ACNS overview.

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 will receive only specific TCP end nodes, not all TCP flows are penalized. After receiving, TCP end nodes freeze their congestion window.

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 (commands calculated by TCP end nodes):

1. $C_{trend} = 1$: "*freeze*". Loss occurred without any sign 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.

3.4 Model verification

According to the testbed introduced in [21] together with [19] we have performed basic model verification. The tests were aimed at verifying basic functionality like commands calculation and the impact of the feedback from nodes in the network. Depicted figures with the simulation results prove the functionality of ACNS feedback from nodes in the network as well as the impact of the carried priority on the whole prioritization process. ACNS system parameters were set according to Table 1. For testing purposes we have randomly generated TCP flow lifetimes for an average of 30 active TCP flows and sampled them over a period of 50 consecutive evenly-spaced measurement periods.

Table 1: 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

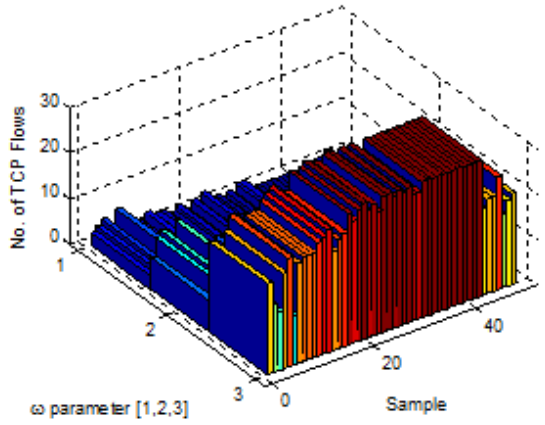

Figure 4: Randomly generated TCP flows without active ACNS.

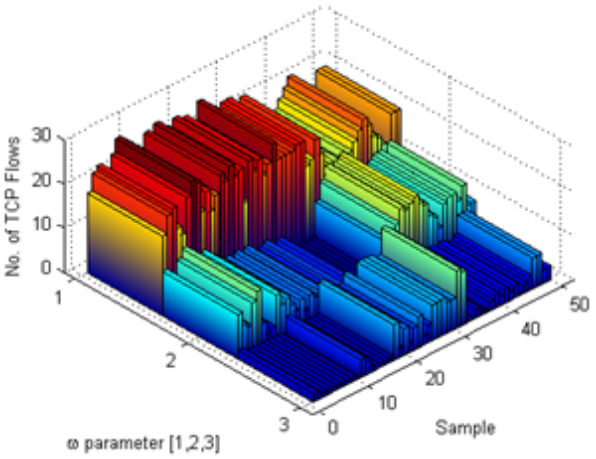

Figure 5: Randomly generated TCP flows with active ACNS.

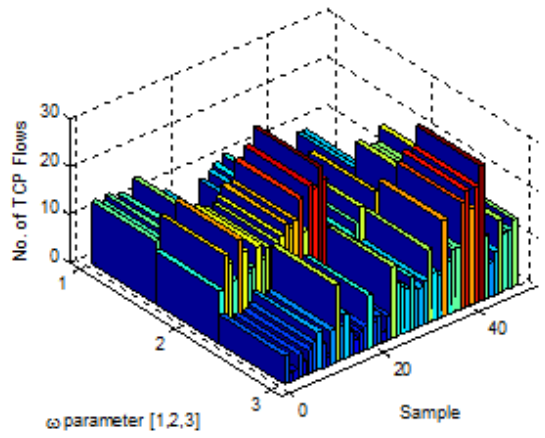

Figure 6: Randomly generated TCP flows with active ACNS and priority of 0.

In the example depicted in Figure 4 we have modeled 30 randomly generated TCP flows without feedback of our congestion prevention ACNS system. As can be seen around sample time 20 a congestion is occurring and the network does not recover for a long period of time until many flows are being dropped around time sample 40. In the example depicted in Figure 5 we have modeled 30 randomly generated TCP flows, with normal distributed priorities. The commands are now being fed back to the end nodes. The results show a slowly increasing tendency of the C parameter as the queue fills up to the point where at around sample number 30 the number of flows with C=3 rises dramatically. The command to decrease the congestion window is fed back to our model and subsequent samples show the effect.

In Figure 6 we used the same model and parameters, but we changed the TCP flow priority distribution uniformly to zero (DSCP zero is considered as best effort). The scenario depicted in Figure 6 illustrates a different distribution of the C parameter, as the queue fills up. With the increasing number of TCP flows assigned C=2 (samples 1 to 25) the congestion is not avoided and the number of flows assigned C=3 increases.

The gradual congestion window decrease afterwards prevents congestion. In this situation all flows have a priority of zero and only TCP flow age is considered. Thereafter flows with lower priorities would be considered for congestion window decrease much sooner than those with higher priorities

4. Integration into protocol stack

In order to use system ACNS in real network it is necessary to modify network layer (IPv4/IPv6) and transport layer (TCP) protocols while keeping the existing header structure of these protocols.

Our approach keeps full backward compatibility with existing IPv4/IPv6 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.

4.1 Network layer

From the network layer point of view ACNS supports multiprotocol environment. According to current standards two version of IP protocol can be identified. The main design part is focused on IPv4 due to its ongoing popularity.

As the design is the same for IPv6, separate section for IPv6 is included as well. The goal of this section is to explain how can be one additional bit required for ACNS functionality acquired in the IPv6 header.

4.1.1 IPv4

Integration in IPv4 header lies in reusing ECN bits with one additional unused bit from field Flags called CMI (Figure 7). Routers are willing to encode more important commands into IPv4 header (Table 2 - NS stands for nonce sum flag) and overwrite existing ones. TCP sender uses messages I2/I3 within one data window. When sending last packet from specific data window, sender uses messages I4/I5 in order to ask routers to encode command in the IPv4 header (saves routers system resources). Messages I6 and I7 are created only by routers.

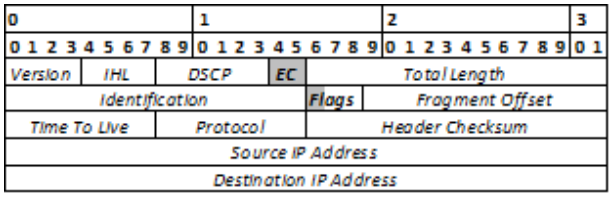

Figure 7: Bits used in IPv4 header.

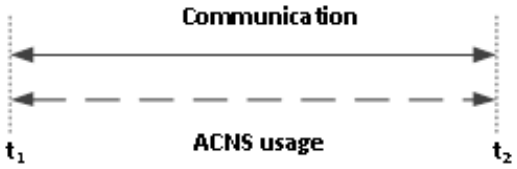

Figure 8: ACNS usage at network layer.

From the network layer point of view the whole communication is considered as one phase because it is not divided into phases as TCP does. While keeping in mind network layer, ACNS can be used during the whole communication (Figure 8).

4.1.2 IPv6

According to RFC2460 IPv6 header does not contain any Flags field. It means that we cannot assign CMI bit directly. In this case allocation of CMI bit will result in using additional nested header in IPv6 header. The disadvantage of this solution lies in the size of the nested header; in order to use one CMI bit another nested header with size of at least eight bytes will have to be used [18].

Different approach introduced in [11] suggests decreasing the size of the field Flow label from 20 bites to 17 bites. Saved three bites can be reused as Flags field where the CMI bit can be allocated directly as in IPv4 header without any overhead. The disadvantage of this approach is its low support across network devices.

4.2 Transport layer

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.

4.2.1 Change upon recipient of acknowledgement

End nodes can receive three different commands within the acknowledgement. Together with these three commands available congestion window (cwnd) changes are defined in (6) 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.

Table 2: ACNS messages encoded in IPv4 header.

#	ECN		CMI	Message
I1	*0*	*0*	*0*	ACNS not supported
I2	*1*	*0*	*0*	ACNS in ECN mode (set by end node), ACNS message: command normal (left by routers), NS = 0
I3	*0*	*1*	*0*	ACNS in ECN mode (set by end node), ACNS message: command normal (left by routers), NS = 1
I4	*1*	*0*	*1*	ACNS supported (set by end node), ACNS message: routers to set command, NS=0
I5	*0*	*1*	*1*	ACNS supported (set by end node), ACNS message: routers to set command, NS=1
I6	*1*	*1*	*1*	ACNS message: command freeze (set by routers)
I7	*1*	*1*	*0*	ACNS message: command fallback (set by routers)

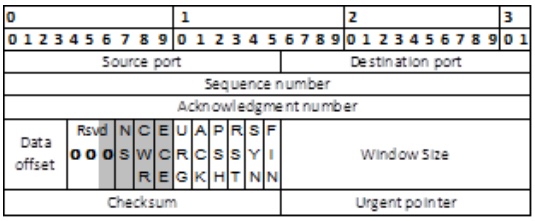

Figure 9: Bits used in TCP header.

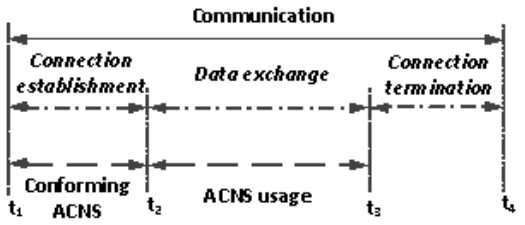

Figure 10: ACNS usage at transport layer.

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

4.2.2 Change upon loss

Using self-calculated trend commands end nodes are able to modify congestion window as defined in (7). After receiving command of 3 end node will decrease the congestion window according to used TCP variant.

$$cwnd = \begin{cases} cwnd_{last} & C_{trend} = 1 \\ cwnd_{last\_x} & C_{trend} = 2 \\ cwnd_{TCP} & C_{trend} = 3 \end{cases} \quad (7)$$

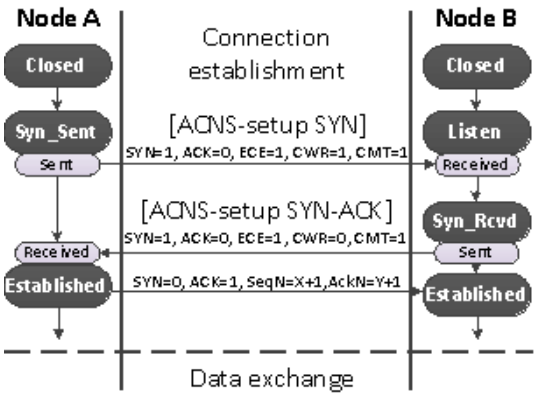

Figure 11: Conforming ACNS during connection establishment.

4.2.3 TCP

Integration in TCP header lies in reusing existing ECN bits and new bit from the reserved field called CMT (Figure 9). These bits allow us to encode and decode all necessary ACNS messages (Table 3 - NS stands for nonce sum flag). From the transport layer point of view the whole communication is divided into 3 phases - connection establishment, data exchange and connection termination (Figure 10). 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 11).

TCP end nodes agree on using ACNS during data exchange however they will not 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

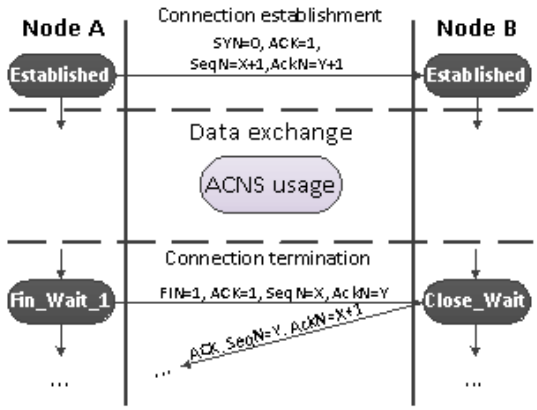

Figure 12: Conforming ACNS during data exchange.

Table 3: ACNS messages encoded in TCP header.

#	CWR	ECE	CMT	Message
T1	0	*0*	*0*	command normal (R), NS=0
T2	*0*	*0*	*0*	command normal (R), NS=1
T3	*1*	*0*	*0*	congestion window reduced (S), NS=0
T4	*1*	*0*	*0*	congestion window reduced (S), NS=1
T5	*0*	*1*	*1*	command freeze (R)
T6	*0*	*1*	*0*	command fallback (R)

Figure 13: High level simulation topology overview.

the used message. ACNS is used during the whole data exchange until the connection termination phase starts (Figure 12).

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 T1/T2 in order to signalize normal congestion window processing. In case of upcoming congestion, TCP receiver can inform TCP sender with message T5 in order to freeze congestion window or with message T6 to apply command fallback. All messages from Table 2 are used only by TCP end nodes.

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.

5. ACNS evaluation

System ACNS was implemented in the network simulator ns-2 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: Hash, Source IP address, Destination IP address, Source port, Destination port, Time of flow add, Time of last flow update, Age, Priority and ACNS compatibility.

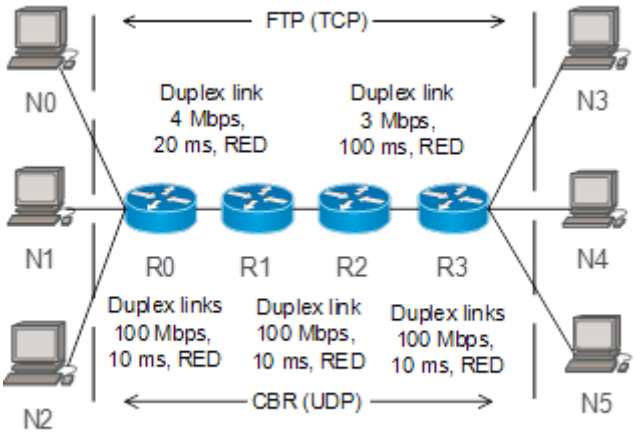

Figure 14: Detailed simulation topology.

Table 4: 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

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 13. Detailed simulation topology is shown in Figure 14. The simulation consisted of 3 concurrent TCP flows and 3 concurrent UDP flows (detailed characteristic in Table 4 and Table 5). All TCP and UDP flows ended at simulation time of 118 seconds when the whole simulation ended. All UDP flows had priority set to 0 (best-effort).

ACNS system parameters were set according to Table 1. Network parameters which were monitored during the

Table 5: 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 6: 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 7: 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

simulations are throughput (maximal, average), RTT (maximal, average), amount of sent data and packet loss.

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

- Table 6 - average and maximal throughput,
- Table 7 - average and maximal RTT,
- Table 8 - amount of sent data,
- Table 9 - packet loss.

For better illustration the comparison of actual throughput and RTT changing in time is shown in the following figures. Figure 15 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 16 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 17 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 18 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 was not used.

According to the simulations results, using ACNS it is possible to increase TCP flows throughput (Figure 15 - without ACNS, Figure 16 - with ACNS) by 44 % which

Table 8: 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 9: Simulation results - packet loss.

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

Table 10: Network performance improvements.

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

lead to increased amount of sent data (44 % increase). Using our new approach TCP flows RTT can be decreased (Figure 17 - without ACNS, Figure 17 - with ACNS) by 7 %. Network performance improvements are summarized in Table 11. All these improvements were achieved with nearly none losses.

6. Conclusions

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. In the future, we would like to focus on the cooperation with IPv6 and further testing in wireless environment.

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] I. Abdeljaouad and H. Rachidi. Performance analysis of modern tcp variants: A comparison of cubic, compound and newreno. *Proceedings of 25th Biennial Symposium on Communications*, pages 80–83, 2010.

[2] L. Arshadi and A. Jahangir. The tcp flow inter-arrival times distribution. *Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS)*, pages 360–365, 2011.

[3] 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.

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

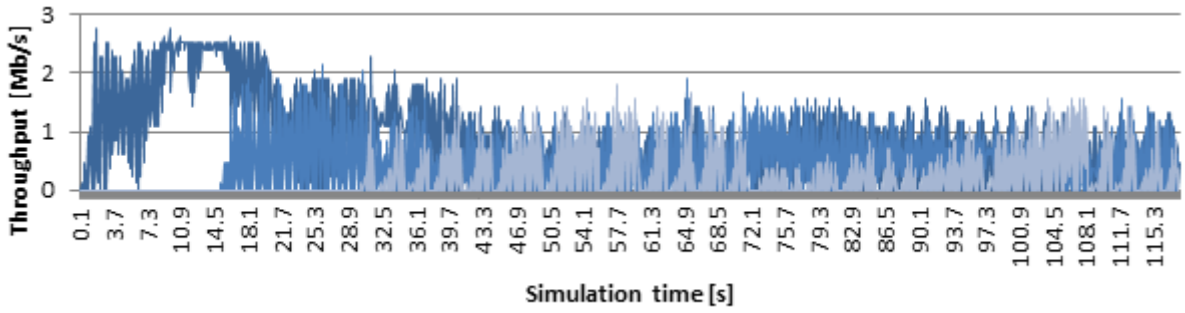

Figure 15: TCP flows throughput (without ACNS).

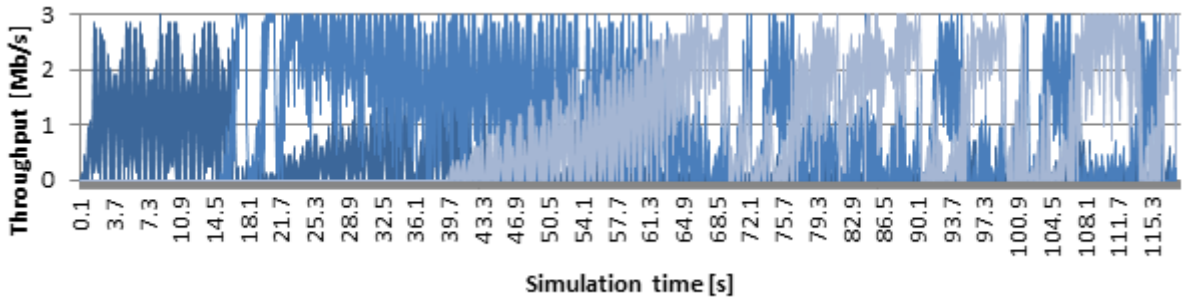

Figure 16: TCP flows throughput (with ACNS).

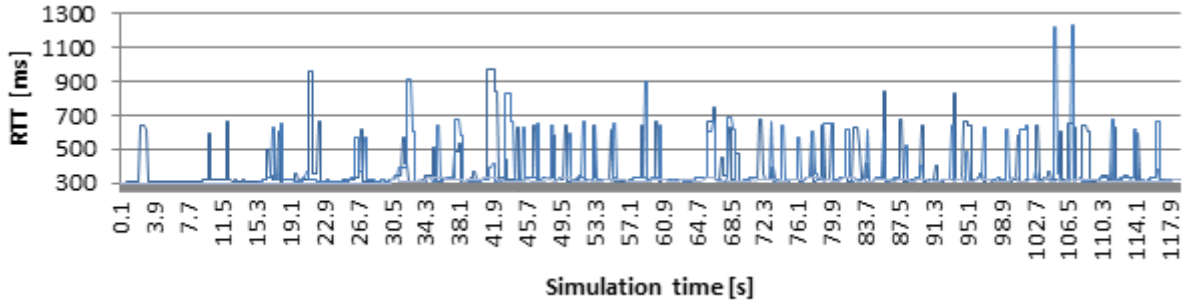

Figure 17: TCP flows RTT (without ACNS).

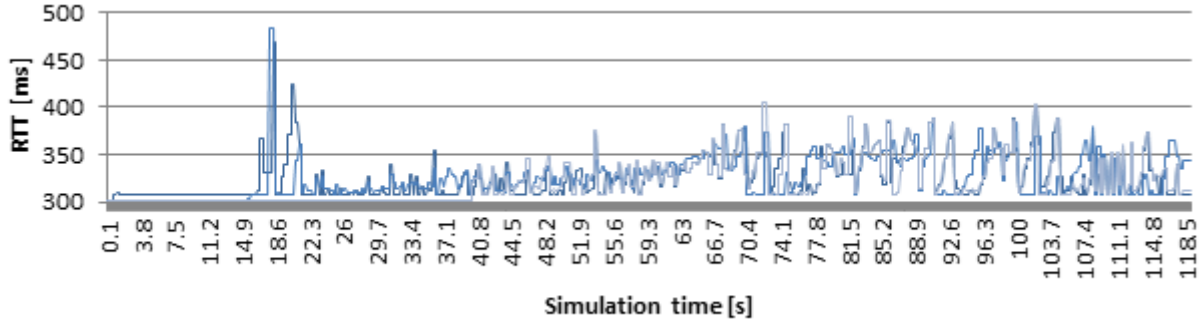

Figure 18: TCP flows RTT (with ACNS).

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

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

[7] S. Henna. A throughput analysis of tcp variants in mobile wireless networks. *Proceedings of the Third International Conference on Next Generation Mobile Applications, Services and Technologies*, 2009.

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

[9] T. Isobe and D. Akashi. Radic-tcp: High-speed protocol applied for virtual private wan. *18th International Conference on Telecommunications (ICT)*, pages 505–510, 2011.

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

[11] S. Krishnan and J. Halpern. Reserving bits in the ipv6 header for future use. *6man Working Group, Internet-Draft*, 2010.

[12] 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.

[13] 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.

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

[15] X. Miao and Q. Feng. Fairness evaluation of the default highspeed tcp in common operating systems. *Proceedings of IC-BNMT2009*, pages 100–105, 2009.

[16] 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.

[17] H. Oda and H. Hisamatu. Compound tcp+ for fairness improvement among compound tcp connections in a wireless lan. *IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, 2010.

[18] H. Rafiee and C. Meinel. Ipv6 deployment and spam challenges. *Internet Computing, IEEE*, pages 22–29, 2012.

[19] L. Roychoudhuri and G. Brewster. On studying the impact of the internet delays on audio transmission. *IEEE Workshop on IP Operations and Management*, pages 208–213, 2002.

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

[21] B. Trammel and D. Schatzmann. Flow concurrency in the internet and its implications for capacity sharing. *Proceedings of the CSWS '12 ACM workshop on Capacity sharing*, pages 15–20, 2012.

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

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

[24] 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.

[25] L. Yee-Ting and D. Leith. Experimental evaluation of tcp protocols for high-speed networks. *IEEE/ACM Transactions on Networking*, 15:1109–1122, 2007.

Selected Papers by the Author

M. Hrubý, M. Olšovský, M. Kotočová. Routing VoIP traffic in Large Networks. In Proceedings of the World Congress on Engineering, WCE 2012, 4-6 July, 2012 Imperial College London, London, U.K, Vol. II. Hong Kong: International Association of Engineers, 2012, pp. 798-803. ISBN 978-988-19252-1-3, 2012.

M. Olšovský, M. Kotočová. TCP with Advanced Window Scaling Option. Lecture Notes in Electrical Engineering - Emerging Trends in Computing, Informatics, Systems Sciences and Engineering, vol. 151, pp. 629-636, ISBN 978-1-14614-3558-7, Springer, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. Algorithm for Dynamic Traffic Rerouting and Congestion Prevention in IP Networks. Lecture Notes in Electrical Engineering - Innovations and Advances in Computer, Information, Systems Sciences, and Engineering, vol. 152, pp. 649-660, ISBN 978-1-4614-3535-8, Springer, 2013.

M. Olšovský, M. Kotočová. TCP with Extended Window Scaling. Lecture Notes in Electrical Engineering - Innovations and Advances in Computer, Information, Systems Sciences, and Engineering, vol. 152, pp. 393-404, ISBN 978-1-4614-3535-8, Springer, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. An Iterative Statistical Method for Congestion Prevention in Transit Networks. In Proceedings of the IEEE 10th Jubilee International Symposium on Aplied Machine Intelligence and Informatics, SAMI 2012. Herľany, Slovakia, January 26-28, 2012, ISBN 978-1-4577-0195-5, 2012.

M. Hrubý, M. Olšovský, M. Kotočová. Methodology for Optimal Link Selection using Multivariant Normal Distribution. In Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2012, May 21-25, 2012, Opatija - Adriatic Coast, Croatia, pp. 577-582. ISBN 978-953-233-069-4, 2012.

M. Hrubý, M. Olšovský, M. Kotočová. Analytical Multiparameter Notification System for TCP Congestion Avoidance. In Proceedings of the Joint International Conferences on Advances in Information Technologies and Communication 2012: ICT, CIT, PECS and EMIE - 2012 Amsterdam, Netherlands, December 2012. Amsterdam: ACEEE, pp. 112-117. ISBN 978-94-91587-03-0, 2012.

M. Hrubý, M. Olšovský, M. Kotočová. Advances in TCP Congestion Prevention. In Proceedings of the IEEE 11th International Symposium on Applied Machine Intelligence and Informatics, SAMI 2013 : January 31- February 2, 2013 Herľany. Slovakia. Piscataway: IEEE, pp. 111–116. ISBN 978-1-4673-5926-9, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. Solving VoIP QoS and Scalability Issues in Backbone Networks. In IAENG Transactions on Engineering Technologies: Special Volume of the World Congress on Engineering 2012, Lecture Notes in Electrical Engineering, Vol. 229, Springer, ISBN 978-94-007-6190-2, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. Towards seamless TCP congestion avoidance in multiprotocol environments. In ACEEE International Journal on Communication 2013, IDES, ISSN 2158-7558 (Online) ISSN 2158-754X (Print). 2013.

M. Olšovský. Multiparameter TCP congestion control. In Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies, Vol. 5, No. 2 (2013) - Bratislava: STU, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. A Novel Technique for TCP Based Congestion Control. Proceedings of the International Conference on Data Communication Networking (DCNET 2013), Reykjavik, Iceland, 29-31 July 2013, ISBN: 978-989-8565-72-3, 2013.

M. Hrubý, M. Olšovský, M. Kotočová. Traffic Flow Optimization and Visualization in MPLS Networks. In Proceedings of the World Congress on Engineering, WCE 2013, 3-5 July, Imperial College London, London, U.K. Vol. II. Hong-Kong: International Association of Engineers, pp. 849–854. ISBN 978-988-19252-8-2, 2013.