ACEEE Int. J. on Communications, Vol. 4, No. 1, July 2013

Towards Seamless TCP Congestion Avoidance in Multiprotocol Environments

Martin Hruby, Michal Olsovsky, 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
Email: {hruby, olsovsky, kotocova}@fiit.stuba.sk

Abstract— **In this paper we explore the area of congestion avoidance in computer networks. We provide a brief overview of the current state of the art in congestion avoidance and also list our extension to the TCP congestion avoidance mechanism. This extension was previously published on an international forum and in this paper we describe an improved version which allows multiprotocol support. We list preliminary results carried out in a simulation environment.**

New introduced approach called Advanced Notification Congestion System (ACNS) allows TCP flows prioritization based on the TCP flow age and priority carried in the header of the network layer protocol. The aim of this approach 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.

Index Terms— **congestion control, command, delay, flow age, loss rate, notification system, performance, priority, queue, TCP, trend, throughput, wired networks, wireless networks**

I. Introduction

Back in 1981 first version of TCP was introduced in RFC793 [1]. To match the increasing traffic requests (mainly focused on bandwidth and delay), it was necessary not to improve only hardware part of the communication networks, but the software part as well. Improvements of TCP, usually called TCP variants or extensions, are mainly focused on the best and most equitable usage of available communication lines [2, 3].

First TCP improvements focused on higher performance were published in [4]. Since 1992 [5], there have been many new TCP variants, which can be divided into two groups based on the type of the access network type – wired and wireless networks. These groups can be further divided into smaller groups whose key element is the way how the congestion is detected. Based on this hierarchy, we can recognize wired network variants like CUBIC, CompoundTCP, Sync-TCP [6, 9, 10] and JTCP, TCP CERL and CompoundTCP+ as wireless network variants [7,8]. All these variants have one thing in common – they don't make any steps for congestion avoidance unless the congestion is detected by the TCP end nodes [13, 16, 17]. Slightly different approach can be identified while using Explicit Congestion Notification (ECN) system. Using this system, end nodes allow nodes in the network to inform them about the situation in the network and give them some kind of feedback [20]. However this feedback is sent to all existing TCP flows apart from any TCP flow characteristic. The only feedback that is sent stands for the immediate decrease of the congestion window. While keeping in mind existing ECN system we have identified following 2 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 in comparison 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 concepts introduced in the following sections are aimed to solve identified issues.

II. Analytical Model

The main idea of our approach ACNS (Advanced Congestion Notification system) is to inform only specific TCP end nodes with command how to modify the size of the congestion window. While notifying only specific group of TCP end nodes, we want to provide more bandwidth to young flows by means of freezing or decreasing the size of congestion window of older TCP flows. Commands sent to TCP end nodes are calculated from weight assigned to each TCP flow. This weight is based on following three key parameters:

1. TCP flow age – various TCP flows can exist in network for different time; long-aged flows are probably greedy data flows while young flows can be probably classified as flows which need to exchange just a small amount of data.
2. TCP flow priority – while keeping the eye on the age of TCP flows sometimes it's necessary to have long-aged TCP flows. Therefore we use the priority carried in the header of network layer protocol as second parameter which influence the final weight of the TCP flow.
3. Queue length – as the penalization is worthy only when there is congestion in the network, the last flow weight calculation input parameter is the actual queue length.

In general there are two situations which can occur while using this new mechanism. First situation is standard 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 in appropriate way. The mechanism of TCP flow weight calculation and command determination is described in

DOI: 01.IJCOM.4.1.1251

section A (TCP flow weight and command calculation).

On the other hand, the second situation represents typical packet loss in the network. It doesn't matter whether there was a loss of data segment or a loss of an acknowledgement – from the TCP end node point of view it is packet loss and therefore the TCP end node did not receive any command from the network. At this point the TCP end node has to determine which command will use based on the commands received in the past. This mechanism is described in section B.

A. TCP flow weight and command calculation

While the TCP communication is passing nodes in the network, it's necessary to calculate the weight of the TCP flow in order to send commands to end nodes. The calculation has three input parameters which are changing in the time and some of them are unique per flow. As we stated before the main parameters are TCP flow age, carried priority and actual queue length.

TCP flow age is unique for every flow and is changing in time. As the age of the TCP flow is theoretically unlimited, this parameter would bring some indeterminism [12] 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 the range of the age values varies from 0 to 1 (including). The comparison of standard and normalized age is shown in [19].

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

Similar normalization is applied to second parameter – priority p. The only difference is that the maximal priority p_{max} is known (according to 6 DSCP bites the maximal priority can be 63) and therefore there is no need for extra function which will do this normalization – it is done within the final TCP flow weight calculation.

Last input parameter actual queue length $\mathcal{A}(q)$ is changing in time and is shared across all flows [22]. It represents the actual usage of the queue and can vary from 0 up to 1.

Final weight ∂ for TCP flow f used for command determination can be calculated using (2) where $\mathcal{F}(p)$ stands for priority part and $\mathcal{T}(f)$ represents the age part. Both subparts can be calculated using (3) and (4).

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

$$\mathcal{F}(p) = \frac{v_p * p_{max}}{p} \tag{3}$$

$$\mathcal{T}(f) = v_a * age(f) \tag{4}$$

These parts need to be independent as the priority in the packet header does not have to be set and used at all. It is possible to put more weight on a specific part (or eliminate the other) by the weight factors - v_p for priority part and v_a for age part. This way allows specific TCP flows prioritization according to the network conditions. Sum of these factors has to be equal to 1 (5).

$$v_p + v_a = 1 \tag{5}$$

Calculated weight ∂ needs to be transformed into command which will be sent to the end nodes of the TCP communication. Final commands can be 1, 2 and 3. Explanation of these commands is provided in the next chapter. As the calculated weight ∂ is a real number, we need to convert it to one of these available commands using simple comparison with 2 thresholds th_{A1} and th_{A2}.

Final command ω for congestion window modification for TCP flow with weight ∂ is determined using (6).

$$\omega = \begin{cases} 1 & \partial < th_{A1} \\ 2 & th_{A1} \leq \partial < th_{A2} \\ 3 & th_{A2} \leq \partial \end{cases} \tag{6}$$

B. Determining command upon packet loss

Approach introduced in section A can be used most of the time but cannot be used when packet loss occurs. On the other hand commands received within acknowledgements can be useful when the loss occurs because they characterize the situation in the network right before the loss. Using these received commands we are able to get more accurate loss classification and determine trend command ω_{trend}. The difference between determining standard commands and trend commands lies in the nodes which do this determination. While standard commands are determined by nodes in the network, trend commands are determined by TCP end nodes, however only during detected loss. This packet loss classification is especially useful in wireless environment where random losses can occur due to interference on air medium.

Firstly TCP end nodes calculate trend weight ∂_{trend} using received commands. As the TCP end nodes receive many commands during the whole TCP communication, all these commands cannot be used for further calculation because they are too old and do not represent the current situation in the network. For further calculation let's suppose we will use only ω_{count} of the latest received commands. Even if we use only few commands we have to distinguish between the times when they were received. This can be achieved by assigning metric to every affected command using the exponential decrease with additional step parameter σ (7) [11, 14, 18].

$$x = e^{-\frac{\lambda}{\sigma}} \tag{7}$$

Calculated metric for every received command is normalized using the sum of metrics of all used commands χ (8).

DOI: 01.IJCOM.4.1.1251

$$\chi = \sum_{\lambda=1}^{\omega_{count}} e^{-\frac{\lambda}{\sigma}} \qquad (8)$$

If P is the list of all received commands the trend weight ∂_{trend} for specific TCP flow calculated by TCP end node is defined as in (9).

$$\partial_{trend} = \sum_{\lambda=1}^{\omega_{count}} \frac{e^{-\frac{\lambda}{\sigma}} . P[\lambda]}{\chi} \qquad (9)$$

Calculated trend weight ∂_{trend} needs to be transformed into trend command ω_{trend} which will be used by TCP end node itself. The trend commands can be 1, 2 and 3. Meaning of these trend commands is explained in the next section. As the calculated trend weight is a real number, we need to convert it to one of these available commands using simple comparison with two thresholds and.

Final trend command with trend weight is defined using (10).

$$\omega_{trend} = \begin{cases} 1 & \partial_{trend} < th_{L1} \\ 2 & th_{L1} \leq \partial_{trend} \leq th_{L2} \\ 3 & th_{L2} < \partial_{trend} \end{cases} \qquad (10)$$

C. Commands overview

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 1.

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 $cwnd$ (12).
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 (13).

Calculated commands:

1. ω_{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 (to ensure different treatment during another loss).
2. ω_{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. ω_{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.

D. Model verification

According to the testbed introduced in [19] together with [15, 21] 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 simulation results prove the functionality of feedback from nodes in the network as well as impact of the carried priority on the whole prioritization procedure.

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 according to [19].

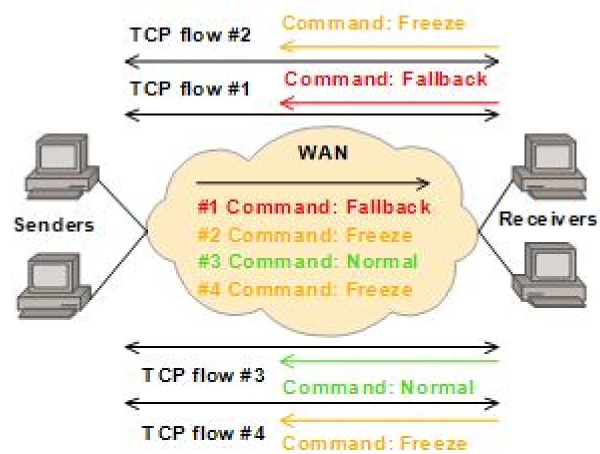

Figure 1. ACNS system overview

TABLE I. 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

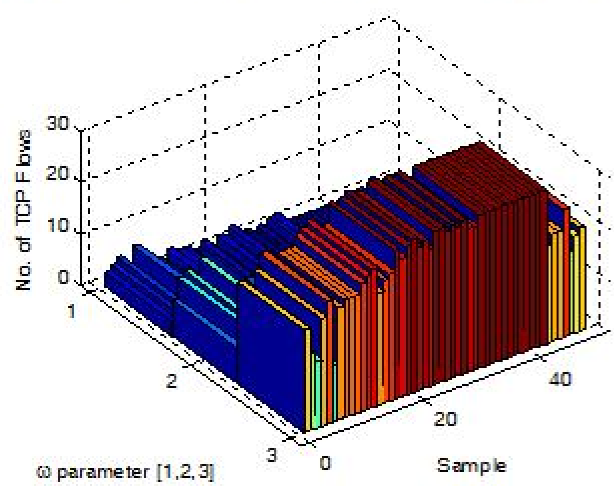

Figure 2. Randomly generated TCP flows without active feedback

DOI: 01.IJCOM.4.1.1251

ACEEE Int. J. on Communications, Vol. 4, No. 1, July 2013

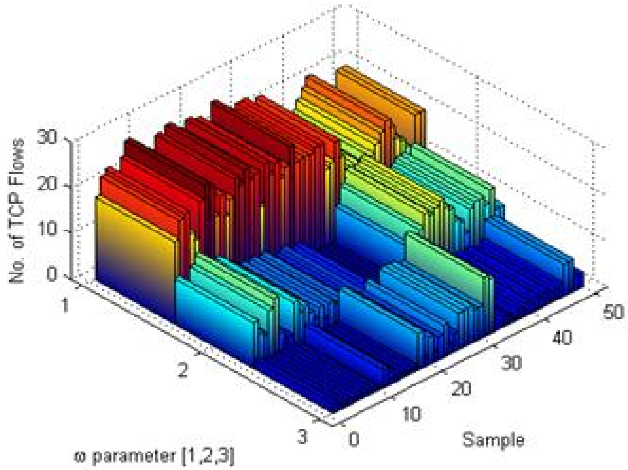

Figure 3. Randomly generated TCP flows with active feedback

In the example depicted in Figure 2 we have modeled 30 randomly generated TCP flows without feedback of our congestion prevention c. 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 3 we have modeled 30 randomly generated TCP flows, with normal distributed priorities (with $\mu = 24$ and $\sigma = 10$). The commands are now being fed back to the end nodes. The results show a slowly increasing tendency of the ω parameter as the queue fills up to the point where at around sample number 30 the number of flows with $\omega = 3$ rises dramatically. The command to decrease the congestion window is fed back to our model and subsequent samples show the effect.

In the following example (Figure 4) we used the same model and parameters, but we changed the TCP flow priority distribution uniformly to zero (DSCP zero is considered best effort). The scenario depicted in Figure 4 illustrates a different distribution of the ω parameter, as the queue fills up. With the increasing number of TCP flows assigned $\omega = 2$ (samples 1 to 25) the congestion is not avoided and the number of flows assigned increases.

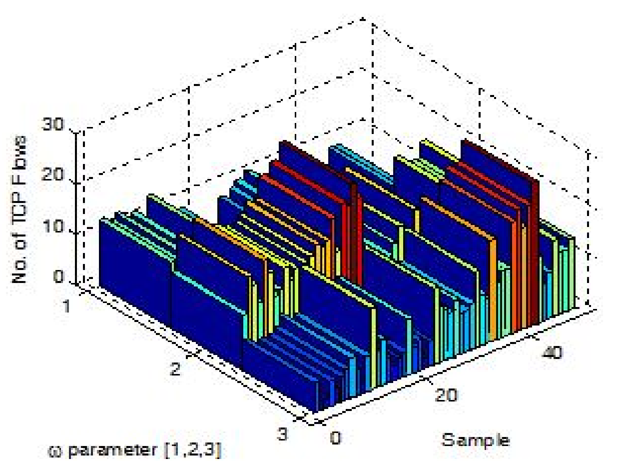

Figure 4. Randomly generated TCP flows (active feedback) - $\mathcal{F}(p)$ = 0

The gradual congestion window decrease afterwards prevents congestion. In this situation $\mathcal{F}(p)$ = 0 because 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.

III. Integration Into Protocol Stack

In order to use system ACNS in real network it's 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.

A. 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.

IPv4

Integration in IPv4 header lies in reusing ECN bits with one additional unused bit from field *Flags* called *CMI* (Figure 5). 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.

Table II. 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)

DOI: 01.IJCOM.4.1.1251

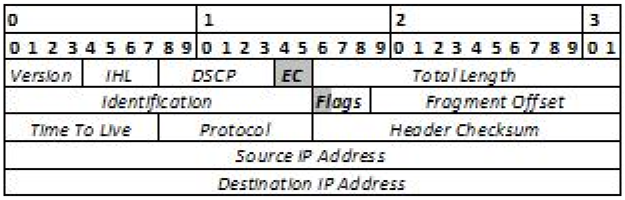

Figure 5. Bits used in IPv4 header

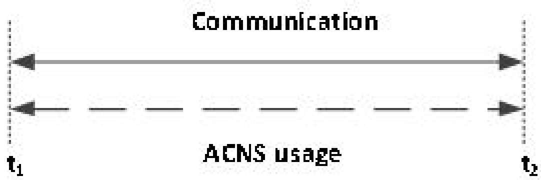

Figure 6. Network layer – ACNS usage

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 6).

IPv6

According to RFC2460 IPv6 header doesn't 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 [23].

Different approach introduced in [24] 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.

B. 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.

Change upon recipient of acknowledgement

As we stated before TCP end nodes can receive three different commands within acknowledgement. Together with these three commands available congestion window (*cwnd*) changes are defined in (11) where $cwnd_{last}$ is defined in (12) and $cwnd_{last\_x}$ in (13). Function *W* stands for congestion window size changing function in time.

$$cwnd = \begin{cases} cwnd_{TCP\_variant} & \omega = 1 \\ cwnd_{last} & \omega = 2 \\ cwnd - cwnd_{last\_x} & \omega = 3 \end{cases} \quad (11)$$

$$cwnd_{last} = W(t\text{-}1). \quad (12)$$

$$cwnd_{last\_x} = W(t - x) \quad (13)$$

After receiving command of 1, TCP end node will be allowed to use its own TCP implementation to calculate new congestion window size value.

Change upon packet loss

Using self-calculated trend commands TCP end nodes are able to modify congestion window as defined in (14).

$$cwnd = \begin{cases} cwnd_{last} & \omega_{trend} = 1 \\ cwnd_{last\_x} & \omega_{trend} = 2 \\ cwnd_{TCP\ variant} & \omega_{trend} = 3 \end{cases} \quad (14)$$

After receiving command of 3 end node will decrease the congestion window according to the used TCP implementation.

TCP

Integration in TCP header lies in reusing existing ECN bits and new bit from the *reserved* field called *CMT* (Figure 7). 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 8). 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 9).

Table III. ACNS messages encoded in TCP header

#	CWR	ECE	CMT	Message
T1	0	*0*	*0*	ACNS message: command normal (receiver), NS = 0
T2	*0*	0	0	ACNS message: command normal (receiver), NS = 1
T3	*1*	0	0	ACNS message: congestion window reduced (sender), NS = 0
T4	*1*	0	0	ACNS message: congestion window reduced (sender), NS = 1
T5	*0*	*1*	*1*	ACNS message: command freeze (receiver)
T6	*0*	1	*0*	ACNS message: command fallback (receiver)

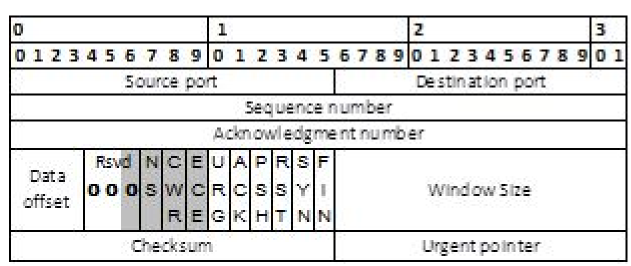

Figure 7. Bits used in TCP header

TCP end nodes agree on using ACNS during data exchange however they won't use ACNS during the initial

DOI: 01.IJCOM.4.1.1251

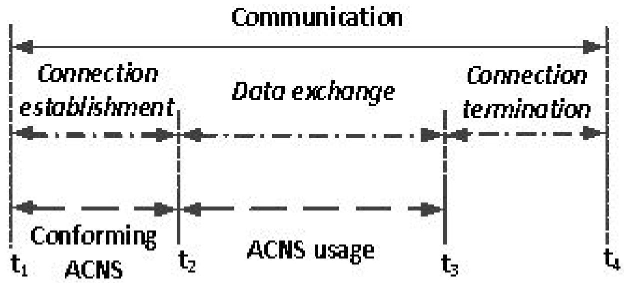

Figure 8. Transport layer – ACNS usage

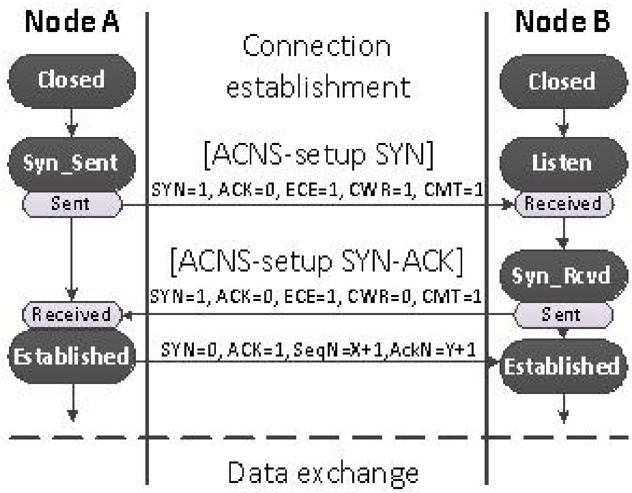

Figure 9. Connection establishment - Conforming ACNS

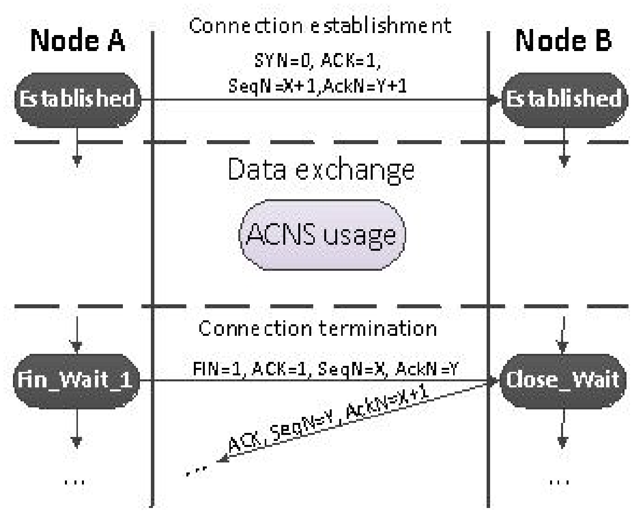

Figure 10. Data exchange - ACNS usage

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 starts (Figure 10).

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

IV. Preliminary 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 4):

- 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 11. Detailed simulation topology is shown in Figure 12. 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 set to 0 (best-effort). ACNS system parameters were set according to Table 1. Network parameters which were monitored during the simulations are throughput (maximal, average), RTT (maximal, average), amount of sent data and packet loss.

Table IV. TCP flows parameters

#					
#	hash	sIP	dIP	sPo	dPo
	addTime	lastTime	comAge	priority	system

DOI: 01.IJCOM.4.1.1251

ACEEE Int. J. on Communications, Vol. 4, No. 1, July 2013

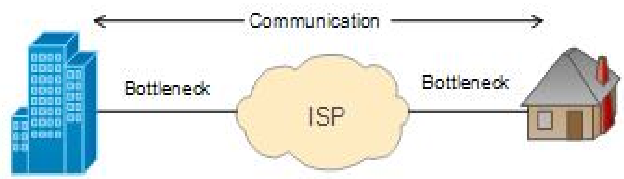

Figure 11. High level simulation topology overview

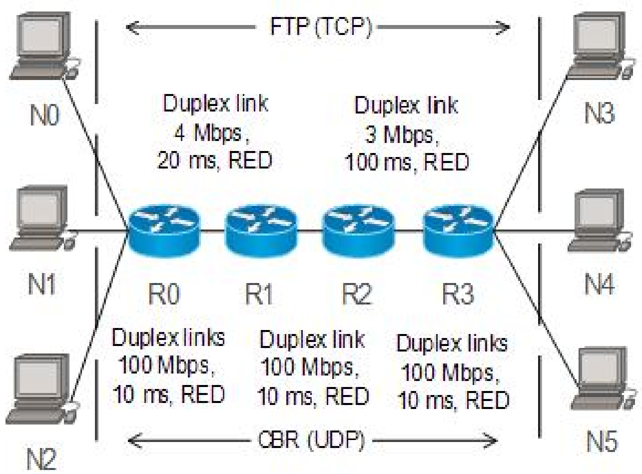

Figure 12. Detailed simulation topology

TABLE V. 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 VI. 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.

For better illustration the comparison of actual throughput and RTT changing in time is shown in the following figures. Figure 13 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.

TABLE VII. 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 VIII. 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 IX. 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 X. SIMULATION RESULTS – PACKET LOSS

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

The same reason is responsible for significant throughput changes at around the 20th and 40th second where the next UDP flows started. Figure 14 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.

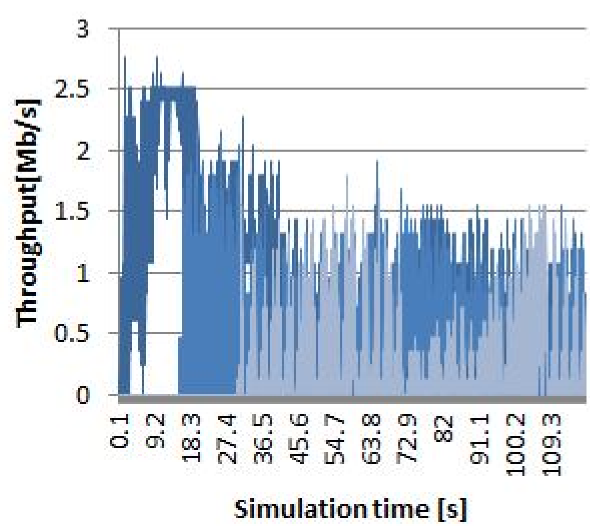

Figure 13. TCP flows throughput (without ACNS)

Figure 15 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 16 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.

According to the simulations results, using ACNS it's possible to increase TCP flows throughput (Figure 13 -

DOI: 01.IJCOM.4.1.1251

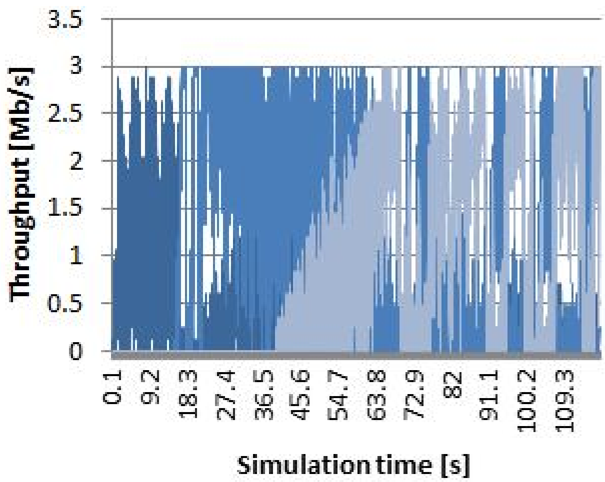

Figure 14. TCP flows throughput (with ACNS)

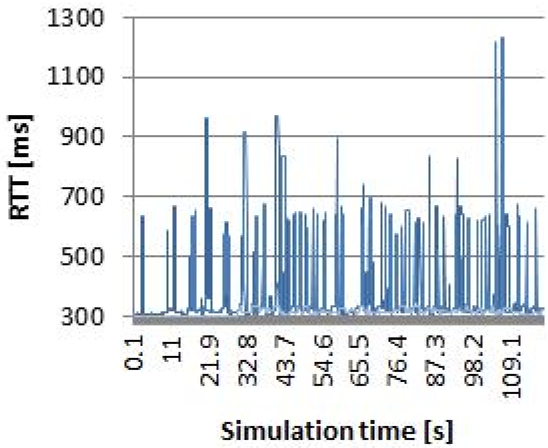

Figure 15. TCP flows RTT (without ACNS)

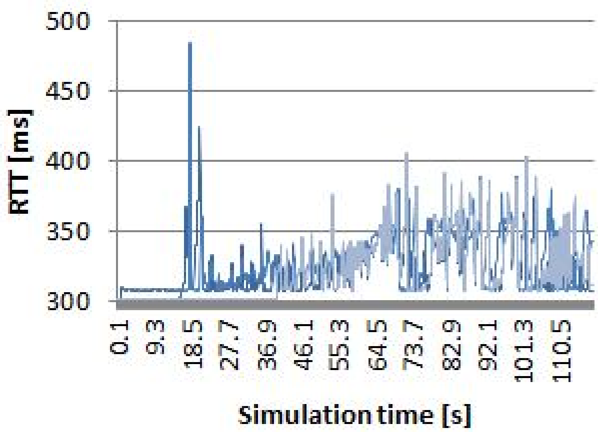

Figure 16. TCP flows RTT (with ACNS)

TABLE XI. NETWORK PERFORMANCE IMPROVEMENTS

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

without ACNS, Figure 14 – 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 15 - without ACNS, Figure 16 – with ACNS) by 7%. Network performance improvements are summarized in Table 11. All these improvements were achieved with nearly none losses.

CONCLUSIONS

In this paper we have introduced an improved version of the 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 variants 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.

ACKNOWLEDGMENT

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] Information Sciences Institute, University of Southern California; RFC793: Transmission Control Protocol, 1981.

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

[3] L. Xu and I. Rhee. CUBIC: A new TCP Friendly high-speed TCP variant. In Proc. Workshop on Protocols for Fast Long Distance Networks, 2005. 2005.

[4] Mirza, M.; Sommers, J.; Barford, P.; Xiaojin Zhu; “A Machine Learning Approach to TCP Throughput Prediction,” Networking, IEEE/ACM Transactions on, vol.18, no.4, pp.1026-1039, Aug. 2010.

[5] Karnik, A.; Kumar, A.; “Performance of TCP congestion control with explicit rate feedback,” Networking, IEEE/ACM Transactions on, vol.13, pp. 108- 120, Feb. 2005.

[6] 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.

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

[8] Todorovic, M. and Lopez-Benitez, N., “Efficiency Study of TCP Protocols in Infrastructured Wireless Networks”, In

DOI: 01.IJCOM.4.1.1251

proceedings of International conference on Networking and Services, ICNS '06, 2006.
[9] Chao, H.J. and Guo, X. "Quality of Service Control in High-Speed Networks", John Wiley & Sons, Ltd. 2005. ISBN 0-471-00397-2.
[10] Welzl, M. "Network Congestion Control - Managing Internet Traffic", John Wiley & Sons, Ltd. 2005. ISBN 978-0-470-02528-4.
[11] 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.
[12] Leon-Garcia, A., & Leon-Garcia, A., "Probability, statistics, and random processes for electrical engineering". Upper Saddle River, NJ, Pearson/Prentice Hall, 2002.
[13] 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.
[14] Malagò, L.; Matteucci, M.; Pistone, G.; "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 (FOGA '11). ACM, New York, NY, USA, pp. 230-242, 2011.
[15] Roychoudhuri, L.; Al-Shaer, E.; Hamed, H.; Brewster, G.B.; "On studying the impact of the Internet delays on audio transmission," IEEE Workshop on IP Operations and Management, pp. 208- 213, 2002.
[16] Kuzmanovic, A. and Knightly, E. W., "TCP-LP: Low-Priority Service via End-Point Congestion Control", IEEE/ACM Transactions on Networking, 2003.
[17] Martin, J., Nilsson, A. and Rhee, I., "Delay-based Congestion Avoidance for TCP", IEEE/ACM Transactions on Networking, June 2003.
[18] 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.
[19] Hrubý, M., Olšovský, M., Kotočová, M.: Analytical Multiparameter Notification System for TCP Congestion Avoidance. In ICT 2012, Proceedings of the Third International Conference on Advances in Information and Communication Technologies. Amsterdam, Netherlands, 22-23 November, 2012.
[20] Kwon, M.; Fahmy, S.; "TCP increase/decrease behavior with explicit congestion notification (ECN)," IEEE International Conference on Communications, ICC 2002, vol.4, pp. 2335-2340, 2002.
[21] Arshadi, L.; Jahangir, A.H.; , "On the TCP Flow Inter-arrival Times Distribution," Computer Modeling and Simulation (EMS), 2011 Fifth UKSim European Symposium on , vol., no., pp.360-365, 16-18 Nov. 2011.
[22] Filipowicz, B. and Kwiecien, J. "Queueing systems and networks, models and applications, technical sciences, bulletin of the polish academy of sciences", Technical Science, Vol. 56 No. 4, 2008
[23] Rafiee, H.; von Lowis, M.; Meinel, C. "IPv6 Deployment and Spam Challenges," Internet Computing, IEEE , vol.16, no.6, pp.22,29, Nov.-Dec. 2012.
[24] Krishnan, S.; Halpern, J.; "Reserving bits in the IPv6 header for future use" 6man Working Group, Internet-Draft, 2010.

DOI: 01.IJCOM.4.1.1251