Acta Informatica Pragensia
2(2), 2013, 1–24, DOI: 10.18267/j.aip.20
Online: aip.vse.cz

Section:
Peer-reviewed papers

Novel methods of utilizing Jitter for Network Congestion Control

Ivan Klimek[1], Marek Čajkovský[1], František Jakab[1]

[1] Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic

{ivan.klimek, marek.cajkovsky, frantisek.jakab}@tuke.sk

Abstract: This paper proposes a novel paradigm for network congestion control. Instead of perpetual conflict as in TCP, a proof-of-concept first-ever protocol enabling inter-flow communication without infrastructure support thru a side channel constructed on generic FIFO queue behaviour is presented. This enables independent flows passing thru the same bottleneck queue to communicate and achieve fair capacity sharing and a stable equilibrium state in a rapid fashion.

Keywords: Congestion control, Transport protocol, Inter-flow communication, Jitter

1 INTRODUCTION

Recent advances in rateless erasure codes [16, 22] enable to separate reliability from rate/congestion control for general purpose unicast transport protocols [15]. Currently for purposes where reliability is not necessary, for example multimedia applications, or when other layers take care of reliability for example through Application Layer Forward Error Correction (AL-FEC) [3]. but congestion control has to be guaranteed - TCP-Friendly Rate Control (TFRC) provides an option [8]. But it does so by modelling the Transmission Control Protocol (TCP) [5] performance as a function of packet loss and Round-Trip Delay Time (RTT) under the same conditions and limiting its rate accordingly. We argue that this approach is sub-optimal because it inherits the TCP performance which is not always satisfactory (For example on Wireless Wide Area Network (WAN) links, or Long Fat Networks (LFN), resp. the issue of Bufferbloat [18] etc.).

TCP flow synchronization occurs when congestion caused packet drop reaches levels that it affects all the flows on the overflowing bottleneck, as TCP understands packet drop as an indication of congestion it reduces its data rate. Because all the affected flows react the same way, the effect is flow synchronization [28] [23]. More generally speaking this behaviour is not TCP-specific, congestion can be utilized for synchronizing into a stable equilibrium state, where instead of aggressively reducing the rate as TCP does, flows stop increasing their rate and only reduce it so that packet loss or queuing doesn’t occur, thus stabilize the overflowing buffer, reaching and holding the full capacity of the bottleneck. Protocol that utilizes this behaviour can be designed [14], the problem with this state is that the capacity is not distributed fairly amongst flows it is just stabilized at whatever utilization the flows reached in the moment bottleneck capacity was reached. While experimenting with this stable state we noticed an interesting phenomena, we were able to on demand break the stable state and increase or decrease jitter levels on all flows passing thru the bottleneck with the change of the data rate of one of the competing flows even if it represented a fraction of the total bottleneck data rate. Thus effectively creating a broadcast communication medium between independent flows from the bottleneck queue. We present a proof of concept and explore the possibility of using this mechanism for congestion control.

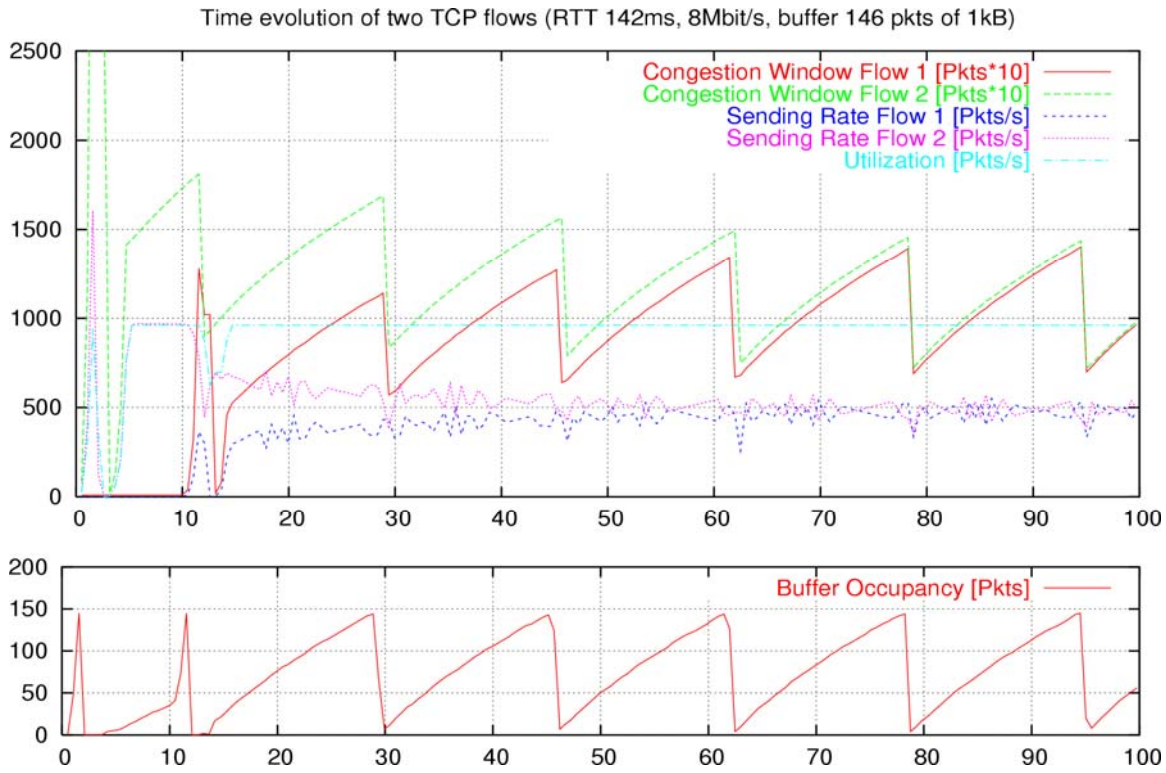

Fig. 1. Example of two TCP flows synchronizing. **X-axis = time [ms], Y-axis = no. of packets/s**

2 JITTER AS CONTROL SIGNAL

Several variants of TCP already proposed using RTT as an additional control signal on top of packet drop.

TCP Vegas [4] aims to improve the end-to-end congestion avoidance mechanism of TCP. The main objective is to estimate the expected bandwidth for the connection in order to control the transmission rate that can avoid network congestion. To achieve this goal, the scheme defines BaseRTT value which represents the minimal round trip time during the transmission to calculate the expected transmission rate of the link. After receiving an acknowledgement, sender continues to update ActualRTT value which represents the current RTT and is necessary to be able to calculate the real transmission rate. If the difference between BaseRTT and ActualRTT is higher than upper bound threshold, congestion may occur since sending rate is too high. Thus, sender decreases one congestion window size. If the difference is smaller than lower bound, sender should increase one congestion window size to utilize the available bandwidth. Else, sender should keep the sending rate stable. TCP Vegas suffers fairness problems when the connections start transmitting at different times. Also, TCP Vegas is not suitable for wireless network since it cannot distinct loss events.

TCP Veno [10] is a loss differentiation scheme for the wireless environment and it is derived from TCP Vegas. This method provides another threshold to differentiate between wireless and congestion losses. Although the performance is improved in wireless environment, the loss differentiation scheme cannot work well when the random loss rate is high. And it still does not solve the fairness issues of Vegas.

TCP Jersey [26] is another enhancement which improves network performance in wireless network. But because it needs router support (ECN) we will not describe it further.

All the mentioned schemes so far used RTT, such congestion estimator cannot perform well when the traffic load gets heavy over reverse links or asymmetric links.

On the other hand, JTCP [24, 25, 7] applies the jitter ratio to differentiate wireless losses from congestion losses and revise the Reno's congestion control scheme to adapt to wireless environments. Jitter ratio [6] is derived from the inter-arrival jitter, which is defined in Real-time Transport Protocol (RTP) [21]. Inter-arrival jitter is the variance of packet spacing at the receiver side and packet spacing at the sender side. In other words, it presents current path's status by the packet-by-packet delay. On packet loss JTCP compares the average jitter ratio with a threshold, which is defined as the inverse of congestion window size. If average jitter ratio is greater than this threshold, JTCP regards the loss event as congestion loss and reduces its congestion window size to one half. Otherwise, the loss event will be viewed as wireless loss. Sender will not reduce the congestion window and will do fast retransmission immediately. The problem with such approach is the setting of the threshold, inverse of congestion window size is a local parameter and does not represent the state of the network. Also, it means that on higher transmission speeds and thus bigger congestion windows the system will be much more sensitive to jitter variance.

Jitter seems to be better suited as an additional control signal than RTT as it describes only the one way delay changes triggered by changed path load, thus it is more stable as it cannot be affected by the load of the reverse path.

3 CHANGING THE PERSPECTIVE

State of the art transport protocols still use the TCP "point of view" - looking at data on packet level [27] [12], even if Forward Error Correction (FEC) protocols slowly gain traction [17] they are no different. The presented design is based on FEC codes and uses their error correcting properties to change the perspective from a packet level to flow level. This abstraction can be compared to the difference in the physics of a particle and a wave. We no longer have to deal with individual packets, retransmit them etc. they are all just pieces of a flow. Where the flow can be modulated as a waveform to create a secondary channel without affecting the primary channel – the information it transmits.

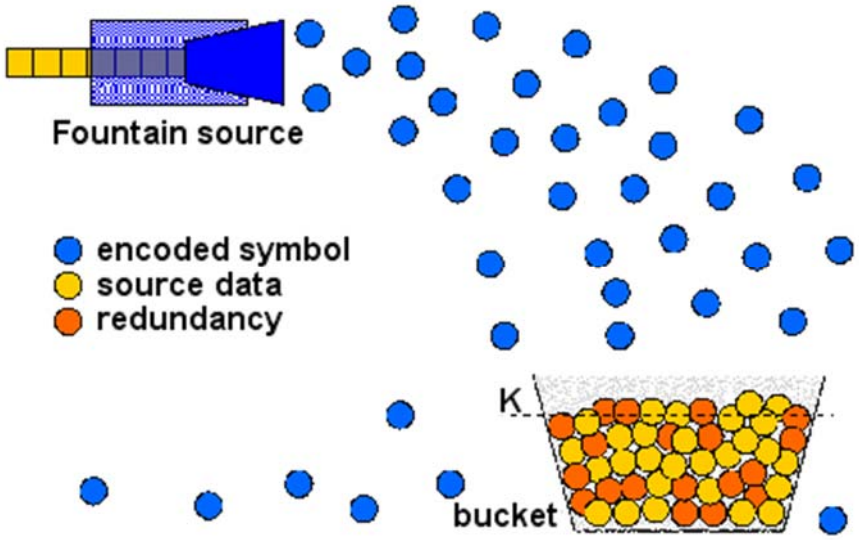

Fig. 2. In digital fountain codes, the encoder produces a potentially infinite number of encoded symbols and the decoder is able to recover the original information from any set of at least K=k(1+e) encoded symbols. [13]

Inter-flow communication is a crucial milestone, it enables fast convergence to a stable network state where resources are allocated with absolute fairness. This is in contrast with currently used TCP-based logic, where flows compete for capacity in a never ending fight for resources without ever reaching a stable state or full resource utilization. All designs that tried to improve this situation did so by utilizing feedback from the network requiring the network to change – routers to add functionality [19],[1]. The presented system is the first as far as authors are aware of that provides the capability for network flows to cooperate in achieving optimal network utilization without the need for any support from the network – thus usable on the currently deployed infrastructure.

Most queues still use the most basic queuing algorithm – First in First out (FIFO) aka DropTail [20]. This fact that FIFO queues are "everywhere" in Internet is considered an issue causing problems such as Bufferbloat [11]. In fact their omnipresence is a crucial enabler for our mechanism to function. Other queuing mechanisms such as Active Queue Management (AQM) (Random Early Detection (RED)[9]/Weighted RED (WRED)[2] etc.) should be also theoretically to some extent supported but we did not so far look deeper into them.

The presented design primarily targets last mile routers as they are where most congestion occurs [11]. This scope limitation enables us to design a simple proof of concept communication system as not too many concurrent flows occur at a last mile bottleneck when compared to a core router bottleneck situation. Figure 3 illustrates the core concept behind our design, independent flows interact at the bottleneck queue. When the ingress packet rate is higher than egress packet rate – buffering occurs. Packets are ordered inside of the queue by their ingress time, if one flow increases its rate, the gap between packets of other flows will be altered creating a measurable change. In the previously mentioned equilibrium state achieved by congestion synchronization [14] a change of the transmitter (TX) inter-packet delta aka packet rate of one flow can on-demand break the queue equilibrium and create jitter measurable by all flows in that queue. The flow can also return back to the original rate, thus a stable/unstable jitter modulation scheme can be constructed.

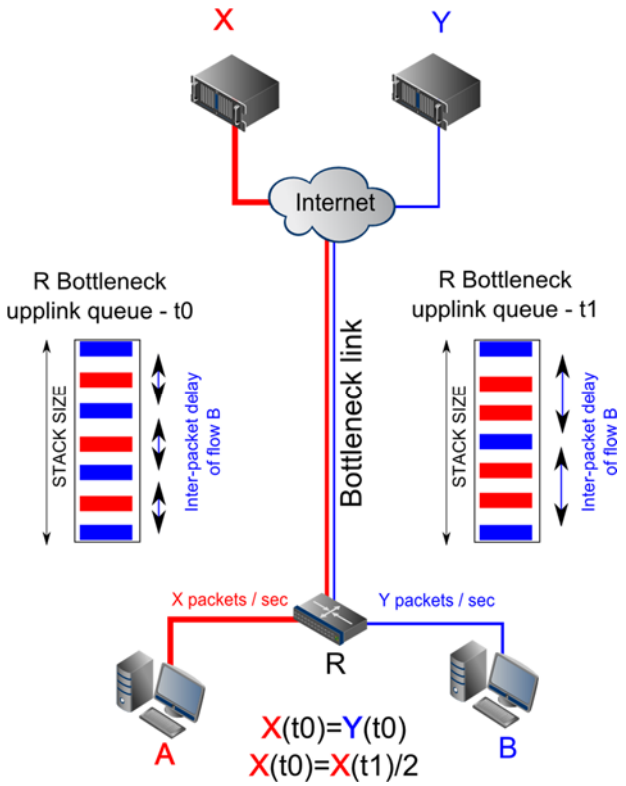

Fig. 3. FIFO inter-flow interaction

4 FIFO THE THEORETICAL COMMUNICATION CHANNEL

FIFO is far from a perfect communication medium, it is not linear as the TX of a frequency or amplitude modulation will not necessarily create a linear response on the receiver (RX) of a different flow. The sampling rate of the RX flow is not stable and not only a function of its packet rate but a function of the current state of the queue that is a superposition of all the different flows passing thru it. Classic modulation approaches such as frequency or amplitude modulation are thus unusable, transmitting special patterns such as patterns with strong auto-correlation properties also doesn't produce stable results as error, erasures and shifts are arbitrary. Even if frequency modulation would be possible it would not be ideal as it would require many samples to be precise. We therefore propose a modulation requiring only a few samples that provides deterministically achievable results – an equilibrium breaking modulation – where flows reach a stable state that is broken and restored in protocol defined time windows.

Because of the mentioned properties a robust coding scheme is required, first of all the true sampling rate is not known therefore the code needs to be self-clocking. The start of the sequence needs to be clearly recognizable, to achieve this the presented code is self-synchronizing by reversing the transmitted sequence every period.

$$xyz \in X^* \Rightarrow xw, xy \in X^* \quad \textbf{(1)}$$

Equation 1: Self-synchronization, a code *X* over an alphabet *A* has a synchronizing word *w* in *A+*

Self-clocking is achieved by utilizing Differential Manchester encoding where a transition is guaranteed at least once every bit, allowing the receiver to perform clock recovery. Differential scheme is preferred as detecting transitions is often less error-prone than comparing against a threshold in a noisy environment as a jitter based modulation is, also the polarity of the transition is unimportant just the presence of a change is detected. The receiver monitors the variance of jitter, where jitter is defined as the inter-packet delta. Variance is derivative of jitter, thus the second derivative of packet arrival time.

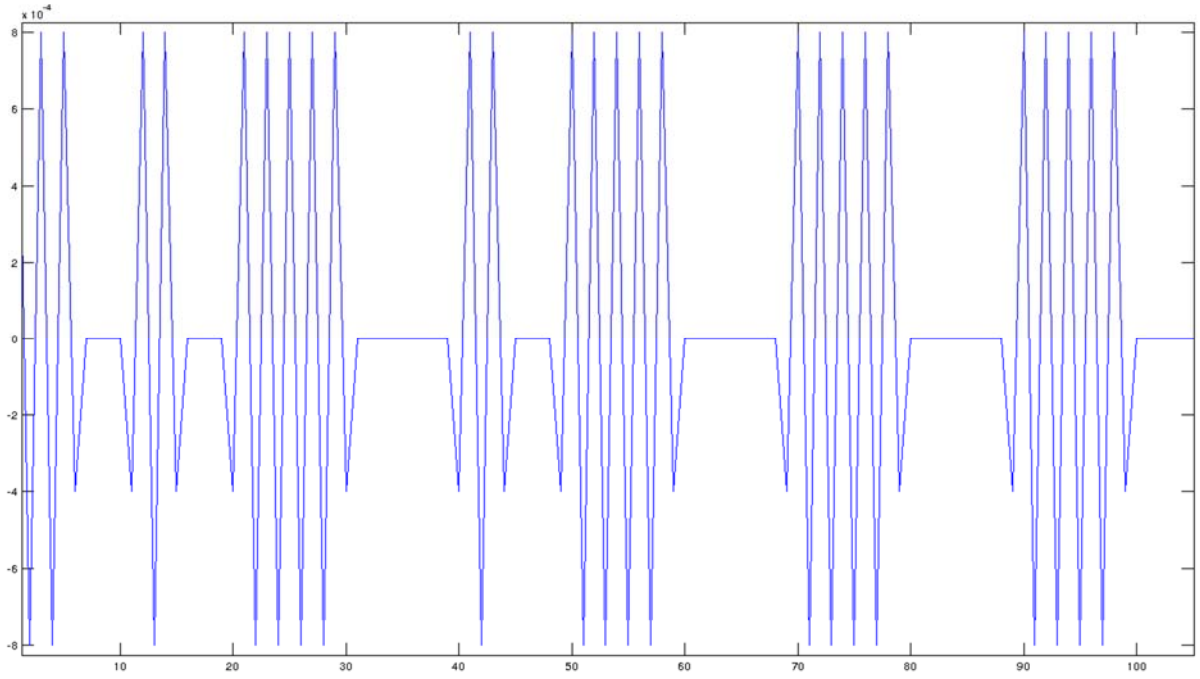

Fig. 4. Sample delta jitter of a receiver flow. **X-axis = jitter [ms], Y-axis = no. of packets**

The variance is measured by taking the zero-crossings of the delta jitter. The number of zero crossing directly reflects the state of the queue – stable/not stable. Because the modulation window is protocol defined, the number of variance changes per window can be interpreted to bits.

The presented design uses 2 changes to encode bit 1, one change to encode bit 0 – the reconstructed signal in Figure 5 is therefore 11001000.

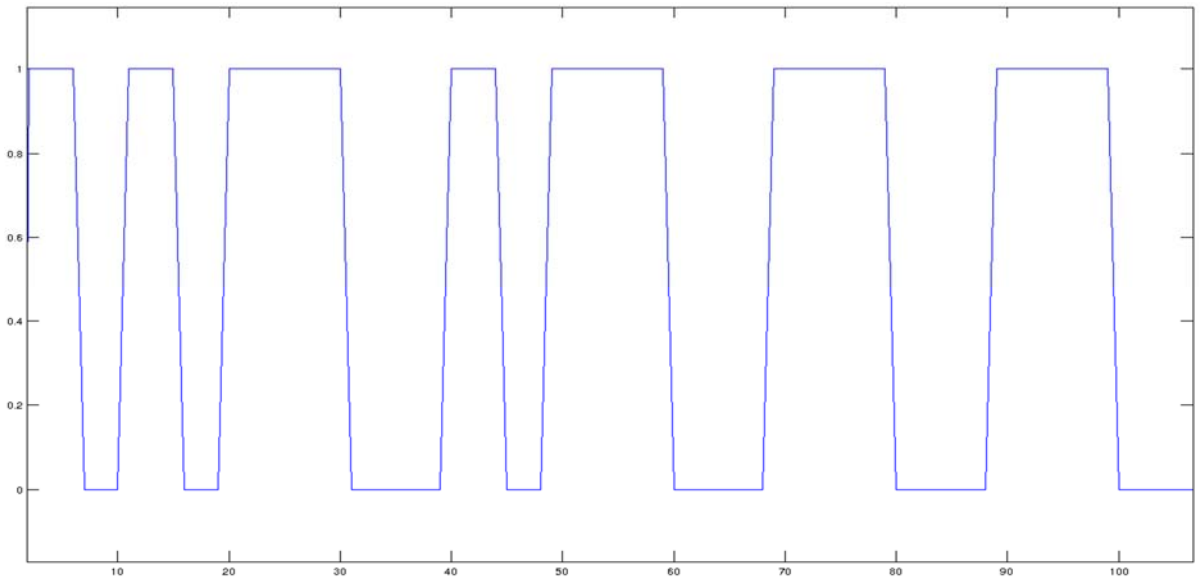

Fig. 5. Sample receiver reconstructed signal - Diff. Manchester encoded. X-axis = jitter [ms], **Y-axis = carrier signal (logical 0 or 1)**

The simulations were performed in ns2 using a star topology with five TX nodes, one router node and one receiver node, each TX node was transmitting one flow. All flows merged at the router node which connection with the receiver node represented the bottleneck link. The modulated flow changed its inter-packet delta between 4ms for the stable state and 2ms for the excited state. The four receiver flows had a constant inter packet delta of 2ms. Protocol defined modulation window was 20ms. Bottleneck link speed was set to match the stable state. Receiver code post-processed the ns2 data in Matlab.

Figure 6 illustrates the bandwidth allocation of the bottleneck per flow, the 4 receiver flows are superimposed as they have the same rate, so the total rate of the not-modulated flows is packet every 0.5ms vs. packet every 2 resp 4ms for the modulated flow. Meaning in average there is 6 times more not-modulated packets than modulated in the queue at every moment.

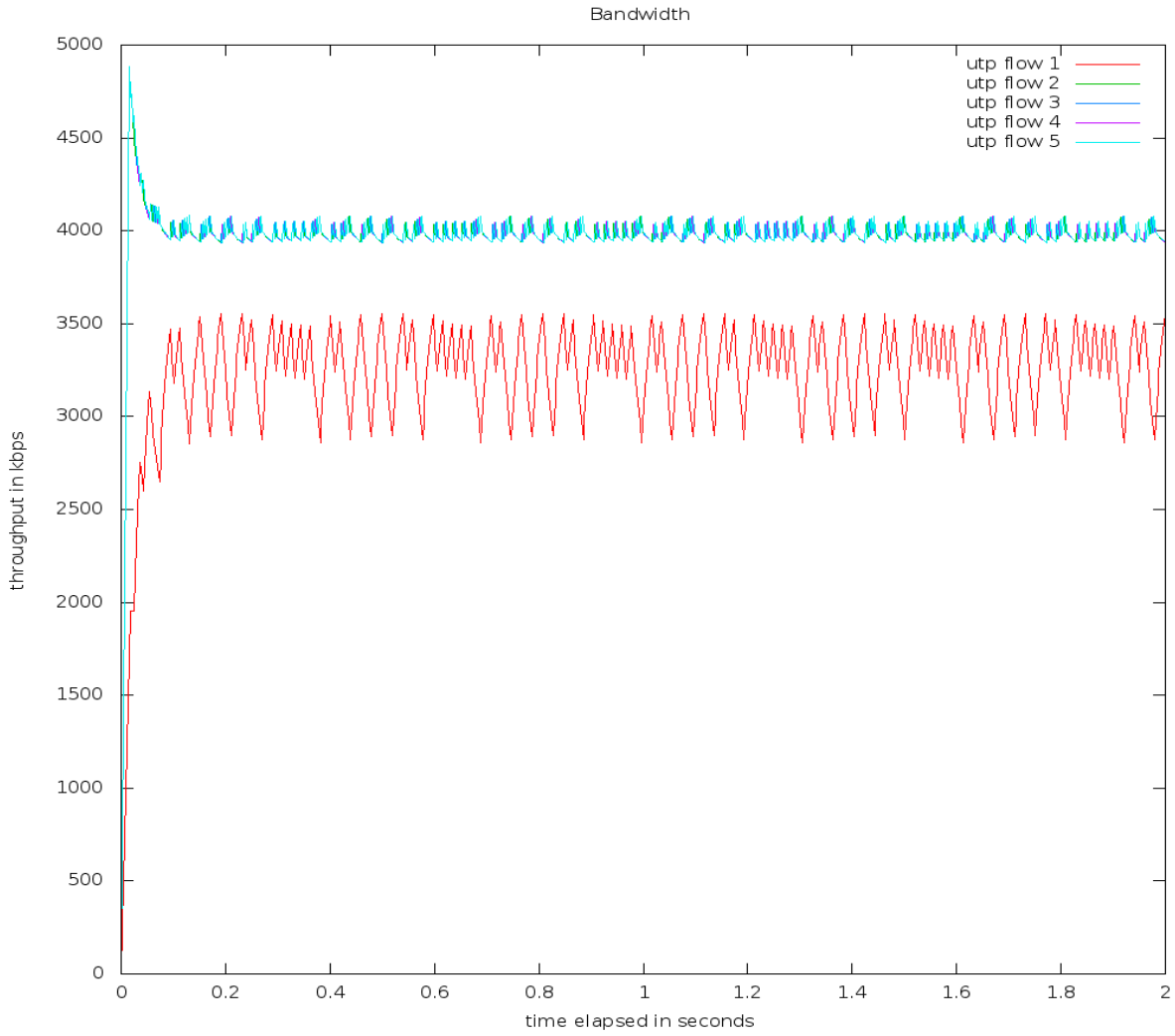

Fig. 6. Bottleneck bandwidth utilization per-flow - red is the modulated flow, blue are the 4 receiver flows superimposed.

To demonstrate that only a fraction of the total queue capacity needs to be modulated for other flows passing thru it be able to receive the signal the following experiment was performed. The not-modulated flows transmitted a packet every millisecond, the modulated flow every 8ms with oscillation of 1ms up/down resulting in 7ms/9ms inter-packet delta for duration of protocol defined window 20ms. Thus in average there was 32 times as many not-modulated packets than modulated in the queue at every moment. Figure 7 illustrates this bandwidth allocation and Figure 8 shows the reconstructed signal. Because the presented modulation only aims to trigger instability in the queue, only a small change is necessary to achieve measurable difference. In contrast to usual modulations where signal is being modulated on a medium, we modulate by creating and disturbing a stable state of the medium.

This approach could be compared to the resonant frequencies known from physics, where even a small periodic driving force can produce large amplitude oscillations. To set the amplitude right we propose to use an Automatic Gain Control on the RX of the modulated flow, because its the RX of the flow that is modulating it can have the knowledge of the fact that modulation should be happening and what

should be transmitted. If the signal is not strong enough because of ambient jitter the RX can immediately notify the TX that amplitude needs to be increased. For the primary information channel the modulation/amplitude itself is invisible as it just oscillates to both direction so in average the rate is constant.

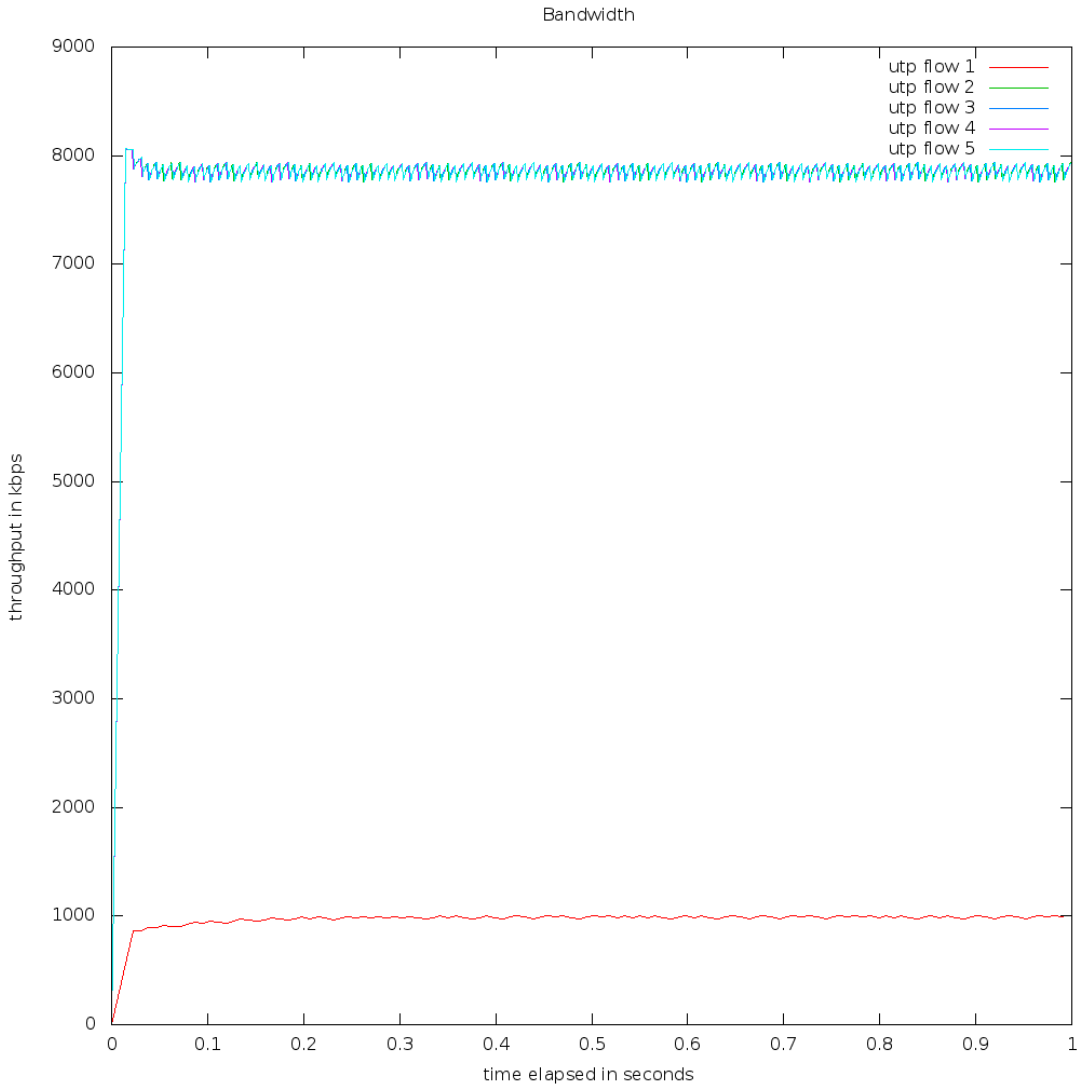

Fig. 7. Bottleneck bandwidth utilization per-flow - red is the modulated flow, blue are the 4 receiver flows superimposed.

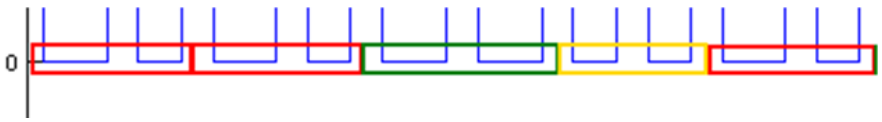

Fig. 8. Reconstructed zoomed color coded signal - red is 1, green and yellow are 0 in up/down position resp.

5 FIFO THE PRACTICAL COMMUNICATION CHANNEL

To verify that the presented theoretical model really works, we implemented it in C and tested "in the wild" on the most worst case scenario we could think of. Worst case meaning a scenario where arbitrary jitter is abundant. We used the topology from Figure 3. Two hosts connected via WiFi to a 3G router, one server was located at the Technical University Košice, the other in Germany – approximately 20ms RTT from the server at our University. Host B had 3 active connections, each at 30ms inter-packet delta, all not-modulated – meaning static rate. Host A had a single modulated flow at base rate of 40ms, with symmetrical amplitude of 10ms, thus one period inter-packet delta was 30ms, the next 50ms and vice versa. The flows from host B and host A met at the shared bottleneck – 3G wifi router uplink. Except of that they had nothing in common, they were destined for different servers and originated from different hosts. Figure 9 shows the jitter measured on one of flow B flows. Figure 10 the signal filtered from the jitter using signal averaging (two pass moving average with window size of half of the protocol defined window).

The performed experiment clearly demonstrated that only a fraction of the total traffic passing thru the bottleneck needs to be modulated for other flows to be able to detect it. In this experiment only the clocking signal was transmitted e.g. all zeros – one change per period. Figure 11 illustrate how does a 1010 signal look – two changes per period marking bit 1.

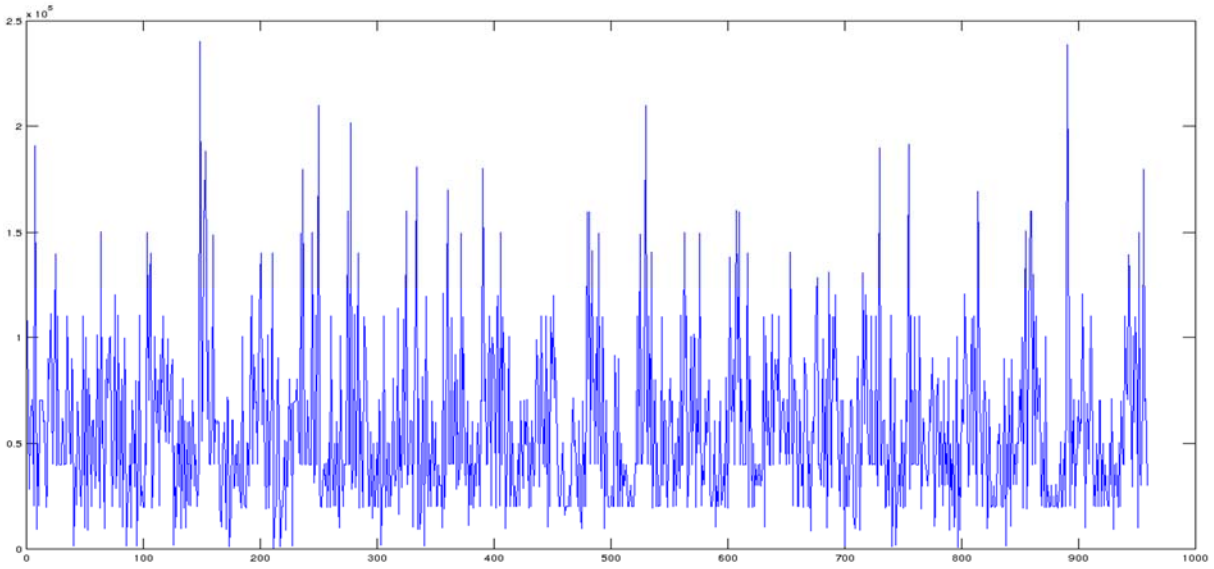

Fig. 9. Jitter of one of the host B flows. Periodic modulation is clearly visible. **X-axis = jitter [ms], Y-axis = no. of packets**

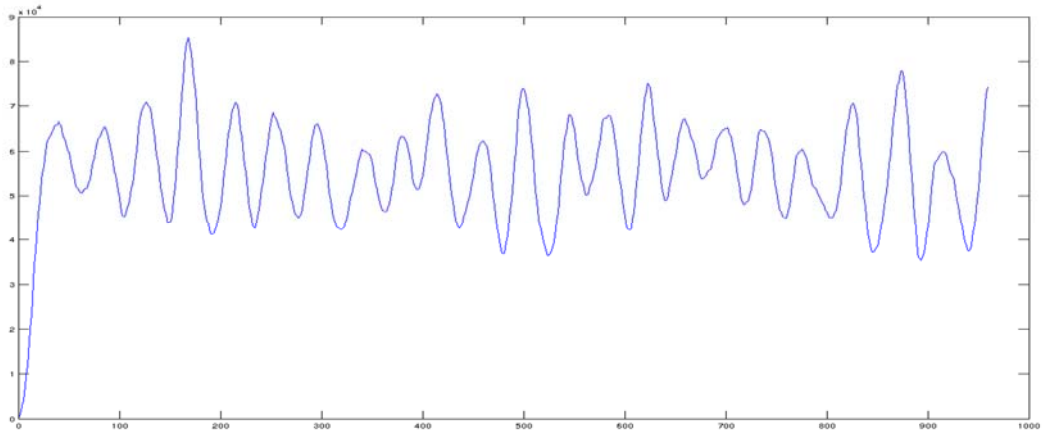

Fig. 10. Filtered signal from the jitter on Figure 9. **X-axis = jitter [ms], Y-axis = no. of packets**

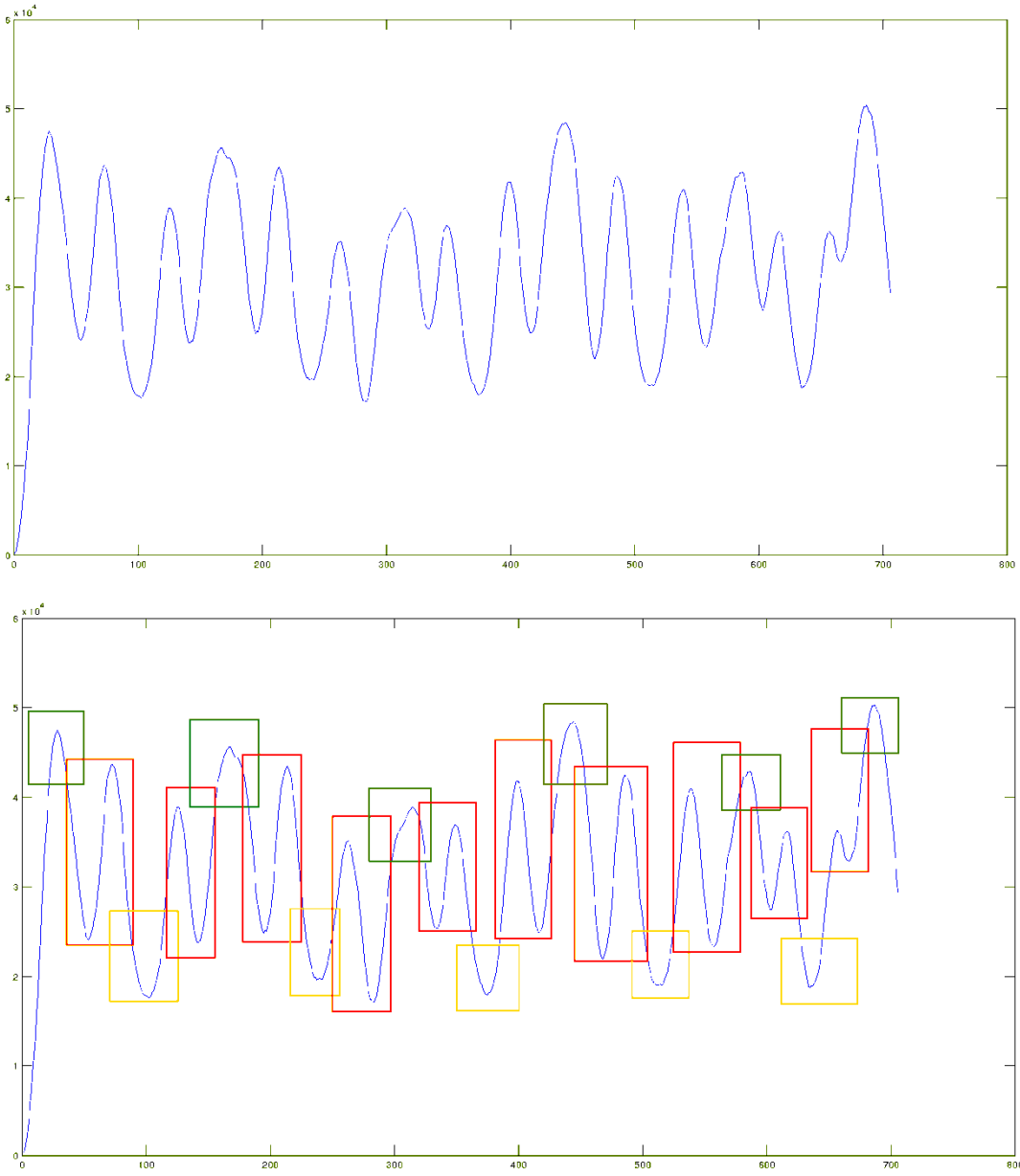

Fig. 11. Signal + colored representation - bit 1, two changes per period - red, bit 0 up position - green, bit 0 down position – yellow. **X-axis = averaged jitter from [ms], Y-axis = no. of packets**

6 DIGITAL VS. ANALOG

As the previous section demonstrated, digital modulation can be achieved. The problem is it would take simply too long for information to be transmitted this way. The protocol defined window in which modulation occurs needs to be at least several tens of milliseconds long so there are enough samples (packets) for signal averaging to eliminate the random noise. If we can transmit only one bit in each protocol defined window, it would take several Round Trip Times on most links to transmit any usable information. We therefore aimed our focus on determining the effects of interactions between flows on the FIFO that could possibly be combined into enough measurable factors that could be used for an analog modulation.

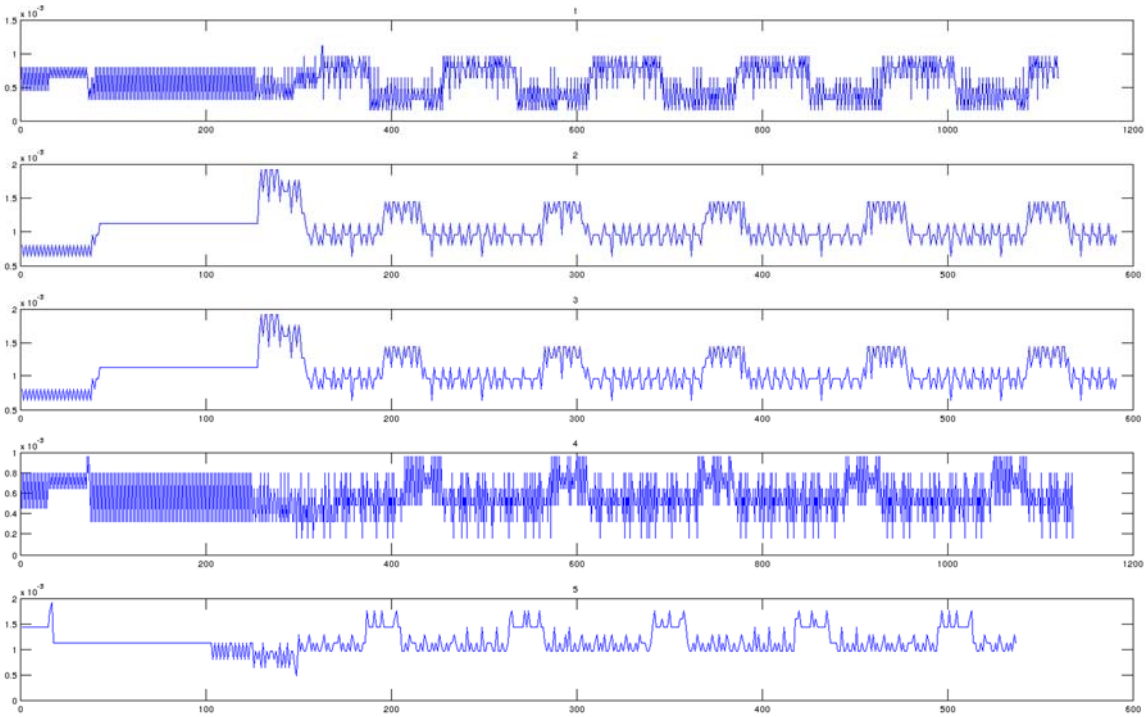

Fig. 12. Jitter of 5 flows interacting at a bottleneck FIFO queue, the first flow from top is the modulated. **X-axis = jitter [ms], Y-axis = no. of packets**

Figure 12 shows the jitter of 5 flows passing thru the same bottleneck. Flow 1 (first from top) is modulated – changing its packet rate in a periodic manner. Flows reached a stable equilibrium state – where they stopped increasing their rate and Flow1 started modulating. Each flow was able to reach a different rate – clearly visible from the jitter graph is that the rate of Flow 2 and 3 are very similar.

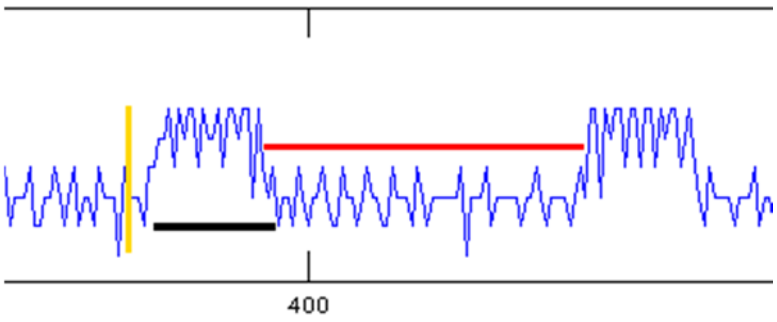

Fig. 13. Zoomed jitter of Flow 2 from Figure 12. Nomenclature for Equation 2 and 3: yellow = jitter amplitude, black = up period length, red = down period length.

Figure 13 shows a zoomed portion of Figure 12, market yellow is the amplitude of the received modulation, black is the length (in packets) of the UP phase and red is the length of DOWN phase. These three variables describe what is going on the FIFO – amplitude is the change created by the modulated flow and is received inversed. When the modulated flow transmits more packets – less packets of other flows will pass thru the FIFO. When the modulated flow transmits less packets – more packets of other flows will pass thru it. The ratio between the UP and DOWN phases corresponds directly to the ratio of the modulated flow rate at higher (amp+ [ms]) and lower rate (amp- [ms]). Jitter samples the state of the queue, the modulated flow changes it by its modulation amplitude thus all the non-modulated flows that pass thru the queue at that time will be subjected to the same level of change – resulting in 1:1 transmission of the amplitude to all flows even if they transmit at different rates.

$$\frac{down\_period\_length}{up\_period\_length} = \frac{(amp+)}{(amp-)} \tag{2}$$

Equation 2: Relation between the period lengths and modulated flow rate, also visible from Figure 13.

$$jitter\_amplitude = (amp+) - (amp-) \tag{3}$$

Equation 3: Relation between jitter amplitude and modulated flow rate, also visible from Figure 13.

Where *interval* is the inter packet gap of the modulated flow and

(amp+) = interval + amplitude

(amp-)= interval - amplitude

In contrast to jitter where the amplitude is received exactly as it is transmitted on bandwidth the effects are specific to each flow and its speed as Figure 14 illustrates. Jitter is the measure of change, a stable jitter amplitude translates into a flow specific amplitude in the "bandwidth domain". A jitter amplitude of X milliseconds translates into a change of Y percent in jitter for that particular flow. Meaning the same level Y of change will be visible as the bandwidth amplitude – Y percent of the flow speed. Also

important to note is that the bandwidth amplitude of the modulated flow is because of the limited capacity of the bottleneck/FIFO matched by the inversed summed amplitude of the other flows. (not visible from Figure 14 as flows are superimposed). This property can be used as a validator of the results from Equation 2 and 3.

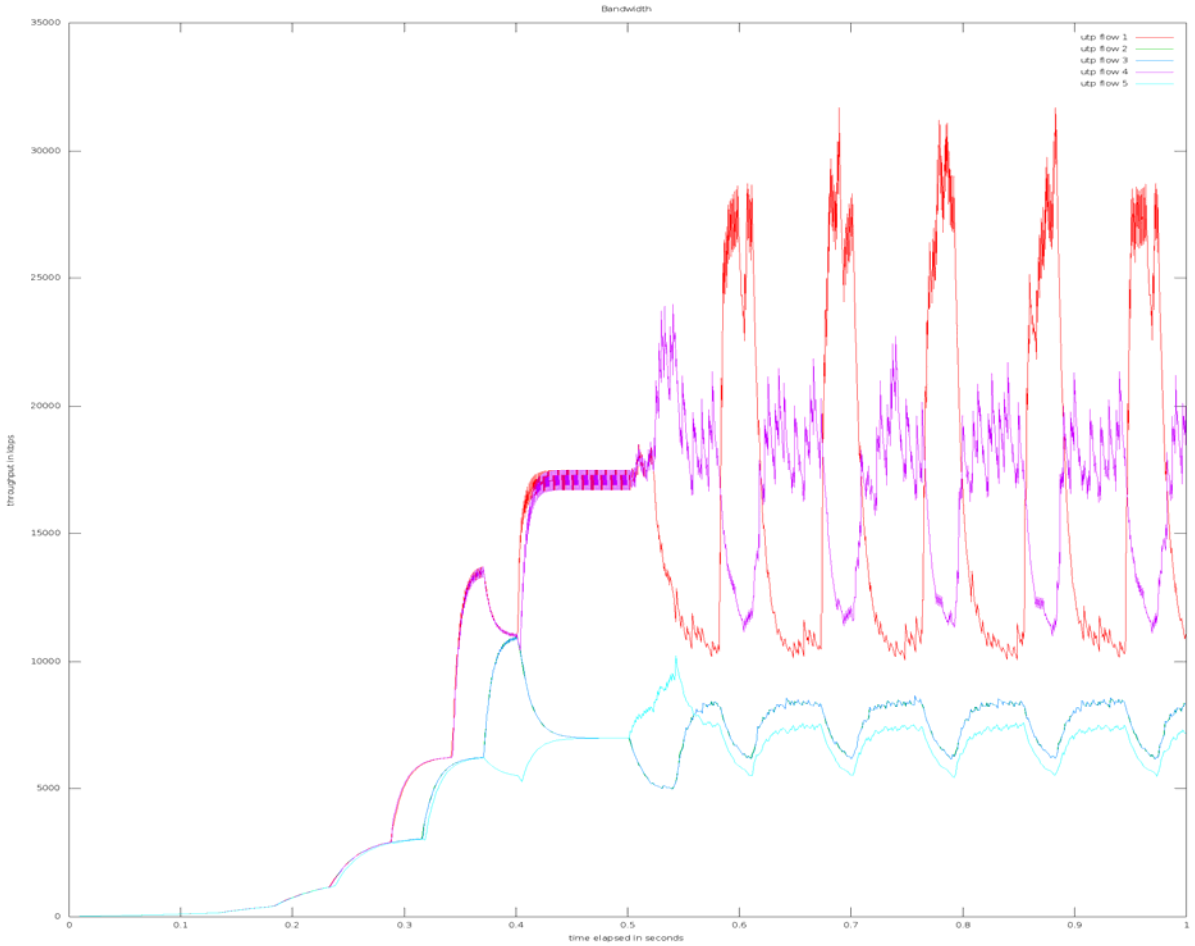

Fig. 14. Bandwidth allocation demonstrating different amplitude size based on the flow speed.

All together these parameters can be used to construct an analog communication scheme, where the modulating flow transmits just a square wave with a constant amplitude and frequency – defined by the protocol defined modulation window. The non-modulated flows can decode the rate and the amplitude of the modulated flow from just two periods worth of samples as shown in Figure 13.

7 CONGESTION CONTROL USING INTER-FLOW COMMUNICATION

We have constructed a simple congestion synchronizing inter-flow communication based, congestion control scheme to demonstrate the possible benefits of communication between independent flows. Synchronization is achieved by exponential search until less packets than expected arrive at the flow RX. The RX holds the logic determining the rate of the flow and decides based on the number of packets it receives, if the expected number of packets per reporting interval is received the rate is

increased, if less packets is received the rate is set to the rate corresponding to the amount of packets actually received – thus at the maximum achievable capacity.

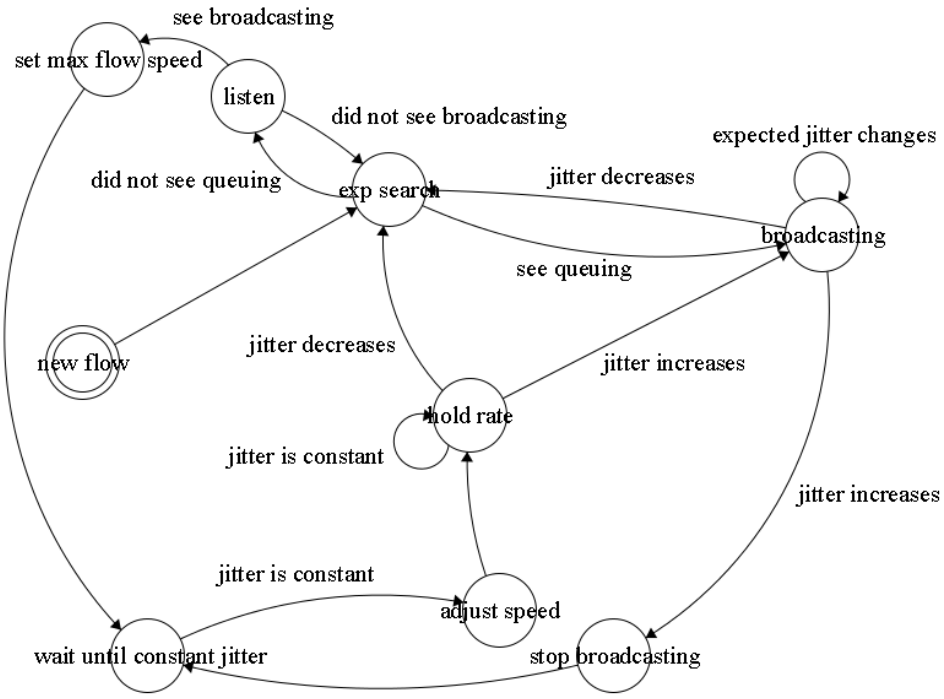

Fig. 15. FSM describing the congestion control algorithm.

The first flow to reach the capacity is because of the aggressiveness of the exponential search expected to be the fastest flow. During the exponential search, the rate is increased in steps, between these steps the rate is stable and the flow listens to possible jitter modulation. If it reaches the capacity but was not able to detect any modulation present it starts to modulate a square wave signal with period of the protocol defined with and a stable amplitude. If another flow is present, it receives the modulation and immediately changes its rate to the decoded rate of the modulating flow. The result of such action is that packets of both flows now arrive at the bottleneck with same inter-packet gaps, thus any effect such as buffering or drop would have the same results on both of them. The result is measurable and rate is set according to it, thus both flows set the same rate and just shared the capacity fairly. The modulated flow immediately detects a change on the network when the other flow changes its rate as less packets will arrive in that interval. To be able to detect when a stable state was achieved it stops modulating. After a stable amount of packets is received for at least two periods, the rate is adapted to the detected rate. If any change would happen such as a new flow or a flow was lost, a change would be detectable. Based on the change an action would be taken, if the jitter would rise and less packets than expected would be received possibly a new flow arrived. The first flow to detect such change would start modulate, the earlier mentioned logic would repeat. If the jitter would drop, possibly a flow was lost. The first flow to detect such change would start exponential search, because of its aggressiveness it is expected the rate would be achieved before other flows could react in the same way. After the new rate would be found, the flow would start modulate it to broadcast it to other flows, that would react and re-sync the capacity.

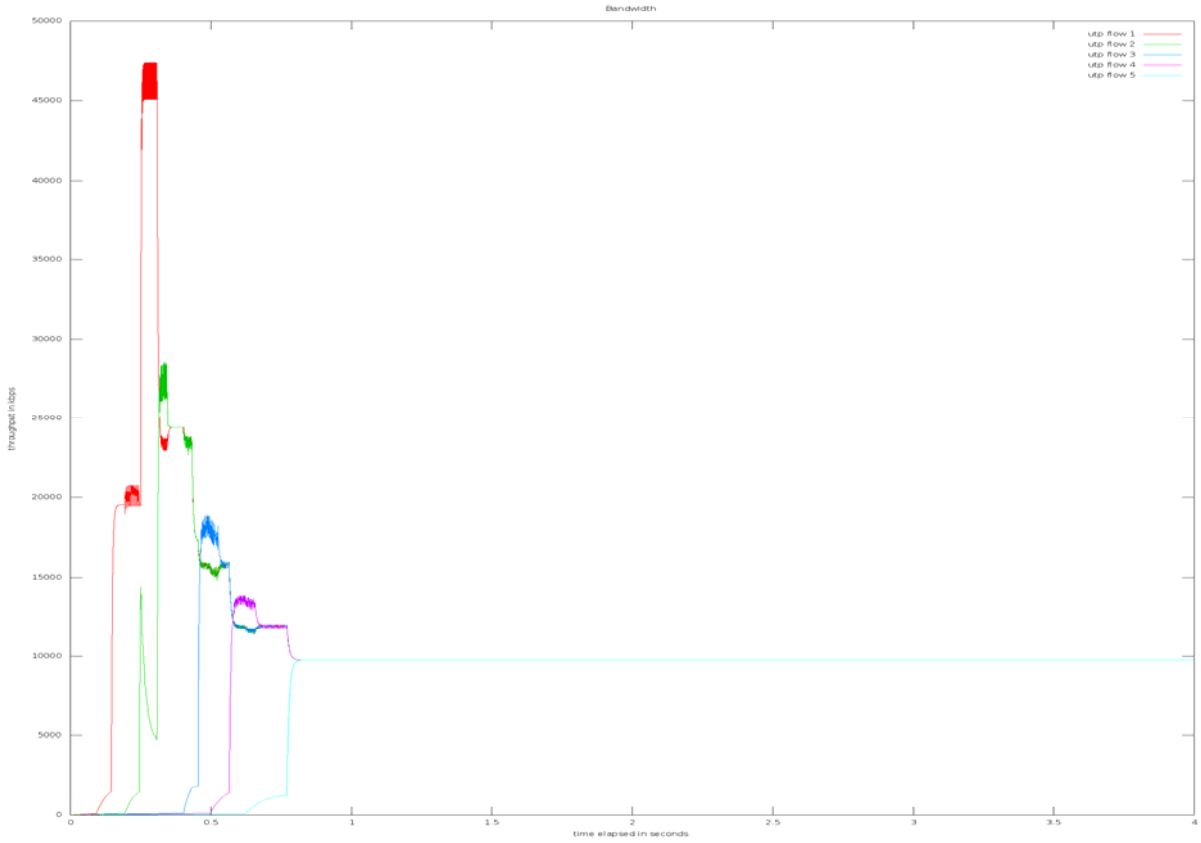

Fig. 16. Simulated capacity sharing with the knowledge of the maximum flow rate.

Figure 16 illustrates the effect of the broadcasted knowledge of the fastest flow rate. If flows know this information, fair and instant capacity sharing is trivial because the FIFO does quantization based on ingress time. If inter packet gaps are the same, the flows will get the same share of capacity. To limit over-buffering it is important to detect stabilized queue state and adapt the transmit rate accordingly in as shortest time as possible.

8 PERFORMANCE EVALUATION

There was performed several experiments using various parameters for queue size. Another parameters which was not changed during experiment was; speed of the exponential search (multiplicative factor was set to 4) and the capacity usage goal (controlling how much capacity should each flow use - set to 90%). One of experiments is shown on figure 17 with queue size set to 20 packets. The simulation runtime was 10 seconds with flows stopping at time 9 seconds. Flows where starting gradually: Flow 1 at time 0, Flow 2 at time 0.3 seconds, Flow 3 at 0.3 seconds, Flow 4 at 0.4 seconds and Flow 5 at 0.5 seconds.

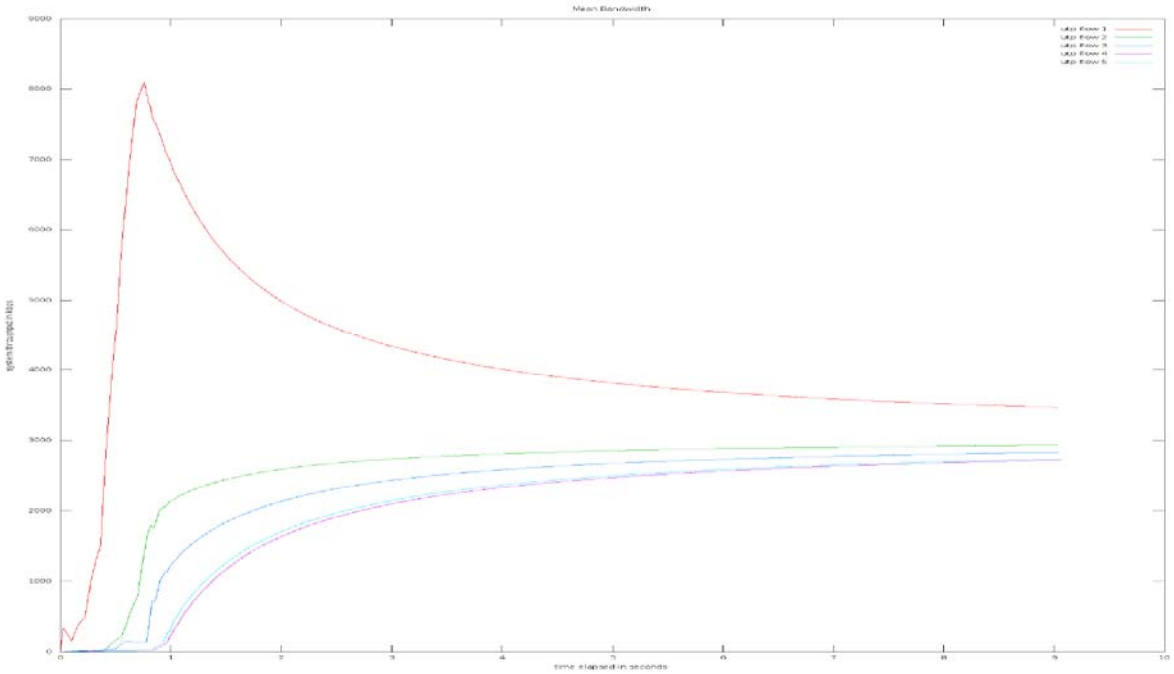

Fig. 17 System throughput with queue size set to 20 packets. **X axis = time elapsed [s], Y axis = throughput [kbps]**

The capacity usage goal is set after the speed of the FIFO equalization - thus after all flows set to the equal goal, they don't set to what they see is the actual throughput but only to the configured capacity usage goal in this example 90 percent of that. As expected the algorithm performs the same on all scenarios - minimal differences was visible between the another (queue size of 20, 50, 100, 1000 and infinite packets) experiments. To compare the presented protocol we believe the best way to see the novelty is to see the existing TCP variants in action and compare visually. Experimental settings (queue size, speed of the exponential search, capacity usage goal) was used on the same values as in previous experiment.

The considered TCP variants where: Binary Increase Congestion control (BIC) and TCP CUBIC: Binary Increase Congestion control. Figure 18 shows TCP BIC system throughput. Figure 19 shows TCP CUBIC system throughput.

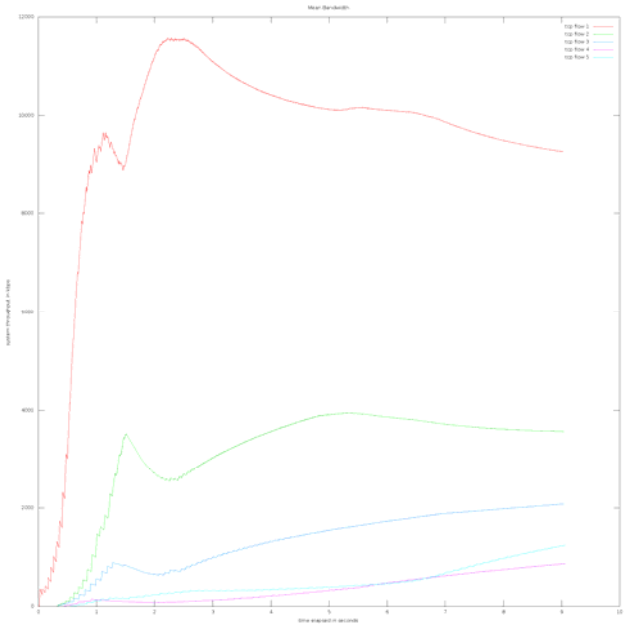

Fig. 18a TCP BIC system throughput for 20 bottleneck queue size. **X axis = time elapsed [s], Y axis = throughput [kbps]**

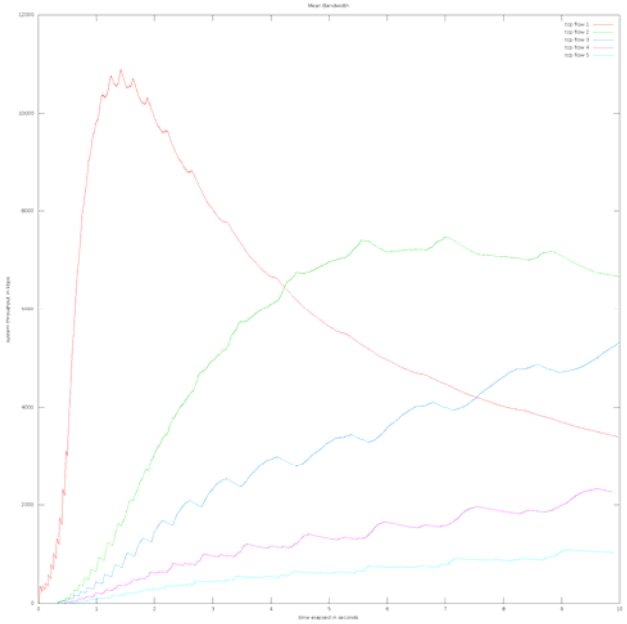

Fig. 18b TCP BIC system throughput for 50 bottleneck queue size. **X axis = time elapsed [s], Y axis = throughput [kbps]**

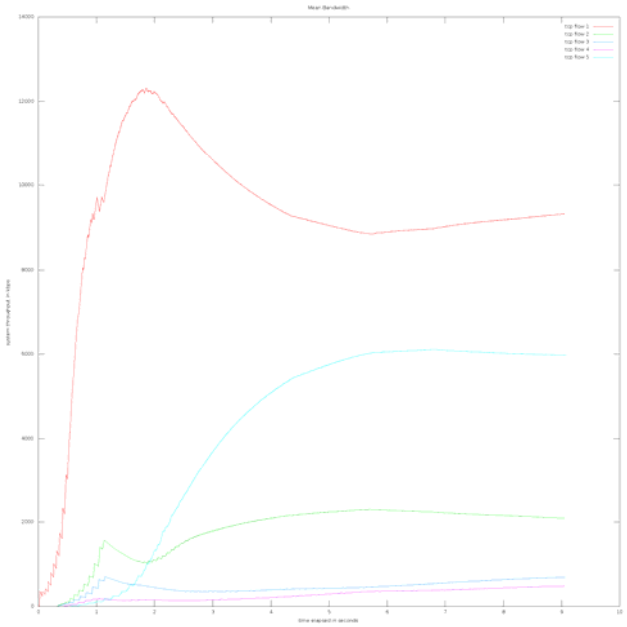

Fig. 18c TCP BIC system throughput for infinite bottleneck queue size. **X axis = time elapsed [s], Y axis = throughput [kbps]**

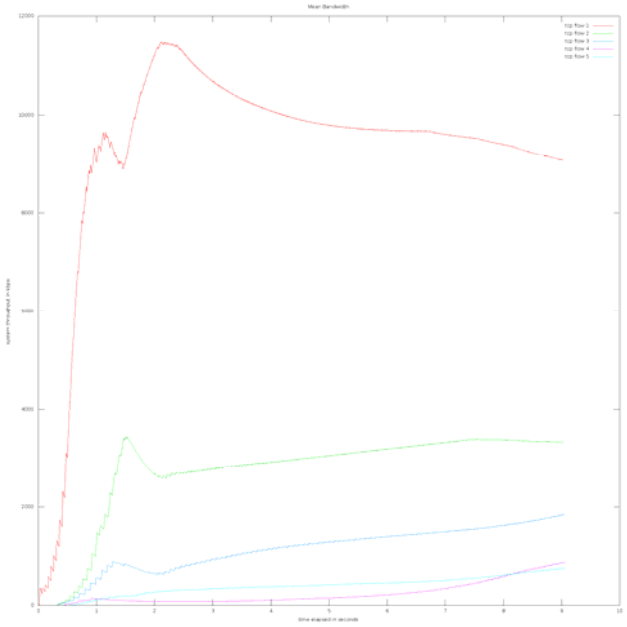

Fig 19a TCP CUBIC system throughput for 20 bottleneck queue size. **X axis = time elapsed [s], Y axis = throughput [kbps]**

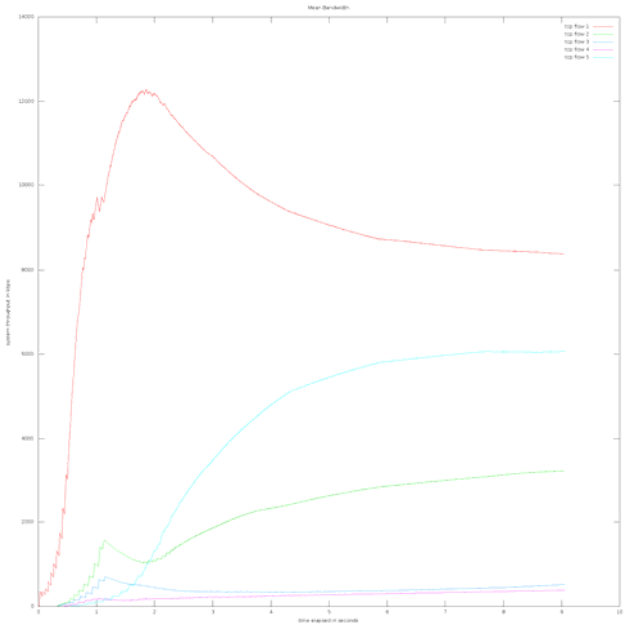

Fig 19b TCP CUBIC system throughput for 50 bottleneck queue size. **X axis = time elapsed [s], Y axis = throughput [kbps]**

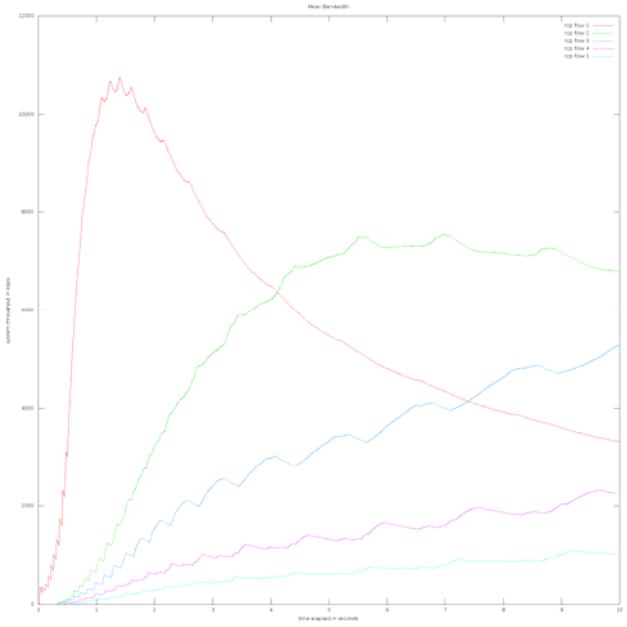

Fig 19c TCP CUBIC system throughput for infinite bottleneck queue size. **X axis = time elapsed [s], Y axis = throughput [kbps]**

After reviewing TCP behaviour, it is clear that none of the variants provide comparable stability, convergence time nor intra-flow fairness to the presented algorithm. Personal opinion of the author is that numeric metrics are useful to compare sub-optimal approaches such as TCP variants in between each other, but hardly useful to compare against a "straight line" which is clearly visible in our approach.

9 CONCLUSION

This paper demonstrated both using simulations and real world measurements that inter-flow communication without any support from the infrastructure can be achieved. We demonstrated proof-of-concepts of both analog and digital modulations over the FIFO jitter side-channel constructed between independent flows sharing the same bottleneck link utilizing the generic FIFO queue behaviour. Further we presented a minimalistic algorithm utilizing this communication to achieve rapid, fair and stable capacity sharing between flows. We hope that the presented proof-of-concepts will ignite wider research interest and drive progress towards a universally optimal transport protocol. Future research focus could include optimization of the analog approximation. Our future research will be aimed towards incorporation of fountain codes as main factors which can possible eliminate the dependency on RTT.

ACKNOWLEDGEMENT

This work is the result of the Project implementation: Competency Centre for Knowledge technologies applied in Innovation of Production Systems in Industry and Services, ITMS: 26220220155, supported by the Research & Development Operational Programme funded by the ERDF.

10 REFERENCES

[1] AYYUB Qazi, I., ZNATI T., ANDREW L.: *Congestion Control using Efficient Explicit Feedback.* In: INFOCOM 2009, IEEE, pp. 10-18. IEEE, 2009. DOI: 10.1109/INFCOM.2009.5061901

[2] BODIN Ulf, OLOV Schelen, PINK Stephen: *Load-tolerant differentiation with active queue management.* In: ACM SIGCOMM Computer Communication Review 30.3 (2000): 4-16. DOI: 10.1145/382179.382180

[3] BOURAS Christos, KANAKIS Nikolaos, KOKKINOS Vasileios, PAPAZOIS Andreas: *AL-FEC for streaming services over LTE systems*. In: Wireless Personal Multimedia Communications (WPMC 1-5.), 2011 14th International Symposium on. IEEE, 2011.

[4] BRAKMO L. S., PETERSON L. L.: *TCP Vegas: end to end congestion avoidance on a global Internet.* In: IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, pp. 1465–1480, 1995. DOI: 10.1109/49.464716

[5] CERF Vinton, DALAL Yogen, SUNSHINE Carl: *Specification of internet transmission control program.* In: INWG General Note, Vol. 72., 1974.

[6] CHEN Shi-Yang, WU Eric Hsiao-Kuang, CHEN Mei-Zhen: *A new approach using time-based model for TCP-friendly rate estimation.* In: Proceedings of the International Conference on Communications (ICC ’03), pp. 679–683, May 2003. DOI: 10.1109/ICC.2003.1204261

[7] CHEN, Jyh-Ming, CHU Ching-Hsiang, WU Eric Hsiao-Kuang, TSAI Meng-Feng, WANG Jian-Ren: *Improving SCTP Performance by Jitter-Based Congestion Control over Wired-Wireless Networks*. In: EURASIP Journal on Wireless Communications and Networking, vol. 2011, 2011. DOI: 10.1155/2011/103027

[8] FLOYD S., KOHLER E., PADHYE J.: *Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC).* In: RFC 4342. March 2006.

[9] FLOYD Sally, JACOBSON V.: *Random early detection gateways for congestion avoidance.* In: Networking, IEEE/ACM Transactions on 1.4 (1993): 397-413. DOI: 10.1109/90.251892

[10] FU Cheng Peng, LIEW S.C: *TCP Veno: TCP enhancement for transmission over wireless access networks.* In: IEEE Journal on Selected Areas in Communications, vol. 21, no. 2, pp. 216–228, 2003. DOI: 10.1109/JSAC.2002.807336

[11] GETTYS Jim, NICHOLS Kathleen: *Bufferbloat: Dark Buffers in the Internet.* In: IEEE Queue 9, no. 11 (2011): 40. DOI: 10.1109/MIC.2011.56

[12] HYUNGSOO Jung, KIM Shin-gyu, YEOM Heon Y., KANG Sooyong, LIBMAN Lavy: *Adaptive delay-based congestion control for high bandwidth-delay product networks.* In: INFOCOM, 2011 Proceedings IEEE, pp. 2885-2893. IEEE, 2011. DOI: 10.1109/INFCOM.2011.5935127

[13] Image Processing Lab, Electronics Department at Politecnico di Torino. Research @ IPL - Digital fountains. [Online: 30.12.2011] http://www1.tlc.polito.it/sas-ipl/research_digital_fountains.php.

[14] KLIMEK Ivan, KELTIKA Marian: *Jitter Utilizing Congestion Estimation (JUICE).* In: Proceedings of Scientific Conference of Young Researchers (SCYR) 2012, FEI TU Kosice, 2012.

[15] KLIMEK Ivan: *Wide Area Network Traffic Optimization - Written work to Doctoral Thesis.* At: Technical University Košice, December 2011.

[16] LUBY M., SHOKROLLAHI A., WATSON M., STOCKHAMMER T., MINDER L.: *RaptorQ Forward Error Correction Scheme for Object Delivery.* In: RFC 6330. August 2011.

[17] MINJI Kim, MÉDARD Muriel, BARROS João: *Modeling network coded TCP throughput: A simple model and its validation.* In: 5th International ICST Conference on Performance Evaluation Methodologies and Tools ValueTools 2011. Newton: EAI Publishing. ISBN 9781936968091.

[18] NAGLE John: *On Packet Switches with Infinite Storage.* In: IEEE Trans. Communications, Vol-35, No. 4, April 1987, pp. 435-438

[19] NANDITA Dukkipati, McKEOWN Nick, FRASER Alexander G.: *RCP-AC: Congestion Control to Make Flows Complete Quickly in Any Environment.* In: INFOCOM 2006. 25 th IEEE International Conference on Computer Communications. Proceedings, pp. 1-5. IEEE, 2006. DOI: 10.1109/INFOCOM.2006.18

[20] NARENDRAN Rajagoplan, MALA C.: *A Comparative Study of Different Queuing Models Used in Network Routers for Congestion Avoidance.* In: Information Technology and Mobile Communication (2011): 431-434. DOI: 10.1007/978-3-642-20573-6_77

[21] SCHULZRINNE H., CASNER S., FREDERICK R., JACOBSON V.: *RTP: A Transport Protocol for Real-Time Application*. In: Internet Engineering Task Force (IETF), RFC 1889, 1996.

[22] SHOKROLLAHI Amin, LUBY Michael: *Raptor Codes: Foundations and Trends(r) in Communications and Information Theory.* In: Boston: Now. ISBN 16-019-8446-4. DOI: 10.1561/0100000060

[23] SO-IN Chakchai: *Loss Synchronization of TCP Connections at a Shared Bottleneck Link.* In: WUSTL Technical Report, 2006.

[24] WU Eric Hsiao-Kuang, CHEN Mei-Zhen: *JTCP: Jitter-Based TCP for Heterogeneous Wireless Networks.* In: IEEE Journal on Selected Areas in Communications, vol. 22, no. 4, pp. 757–766, 2004. DOI: 10.1109/JSAC.2004.825999

[25] WU Eric Hsiao-Kuang, CHENG Yu-Chen: *JRC: jitter-based rate control scheme for wired-wireless hybrid network.* In: International Journal Pervasive Computing and Communications, vol. 3, no. 3, pp. 322-337, 2008. DOI: 10.1108/17427370710856264

[26] XU K., TIAN Y., ANSARI N.: *TCP-Jersey for Wireless IP Communications.* In: IEEE Journal on Selected Areas in Communications, vol. 22, no. 4, pp. 747–756, 2004. DOI: 10.1109/JSAC.2004.825989

[27] YUNHONG G., GROSSMAN R. L.: *End-to-End Congestion Control for High Performance Data Transfer.* In: IEEE/ACM Transaction on Networking, 2003.

[28] ZHANG Lixia, SHENKER Scott, CLARK Daivd D.: *Observations on the dynamics of a congestion control algorithm: The Effects of Two Way Traffic*. In: ACM SIGCOMM '91, Zurich, 1991.DOI: 10.1145/115992.116006.