

FSE-NG for managing real time media flows and SCTP data channel in WebRTC

Julius Flohr, Ekaterina Volodina and Erwin P. Rathgeb

University of Duisburg-Essen
Computer Networking Technology Group
45326 Essen, Germany

{julius.flohr, ekaterina.volodina, erwin.rathgeb}@uni-due.de

Abstract—WebRTC is the upcoming W3C and IETF standard for peer to peer real time communication in web browsers. It allows users to simultaneously transfer audio and video material via the Real Time Transport Protocol (RTP) as well as arbitrary data via the Stream Control Transmission Protocol (SCTP). These two transport protocols provide Quality of Service (QoS) features which are mutually exclusive in networks without Active Queue Management (AQM): RTP tries to maintain low end-to-end delays, while SCTP tries to maximize its throughput. This contradiction leads to a poor user experience for real-time flows due to excessive queuing delay caused by SCTP's loss based Congestion Control (CC). We overcome this issue by extending the Flow State Exchange (FSE) approach by *Islam et al.* to also support flows with loss based CC. By means of simulation, we show that our approach keeps end-to-end delays minimal while also allowing the enforcement of different flow rate priorities.

I. INTRODUCTION & MOTIVATION

WebRTC is an open standard for real time communication implemented in every major web browser. It allows transfer of delay sensitive real time audio and video material using the *Secure Real time Transport Protocol* (SRTP) [1], as well as transport of arbitrary data using the *Stream Control Transmission Protocol* (SCTP) [2]. Combining these two transport protocols in one user land application creates new challenges for Internet congestion control (CC).

A saturated SCTP sender can drastically decrease the quality of the real time communication due to SCTP's loss based congestion control which induces large amounts of queuing delay. However, RTP's delay based congestion control tries to minimize end to end delays in order to enable real time communication. Overcoming these conflicting goals with classical per stream congestion control schemes is hard. Previous work has shown that it can be beneficial to employ a coupling mechanism between the different congestion control instances. This mostly concerns the enforcement of different bandwidth priorities [3] [4], but these solutions are limited to either rate based or window based congestion control schemes and can not mix delay based and loss based congestion control mechanisms. Our previous work [5] has shown that it is possible to mix these congestion control properties in one coupling mechanism in order to prevent an excessive buildup of queuing delay in the network.

There is currently no solution which is capable of reducing queuing delay buildup while also allowing to enforce different priorities for flows. This work aims at combining the two aforementioned approaches in order to reach both goals at the same time.

In this work we extend the existing approach of the *Flow State Exchange* (FSE) by *Islam et al.* [3] with ideas introduced in our previous work [5]. This results in a system which provides the same functionalities as the classical FSE approach but also integrates flows controlled by a window based congestion control while also maintaining low queue levels.

The rest of this paper is structured as follows: Chapter two briefly covers related work. In chapter three we introduce the Flow State Exchange algorithm and our extension to it. Chapter four presents simulation results while chapter five concludes this work.

II. RELATED WORK

Latency is a critical performance parameter for Quality of Experience (QoE) of real time communication in WebRTC.

There are efforts to adapt delay based congestion control algorithms to SCTP, such as *Low Extra Delay Background Transport* (LEDBAT) [6] and *Hamilton Delay-Based Congestion Control* (HD) [7], but previous results by *Becke et al.* [8] show that the improvements in delay are insufficient in a WebRTC context.

Combining the congestion controls of multiple streams can offer several performance benefits: Reduction of overall latency and loss, as well as improvements in fairness and enforcement of different rate priorities. Several approaches are known in literature which propose to do that. Some of them, such as [9] [10], deal with the coordination of concurrent TCP connections, others [11] combine the congestion controllers of RTP flows. There is one approach [4] that proposes to couple all flows independent of transport type, however it is hard to implement and does not focus on improving application to application (A2A) delay. There are currently no proposals which combine loss based and delay based congestion controllers.

III. FSE-NG APPROACH

Our approach is based on [3] by *Islam et al.*. The *Flow State Exchange* (FSE) algorithm consists of two different entities:

Shared Bottleneck Detection (SBD) is used to assign flows that traverse the same bottleneck to a group. The FSE itself is a centralized entity which manages the total bandwidth available (per bottleneck) according to predefined weights.

In case of WebRTC, SBD is not required, since in most use cases SCTP and RTP media material is multiplexed into the same DTLS connection and therefore shares the same 5-tuple. In its current form, the FSE algorithm as described in [11] describes how to couple multiple RTP flows - and neither SCTP flows, nor other loss based congestion control algorithms.

A. Introducing FSE

In order to be able to understand the remainder of this paper, further information will be provided on how the rate calculation in the FSE is realized. When a flow i starts, it registers itself at the FSE with the congestion controller's initial rate $FSE_{R_i} = R_{min}$ and adds it to the sum of all media flow sending rates (total current output rate) S_{CR} . Every time the congestion controller of the flow calculates a new sending rate CCR_i , S_{CR} is updated:

$$S_{CR} = S_{CR} + CCR_i - FSE_{R_i} \quad (1)$$

Therefore, S_{CR} represents the total available bandwidth for all media flows.

After that, the FSE calculates the new sending rate for every flow according to its priority P_i , where P_{total} is the sum of all flow priorities.

$$FSE_{R_i} = \frac{S_{CR} \cdot P_i}{P_{total}} \quad (2)$$

B. Challenges and Pitfalls

When integrating window based flows in to the existing FSE algorithm, several pitfalls have to be avoided:

a) *Updating available bandwidth:* When determining the bottleneck capacity, it is important to do this without causing additional queuing delay. Therefore, no congestion control information from the loss based flows is used to update S_{CR} . This is similar to [5].

b) *Converting bit rates to congestion windows:* The FSE algorithm operates on rates, while the standard congestion control algorithms of SCTP and TCP operate on congestion windows ($CWND$), i.e. a maximum amount of data a stream may have in flight before receiving an acknowledgment. There are approaches which describe the relation between throughput and window size for standard TCP [12], however in the case of FSE this is relatively trivial: The complexity when calculating TCP's throughput based on a congestion window stems from periodic losses introduced by TCP's capacity seeking strategy. Our FSE implementation limits the sending rate of loss based flows by calculating an upper limit for congestion window of every flow ($CWND_{max_i}$). This limit is calculated using information derived from the delay based congestion controls of the real time media material. This means that the congestion window of the flow is limited before losses caused by overfull

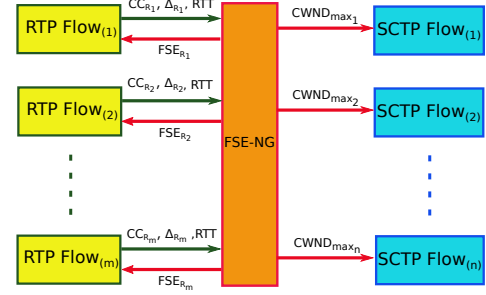


Fig. 1. FSE-NG Architecture

queues occur, making the behavior of the loss based sender more like BBR [13] than a classical loss based congestion control, in the sense that it tries to keep $CWND_{max}$ at the flow's share of the bandwidth delay product. Thus, the $CWND_{max}$ of a loss based flow i can simply be approximated by the bandwidth delay product of the target rate R_i and the base round trip time (RTT) of the link RTT_{base} :

$$CWND_{max_i} = R_i \cdot RTT_{base} \quad (3)$$

However, since we are computing an upper limit of the congestion window, the loss based flow still reacts to transmission errors and packet losses. Thus, the rates FSE-NG calculates for window based flows can be understood as upper limits. The actual throughput can be smaller if the a flow's congestion control algorithm makes the decision not to use the entire congestion window available.

C. FSE-NG

The basic idea behind the extended Flow State Exchange algorithm is to get an estimate of the available bottleneck capacity (S_{CR}). This is then shared among all flows according to their priority(p). Figure 1 shows an overview of its general architecture. Each RTP flow provides congestion control information (current RTT , calculated sending rate CCR and Δ_R) to the FSE-NG instance which then uses it to calculate new output rates for each RTP and SCTP flow. Where CCR is the absolute sending rate calculated by the congestion control and Δ_R is the relative change in sending rate the congestion controller would like to apply. This distinction has to be made since CCR can be limited by the media source (e.g. maximum video resolution of a webcam). Please note that no congestion control information from the loss based flows is used for this calculation. The signal used for estimation is therefore purely delay based.

Algorithm 1 describes the flow update mechanism of the improved FSE algorithm in detail. The process of flow registration and deregistration remains unchanged with one exception: If the last RTP flow leaves, the window limits of all window based flows are removed since no updated congestion control information is available. During initialization, the sum of all calculated rates (S_{CR}) is initialized to zero. As described in the original FSE publication [3], RTP flows add or remove their (initial) sending rates from the sum. To be able to perform the

conversion between sending rates and a congestion window, an estimate of the path's base round trip time is required (RTT_{base}). This estimate is initialized to infinity.

Assume a RTCP feedback packet j is received at a RTP sender. This packet carries congestion control information such as estimates for queuing delay, ECN and packet loss ratios which are used by the sender's congestion control algorithm to compute a new, desired absolute sending rate (CCR), a relative change in sending rate Δ_R as well as an estimate of the current RTT . This information is passed to the FSE-NG implementation as input parameters. Output of the algorithm is a sending rate FSE_i for every RTP flow, and a $CWND_{max_i}$ for every SCTP flow i . With the absolute rate CCR and the associated relative change in sending rate Δ_R computed by the RTP congestion control, it is possible to obtain an estimate of the bottleneck's bandwidth (SCR): Depending on whether there is currently at least one loss based saturated flow active, either the difference in sending rate compared to the last RTCP update packet, or the sum of the difference and Δ_R is added to the estimate. This causes SCR to converge to the bottleneck capacity but prevents it from overestimating the bandwidth when it is not actually used by any flow (Algorithm 1, lines 3-7). It is possible to calculate the shares of each RTP flow using the same rate calculation scheme as proposed in the initial FSE publication. There, a share of the total bandwidth is allocated that is proportional to the flow's priority divided by the sum of all priorities in the flow group (including all window based CCs). This corrected flow rate is then passed to the congestion controllers of the individual flows. However, one has to be aware of the maximum possible output rate of the current flow (R_{max}) (Algorithm 1, line 9). When tracking all calculated rates of all RTP flows, it is possible to calculate

Algorithm 1 FSE-NG Algorithm

Input: $CCR_j, \Delta_{R_j}, RTT_j$

Output: $FSE_i, CWND_{max_i}$

initialization:

1: $SCR := 0$

2: $RTT_{base} := \infty$

upon arrival of RTCP message i for flow j :

3: **if** no window based flows present or all app. limited **then**

4: $SCR = SCR + CCR_{ij} - FSE_{i,j-1}$

5: **else**

6: $SCR = SCR + CCR_{ij} - FSE_{i,j-1} + \Delta_{R_i}$

7: **end if**

$SR_{RTP} = 0$

8: **for all** RTP flows k **do**

9: $R_k = \min(\frac{P_k \cdot SCR}{P_{total}}, R_{max})$

10: $SR_{RTP} = SR_{RTP} + R_k$

11: **end for**

12: $SR_{SCTP} = SCR - SR_{RTP}$

13: $RTT_{base} = \min(RTT_{base}, RTT_j)$

14: **for all** (window based flows k in FG) **do**

15: $CWND_{max_k} = \frac{P_k \cdot SR_{SCTP} \cdot RTT_{base}}{P_{wnd}}$

16: **end for**

the share of the total bandwidth that can be assigned to the loss based flows (Alg. 1, line 12). This bandwidth is shared among all loss-based flows using the same algorithm as for the RTP flows, however here the sum of only the window based flows is used (P_{wnd}). Now these rates have to be converted to actual congestion windows. This done using equation 3. For this, we assume that the base RTT is equivalent to the minimal RTT we observe during the lifetime of the connection. This can be done due to the fact that the RTP congestion control algorithm uses the same assumption in its design.

IV. RESULTS

We have implemented our algorithm in the OM-NeT++/INET [14] discrete event simulation framework. The simulation scenario consists of a simple dumbbell network topology with varying base delay and bottleneck capacity. In our simulations we used the RTP/NADA implementation we already used in [5] and which has been developed using feedback from the original NADA authors. Our SCTP implementation shares most of its code base with the kernel of the FreeBSD¹ operating system. In this scenario, two RTP video streams controlled by the NADA congestion control algorithm, together with one SCTP (default CC) flow, are managed by our improved FSE-NG implementation.

We used a saturated traffic source for SCTP and the Perfect Video Codec from the Synccodecs² adaptive synthetic video codec for RTP. This codec is also used at the IETF for benchmarking different RTP congestion control algorithms. In the scenario evaluated, three equally weighted flows share a bottleneck-link with 3.5MBit/s and 50ms one-way delay (OWD). The router at the bottleneck link has a buffer capacity of 100 packets (340ms maximum queuing delay for full-sized segments) and uses a standard droptail queue management. The RTP flows are started at time $t = 0\text{s}$ while SCTP begins a saturated data transfer at $t = 15\text{s}$. RTP flow 2 stops transmission at $t = 40$, RTP flow 1 at $t = 60\text{s}$ and SCTP at $t = 70\text{s}$.

Figures 2 and 3 depict throughput and delay of all flows coupled with our improved version of the the FSE-NG algorithm. All flows have the same priority and obtain an optimal share of bandwidth. Even when RTP flow 2 terminates at $t = 40\text{s}$, the bandwidth that becomes available is immediately shared among the remaining flows while also respecting the maximum output rate R_{max} .

Since SCTP's maximum congestion window is directly controlled by delay based congestion control algorithms, it converges to a rate which saturates the available link capacity but does not cause any extra queuing delay. Instead of several hundred milliseconds end-to-end delay (with standard FSE), it now quickly converges to an operating point below 100ms where real-time communication is possible. When the second RTP flow terminates at $t = 60\text{s}$, SCTP's $CWND_{max}$ is removed and the queuing delay immediately rises again.

¹<https://www.freebsd.org/de/>, accessed 2nd of May 2018

²<https://github.com/cisco/synccodecs>, accessed 2nd of May 2018

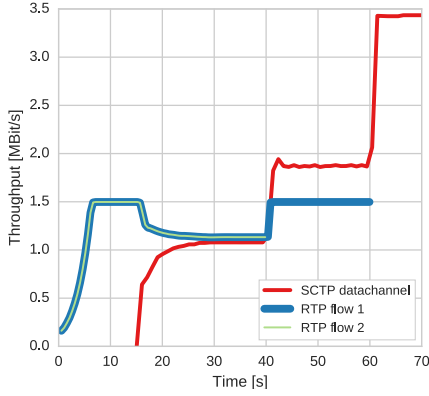


Fig. 2. Throughput of FSE-NG. 3.5Mbit/s bottleneck, 50 ms OWD, 340 ms max. queuing delay

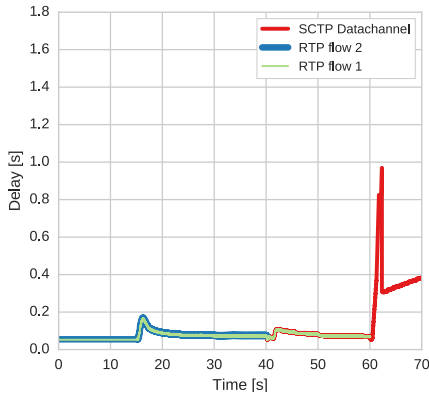


Fig. 3. A2A delay of FSE-NG. 3.5Mbit/s bottleneck, 50 ms OWD, 340 ms max. queuing delay

A. Parameter Study

We have performed an exhaustive parameter study in order to ensure that our proposed algorithm works in a wide range of scenarios. We explored the parameter space in various dimensions: Bottleneck capacity, one way propagation delay, bottleneck router queue length as well as various random background traffic patterns and flow priorities. We found that our algorithm significantly improved the end to end delays experienced while minimally impacting SCTP throughput for small BDP links. On high BDP links, our algorithm was able to increase SCTP throughput since packet losses caused by network congestion are prevented.

V. CONCLUSION AND FUTURE WORK

In this work we present an improvement of the existing Flow State Exchange algorithm, called FSE-NG. This allows not only the coordination and prioritization of rate based real time media traffic originating from one sender, but also the inclusion of arbitrary window based transport protocol flows. Our simulation results show that this scheme works for different bottleneck scenarios as well priorities while drastically

reducing queuing delay experienced in every scenario tested. This allows real time communication in many scenarios where it would not have been possible before due to poor user experience. Our research suggests that this scheme works for a large parameter space and is generally applicable to common WebRTC scenarios.

Future work includes evaluation of the FSE-NG algorithm with other congestion control algorithms. Especially, it is interesting to see whether delay-based control approaches [6] [7] offer any further reduction in queuing delay.

ACKNOWLEDGMENT

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) RA1044/6-1.

REFERENCES

- [1] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," RFC 3711 (Proposed Standard), Internet Engineering Task Force, Mar. 2004, updated by RFCs 5506, 6904. [Online]. Available: <http://www.ietf.org/rfc/rfc3711.txt>
- [2] R. Stewart, "Stream Control Transmission Protocol," RFC 4960 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 6096, 6335, 7053. [Online]. Available: <http://www.ietf.org/rfc/rfc4960.txt>
- [3] S. Islam, M. Welzl, D. Hayes, and S. Gjessing, "Managing real-time media flows through a flow state exchange," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 112–120.
- [4] H. Balakrishnan and S. Seshan, "The Congestion Manager," RFC 3124 (Proposed Standard), Internet Engineering Task Force, Jun. 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3124.txt>
- [5] J. Flohr and E. P. Rathgeb, "Rosiee: Reduction of self inflicted queuing delay in webRTC," in *Teletraffic Congress (ITC 29), 2017 29th International*, vol. 3. IEEE, 2017, pp. 7–12.
- [6] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," RFC 6817 (Experimental), Internet Engineering Task Force, Dec. 2012. [Online]. Available: <http://www.ietf.org/rfc/rfc6817.txt>
- [7] L. Budzisz, R. Stanojevic, R. Shorten, and F. Baker, "A strategy for fair coexistence of loss and delay-based congestion control algorithms," *IEEE Communications Letters*, vol. 13, no. 7, 2009.
- [8] M. Becke, E. P. Rathgeb, S. Werner, I. Rüngeler, M. Tüxen, and R. Stewart, "Data channel considerations for RTCWeb," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 34–41, 2013.
- [9] J. Touch, "TCP Control Block Interdependence," RFC 2140 (Informational), Internet Engineering Task Force, Apr. 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2140.txt>
- [10] M. Savorić, H. Karl, M. Schläger, T. Poschwatta, and A. Wolisz, "Analysis and performance evaluation of the efcm common congestion controller for tcp connections," *Computer Networks*, vol. 49, no. 2, pp. 269–294, 2005.
- [11] S. Islam, M. Welzl, and S. Gjessing, "Coupled congestion control for RTP media," Internet Engineering Task Force, Internet-Draft draft-ietf-rmcat-coupled-cc-07, Sep. 2017, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-ietf-rmcat-coupled-cc-07>
- [12] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.
- [13] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh *et al.*, "BBR: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.
- [14] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 60.