Awesome title here

Managing Real- Time Video and Data Flows with Coupled Congestion Control Mechanism

Tobias Fladby

Thesis submitted for the degree of
Master in programming and system architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2021

Awesome title here

Managing Real- Time Video and Data Flows with Coupled Congestion Control Mechanism

Tobias Fladby

Awesome title here

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Contents

Preface

Chapter 1

Introduction

1.1 Problem statement

1.2 Contributions

1.3 Research questions

1.4 Organization

Chapter 2

Background

2.1 WebRTC architecture

2.1.1 Real- time communication

2.1.2 Standardization

2.1.3 User API

Services

RTCPeerConnection API

DataChannel API

2.1.4 Signalling

NAT

ICE Framework

TURN

STUN

SDP

2.1.5 Encryption

TLS

DTLS

2.1.6 Usage

2.1.7 Browser engine

Aquiring WebRTC statistics

2.2 Transport protocols

2.2.1 TCP

2.2.2 UDP

2.2.3 RTP and RTCP

2.2.4 SCTP

2.3 Congestion control

latency in order to assure a good user experience.

History and previous research [cite relevant stuff, like congestion collapse]has shown that protocols should employ mechanisms that limit the amount of data sent per second to a reasonable level in order to avoid congestion as well as keep the latency low.

2.3.1 Google Congestion Control

RTP by itself only provides simple end- to- end delivery services for multimedia[cite RTP standard], since real- time communication requires congestion control it must implemented on top of RTP. Chromium's WebRTC implementation uses an algorithm called Google Congestion Control [1] to provide the mechanism. It consists of two controllers, one loss- based and one delay- based. The loss- based controller located on the sender- side, uses loss rate, RTT and REMB[Cite REMB message definition] messages to compute a target sending bitrate. The delay- based controller can either be implemented on the receiver- side or sender- side. It uses packet arrival info to compute a maximum bitrate which is passed to the loss- based controller. The actual sending rate is set to the minimum of the two bitrates.

The loss- based controller

The loss- based controller is run every time a feedback message from the receiver- side is received. If more than 10% of packets have been lost when feedback is received the controller decreases the estimate. If less than 2% is lost it will increase the estimate under the presumption that there is more bandwidth to utilize. Otherwise the estimate stays the same.

The delay- based controller

The delay- based controller consists of several parts: pre- filtering, an arrival- time filter, an over- use detector and a rate controller.

Pre- filtering is used to make sure that channel outages, events unrelated to congestion are not interpreted as congestion. Packets will naturally be delayed when a channel outage occurs so without this filter the algorithm would unnecessarily lower bitrate, thus lowering the quality of the communication for no reason. A channel outage will cause the packets to be queued in network buffers, thus when the channel is restored the packets will arrive in bursts. The filter utilizes the fact that the packet groups will arrive in bursts during a channel outage and merges them under such conditions.

The arrival- time filter is responsible for calculating the queueing time variation which is an estimation of how the delay is developing at a certain time. The goal of the over- use detector is to produce a signal that drives the state of the remote rate controller. The goal of the over- use detector is to compare the queueing time variation obtained as output from the arrival-

time filter with a threshold. If the estimate is above the threshold for a certain amount of time and not sinking it wil signal the rate contol.

Performance

2.3.2 NADA

2.3.3 Scream

2.3.4 Problems with combined controls

2.4 Coupled Congestion Control

Coupled congestion control [2] is a mechanism that aims to improve the delay, loss and fairness for flows travelling over the same bottlenecks by combining their congestion controls. The system has two main components, the Shared Bottleneck Detection(SBD) and the Flow State Exchange(FSE).

2.4.1 Managing flows

2.4.2 The Flow State Exchange

The FSE can be described as a manager that maintains information about the flows and hands out the allowed bit rate for each flow depending on several factors. Firstly, it takes into account how many other flows that are sharing a bottleneck, to make this possible every flow register themselves with the FSE and SBD when they start. For each flow FSE also keeps stores a priority number, the rate used by the flow and the desired rate of the flow.

2.5 Shared Bottleneck Detection

The SBD is an entity that is responsible for determining which flows are traversing the same bottleneck. In [2] three methods for deriving if flows share the same bottleneck are mentioned.

2.5.1 Multiplexed flows

One way is through comparing multiplexed flows. Since the flows with the same five- tuple will be routed along the same path, SBD can assume that they share the same bottleneck. However this method cannot be used for coupled congestion controllers with one sender talking to multiple receivers, given that they will not have the same five- tuple. Since WebRTC uses both SRTP and SCTP multiplexed on UDP, this implies they have the same five- tuple and that the first method will work.

2.5.2 Measurement

One might also use measurements of e.g. delay and loss and look at correlations to derive if flows have a shared bottleneck.

2.5.3 Configuration

Chapter 3

Design

Chapter 4

Implementation

Chapter 5

Evaluation

5.1 Testbed

5.2 Experiments

Chapter 6

Conclusion

6.1 Research Findings

6.2 Further work

6.3 Closing remarks

Bibliography

[1] Stefan Holmer et al. *A Google Congestion Control Algorithm for Real-Time Communication*. Internet-Draft draft-ietf-rmcat-gcc-02. Work in Progress. Internet Engineering Task Force, July 2016. 19 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-rmcat-gcc-02.

[2] Safiqul Islam, Michael Welzl and Stein Gjessing. *Coupled Congestion Control for RTP Media*. RFC 8699. Jan. 2020. DOI: 10.17487/RFC8699. URL: https://rfc-editor.org/rfc/rfc8699.txt.