NADA Fairness: Analyzing the Future of WebRTC

Serena Vincent (*Spelman College)*, Adithya Phillip, Rukshani Athapathu, Justine Sherry, Srinivasan Seshan
{svincent}@scmail.spelman.edu, {dathapat}@andrew.cmu.edu, {aphilip, sherry, srini}@cs.cmu.edu
Carnegie Mellon University

1. INTRODUCTION

Real-time communications are increasing over the web and there is no standard congestion control algorithm defined for WebRTC (eg., video conferencing and live stream), leaving services to find their own CCA to use when implementing their platform. However, there is no standard congestion control algorithm (CCA) defined for WebRTC, leading to services having to create or find their own CCA to use on their platform. Hence, the IETF is considering three algorithms to become the standard for WebRTC: SCReAM [1], GCC [5], and Network-Assisted Dynamic Adaption (NADA) [6]. NADA's core congestion algorithm creates a unified congestion signal, including packet loss, queuing delay, and explicit congestion notification (ECN) markings [3]. NADA takes this approach in hopes of being able to react to fast changes in the network, allow for weighted bandwidth sharing for multiple competing video streams, and sustain a significant amount of bottleneck bandwidth when competing with TCP [6]. NADA is one of the algorithms being considered, but we do not know if it is fair to other CCAs when a NADA flow competes with a legacy CCA. Earlier work on congestion control intends to satisfy two criteria for real-time media traffic: TCP-friendliness (outgoing rate is equal to a comparable TCP flow) [4] and media-friendliness (media streaming rate stays smooth) [2]. Our focus is on the former, TCP-friendliness, and it leads to the question: *Is NADA fair to other algorithms deployed on the Internet?*

We answer this question by studying NADA in a controlled ns-3 environment against TCP Reno, TCP Cubic, and itself, varying testbed configurations to understand if NADA is fair enough to be considered the standard algorithm for WebRTC. We found that NADA usually takes less than its fair share of bandwidth, but in some situations, like competing against multiple NewReno flows, can take more than its fair share.

2. SETUP

We used a discrete event network simulator, ns-3, to model a simplified version of real-time media congestion and evaluate real-time media CCAs in a simulation environment. This ns3-rmact implementation is created by Cisco and sends fake codec data to simulate a RTC environment. A sender application, `RmcatSender`, sends media packets of fake video codec data to the receiver application, `RmcatReciever`. The receiver application gets a sequence of packets and timestamp information, sending it back to the sender in feedback packets. The CCA running on the sender processes feedback information to obtain the bandwidth estimation, using this information to estimate to control the fake video encoder. We used point-to-point wires in ns-3 to implement a dumbbell network topology. The simulation environment consists of a 10 Mbps bottleneck link that runs for 100 seconds. We configure NADA with a maximum encoding rate of 20 Mbps and a minimum encoding rate of 150 Kbps.

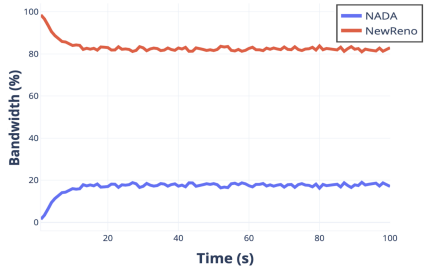

Figure 1: Bandwidth (BW) share for NADA vs TCP NewReno at default parameters

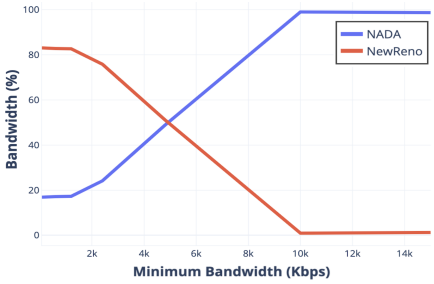

Figure 2: NADA v TCP NewReno BW share when varying NADA's minimum BW parameter

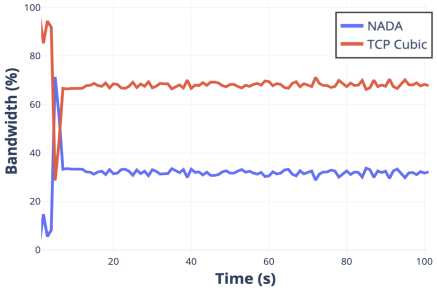

Figure 3: BW share for NADA v Cubic flows with default parameters

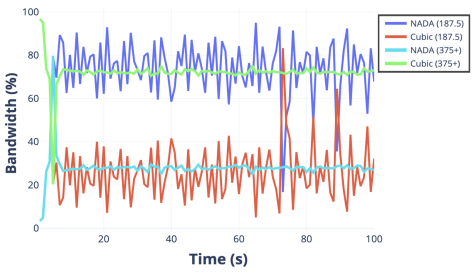

Figure 4: BW share for NADA v Cubic flows with a buffer size of 187.5 and 375+ packets with default parameters

3. RESULTS

NADA vs TCP NewReno

Depending on the configuration, NADA can take as little as 17% and as much as 99% of the fair share. Hence, NADA's behavior ranges from very unfair to very fair (generous). Figure 1 shows the percentage in bandwidth for two competing flows sharing the same bottleneck link. NADA is much more complacent when using the bottleneck link and performs poorly when competing against NewReno.

When testing NADA vs. TCP NewReno with varying buffer sizes, we found that varying the queue size for a small bottleneck bandwidth (1.5 Mbps) can increase the NADA rate. However, when the link is at the default link rate (10 Mbps), NADA stays at the same rate, continuously taking a bit more than 17% of the bandwidth versus the 83% for NewReno.

Surprisingly, an increase in NADA's minimum bandwidth value can make NADA extremely unfair. Figure 2 shows competing NADA and NewReno flows, doubling from 75 Kbps for each experiment. Both flows take their fair share of the bandwidth when the minimum is around 5 Mbps, but the higher the minimum, the greater amount of bandwidth NADA takes.

We also found that the more NewReno flows that compete with a NADA flow, the *more* bandwidth NADA takes. Figure 5 displays this surprising finding, demonstrating that NADA takes around 80% of the bandwidth, leaving only 20% for the five NewReno flows. This shows NADA is unable to fairly allocate bandwidth to multiple competing flows.

NADA vs TCP Cubic

When NADA's core algorithm competes with TCP Cubic, it continues to take less than its fair share, only consuming around one-third of the bottleneck link, as seen in Figure 3.

For testing NADA against Cubic with varying queue sizes, we found that NADA continuously takes less than its fair share of bandwidth for all buffer sizes above 375 packets, but for a buffer size below that, the queue size is too small to accommodate NADA and Cubic at steady rates, as seen in Figure 4.

Varying NADA's minimum bandwidth produces similar results as compared to TCP NewReno, fair around 5 Mbps, but increasingly becomes more unfair and takes more bandwidth away from TCP Cubic.

NADA vs NADA

Figure 6 shows NADA versus itself with one fixed RTT flow and another with a varying RTT, finding that a NADA flow with a fixed RTT is able to accommodate other NADA flows with sluggish responses. This is unusual when comparing to NewReno, as NewReno is known for being unfair to competing flows with higher RTTs. Knowing this, NADA streams with varying response times will be able to coexist with each other if deployed on the Internet more fairly than NewReno flows coexist with each other.

4. CONCLUSION & FUTURE WORK

We seek to understand if NADA is fair enough to be considered the standard for WebRTC and if it can be deployed across the Internet. We tested its fairness against other widely deployed algorithms while varying various network and protocol properties and found NADA does not take its fair share of the bottleneck link when competing against a few widely deployed algorithms on the Internet. In future work, we plan to test this algorithm against more CCAs (e.g., BBR, SCReAM, GCC) and consider varying other network properties (e.g., measuring packet loss). We also plan to investigate why NADA is aggressively taking bandwidth when competing with multiple Reno flows.

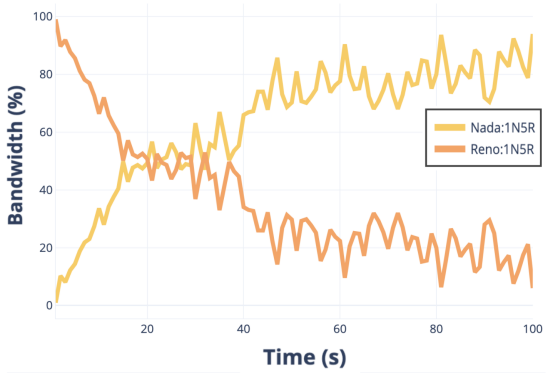

Figure 5: Rate for one NADA vs five Reno flows sharing a bottleneck link with default parameters

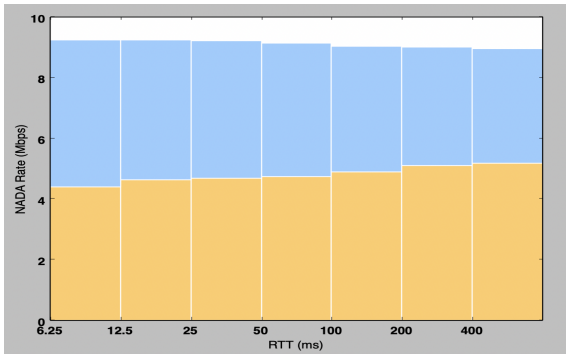

Figure 6: Rate for two competing flows sharing a bottleneck link with a fixed (yellow) and varied (blue) RTT flow

5. REFERENCES

[1] I. Johansson, Z. Sarker, and Ericsson AB, "Self-Clocked Rate Adaptation for Multimedia," RFC 8298 (Experimental), Dec. 2001.

[2] J. Yan, K. Katrinis, M. May, and B. Plattner, "Media- and TCP-friendly congestion control for scalable video streams," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 196–206, Apr. 2006.

[3] K. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168 (Proposed Standard), Sep. 2001.

[4] S. Floyd, M. Handley, J. Pahdye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," RFC 5348 (Proposed Standard), Sep. 2008.

[5] S. Holmer, H. Lundin, G. Carlucci, L. De Cicco, and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication," Internet-Draft (Informational), Jan. 2017.

[6] X. Zhu, R. Pan, M. Ramalho, and S. Mena, "Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media," RFC 8698 (Experimental), Feb. 2020.