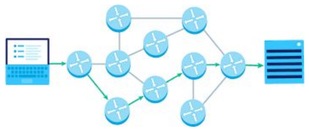
Congestion Control with TCP Hybla

By: Julia Gersey, Audrey Kim, Vasu Ramanujam, Lisa Shen

Introduction

With millions of people connecting to the internet daily, millions of devices are vying for their fair (or unfair) share of bandwidth. **Congestion control algorithms** (CCA) determine the allocation of bandwidth and prioritizes what is fair.



However, the parameters of the network affect the way certain algorithms adapt to sharing the network. These parameters can include **round trip time (RTT)** (the amount of time it takes for a packet of data to go from the sender to the receiver and back), **bandwidth,** and number of **flows** (connections of packets being sent).

TCP Hybla

Created in 2004, TCP Hybla [1] is a **congestion control algorithm** that optimizes for satellites. Typically, when competing against other CCAs, satellite connections are disadvantaged due to their long **RTT**, reducing the number of updates the CCA can make. TCP Hybla solves this problem by scaling another common algorithm, TCP Reno, to an arbitrary reference RTT of 25 ms. After each time interval of length equivalent to the RTT, Hybla updates by a ratio of $(RTT/RTT_0)^2$, where RTT is the current RTT and RTT_0 is the assumed reference RTT, as opposed to 1 (the update size of TCP Reno) [2].

Problems

With the variety of CCAs proposed and deployed throughout the years, how do we ensure fairness?

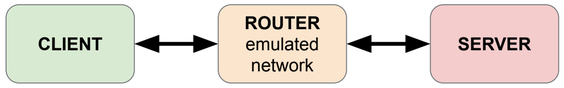
How does Hybla compete with itself for bandwidth with different parameters? How does this help answer the question of fairness?

Carnegie Mellon University

School of Computer Science

Methods

We logged into the remote testbed using ssh and emulated controlled traffic workloads.



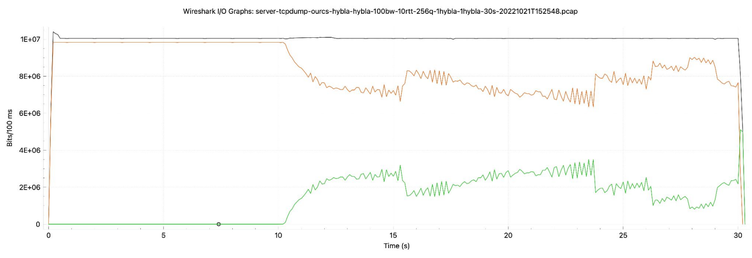
Configure:	*Configure:*	*Capture*
Two batches of flows, congestion control algorithm, start time, duration	Bandwidth, delay, and queue size	*tcpdump* trace

By changing the bash script, we could evaluate how Hybla performs against itself while changing the following parameters: CCA, Bandwidth, Round Trip Time, Number of Flows, and Duration. We initialized half the flows to begin at 0 seconds and half to begin after 10 seconds.

Results

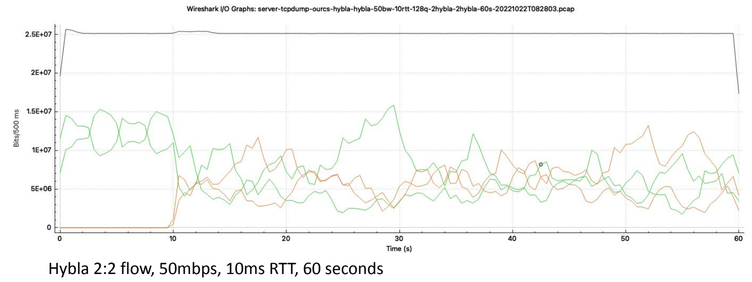
Hybla vs. Hybla

During our first run on Hybla v. Hybla, we noticed a time frame of 30s was inadequate to observe convergence.



Hybla 1:1 flow, 100mbps, 10ms RTT, 30 seconds

Thus, we ran subsequent experiments for a duration of 60s and switched the flows to 2 each.

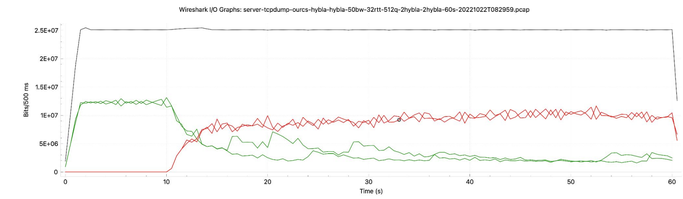


Hybla 2:2 flow, 50mbps, 10ms RTT, 60 seconds

With an RTT under 25ms, Hybla doesn't modify the update function, and thus shares bandwidth relatively equally.

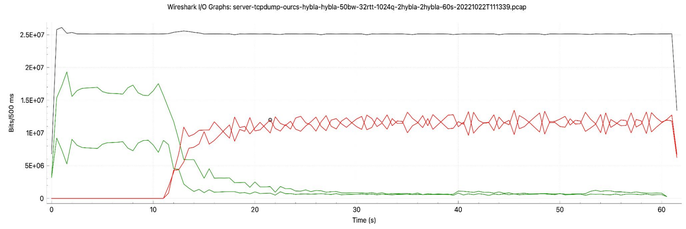
Results (cont'd)

Through the following run of the experiment, we noticed that as we increase the RTT past 25ms, the normalization between Hybla and Hybla reach an unfair distribution. We hypothesized that the RTT estimate of the late-coming flow was approximately 3 times larger, making it more aggressive.



Hybla 2:2 flow, 50mbps, 32ms RTT, 60 seconds, queue = 2BDP

To test this hypothesis, we increased the queue size and expected the ratio to become more disparate. The graph below shows this radicalization of unfairness.



Hybla 2:2 flow, 50 mbpss, 32 ms RTT, 60 seconds , queue = 4BDP

Conclusion

The average ratio of the throughput of flows in the experiment with a queue size of 2BDP was about 1:4 in the late-coming flow's favour, whereas increasing the queue to a size of 4BDP led to it being about 1:15.
Through experimentation with how the change of parameters influence the disparity and unfairness of the Hybla Congestion Control Algorithm, we determined that Hybla's fairness depends on the initial estimation of RTT. It appears that Hybla doesn't recalculate RTT and therefore, late-coming flows are more aggressive due to a higher RTT estimate.

References

[1] Carlo Caini and Rosario Firrincieli. 2004. TCP Hybla: a TCP enhancement for heterogeneous networks. Int. J. Satell. Commun. Netw. 22, 5 (September 2004), 547–566. https://doi.org/10.1002/sat.799
[2] Dordal, Peter. "Newer TCP Implementations." ;Newer TCP Implementations - An Introduction to Computer Networks, Desktop Edition 2.0.9, https://intronetworks.cs.luc.edu/current2/html/newtcps.html#tcp-hybla.