NANYANG TECHNOLOGICAL UNIVERSITY

School of Computer Science and Engineering (SCSE)

SC4052: Cloud Computing

Academic Year: 2024-2025, Semester 2

Assignment 1

M Hisham B Khairul A (U2121992E)

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Contents

1. AIMD Parameter Tuning

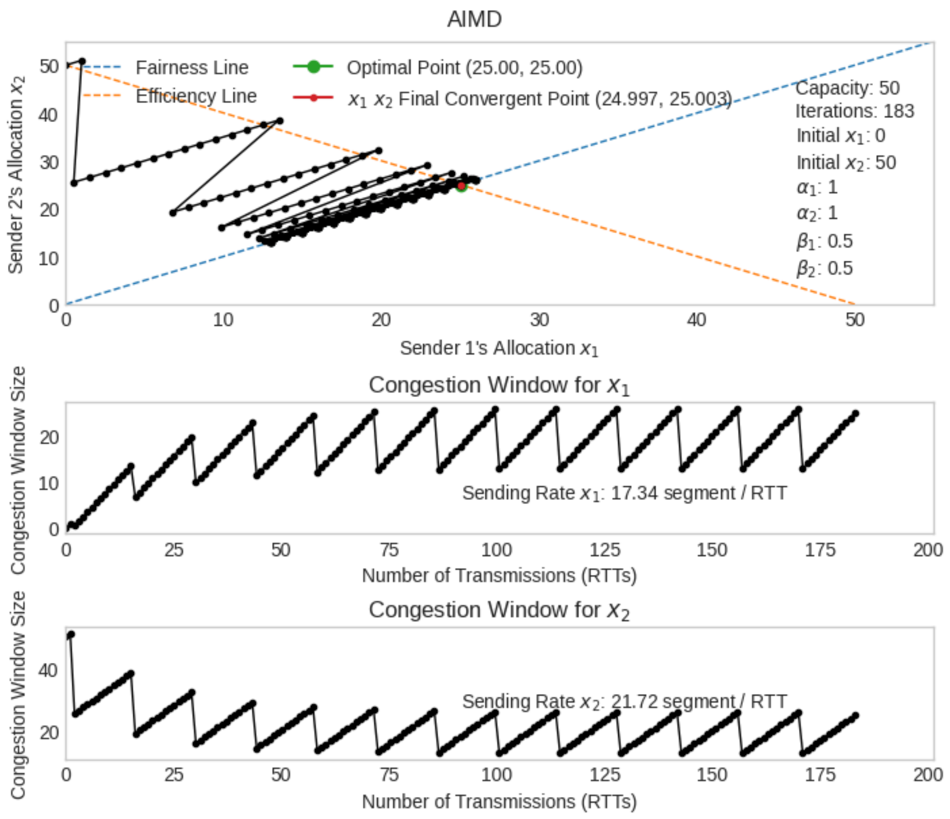
The default - Additive Increase Multiplicative Decrease (AIMD)

AIMD is a common mechanism in TCP, used by senders to determine the optimal sending rate while minimizing packet drops/congestion. Its process is as follows:

- Senders start with a window size *cwnd,* the number of packets sent out at any time.
- For each time slice, the window size is **increased additively** by a factor α.
- This occurs until packet drops/congestion is detected via multiple ACKs.
- Then, it is **decreased exponentially** by a factor β.

This mechanism leads to an eventual convergence of sending rates for senders, where the rates are **fair** (each sender gets an equal rate in the network).

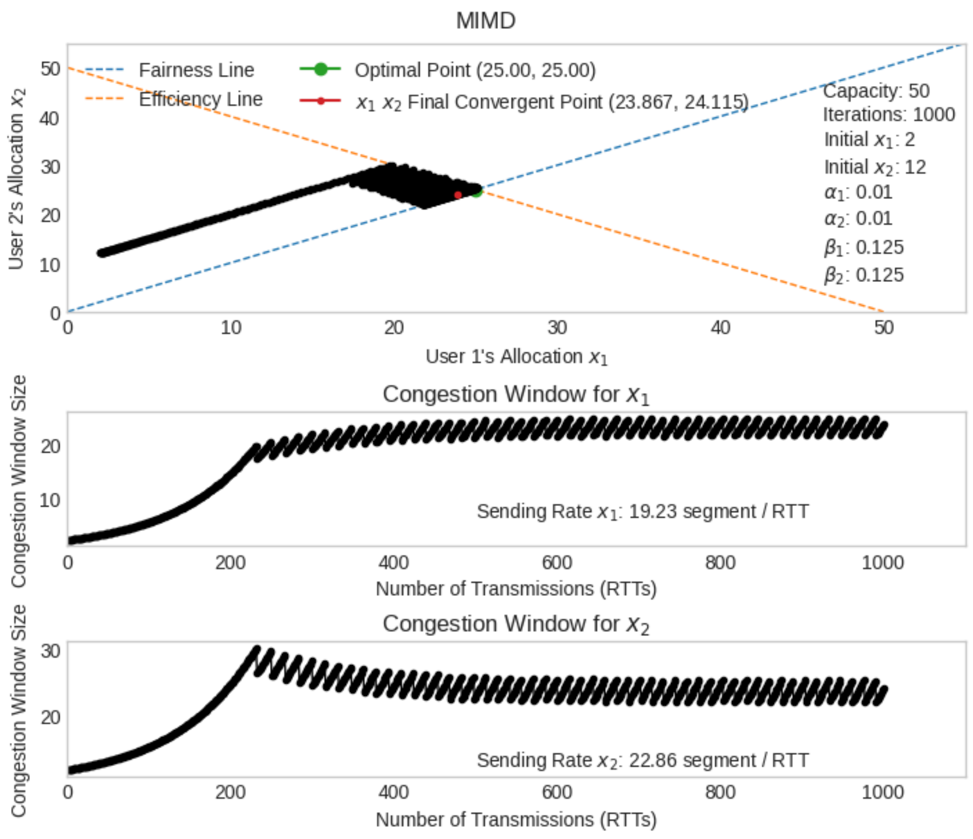
To demonstrate this, we look at a simple case of determining the sending allocations for 2 equal senders using AIMD. We assume a network capacity of 50, and use the default values $\alpha=1, \beta=0.5$. Even in the most extreme case where one sender starts with all the capacity, the rates converge fairly after 183 iterations:



However, we can also see that AIMD does **not** result in an **efficient** sending rate; the average total sending rate is 39.06, far below the maximum of 50. This is because of a small additive rate and high multiplicative rate, resulting in each sender spending most of the time slowly increasing, and thus below the maximum rate.

Multiplicative Increase Multiplicative Decrease (MIMD)/Scalable TCP

A similar algorithm used in TCP congestion control is MIMD, where a **multiplicative increase** is used instead of an additive increase. To demonstrate this, we use the common values of $\alpha=0.01$, $\beta=0.875$:



Even after reaching the limit of 1000 iterations, the rates do not converge, meaning that the algorithm is **not fair**. However, we note that the total average sending rate is better than for AIMD at 42.09, meaning that MIMD may be **more efficient**. Furthermore, in real-world high throughput scenarios with large congestion windows, MIMD can ramp up faster than AIMD [1], providing more consistent throughput.

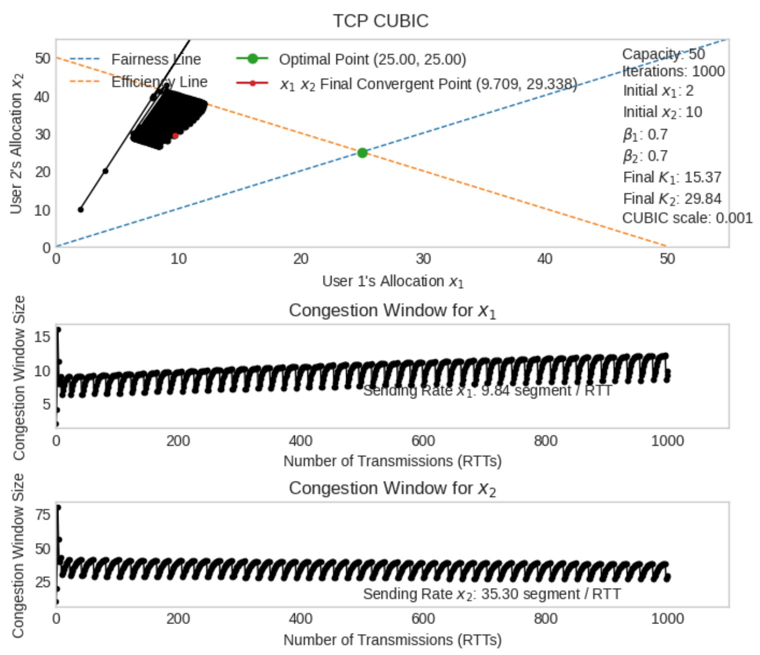
TCP CUBIC

TCP CUBIC is an alternative algorithm which accounts for time taken since the last congestion event, and the previous maximum window size. It uses the following formula [3]:

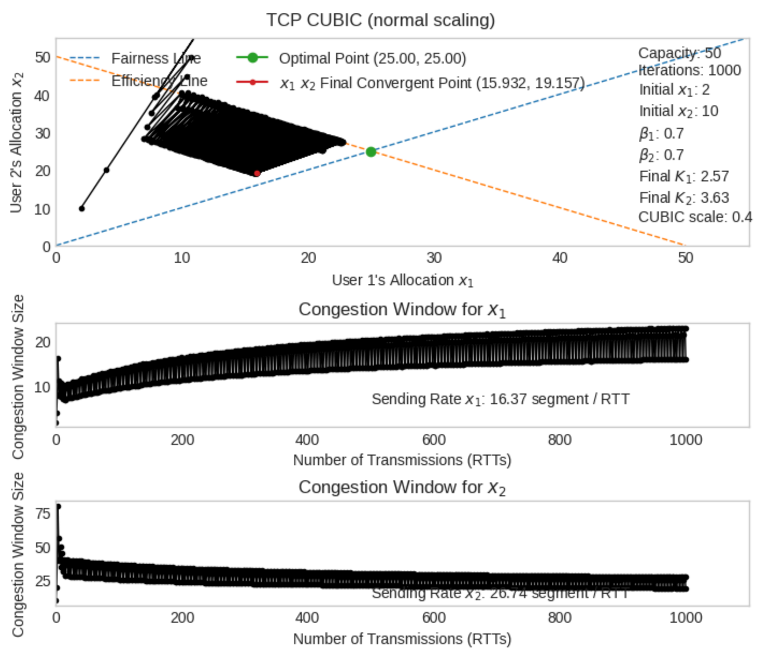
$$cwnd = C(T-K)^3 + w_{max}$$
$$\text{where } K = \sqrt[3]{\frac{w_{max}(1-\beta)}{C}}$$

Using a cubic function means that there is an **inflection point** set to the previous max window size. Before this inflection point, there is fast growth. After it, growth starts slow but quickly increases. The shape of a cubic graph indicates that most time is spent near the inflection point which results in more network stability.

We can use a small constant scaling of 0.001 to show the slowing of growth near the inflection point, and use the standard scaling of 0.4 to demonstrate its more typical behaviour:



Note that demonstrating its bandwidth-searching behaviour is not possible in this graph, as we only set a constant network capacity. However, in a real-world situation with fluctuating network capacity, this algorithm is able to search for additional capacity while remaining mostly stable.



2. TCP For Data Centres

Introduction

The previous algorithms were explored under light and ideal conditions, with constant network capacity and only 2 equal senders with constant sending rates.

In the real world, a network is likely to have more than 2 senders, fluctuating network capacity, and more "bursty" sending behaviour. In a data centre environment, traditional TCP congestion algorithms will face even more challenges, with the need to **maximally exploit high bandwidth** for efficiency, maintain **low latency/congestion**, and even assign sending allocation based on **customer priorities**.

We will look at John Ousterhout's paper, "It's Time to Replace TCP in the Datacenter" [4] to shed more light on the challenges of standard TCP algorithms in data centres. We will also look at adaptations made by modern TCP algorithms to mitigate such challenges, as well as alternatives to TCP itself.

TCP In Data Centres

The following points are brought up by John Ousterhout regarding the issues of TCP specifically in data centre environments:

Stream-based. TCP treats data as a byte stream, while data center applications use discrete messages. This mismatch causes inefficiencies in message processing and load balancing.

Connection-oriented. TCP's reliance on persistent connections introduces overhead with thousands of active connections. Short-lived serverless workloads make handshaking costs harder to amortize.

Bandwidth-sharing. TCP's fairness model slows down all senders in congested networks, reducing efficiency. A run-to-completion approach would be better but isn't natively supported.

Sender-driven congestion control. TCP relies on sender-side control, but modern switches use priority queues. High bandwidth also makes congestion signals outdated before they take effect.

In-order delivery. TCP expects in-order packets, but data centers use packet spraying, leading to unnecessary congestion signals.

Stanford TCP/DCTCP

Stanford's DCTCP modifies TCP to use Explicit Congestion Notification (ECN) [5]. ECN is a mechanism to mark packets when congestion occurs, allowing downstream nodes to gauge the extent of congestion. It is able to deliver similar performance to TCP while using ~90% less buffer space, and performs better under bursty loads [5]. It requires routers/switches to be specially configured for ECN marking, while data centres do have. One downside is that the ECN functionality overrides functionality meant for integrity signalling [6], which may lead to less reliable networking.

TCP ex-Machina: Computer-Generated Congestion Control

Weinstein and Balakrishnan of MIT [7] discuss the use of a specialized program called Remy for constructing end-to-end congestion algorithms. The program takes as input assumptions about the network's operation environment (such as upper/lower speed limits, delays, queue sizes) and the traffic model/behaviour for the network, and outputs offline congestion control algorithms.

The paper finds that networks with broadly similar characteristics as was inputted, found better performance with Remy algorithms, versus typical algorithms such as TCP Cubic/Vegas. However, the paper largely mentions non-data centre environments, such as the web or cellular networks, where latency and efficiency requirements may be lower. Furthermore, it does not take advantage of hardware-tier adaptations which are available in data centres, such as DCTCP.

Homa

Ousterhout's paper highlights a possible alternative to TCP called Homa [5], aimed at mitigating the issues brought up. Homa is an RPC-focused protocol which is based on messages rather than bytestreams. This is more efficient as threads are able to read exactly as much data as they need, which allows for safe socket multiplexing. It also allows for load balancing based on run-to-completion, which can result in more efficient program execution. Homa is also connectionless, mitigating connection overhead, and allows for out-of-order packet flow, which may allow additional optimizations.

However, Homa is a completely different protocol from TCP, making adoption a very difficult task as the majority of networks use TCP.

Conclusion

The most optimal TCP congestion control in data centres is likely to come from a mix of hardware-specific adaptations such as DCTCP, and programmed congestion algorithms such as TCP ex-Machina. Furthermore, as data centres control and have constant insight into their entire network stack, they can also take advantage of dynamic/on-the-fly congestion management, using machine learning/AI based on network performance and behaviours.

Despite this, it is likely that a complete protocol overhaul such as Homa will bring a larger increase in performance for data centres. While adaptation to new protocols is nigh-impossible for most entrenched networks, new data centres can consider using Homa internally.

References:

[1] C. T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," in *PFLDnet 2003*, 2003. [Online]. Available: https://datatag.web.cern.ch/papers/pfldnet2003-ctk.pdf.

[2] S. Floyd, "HighSpeed TCP for large congestion windows," IETF, RFC 3649, Dec. 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3649.txt.

[3] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," 2005. [Online]. Available: https://web.archive.org/web/20150726051512/http://www4.ncsu.edu/~rhee/export/bitcp/cubic-paper.pdf.

[4] S. S. Lall, A. Ousterhout, K. Winstein, and H. Balakrishnan, "TCP ex Machina: Computer-generated congestion control," 2022. [Online]. Available: https://arxiv.org/pdf/2210.00714.

[5] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (DCTCP)," 2010. [Online]. Available: https://web.stanford.edu/~balaji/papers/10datacenter.pdf.

[6] M. Welzl and G. Armitage, "Problem statement for the introduction of congestion control in the RFC series," IETF, RFC 7560, Jul. 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7560.

[7] K. Winstein and H. Balakrishnan, "TCP ex Machina: Computer-generated congestion control," 2013. [Online]. Available: https://web.mit.edu/remy/TCPexMachina.pdf.

Appendix:

Code: Attached as Jupyter Notebook