**A logo with text on it

AI-generated content may be incorrect.**

SC4052 Cloud Computing AY2024/2: Assignment 1

**Lee Alessandro (U2120619H)**

# Literature Survey

Transmission Control Protocol (TCP) is the primary transport protocol for the Internet, but modern datacentre environments surface challenges that TCP struggle to address. Datacentres feature high-bandwidth, low-latency network requirements with differing traffic patterns from traditional networks like the Internet [1], that TCP fail to tackle. I will review the different congestion control algorithms, such as the Additive Increase, Multiplicative Decrease (AIMD) and Multiplicative Increase, Multiplicative Decrease (MIMD) algorithms.

AIMD forms the foundation of TCP’s congestion control, which increases the congestion window linearly by a fixed amount per Round Trip Time (RTT) when no congestion is detected, and decreases multiplicatively when congestion is detected. While AIMD achieves fairness and stability in traditional networks, it is highly limited in datacentre environments since its linear mechanism limits the rate of recovery after congestion events, leading to link underutilisation. It is shown that standard TCP would require unrealistically low packet loss rates of at most once every 2 or 3 hours in order to achieve a high steady throughput [2].

MIMD is an attempt to respond more aggressively to available bandwidth by growing the congestion window exponentially instead of linearly, addressing the constraint of link underutilisation by AIMD. High Speed TCP (HSTCP) [2] and Scalable TCP [3] include parts of multiplicative increase to improve performance in high-bandwidth environments. Pure MIMD approach, however, was not adopted because it can be unstable and unfair in mixed-flow environments [4].

Recent approaches like TCP ex Machina [5] use machine learning techniques to automatically create congestion control algorithms. The approach termed ‘Remy’ produces algorithms that outperform manually crafted protocols. Data Centre TCP (DCTCP) [6] displays a significant advancement in congestion control by leveraging on Explicit Congestion Notification (ECN) signals to make better congestion assessments. This allows it to maintain high throughput and keep queues short. However, it would require ECN support in network switches.

AIMD provides stability and fairness, but is largely limited in high-bandwidth environments and thus incentivising research in MIMD algorithms. The remainder of this report will explore the performance of Hybrid AIMD-MIMD and Hybrid AIMD-MIMD power approaches through simulation. I will show that these hybrid algorithms can better meet the needs of modern datacentre environments that pure AIMD or pure MIMD.

The hybrid AIMD-MIMD approach combines linear and exponential growth based on the current window size. When the window size is small (cwnd < w\_low), it behaves exactly like AIMD, for stability and fairness. When the window size is large (cwnd > w\_high), it behaves exactly like MIMD for responsiveness. Between w\_low and w\_high, it uses linear interpolation between AIMD and MIMD.

# Experimentation

This section presents experimental results comparing four TCP congestion control algorithms: AIMD, MIMD, Hybrid AIMD-MIMD and Hybrid Power. The hybrid approaches were designed to overcome the limitations of AIMD and MIMD in datacentre environments.

## Algorithm implementation

 **AIMD** (α=1.0, β=0.5): Increases window by a constant value per RTT, decreases multiplicatively on congestion.

 **MIMD** (α=0.05, β=0.5): Increases window proportionally to window size, providing faster growth.

 **Hybrid AIMD-MIMD** (α\_add=0.5, α\_mult=0.05, w\_low=10.0, w\_high=30.0, β=0.4): Uses AIMD for small windows, MIMD for large windows, with linear interpolation between thresholds.

 **Hybrid Power** (α\_add=0.5, w\_low=10.0, power=1.5, β=0.4): Scales increase rate using a power function when windows exceed threshold.

## Experimental Setup

**Scenarios**

**Two scenarios** were created to emulate differing environments:

* **Basic:** Homogeneous RTTs, static flows
* **Heterogeneous:** Varied RTTs (rtt\_heterogeneity=0.5)

For each scenario, I evaluated performance with a total of **four differing concurrent flows that are competing for the same bottleneck capacity**; 5, 10, 20, and 50 concurrent flows sharing a 100 Mbps link (simulated with capacity=100), representing different degrees of contention common in datacentres.

**Fixed variables** across scenarios are:

* Iteration = 300: Representing 300 RTT cycles
* Capacity = 100: Representing the bottleneck link’s bandwidth of 100 Mbps

**Jain’s Fairness Index**

Used to measure bandwidth allocation fairness, and it ranges from 1/n (worst case) to 1 (perfect fairness). This provides a quantitative measure of resource allocation equity.

A black math symbols with a white background

AI-generated content may be incorrect.

**Convergence Criteria**

An algorithm was considered **converged** when it maintained fairness ≥0.9 and utilization ≥0.8 for at least 5 consecutive iterations.

**Total runs**

With 4 algorithms and 2 differing scenarios across 4 differing concurrent flows tested, there will be a total of 4\*2\*4 = 32 runs in total.

## Performance Metrics

 **Fairness:** Final Jain’s Fairness Index

 **FairnessUAC:** Average fairness throughout simulation

 **Utilization:** Final link utilization

 **UtilizationUAC:** Average utilization throughout simulation

 **Oscillation:** Fluctuation amplitude in utilization

 **ConvergenceRate:** Proportion of cases achieving convergence

A composite score was calculated as:

**Score = 0.3\*Fairness + 0.3\*Utilization + 0.2\*FairnessUAC + 0.1\*UtilizationUAC + 0.1\*(1-Oscillation)**

## Key Results

Overall performance across 32 runs

 **Hybrid AIMD-MIMD:** Score = 0.8561

 **AIMD:** Score = 0.8341

 **Hybrid Power:** Score = 0.7916

 **MIMD:** Score = 0.6899

Best Algorithm by Scenario (Basic vs Heterogeneous – average score across four concurrent flows)

** Basic:** Hybrid AIMD-MIMD**,** Score = 0.8771

** Heterogeneous:** Hybrid AIMD-MIMD**,** Score = 0.8352

Analysis on the figures

The radar chart analysis (Figure 1) visually confirms that Hybrid AIMD-MIMD achieved the most ideal and balanced performance across all performance metrics in all 32 runs. While Hybrid AIMD-MIMD may fall short to AIMD in terms of fairness, it has outperformed all other algorithms in the 5 other performance metrics. This suggests that the adaptive switching between additive increase and multiplicative increase provides a balance across all performance metrics tested. On the other hand, the Hybrid Power algorithm as compared to AIMD displayed a shortfall in fairness and fairnessUAC, trading it for better wholistic performance across other metrics.

Figure 2 outlines how Hybrid AIMD-MIMD outperformed all the other algorithms in a heterogenous scenario (varying RTTs) with 50 concurrent flows. Hybrid AIMD-MIMD achieved the highest fairness and utilisation, while managing to converge at the earliest iteration out of all other algorithms. Furthermore, the Hybrid Power algorithm was the next best algorithm, outperforming the traditional AIMD and MIMD algorithms.

A diagram of a graph

AI-generated content may be incorrect.

Figure 1: Radar Chart Analysis of Algorithm Performance across Performance Metrics

A screenshot of a graph

AI-generated content may be incorrect.

Figure 2: Analysis of all 4 algorithms in Heterogeneous Scenario with 50 concurrent flows

# Convergence Proof

The Hybrid AIMD-MIMD algorithm can be modeled as a linear dynamical system with the window size updating at each congestion event following the formula:

where is the system matrix defining how the window size changes over time:

A black background with white text

AI-generated content may be incorrect.

has two components, the Diagonal Matrix corresponding to Multiplicative Decrease and the Rank-1 Update corresponding to Additive Increase (AI) and Multiplicative Increase (MI).

According to the Perron-Frobenius Theorem, these matrices have a unique positive eigenvector associated with the dominant eigenvalue. Thus, any positive vector repeatedly multiplied by will eventually align with this eigenvector to ensure convergence.

My simulation confirms the theoretical findings, where after 155 iterations:

* **Jain’s fairness index** gradually increased from 0.64 to 0.99 (near perfect fairness)
* **Window Size converges** from initial [2.0, 8.0, 20.0] to [5.99, 5.99, 6.00]
* **Eigenvector (normalized final window size)** reaches [0.33, 0.33, 0.33] (closely matching the theoretical Perron-Frobenius eigenvector)
* **L2 Distance (Euclidean Distance)** which measures how close the window size vector

is to the Perron-Frobenius eigenvector of the final system matrix decreases over iterations, meaning that the window sizes are converging towards the theoretical equilibrium given by the eigenvector.

A group of graphs showing different types of events

AI-generated content may be incorrect.

Figure 3: Graphs depicting convergence after 155 iterations

# References

[1] J. Ousterhout, “It’s Time to Replace TCP in the Datacenter,” arXiv:2210.00714 [cs.NI], Jan. 2023. [Online]. Available: <https://arxiv.org/pdf/2210.00714>

[2] S. Floyd, “HighSpeed TCP for Large Congestion Windows,” Internet draft: draft-floyd-tcp-highspeed-02, Work in progress, Feb. 2003. [Online]. Available: <https://www.evl.uic.edu/eric/atp/HighSpeedTCP.pdf>

[3] T. Kelly, “Scalable TCP: Improving performance in highspeed wide area networks,” in Proceedings of the PFLDnet 2003 Workshop, Feb. 2003. [Online]. Available: <https://www.cs.unc.edu/~jasleen/Courses/Fall14-631/papers/pfldnet2003-ctk.pdf>

[4] E. Altman, K. E. Avrachenkov, and B. J. Prabhu, “Fairness in MIMD congestion control algorithms,” in Proc. IEEE 24th Annu. Joint Conf. IEEE Computer and Communications Societies (INFOCOM), vol. 2, 2005, pp. 1350–1361. doi: 10.1109/INFCOM.2005.1498360.

[5] K. Winstein and H. Balakrishnan, “TCP ex Machina: Computer-generated congestion control,” in Proceedings of the ACM SIGCOMM 2013 Conference, Hong Kong, China, Aug. 2013, pp. 123–134. doi: 10.1145/2486001.2486020

**[6]** M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, “Data Center TCP (DCTCP),” Proceedings of the ACM SIGCOMM 2010 Conference, New Delhi, India, Aug. 2010, pp. 63–74.

# Appendix

Code: <https://github.com/Alexlaaa/Cloud-Computing/blob/main/SC4052_Assignment%201_LeeAlessandro_U2120619H.ipynb>