Dynamic Time Based Congestion Control:

SC4052 Cloud Computing

AY24/25 Assignment 1

Lee Bo Hua (U2122595D)

Github Link:

https://github.com/bohua12/Dynamic-Time-Based-congestion-Control

Table of Contents

1 Literature Review on TCP for Data Centers

TCP is everywhere – from devices as small as our mobile phones, to data centers spanning several hectares. This protocol has been in place since the 1970s and the fundamentals have remain largely unchanged. The needs and demands of a modern networking environment, however, have evolved drastically. TCP have been praised for its reliability and congestion control mechanisms, and stream-oriented protocol. However, these features prove to be a significant overhead in environments that demand low latency and high throughout [1].

One such environments are datacenters. Several experts have argued against the use of TCP due to its inherent characteristic such as stream orientation, connection orientation, bandwidth sharing strategies and it's congestion control mechanism. These characteristics leads to are acceptable in office or personal uses but lead to significant bottlenecks in datacenters where high speed and low latency are of paramount importance. [2] These experts advocate for a change in data center transport protocol. However, this faces significant challenges due to large scale integration and substantial infrastructure changes required. [3]

However, these limitations should not deter efforts on improving on the latency on throughout of data centers! Traditionally, TCP uses the Additional Increase Multiplicative Decrease (AIMD) for congestion control. AIMD gradually increases congestion window by a fixed amount when no congestion is detected, allowing connections to utilize more bandwidth gradually. Conversely when congestion is detected, congestion window is scaled down by a multiplied factor.

However, this approach is inherently naïve, as it assumes that network traffic behaves consistently all the time. AIMD Fails to consider that traffic behavior varies with traffic spikes, predictable variations of traffic, and time of day. I feel that a more intelligent congestion control algorithm can be created to better handle congestions by taking into account these factors.

I propose an approach called **Dynamic Time Based congestion Control** (DTBC) that builds upon AIMD. However instead of a consist scale, the scale in which congestion window is changes is dynamically adjusted by the time of the day

2 Experiment

I will be implementing both AIMD and DTBC twice – one for in depth experimentation between 2 users and one for experimentation for scale, with up to 1000 users

2.1 Initialization

2.1.1 Network Traffic

I had the grand ambition of looking for data center datasets that models traffic over time. However, that task proved to be extremely difficult due to the lack of datasets that fits my needs. For those that I managed to find [5], these datasets are way too complex for my experiment. Hence, I had to simulate traffic in 24 hours with reasonable justification.

Time Period	Traffic Load	Justification
0000-0600	0.4-0.5	Lowest activity as most of the population are sleeping
0700-0900	0.6-0.9	Rising business activities
1000-1100; 1800	1.0	Moderate business activities
1200-1700	1.1-1.3	Peak business activity
1800-2300	0.5-1.0	Declining business activity

2.1.2 User Traffic

For similar reasons as Network Traffic, I had to simulate user traffic. I generated dynamic traffic loads for multiple users over 100 hours. For each hour, there are time-based variations based on the below factors. For the 2nd part of the experiment, I have generated 2,6,10,100,1000 users

1) User-Specific demand level
    a. Assuming '**C**' units of network traffic are assigned to 2 users
    b. Each user is assigned a unique average demand, ranging from 50-80% of **C**
2) Hourly traffic mapping
    a. **Hourly Load Factor** of traffic demand is derived based on the time of the day, scaled to the traffic load as
    b. A **variation** of +- 10% is introduced to simulate traffic fluctuations

With the formula as shown below

$$D_{\{u,h\}} = (C \times r_u \times L_h) \times (1 + V)$$

$$D_{\{u,h\}} = Final\ demand\ for\ user\ u\ at\ hour\ h$$
$$C = total\ network\ capaacity$$
$$R_{\{u\}} = User\ specific\ demand\ ratio\ (0.5 - 0.8)$$
$$L_{\{h\}} = Hourly\ Load\ factor$$
$$V = Variation\ factor, where\ V \sim U(-0.1,0.1)$$

2.2 Implementation of AIMD

My implementation of AIMD is inspired by Prof Tan Chee Wei's Matlab code shown during tutorial. The important thing to take note of is the α and β values.

Variable	Initial Value	Formula for change	Remarks
α	1	NIL (Fixed value)	Increase congestion window size by α during the Additive Increase (AI) phase
β	0.5	NIL (Fixed value)	Scale congestion window size by factor of β during the Multiplative Decrease phase

2.3 Implementation of DTBC

Let's introduce a new variable here, **σ** which represents the current Demand Factor as determined by the traffic of the time of the day

Variable	Initial Value	Formula for change	Remarks
α	1	**α** / **σ**	Increase congestion window size by α during the AI Phase
β	0.5	1 – ((1- **β) * σ)**	Scale congestion window size by factor of β during the MD phase
σ	[0.4,1.3]	-	Changed based on time of the day

3 Discussion of results

3.1 In-depth comparison between 2 users

The results from the low traffic simulation between AIMD (Figure 1) and DTBC (Figure 2) shows a **clear advantage for DTBC**. I have chosen to show this scenario instead of higher traffics, as it most clearly illustrates DTBC's **superior utilization efficiency**.

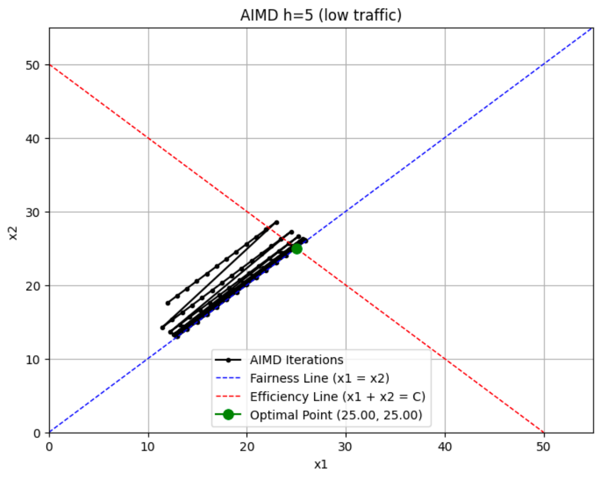

Figure 1: Simulation for AIMD

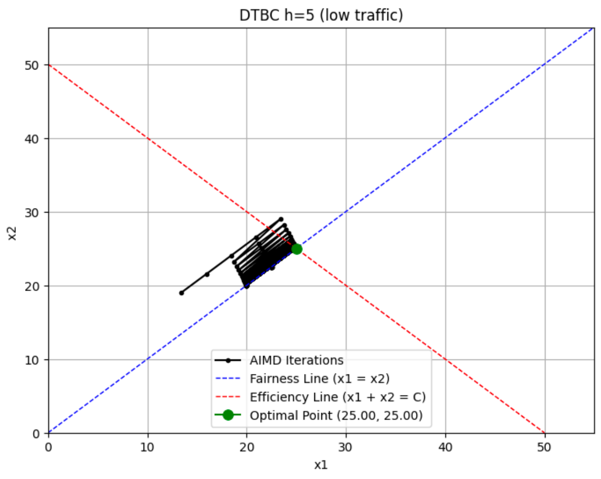

Figure 2: Simulation for DTBC

3.2 Comparison between multiple groups of users

We have used Utilization Index, Jain's Fairness Index and Convergence Iteration as key metrics to evaluate performance of AIMD and DTBC. The results from scaling number of users **confirm that DTBC outperforms AIMD in both utilization and fairness**. Across all number of users, DTBC **consistently achieve higher utilization and fairness** as seen in Figure 3 and 4. This is demonstrates DTBC's effectiveness in adjusting window sizes based on time of the day

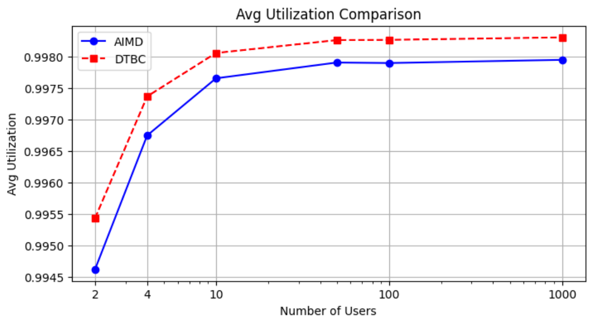

Figure 3: Utilization Index

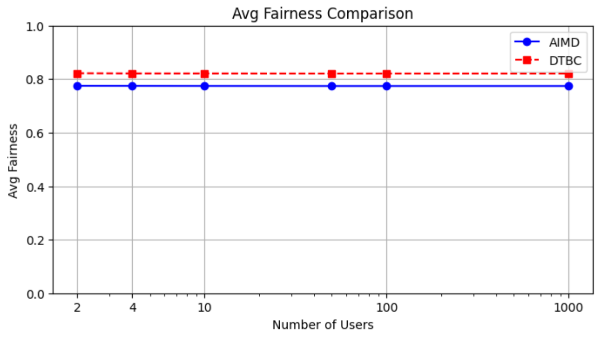

Figure 4: Jain's fairness Index

However, the **convergence index for DTBC is significantly higher than AIMD**. This indicates that DTBC takes longer to stabilise than AIMD. (Refer to appendix figure 5)

4 Convergence Proof for DTBC Algorithm

$$W(k+1) = A(k) * W(K)$$

Where A(K) is a system matrix that determines how the window size evolves over time. It consists of a diagonal matrix D(K) representing MD, and a rank-1 update, R(K) representing AI. Thus, the update rule can be written as

$$w(k+1) = \left(D(k) + R(k)\right) * w(k)$$

According to Perron-Forbenius Theorem, these matrices have a unique dominant eigenvector associated with the dominant eigenvalue that ensures eventual convergence. Since AIMD is already proved to be convergent, DTBC, with a dynamically adjusted A(K) and D(K), **can still converge.**

5 Conclusion

This study successfully introduced and compared AIMD and DTBC based on network and user traffic simulations. It clearly demonstrated DTBC's superior performance as it's utilization and fairness across all user groups is consistently greater. This proves that the DTBC algorithm, although simple, is **novel** and can effectively dynamically adjust congestion window based on the time of the day. DTBC is also **scalable** due to performance improvement over AIMD being consistent and not declining.

There are, however, slightly worst convergence performance for AIMD which I feel is less of an issue, as long as there is some form of eventual convergence.

Beyond this, there are opportunities for further improvement. For example, by using sequential recommendation models to predict and model real-time network conditions.

This approach is meant to be used in complement to other Congestion Control algorithms as it only considers one time dependent aspect. Although DTBC proved to have significantly better utility and fairness than AIMD, it is best deployed alongside other algorithms that consider other critical factors, like other time dependent aspect, or even ML Techniques that can further mine patterns in traffic time variations [4]

6 References

[1] Khalil, Mohsin, and Farid Ullah Khan. “Exploration of TCP Parameters for Enhanced Performance in a Datacenter Environment.” *ArXiv (Cornell University)*, 1 Nov. 2018, pp. 65–69, https://doi.org/10.1109/iceese.2018.8703553. Accessed 27 Feb. 2025.

[2] Ousterhout, John. *It’s Time to Replace TCP in the Datacenter*. 19 Jan. 2023, arxiv.org/abs/2210.00714, https://doi.org/10.48550/arXiv.2210.00714. Accessed 28 Feb. 2025.

[3] Team, SimBricks. “Challenges in Evaluating New Data Center Network Protocols - SimBricks.” *Simbricks.io*, 2024, www.simbricks.io/blog/homa-evaluation-challenges.html. Accessed 28 Feb. 2025.

[4] Winstein, Keith, and Hari Balakrishnan. “TCP Ex Machina.” *ACM Special Interest Group on Data Communication*, 27 Aug. 2013, https://doi.org/10.1145/2486001.2486020.

[5] Theophilus, Benson. “Data Set for IMC 2010 Data Center Measurement.” *Wisc.edu*, 2025, pages.cs.wisc.edu/~tbenson/IMC10_Data.html

7 Appendix

Do refer to my Jupyter notebook titled DTBC_2users.ipynb or DTBC_scaled.ipynb either in the submission or GitHub link for the code and other graphs, as there are too many to put in here.

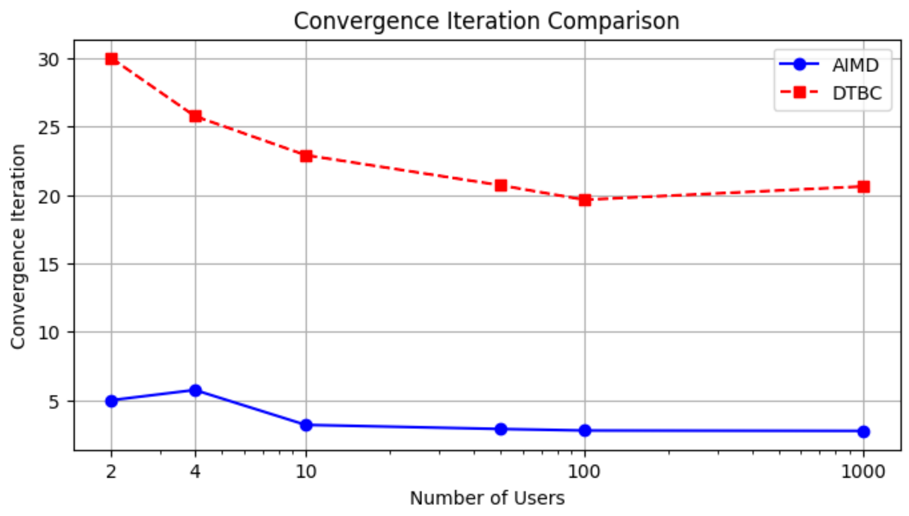

Figure 5: Convergence Iteration Comparison