CS 6043 – Computer Networking

A Simulation-based Study of Additive increase/multiplicative decrease (AIMD)

Submitted by:
Gautham Pothana
Jithesh Kumar Kasi

Introduction

- Proposed by Chiu and Jain.
- Congestion avoidance algorithm.
- When network capacity is not achieved, the AIMD method employs a variable to raise each user's allotment. When the network capacity becomes overburdened, a variable seeks to reduce each user's allotment to less than that capacity.
- Formula

$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } y(t) = 0, \text{ Increase} \\ b_D * x_i(t) & \text{if } y(t) = 1, \text{ Decrease} \end{cases}$$

Where,

t = current time value

$x_i(t)$ = i-th user's transmission rate at time t

a_I = additive increase parameter

bD = multiplicative decrease parameter

Introduction (cont.)

- Characteristics of the algorithm
  - **Efficiency**: For loads with different users, ideally the load should stay on the efficiency line (grid capacity) of the graph. All elements below the line are underloaded and all elements above are overloaded.

$$X(t) = \sum x_i(t)$$

$$X(t) > X_{goal} = overload$$

$$X(t) < X_{goal} = underload$$

where:

$x_i(t)$ = the i-th user's transmission rate
$X(t)$ = the sum of every user's transmission rates
X_{goal} = desired load level

Introduction (cont.)

- Characteristics of the algorithm
  - **Fairness**: In an ideal world, different users should contribute the same amount to the total network load, or "same bottleneck".

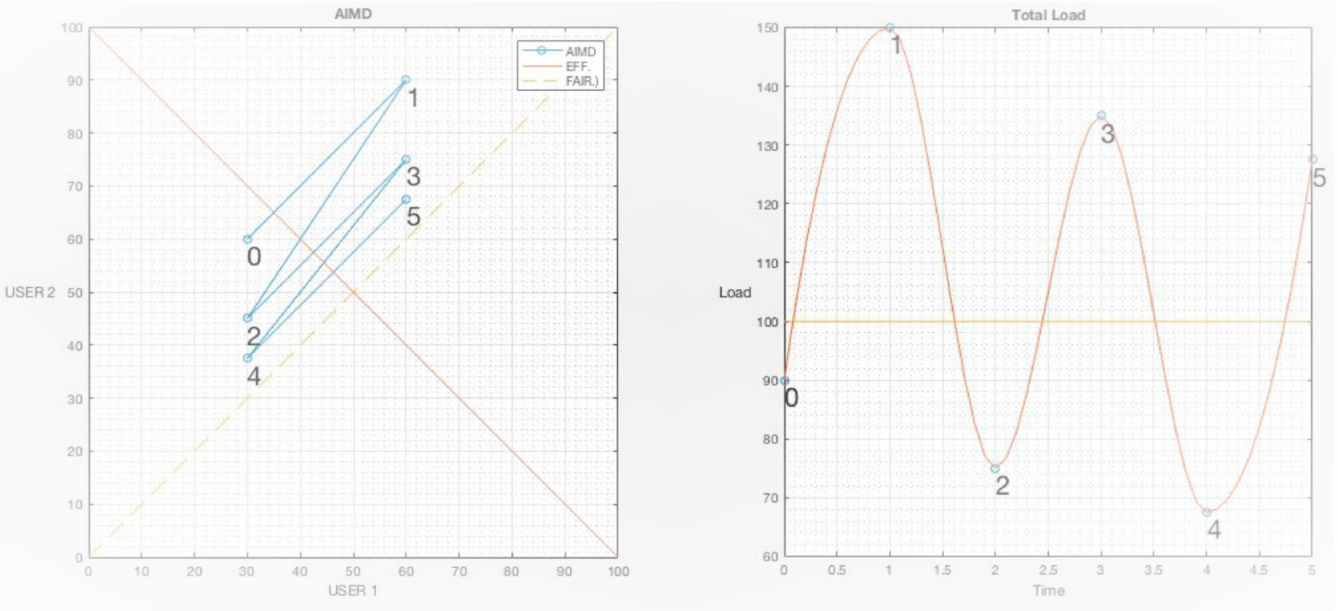
$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)}$$

where:

$$x_i = \text{the } i\text{-th user's transmission rate}$$
$$n = \text{the number of users on the network}$$

Detailed Description

- The aim is to get our load values as near to ideal efficiency and fairness as feasible [i.e. an allocation of (50, 50) in this case].



Detailed Description(contd.)

- **Load Distribution:** A prerequisite is that the centralized system must be fully aware of the load in the system.
- **Convergence:** Whatever your initial starting point for load balancing, you need a congestion avoidance method that helps you converge on the efficiency-fairness line of the graph.
- **Responsiveness:** It is the amount of time it takes for load values to arrive at the "desired" location.

$$t_e = \begin{cases} \frac{log\left(\frac{an+(b-1)X_{goal}}{an+(b-1)X(0)}\right)}{log(b)} & b > 0 \\ \frac{X_{goal}-X(0)}{an} & b = 0 \end{cases}$$

where:

$$\begin{aligned} a &= \text{the additive increase parameter} \\ n &= \text{the number of users on the network} \\ b &= \text{the multiplicative decrease parameter} \\ X_{goal} &= \text{the total network capacity} \\ X(0) &= \text{the sum of every user's original transmission rate} \end{aligned}$$

Detailed Description(contd.)

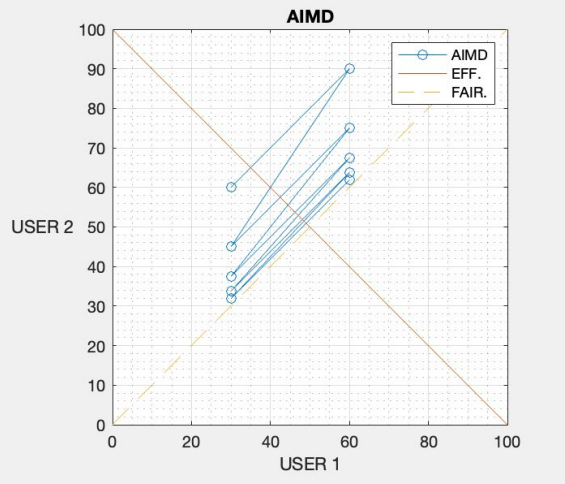
- **Smoothness:** The degree to which the load value oscillates around the efficiency line as it approaches the "target" position.

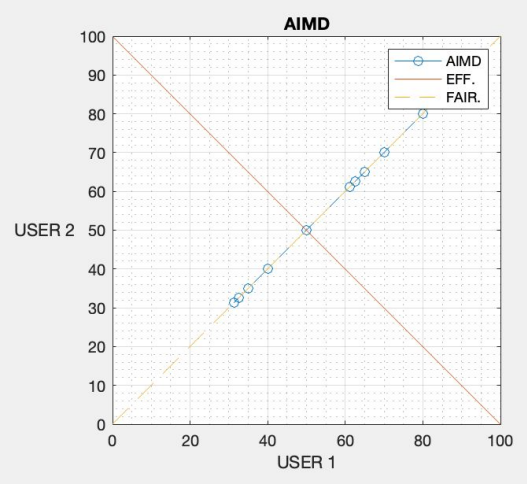
$$s_e = |an + (b-1)X_{goal}|$$

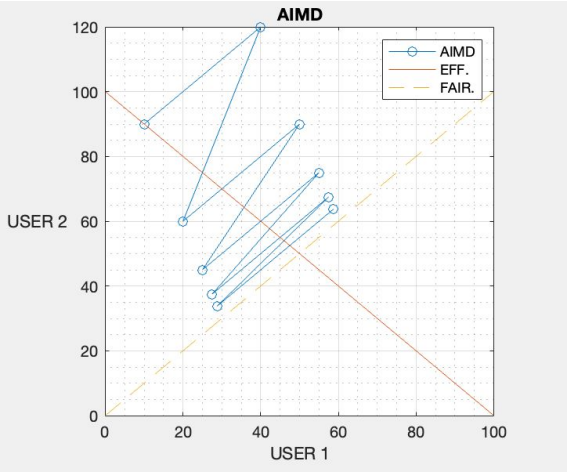
where:

$$\begin{aligned} a &= \text{the additive increase parameter} \\ n &= \text{the number of users on the network} \\ b &= \text{the multiplicative decrease parameter} \\ X_{goal} &= \text{the total network capacity} \end{aligned}$$

Detailed Description(contd.)



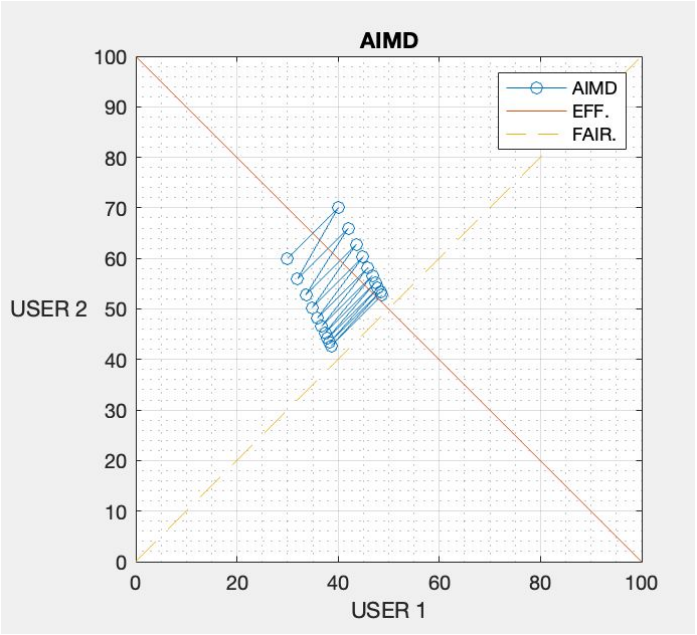




(a) (30, 60) - A little Fair (b) (50, 50) - Perfectly Fair & efficient (c) (10, 90) - Very Unfair

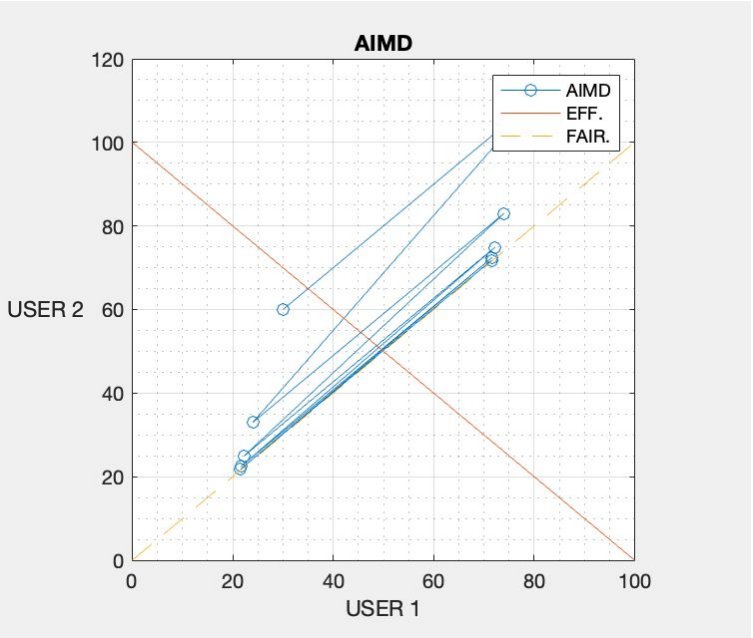
Results/ Observations/ Insights

- **Better Smoothness:** Smaller a value and higher b value gives low oscillations which means better smoothness for the network users. This translates to larger convergence time as it takes more iterations of the code to converge.



Results/ Observations/ Insights

- **Better Convergence:** Larger a value and lower b value gives higher oscillations of user's network usage. This gives less smoothness and faster convergence time and less iterations of code.



Summary

- Effective congestion control algorithm.
- Uses: TCP, STCP, OSI Transport class.
- An optimal trade-off between smoothness and convergence occurred when both a and b were decreased.
- The Responsiveness formula sometimes resulted in negative or non-existent values.
- CUBIC algorithm.

References

[1] https://www.geeksforgeeks.org/aimd-algorithm/

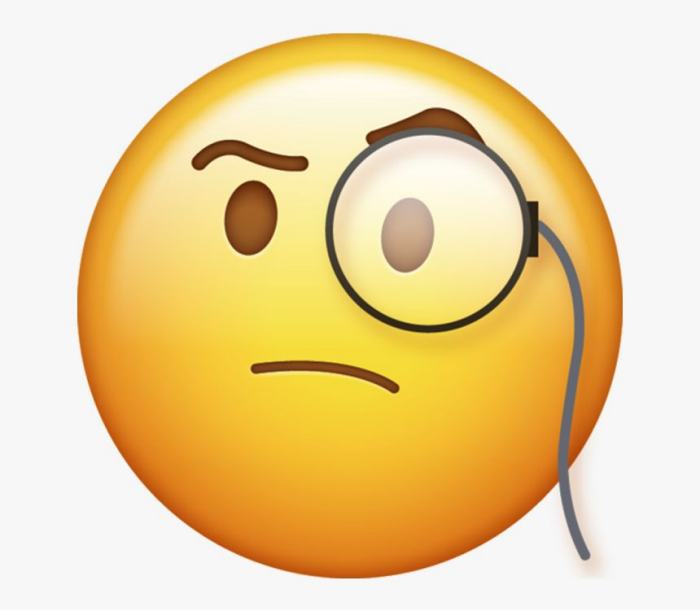
[2] https://www.cs.princeton.edu/courses/archive/fall16/cos561/papers/Cubic08.pdf

[3] https://www.cse.wustl.edu/~jain/papers/ftp/cong_av.pdf

[4] https://en.wikipedia.org/wiki/Additive_increase/multiplicative_decrease

[5] https://www.researchgate.net/publication/255591510_The_new_AIMD_congestion_control_algorithm

ANY QUESTIONS?




Thank You!