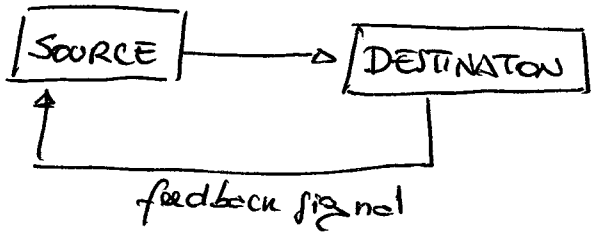
TCP

TCP provides end-to-end flow control in The Internet. It is based on the general philosophy of having Intelligent edge - Dumb Core namely, control Tasks should not be handled by routers if they can be handled by end-to-end devices.

It is a CONTROL protocol using the general idea of feedback:



The feedback signal is sent in terms of packet delay & packet losses experienced by end-hosts. Based on this information the source adjusts the rate to maximize flow while guaranteeing some fairness among different users.

Transmission rate is adjusted dynamically varying the widow size of packets not yet acknowledged.

Large sliding window => aggressive, have more packets in queues
Small sliding window => conservative, have less packets in queues

SIZE of sliding window indicates # of "active" packets still without ack from receiver. Determining the size of this window is The key for optimal operation of the network to guarantee maximum flow & fairness.

AIMD protocol: packet losses & delay are used to detect congestion.
Rate increases linearly until congestion is detected.
Once detected, transmitter decreases rate by a multiplicative factor.

EXAMPLE: increase window by fix amount every round-trip time
if congestion occurs decrease window by cutting it by 1/2.

time sbt = packet received time + ack time = round trip time

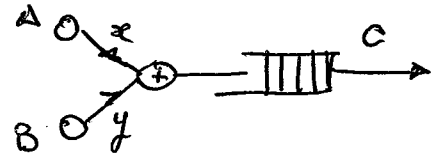
Determine size of window:

$$w(t+1) = \begin{cases} w(t) + a & \text{if not congestion} \\ w(t) \cdot b & \text{if congestion} \end{cases} \qquad a>0 \quad b<1$$

Notice the AIMD requires a congestion mechanism notification (from packet losses)

AIMD converges to a fair solution for N users sharing one queue.

Consider Two sources sharing one link of capacity C.

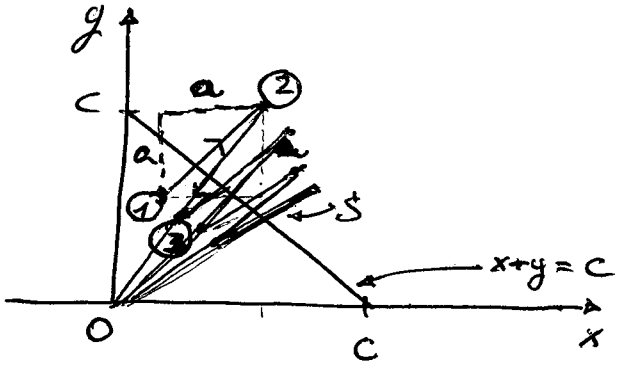


when the sum of rates $x+y > C$ the queue arrival rate > departure rate and eventually queue overflows $\Rightarrow$ congestion.

$x \to x/2$
$y \to y/2$] users cut the rate by 1/2

Eventually they approach the set S where the rates are equal $y = x$.

Example:



Increase rate by a, move 1 → 2
Multiply rate by b, move 2 → 3
eventually converge on the line

$$\begin{cases} y = x \\ y \cdot b = x \cdot b \\ y + a = x + a \end{cases}$$

This works also for more than 2 flows.

Sources do not need to know # flows

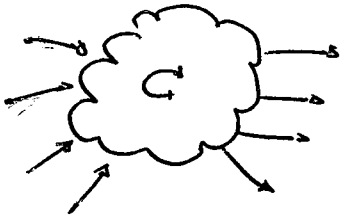
It adjusts rate around value C in a fair way.

Make sure you understand dynamics explained in this note

Note: when move (1) → (2)
you move along a (3) → (4)
line of 45° angle (5) → (6) ...
with x axis

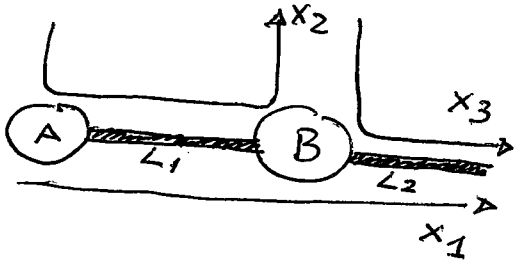
when move (2) → (3)
you move along (4) → (5)
line passing (5) → (6) ...
through origin

AIMD looks like a reasonable thing to do if we think at the whole internet as a single resource shared by many users.



In practice, however, we have a system of many queues and many source-destination pairs. How can we analyze performance in this case? How can we balance between throughput and rate?

Let's start with a simple example:
2 Nodes (A, B), 3 Flows ($x_1 x_2 x_3$), 2 links (L_1, L_2)



Link constraints

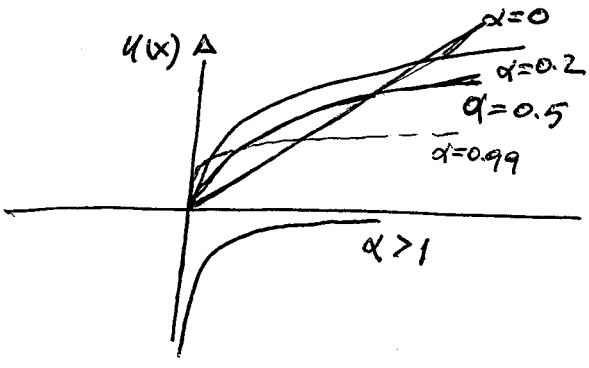
$$\begin{cases} x_1 + x_2 \le 1 & \text{on link } L_1 \\ x_1 + x_3 \le 1 & \text{on link } L_2 \end{cases}$$

Examples of feasible rates:
$(0,1,1) \Rightarrow \sum \text{rates} = 2$ but unfair
$(1/2, 1/2, 1/2) \Rightarrow \sum \text{rates} = 1.5$ but fair

To balance between throughput & fairness we pick a utility function that we try to maximize. The form of the utility function is critical to control both performance & fairness.

For example let: $\boxed{u(x) = \frac{1}{1-\alpha} x^{(1-\alpha)}}$ $\alpha \neq 1$

This function is increasing & concave exponent $\boxed{\alpha}$ controls concavity imposing a "law of diminishing returns" For $\alpha = 0$ is a line (it is always better to increase rate)



$\alpha > 0$ is concave (initially a lot of gain to increase rate, but less and less gain for large x)

We now want to maximize the sum-utilities for all flows rather than the sum-flows

the choice of α will determine whether the optimal solution will be biased towards performance ($\alpha \to 0$) or fairness ($\alpha \to \infty$)

OPTIMIZATION PROBLEM

$$\max \quad u(x_1) + u(x_2) + u(x_3) = \frac{1}{1-\alpha}\left(x_1^{1-\alpha} + x_2^{1-\alpha} + x_3^{1-\alpha}\right)$$

$$\text{s.t} \begin{cases} x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \end{cases}$$

Since $u(x)$ is increasing we can saturate the constraints and let

$$\left.\begin{aligned} x_1 + x_2 &= 1 \\ x_1 + x_3 &= 1 \end{aligned}\right\} \quad \begin{aligned} x_1 &= 1 - x_2 \\ x_3 &= -x_2 \end{aligned}$$

It follows that we want to maximize:

$$\frac{1}{1-\alpha}\left[(1-x_2)^{1-\alpha} + x_2^{1-\alpha} + (1-x_2)^{1-\alpha}\right]$$

$$= \frac{1}{1-\alpha}\left[2(1-x_2)^{1-\alpha} + x_2^{1-\alpha}\right]$$

Taking derivative $= 0$, we have:

$$\frac{1}{1-\alpha}\left[(1-\alpha)x^{-\alpha} + 2(1-\alpha)(1-x)^{-\alpha}(-1)\right] = 0$$

$$x^{-\alpha} - 2(1-x)^{-\alpha} = 0$$

$$x^{-\alpha} = 2(1-x)^{-\alpha}$$

$$x = 2^{-1/\alpha}(1-x)$$

$$x = 2^{-1/\alpha} - 2^{-1/\alpha}x$$

$$x(1+2^{-1/\alpha}) = 2^{-1/\alpha}$$

$$x = \frac{2^{-1/\alpha}}{1+2^{-1/\alpha}} = \frac{2^{-1/\alpha}}{1+\frac{1}{2^{1/\alpha}}} = \frac{1}{2^{1/\alpha}+1}$$

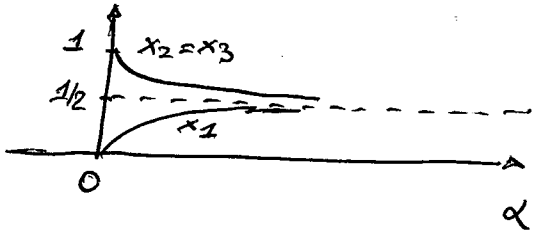
The solution is:

$$\begin{cases} x_1 = 1 - x_2 \\ x_3 = x_2 \\ x_2 = \frac{1}{1+2^{1/\alpha}} \end{cases}$$

for $\alpha = \infty \Rightarrow \begin{cases} (x_1, x_2, x_3) = (1/2, 1/2, 1/2) \\ u(x_1)+u(x_2)+u(x_3) = 1.5 \end{cases}$

$\alpha = 0 \Rightarrow \begin{cases} u(x_1)+u(x_2)+u(x_3) = 2 \\ (x_1, x_2, x_3) = (0, 1, 1) \end{cases}$

By varying α, we go from maximum throughput ($\alpha = 0$) rate to completely fair rate ($\alpha = \infty$)



$$\Leftarrow \begin{cases} x_1 = 1 - \frac{1}{2^{1/\alpha}+1} \\ x_2 = x_3 = \frac{1}{1+2^{1/\alpha}} \end{cases}$$

By varying α we go from maximizing the sum of the rates $x_1+x_2+x_3 = 1+1+0 = 2$ to maximizing the minimum rate $x_1 = 1/2$, $x_2 = x_3 = 1/2$

In general: we can find the optimal link rates by solving an optimization problem subject to link budget constraints. But this requires knowing the whole network and solving the problem off-line. Can we find an analogous distributed solution (maybe approximate) that finds a similar optimum operating point for the network?