Congestion Control

Definition. Internet is a shared network between various heterogeneous users with distinct requirements from the Internet. The goal of congestion control protocols is to provide means to share this resource (Internet) in a "reasonable" sense via distributed implementation.

Toy question.

![Diagram](sender to network to receiver)

Sender wishes to send data to its receiver through a network whose capacity is unknown. Only feedback sender can possibly have is whether the sent data has reached receiver or not. If sender sends data at a low rate, resources may not be well-utilized. Else, there may be a lot of congestion. The sender needs to decide how to find appropriate rate. On the other hand, network needs to decide how to allocate resources among various users accessing it.
We shall address this question as follows. First we will discuss the (idealized version) of the existing protocol/algorithm called TCP. We shall study its performance by means of simple "fluid model" of it. We will find that the algorithm, somewhat surprisingly, achieves excellent performance in terms of it sharing resources. Specifically, in some sense the allocation obtained will be

* efficient: most resource will be utilized
* fair: everyone gets "reasonable" share of the resource.

This fluid model analysis (for single link, toy model scenario) will motivate us to investigate this further. We shall go through a systematic study of this question in the context of resource allocation and understand, somewhat co-incidental, relation between TCP algorithm and the "primal" algorithm of the resource allocation problem. We shall end by investigating relation between efficiency of such algorithm by means of "fluid model".
Congestion Control Mechanism. There are two building blocks of the congestion control.

1. Drop-detection. It allows sender to recognize congestion in the network.

2. Rate-control. It allows sender to change rate so as to "match" available bandwidth.

These combined protocol is called Transmission Control Protocol (TCP).

History. Internet came out of a DARPA funded ARPANET project. The aim of the project was to allow heterogeneous entities to operate over a network with little or no coordination between each other.

Clearly there were congestion control protocols in those networks. The older versions of protocol had issues: Specifically, they led network to reach "congestion collapse" often. A popular documented example: 36 Kbps \rightarrow 40 bytes speed between UCB \& Lawrence Berkeley lab in Oct. 1986!
Primary reason: lack of a mechanism that allowed network to "get out of" congestion.

Current TCP, which is a variant of protocol proposed by Van Jacobson (1988), overcomes this by means of "packet-conservation" law — unless a sent packet is acknowledged, do not send new packet.

Sender Algorithm in TCP.

Parameters:
- $STHRESH$: threshold for slow-start
- $W_{max}$: max. window-size
- $CW$: current window-size

Algorithm operates in two phases: slow-start followed by congestion avoidance. The goal of slow-start is to quickly recognize the "available" bandwidth and then the goal of congestion avoidance is to stay around it unless network congestion changes.
Rate-control Mechanism.

Slow-start:

- set \( STHRESH = \frac{W_{max}}{2} \).
- Start with \( CW = 1 \), set packet-counter \( P = 0 \).
- Each time send packets between \( P, P + CW \).
- When acknowledgement for \( P + 1 \) is received, increase \( CW \) to \( CW + 1 \), till \( CW < STHRESH \) or packet-loss detected.
- If packet-loss, set \( STHRESH \) to \( STHRESH / 2 \); \( CW \) to \( CW / 2 \).

- Continue till \( CW < STHRESH \) in above manner. After that, enter congestion avoidance phase.

Congestion avoidance:

- For every "ack" increase \( CW \) to \( CW + \frac{1}{CW} \).

- For every packet-loss detected,
  set \( STHRESH \) to \( CW / 2 \) and \( CW \) to \( 1 \).
  Go to slow-start.
Drop detection mechanism.

- Essentially, a packet is dropped when an acknowledgment is not received within a certain "time-out".

Receiver's algorithm.

- Send acknowledgements on receiving packet. That is, the identifier or number of the packet that the sender is expecting next.

The above algorithm implies that one way to detect packet drop is if multiple acks from receiver for the same packet or no ack within a certain time-out.
Fluid-model for TCP. We consider deterministic dynamic evolution equations for rates under TCP. Let's consider only congestion avoidance phase at it is the stage that primarily governs the throughput.

Let $W_i(t)$ be window size of user $i$ at time $t$. Let $T_i$ be Round-trip time for user $i$. Then, rate

$$x_i(t) \approx \frac{W_i(t)}{T_i}.$$  

Let $q_i(t)$ be fraction of packets dropped along path of $i$. Then the drop-rate is

$$\approx x_i(t-T_i) q_i(t).$$

And, acknowledgement rate is

$$\approx x_i(t-T_i) (1-q_i(t)).$$
The $W_i(t)$ changes as follows:

$$\frac{dW_i(t)}{dt} = (1 - q_i(t)) \frac{1}{W_i(t)} \cdot x_i(t-T_i)$$

$$- q_i(t) \cdot \beta W_i(t) \cdot x_i(t-T_i).$$

Since, $x_i(t) = \frac{W_i(t)}{T_i}$, then

$$\frac{dx_i(t)}{dt} = \frac{1-q_i(t)}{T_i} \cdot x_i(t-T_i) \cdot \frac{x_i(t)}{x_i(t)}$$

$$- q_i(t) \cdot \beta T_i \cdot x_i(t) \cdot x_i(t-T_i).$$

Setting $x_i(t-T_i) \approx x_i(t)$,

$$\frac{dx_i(t)}{dt} = \frac{1-q_i(t)}{T_i} - \beta T_i \cdot q_i(t) \cdot x_i^2(t)$$

"Usually," $q_i(t)$ is small or else system will be driven to that regime. Therefore, let $-q_i(t) \approx 1$. 
That is:
\[
\frac{dx_i(t)}{dt} = \frac{1}{T_i} - \beta T_i q_i(t) x_i^2(t)
\]

and in the "equilibrium" or fixed point scenario we have
\[
x_i^* = \frac{1}{T_i} \cdot \sqrt{\frac{1}{\beta q_i}}
\]

**Question:** how good (or bad) is such fixed point?

Let us consider a simple scenario: single link.

Let there be $n$ flows sharing a one link of capacity $C$.
Let its buffersize be $B$. As before $T_i$ be round trip time for flow $i$ and $\beta$ be as before. Since all flows pass through
only one link, they all have same drop probability. (In the fixed point). Therefore, from above in the fixed point:

$$x_i = \frac{1}{T_i} \sqrt{\frac{1}{\beta q}} \quad \text{for all } i.$$ 

Let $U = \sum x_i$: total utilized capacity of link.

$U$ should be at most $C$, but question is if it's too small compared to $C$: from above,

$$U = \sum x_i = \frac{1}{\sqrt{\beta q}} \cdot \left( \sum \frac{T_i}{T_i} \right)$$

If we imagine the single link as a queue, then the drop probability can be approximated (using $M/M/1$ formula) as:

$$q_i = \left( \frac{U_i}{C} \right)^B \equiv \sigma_i^B$$
Therefore, we have

\[ U = \left( \sum_{i} \frac{1}{T_i} \right) \cdot \frac{1}{\sqrt{\beta}} \cdot s^{-3/2} \]

That is,

\[ s^{3/2} = \frac{1}{\sqrt{\beta}} \left( \sum_{i} \frac{1}{T_i} \right) \cdot \frac{1}{U} \]

\[ = \frac{1}{\sqrt{\beta}} \left( \sum_{i} \frac{1}{T_i} \right) \cdot \frac{1}{gC} \]

Then,

\[ s^{3/2 + 1} = \frac{1}{C} \cdot \frac{1}{\sqrt{\beta}} \left( \sum_{i} \frac{1}{T_i} \right) \]

Then with \( \ln s = \ln (1-(1-s)) \approx -(1-s) \) approximation,

\[ (1-s) \approx \frac{2}{B} \ln C - \frac{2}{B} \ln \left[ \frac{1}{\sqrt{\beta}} \left( \sum_{i} \frac{1}{T_i} \right) \right] \]
Therefore, for fixed $\beta$, $T_i$'s and $C$ large enough, as $\beta \to \infty$,

$$\beta \to 1 \iff U \to C$$

Summary: Single link fixed point analysis suggests that for large enough capacity and buffer size, TCP allocates rates to flows sharing the link so that

* Total allocation is close to $C$, and

* each flow gets roughly $\frac{1}{T_i}$ fraction of $C$. 
Resource allocation. Consider a toy resource allocation problem. Given a cake we wish to divide it among n children. What is the fair solution?

A nice answer. Let a knife move on cake from left to right. When someone shouts “stop” s/he gets “the piece”. Rational behavior for person is to look for \( \frac{1}{n} \) cut.

Single-link model. Let \( C \) be capacity of a link which needs to be shared by \( R \) users each having demands \( f_1, \ldots, f_R \).

Problem: Assign rates \( x_1, \ldots, x_R \) so that

\[
\sum_{i=1}^{R} x_i \leq C
\]

\( x_i \in [0, f_i] \) ; \( 1 \leq i \leq R \)

and “utility (\( x_1, \ldots, x_R \))” is maximized.
Optimization:

$$\max \sum_{i=1}^{R} U_i(x_i)$$

subject to

$$0 \leq x_i \leq f_i ; \quad i \in \mathcal{R}$$

$$\sum_{i=1}^{R} x_i \leq c.$$ 

Question: how to decide utility, $U_i(\cdot)$?

Max-min fair: single-link.

Assign rates so that min rate is maximized.

Equivalent, $(x^*_i) ; \quad 1 \leq i \leq R$ is called max-min fair if for any $(i,j)$ with $x^*_i > x^*_j$ only if $x^*_j = f_j$. 
Max-min fair network.

Given network graph $G = (V, E)$ with capacities $(C_e)_{e \in E}$. Let users $1, \ldots, R$ have demands $f_1, \ldots, f_R$. All users have pre-decided paths through network. Allocate rates $x_1^*, \ldots, x_R^*$ so that

$$\begin{cases} 
    x_i^* \in [0, f_i] & 1 \leq i \leq R \\
    \sum_{i : e \ni i} x_i^* \leq C_e & \text{for all } e \in E
\end{cases}$$

and for any other $y_1, \ldots, y_R$ satisfying (x) if $y_i > x_i^*$ for some $i$ then there exists $j$ s.t. $y_j < x_j^* \leq x_i^*$.

Intuition. From max-min fair solution, the only way for some user to become "richer" is to make a poor user "poorer".
$\alpha$-Fair Utility. A class of fairness notion was introduced by Mo-Walrand (1999). It is based on defining utility functions

$$U(x) = \begin{cases} 
-w \cdot \frac{x^{1-\alpha}}{1-\alpha} & \alpha \in (0, \infty) \setminus \{1\} \\
\log x & \alpha = 1.
\end{cases}$$

Corresponding $\alpha$-fair allocation is a solution to the following optimization problem:

**RAC:** $\max_{i=1}^{R} \sum U(x_i)$

subject to $\sum_{i : e \in i} x_i \leq C_e, \; \forall e \in E,$

$x_i \in [0, f_i]$.

We will consider $f_i = \infty$ : all users want as high rate as possible.
Examples.

I. Minimum-delay fair. \( (\alpha = 2) \).

\[
U(x) = -W \cdot \frac{x^{1-2}}{1-2} = \frac{W}{x}.
\]

Delay \( \alpha \frac{1}{\text{rate}} \); therefore, it's min-delay fair.

II. Proportional fair. \( (\alpha = 1) \)

\[
U(x) = W \log x
\]

In case of single-link, \( x^* \propto W \).

III. Max-min fair. \( (\alpha = \infty) \)

\[
U(x) = -W \frac{x^{1-\infty}}{1-\infty} \quad \text{as} \quad x \to \infty.
\]

It approaches max-min fair.
Algorithm. We wish to design algorithms for solving RAC to obtain fair ra allocation. Let’s re-write RAC.

\[ \text{RAC.} \quad \max \sum_{i=1}^{n} u_i(x_i) \]

subject to \[ M x \leq C \]
\[ x \geq 0. \]

Here, \( M = [M_{e,i}] \) is routing matrix; \( M_{e,i} = 1 \) only if \( i \) passes through edge \( e \).

\[ \text{RAC1.} \quad \max \sum_{i=1}^{n} u_i(x_i) - \sum_{e \in E} \int_{0}^{y} f_e(s) \, ds \]

Here, constraint \( y = M x \leq C \) is replaced by means of penalty function, \( f_e(\cdot) \), \( e \in E \).

Properties of \( f_e(\cdot) \): \( \int f_e(s) \, ds \to \infty \) as \( y \to \infty \).

\( f_e(\cdot) \) is non-negative, non-decreasing, continuous, differentiable.
Properties of RAC/RAC1.

Note that $U_i(\cdot)$ is strictly concave for any $\alpha$-fair utility. That is,

$$U_i(\alpha x + (1-\alpha) y) > \alpha U_i(x) + (1-\alpha) U_i(y); \ x \neq y.$$ 

Now, $g(y) = \int_0^y f_e(x) \, dx$ is convex. Because,

$$g_e'(y) = f_e(y) ; \quad g_e''(y) = f_e'(y) \geq 0$$

Therefore,

$$\sum U_i(x_i) - \sum_{i \in e} \int_0^{y=\sum x_i} f_e(s) \, ds$$

is strictly concave.

Thus, both RAC and RAE1 are concave maximization.

Additionally, we will assume that (a) $U_i(x) \to -\infty$ as $x \to 0$;

(b) $\sum U_i(x) - \sum_{i \in e} \int_0^{y=\sum x_i} f_e(s) \, ds \to -\infty$ as $\|x\| \to \infty$.

Then, the maximization problems RAC/RAC1 have unique maximizer in the interior of set $\{x \geq 0\}$. We will consider RAE1 first.
Property of optimal Soln. of RAC1.

Let, \[ V(x) = \sum_{i \in E} u_i(x_i) - \sum_{s} \int_{0}^{s} f_e(s) \, ds. \]

Since, maximizer is in the interior; we have that optimal solution must satisfy the following:

\[ \frac{\partial V}{\partial x_i} = 0 \quad \text{for all } i; \]

That is,

\[ u_i'(x_i) = \sum_{e: i \in E} f_e \left( y_e = \sum_{j \in E} x_j \right) + i \]

A solution of these equations will be the maximizer. Next, we wish to find an iterative distributed algorithm for solving these equations.
Distributed algorithm for RAC1.

Algorithm is iterative. Suppose initial solution is $x_0 = 0$. Let, at iteration $k$, solution is $x_k$. To obtain $x_{k+1}$, we will do the following.

- Links $e \in E$ compute $y_e(k) = \sum_{i : i \in e} x_i(k)$

  Corresponding “penalty”, $p_e(k) = f_e(y_e(k))$.

  “Price” for user $i$ is $q_i(k) = \sum_{e : i \in e} p_e(k)$.

- Gradient algorithm:

  $$x(k+1) = x(k) + \alpha_k \nabla V(x(k))$$

  But $\nabla V(x) = [u'_i(x_i) - q_i(x)]$
Lemma. Gradient algorithm converges to global optimal if $\alpha_k \to 0$ and $\sum \alpha_k = \infty$.

Proof. We will describe proof based on continuous approximation of gradient dynamics.

That is,

$$\frac{dX_i(t)}{dt} = U_i'(x_c(t)) - q_c(x_c(t)) = \frac{\partial V(x(t))}{\partial x_i}.$$

Consider, Lyapunov function

$$L(t) = V(x(t)).$$

$$\dot{L}(t) = \frac{dV(x(t))}{dt} = \sum_{i=1}^{R} \frac{\partial V(x(t))}{\partial x_i} \frac{dx_i}{dt}$$

$$= \sum_{i=1}^{R} \left( \frac{\partial V(x(t))}{\partial x_i} \right)^2$$

Then

$$\dot{L}(t) = 0 \iff \frac{\partial V(x(t))}{\partial x_i} = 0 \quad \forall i.$$
As per above discussion, this implies that the algorithm converges only when \( \frac{\partial V}{\partial x_i} \rightarrow 0 + i \).

That is, the convergent point is the unique optimal solution. This completes the proof. Q.E.D.

**Exact Penalty Function.** The algorithm above establishes that we have distributed solution for finding approximate solution to the RAC problem (i.e. RAE1). Clearly, utility function is always approximate so practically solving RAE1 is "good enough." However, we would like to obtain exact solution to RAC if possible. We do so next by means of **Exact Penalty Function/Adaptive Penalty Function**.

**Basic idea.** Change penalty function depending on how the solution to RAE1 appears. Specifically, consider the following form of penalty function:

\[
fe(y_e, \hat{c}_e) = \left( \frac{y_e}{\hat{c}_e} \right)^B e^{\frac{-C}{\hat{c}_e}} \rightarrow \begin{cases} 
0 & y_e > \hat{c}_e \\
1 & y_e = \hat{c}_e \\
0 & y_e < \hat{c}_e 
\end{cases}
\]
\[
\frac{d \hat{C}_e}{dt} = \alpha_e \left( C_e - y_e \right)^+ \hat{C}_e = \begin{cases} 
C_e - y_e & \hat{C}_e > 0 \\
\max \left( C_e - y_e, 0 \right) & \hat{C}_e = 0
\end{cases}
\]

**Algorithm.**

\[
\hat{x}_e(t) = K_e(x_i(t)) \left[ U_e(x_i(t)) - q_e(x_i(t)) \right]
\]

where, \( q = M p \);

\[
P_e(x(t)) = f_e \left( \sum_{i \in \mathcal{E}} x_i(t) \right)
\]

**AND**

\[
f_e(x) = \left( \frac{8}{\hat{C}_e} \right)^{Be} \text{ with }
\]

\[
\frac{d \hat{C}_e}{dt} = \alpha_e \left( C_e - y_e \right)^+ \hat{C}_e \quad ; \quad y_e = \sum_{i \in \mathcal{E}} x_i
\]

**Theorem.** The algorithm converges to the optimal solution of RAC if \( M \) has full rank.

**Proof.** See, page 30-31 of book [Srikant].
Dual of RAC. Recall that

\[ \text{RAC: } \max \sum u_i(x_i) \text{ subject to } y = Mx \leq c. \]

Lagrangian. \( x \geq 0, \lambda \geq 0 : \)

\[ L(x; \lambda) = \sum_i u_i(x_i) - \sum_e \lambda_e (y_e - c_e) \]

Dual. \( \lambda \geq 0 \)

\[ D(\lambda) = \max_{x \geq 0} L(x; \lambda) \]

\[ = \max_{x \geq 0} \left[ \sum_i u_i(x_i) - \sum_e \lambda_e y_e \right] + \sum_e \lambda_e c_e \]

Problem: \( \min_{\lambda \geq 0} D(\lambda) \).

By strong duality, the cost of above solution is the same as that of RAC.
Now, to evaluate $D(\lambda)$, note that
\[
\frac{\partial L}{\partial x_i} = 0. \quad \text{That is, } \quad \mathcal{U}_i(x_i) - \sum_{e:i \in e} \lambda_e = 0.
\]

By complimentary slackness condition, we have
\[
\lambda_i(y_e - c_e) = 0 \quad \text{for all } e.
\]

Algorithm for Dual.
\[
x_i = \left( \mathcal{U}^T \right)^\dagger \left( \sum_{e:i \in e} \lambda_e \right)
\]
\[
\frac{d\lambda_e}{dt} = h_e(\lambda_e)(y_e - c_e)^+_{\lambda_e}
\]

Theorem. The above gradient algorithm converges to the optimal solution if the routing matrix $M$ is of full rank.

Proof. Can be found on page 29, [Srikant].
Primal-Dual algorithm.

Primal step. \[ \frac{dx_i(t)}{dt} = k_i(x_i(t)) \Big( u_i'(x_i) - q_i(x_i(t)) \Big) \]

Dual step. \[ \frac{dp_e(t)}{dt} = k_e(p_e) \Big( y_e - c_e \Big)^+ \]

Theorem. The primal-dual algorithm converges to the optimal solution of RAE. \( \mathbf{v} \)

Proof. Can be found, page 28, [Srikant].

Next, we will identify congestion control algorithm with primal step and the packet-drop mechanism with dual algorithm.

To start, we recall the TCP "fluid model" we had earlier:

\[ \frac{dx_i(t)}{dt} \approx \frac{1}{T_i} - \beta q_i(t) \Gamma_i \cdot x_i^2(t). \]
Usually, \( q_i(t) \) is small. Then, \(-q_i(t) \approx 1\): 

\[
\frac{dx_i(t)}{dt} = \frac{1}{T_i} - \beta T_i q_i(t) x_i^2(t)
\]

\[
= \beta x_i^2(t) T_i \left[ \frac{1}{\beta T_i^2 x_i^2(t)} - q_i(t) \right]
\]

\[
\dot{x}_i(t) = K_i(x_i(t)) \left[ \frac{U_i'(x_i(t))}{x_i} - q_i(t) \right]
\]

That is: \( U_i'(x) = \frac{1}{x^2} - \left[ \frac{1}{\beta T_i^2 x^2} \right] \)

That is: \( U_i(x) = -\frac{W_i}{x} \),

where \( W_i = \frac{1}{\beta T_i^2} \).

That is, it is \( \alpha = 2 \) fair.

Thus, rate-control \( \equiv \) primal-algorithm.
Now, queue at links of the network evolve as

\[ \dot{Q}_e(t) \approx \left( \frac{Y_e(t) - C_e}{Q_e(t)} \right)^+. \]

Then, marking/drop probability

\[ P_e(t) \approx \frac{Q_e(t)}{B} \]

That is, \( P_e(t) \approx \frac{1}{B} \left( \frac{Y_e(t) - C_e}{Q_e(t)} \right)^+ \)

We note that net-drop probability,

\[ q_e^d(t) \approx \sum_{e: ice} P_e(t) \quad \text{because} \]

\[ 1 - q_e^d(t) = \prod_{e: ice} (1 - P_e(t)) \approx 1 - \sum_{e: ice} P_e(t) \]

Thus, dual-algorithm = queue drop mechanism.
Summary. Internet congestion control protocols (rate-control, TCP and packet-drop mechanism), though designed with very different reasoning, have resulted in solving certain "fair-rate allocation" problem.

Next, we will study effect of such protocols on the throughput property of the network.