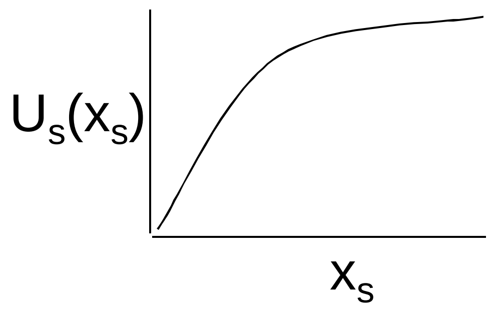
Optimization-based approach to congestion control

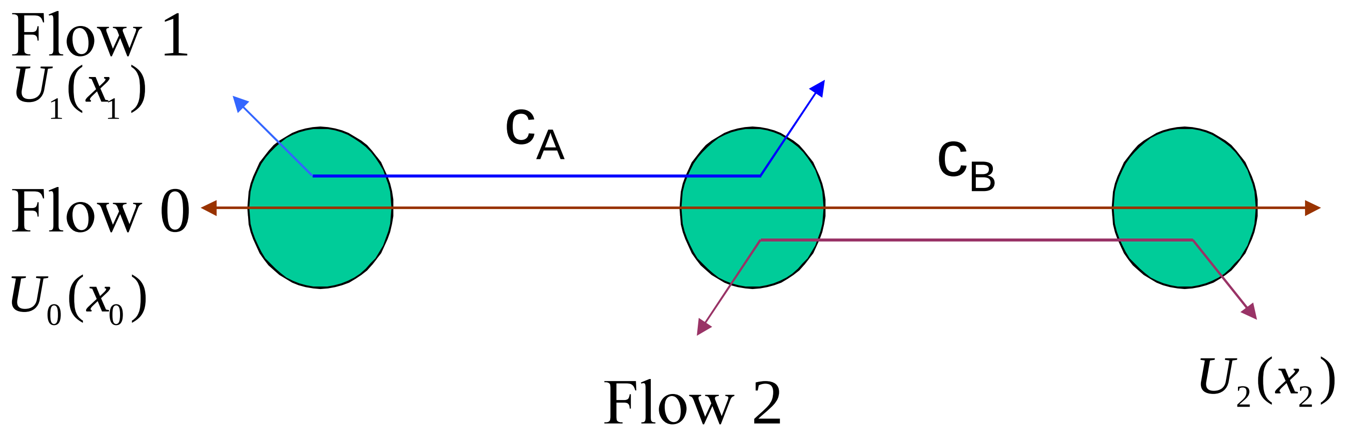
Resource allocation as an optimization problem:

- how to allocate resources (e.g., bandwidth) to optimize some objective function
- may not be possible that optimality exactly obtained but…
  - optimization framework as means to explicitly steer network towards desirable operating point
  - practical congestion control as distributed asynchronous algorithms to solve optimization problem
  - systematic approach towards protocol design

Model

- network: links $\{l\}$, capacities $\{c_l\}$
- flows S: $(L(s), U_s(x_s)), s \in S$
  - $L(s)$ - links used by flow s
  - $U_s(x_s)$ – utility, strictly concave function of flow rate x_s





Kelly’s system problem

$$\max_{x_0,x_1,x_2} \sum_{i=1}^{3} U_i(x_i)$$

subject to

$$\begin{aligned} x_0 + x_1 &\leq c_A \\ x_0 + x_2 &\leq c_B \\ x_i &\geq 0 \end{aligned}$$

Optimization Problem

$$\max_{x_s \geq 0} \quad \sum_s U_s(x_s)$$

“system” problem

$$\text{subject to} \quad \sum_{s \in S(l)} x_s \leq c_l, \forall l \in L$$

- maximize system utility (note: all sources “equal)
- constraint: bandwidth used less than capacity
- centralized solution is impractical:
  - must know all utility functions
  - must control all sources
  - we’ll see: congestion controller as distributed asynchronous algorithm to solve this problem

Outline

- What are good choices for utilities? How fair among users?
- Is there a distributed algorithm to approximate the optimization (source algorithm, price function)?
- What utility function is optimized by TCP?

$\alpha -$ fair utility functions

$$U(x) = \begin{cases} \frac{x^{1-\alpha}}{1-\alpha}, & \alpha \geq 0, \alpha \neq 1 \\ \ln x, & \alpha = 1 \end{cases}$$

$$U'(x) = \frac{1}{x^{\alpha}}, \qquad \alpha \geq 0$$

Special case: Max-min fairness

Rates $\{x_r\}$ max-min fair if for any other feasible rates $\{y_r\}$, if $y_s > x_s$, then $\exists$ p, such that $x_p \leq x_s$ and $y_p < x_p$

Can only increase a rate by decreasing a lower rate

Corresponding utility function?

$$U_r(x_r) = \lim_{\alpha \to \infty} \frac{x_r^{1-\alpha}}{1-\alpha}$$

Other special cases

- proportional fairness, $\alpha = 1$
  - $U_r(x_r) = \ln x_r$
  - weighted proportional fairness if $U_r(x_r) = w_r \ln x_r$

- rates $\{x_r\}$ are minimum potential delay fair if $U_r(x_r) = -w_r / x_r$

  Interpretation: if w_r is file size, then w_r / x_r is transfer time; optimization problem is to minimize sum of transfer delays

Recall: Optimization Problem

- network: links $\{l\}$, capacities $\{c_l\}$
- flows S: $(L(s), U_s(x_s)), s \in S$
  - $L(s)$ - links used by flow s
  - $U_s(x_s)$ – utility, strictly concave function of source flow x_s

$$\max_{x_s \geq 0} \quad \sum_s U_s(x_s)$$

$$\text{subject to} \quad \sum_{s \in S(l)} x_s \leq c_l, \forall l \in L$$

"system" problem

- constraints of form $\boldsymbol{Rx} \leq \boldsymbol{c}$,

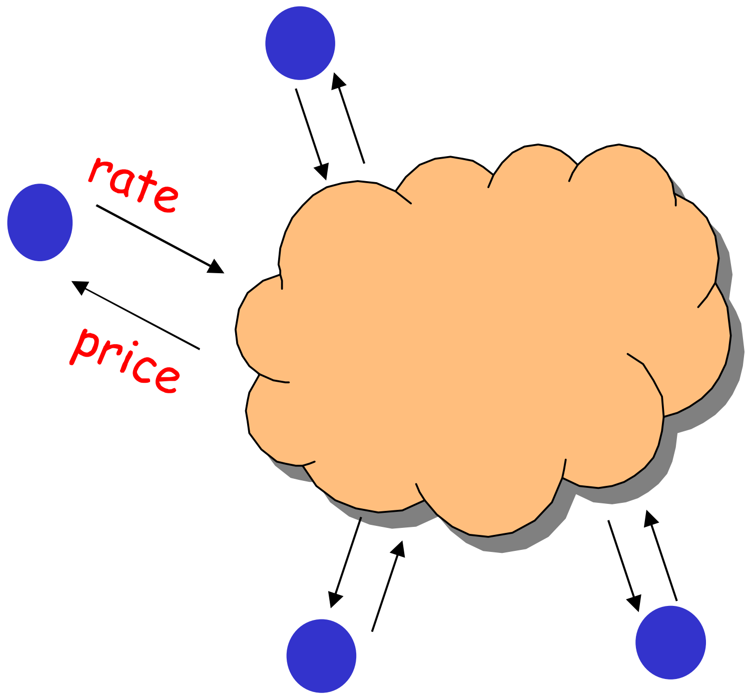
  $\boldsymbol{R}$, routing matrix

Distributed algorithm

Optimization problem decouples into (Kelly):

- greedy optimization problem for every session r

  $\max U_r(x_r) - q_r x_r$
- price q_r given by network as function of rates, $q_r = \sum p_l$ where p_l is price for link l
- provides solution to system problem



Remove constraints: primal approach

- consider following problem

$$V(\mathbf{x}) = \sum_{r} U_r(x_r) - \sum_{l \in L} g_l(\sum x_s)$$

- $g_l(y)$ – penalty function
  - g_l (y) non decreasing, convex and

    g_l (y) $\rightarrow \infty$ as y $\rightarrow \infty$

  - f_l (y) = dg_l (y)/dy

max V(**x**)

$$\frac{\partial V}{\partial x_r} = 0, \quad r \in S$$

$$U_r'(x_r) - \sum_{l:l\in r} f_l(y_l) = 0, \quad r \in S$$

$$y_l = \sum_{s:l\in s} x_s, \quad l \in L$$

- $p_l(t)$ price of link l at time t

$$p_l(t) = f_l(y_l(t))$$

$$U_r'(x_r) - \sum_{l \in r} p_l = 0, \qquad r \in S$$

- optimization problem decouples into:
  - greedy optimization problem for every session

$$\max U_{\grave{0}}(x_{\grave{0}}) - q_{\grave{0}} x_{\grave{0}}$$

  - price q_s given by network as function of rates

$$q_{\grave{0}} = \sum_{l \in r} p_l$$

Source Algorithm

- source performs gradient descent:

$$\dot{x}_r(t) = k_r(x_r(t))(U'_r(x_r(t)) - q_r(t))$$

$$x_r(t+1) = x_r(t) + \dot{x}_r(t)$$

- $k_r(\cdot)$ nonnegative nondecreasing function
- above algorithm converges to unique solution for any initial condition
- can show convergence using Lyapunov functions
- example: q_r – loss probability

Example: Proportionally-Fair Controller

If utility function is

$$U_r(x_r) = w_r \ln x_r$$

then a controller that implements it is given by

$$\dot{x}_r = \kappa_r(w_r - x_r q_r)/x_\mathrm{r}$$

Price functions

- drop packets when sending rate exceeds link capacity (bandwidth)

$$p_l = f_l(y_l) = \frac{(y_l - c_l)^+}{y_l}$$

Price functions

Mark packet when queue length (bits) is at least B

- example - M/M/1 queue

$$p_l = f_l(y_l) = \begin{cases} (y_l / c_l)^B, & y_l < c_l \\ 1, & y_l \geq c_l \end{cases}$$

Pr(queue length >= B)

Price functions

Drop packets when queue is full

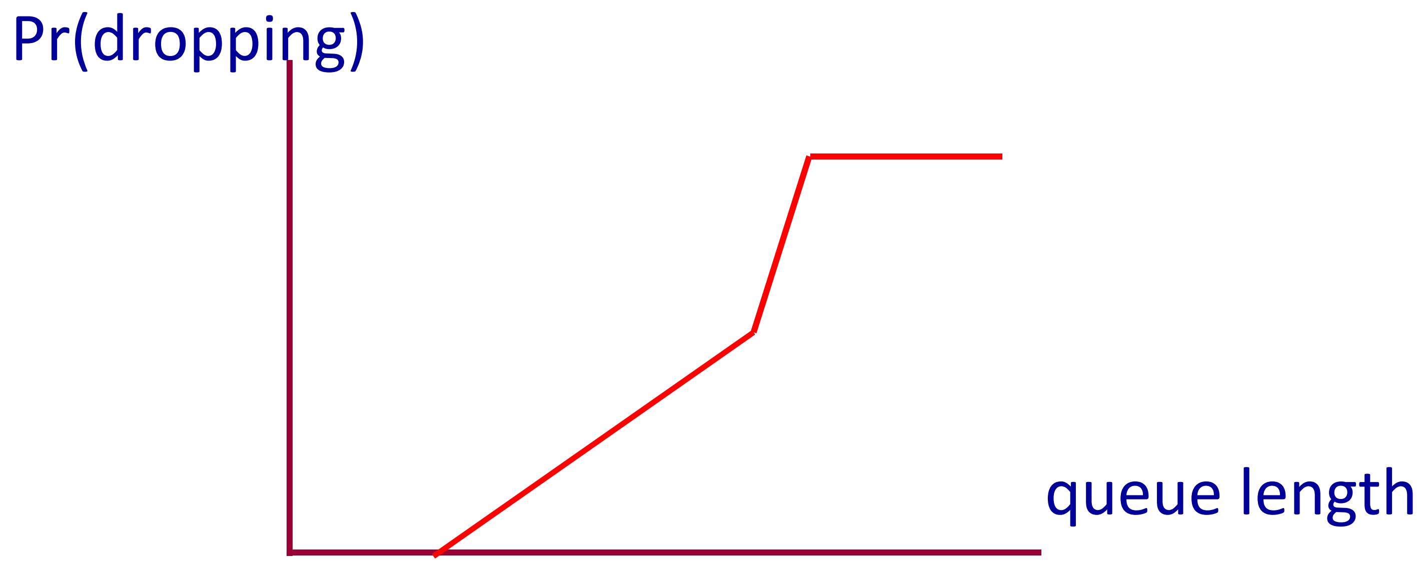
- example - M/M/1/B queue

$$p_l = f_l(y_l) = \frac{1-\rho}{1-\rho^{B+1}}\rho^B$$

Pr(queue overflow)

where $\rho = y_l / c_l$

Random Early Detection (RED)



- simplified view: $p_l = f_l(q_l)$
- issues with only dropping when buffer is full
  - unfair among flows
  - synchronized "hold back"
- RED: probabilistically drop packets, depends on queue length

Reverse engineering TCP

- condition for optimality is

$$U'_i(x_i) - p = 0$$

or

$$x_i = U_i'^{-1}(p)$$

if we have an expression for x_i, we can use to obtain U_i

Case study: TCP-Reno

- TCP-Reno in equilibrium:

$$\frac{2}{2 + T_i^2 x_i^2} = p$$

- utility function:

$$U_i(x_i) = \frac{\sqrt{2}}{T_i} \arctan \frac{x_i T_i}{\sqrt{2}}$$

Case study: Simplified TCP-Reno

- suppose

$$x = \frac{\sqrt{2(1-p)}}{T\sqrt{p}} \approx \frac{\sqrt{2}}{T\sqrt{p}}$$

- then,

$$U(x) = -\frac{2}{T^2 x}$$