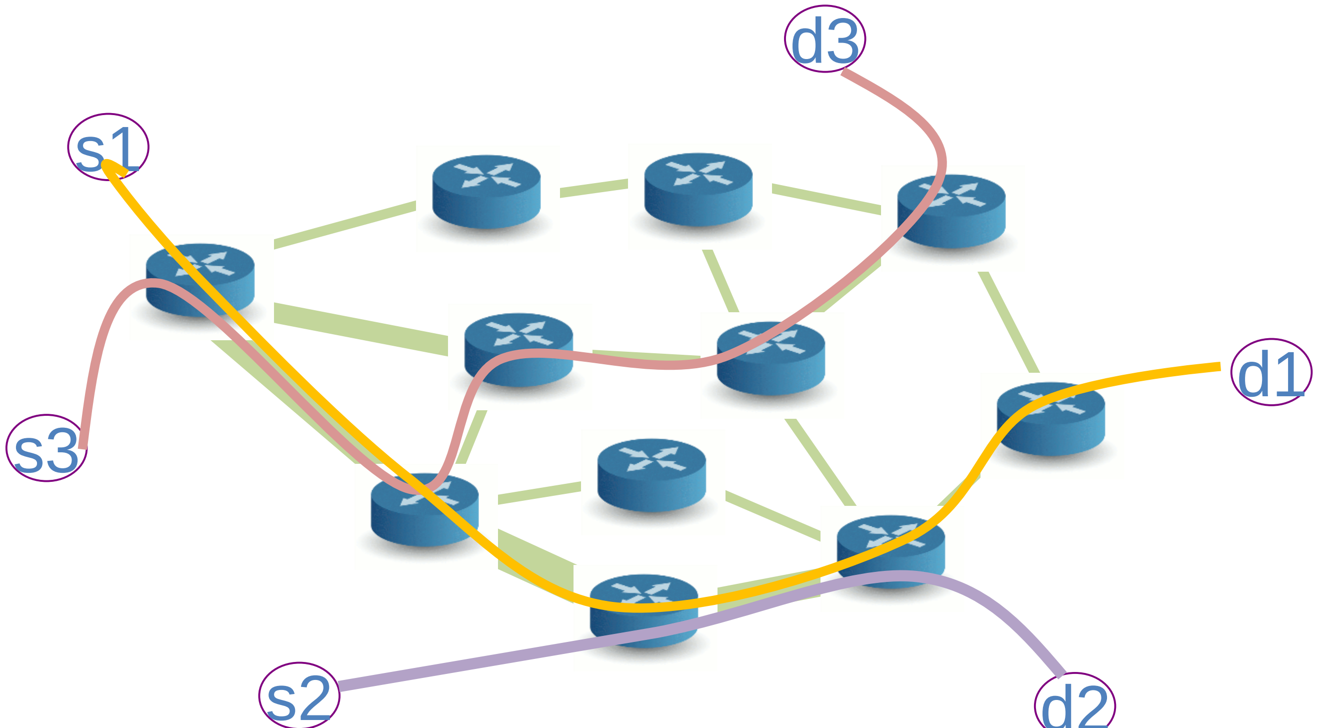
Networks: distributed control and emerging phenomena

(formerly: Networks, algorithms and probability)

Laurent Massoulié

laurent.massoulie@inria.fr

Distributed control for data transport



- ❑ How to assign bandwidth in networks
  - ❑ Understanding TCP, the protocol regulating most Internet traffic

  → Convex optimization theory & dynamical systems

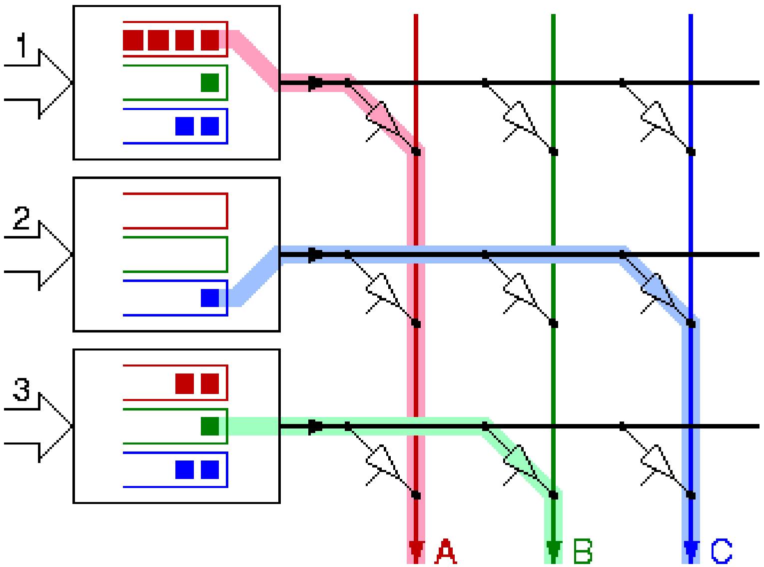
Distributed control for data transport

How to manage collisions (i.e. lost transmissions because of interference) between wireless transmitters

- ❑Aloha and Ethernet protocols
- →Markov chains and criteria for ergodicity

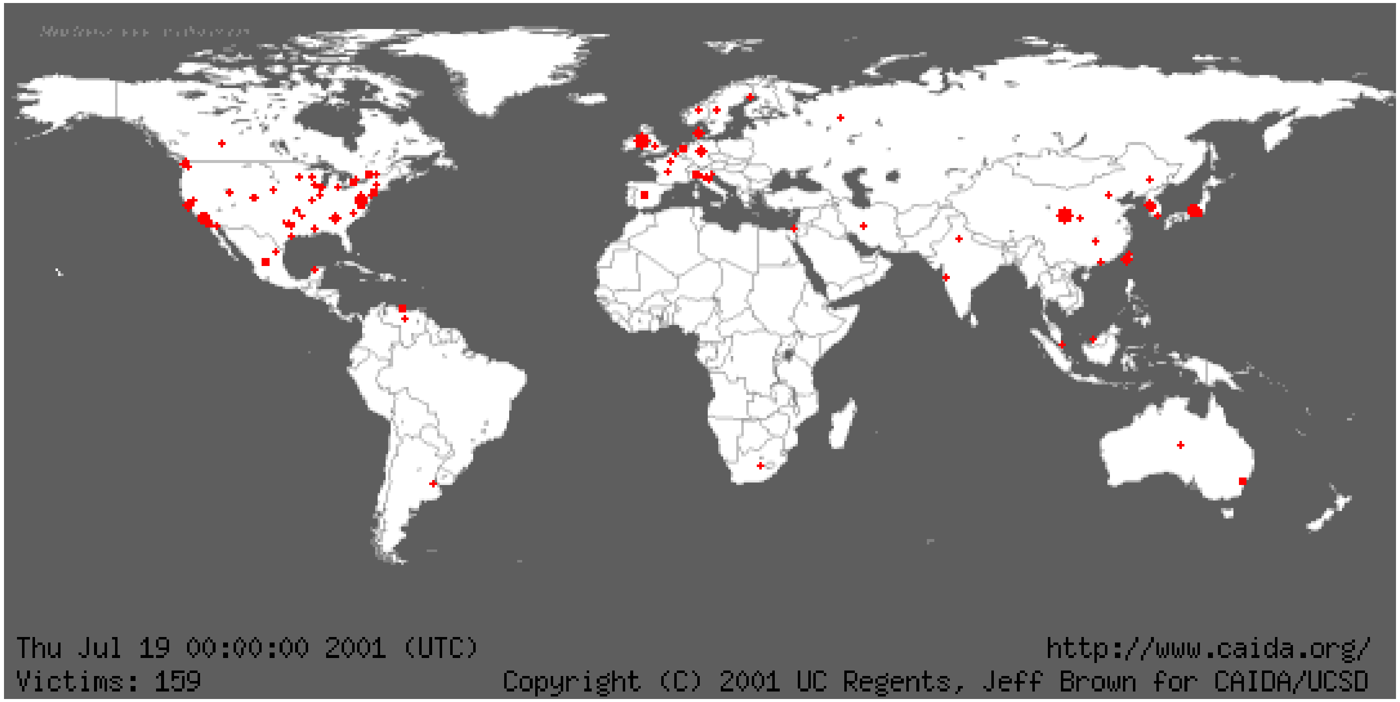
Distributed control for data transport

Crossbar switch with input queues:



- How to schedule transmissions in switches, and multi-hop wireless networks
  - Max-weight & backpressure algorithms

Network epidemics



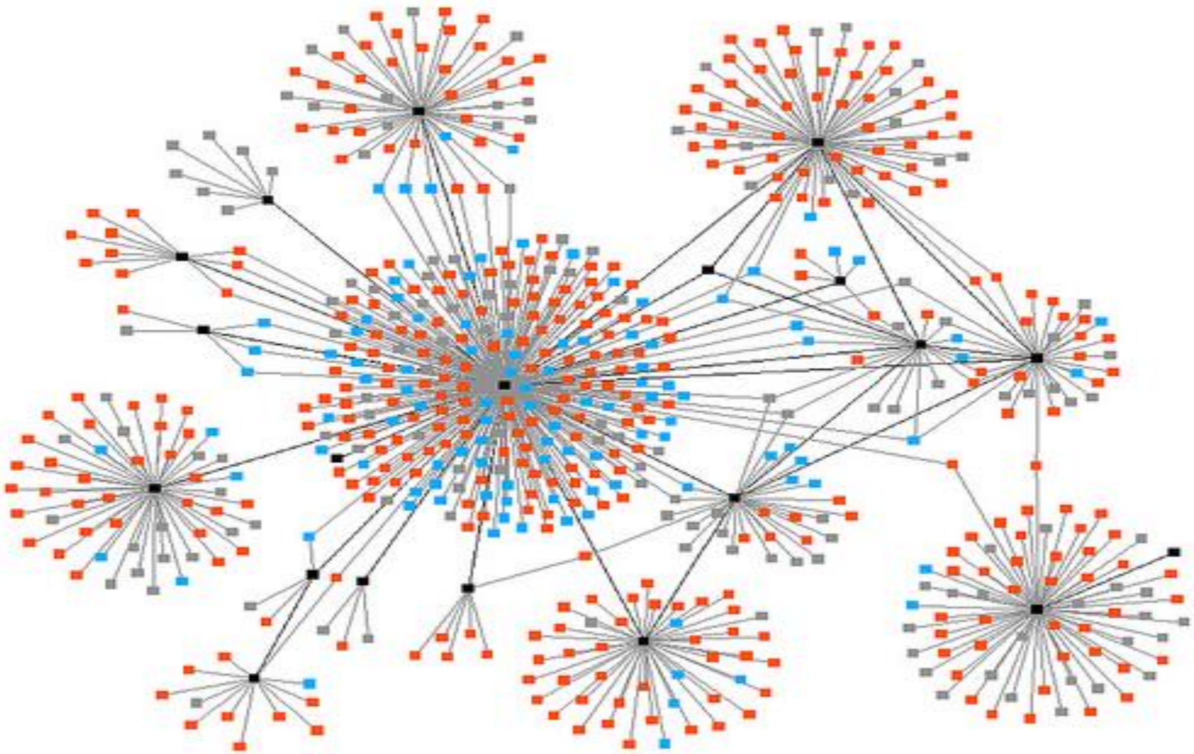
Spread of “CodeRed” Internet worm, 2001

Spread of a picture on facebook
https://www.facebookstories.com/stories/2200/

Network epidemics

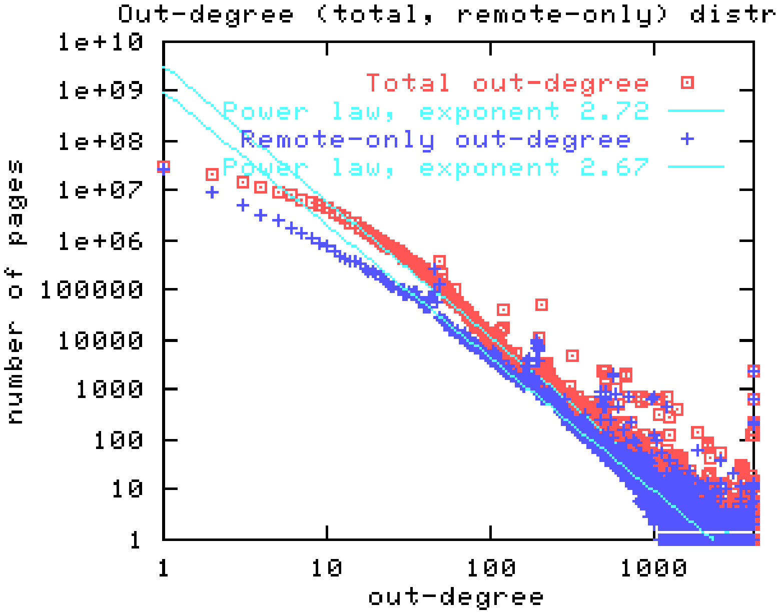
- What makes an epidemic potent or weak
  → random graphs, branching processes and phase transitions
- What features of network topology affect epidemic outbreak
  → graph topology descriptors, comparison of Markov chains by “coupling”
- How to maximize size of outbreak
  → submodular functions and greedy maximization

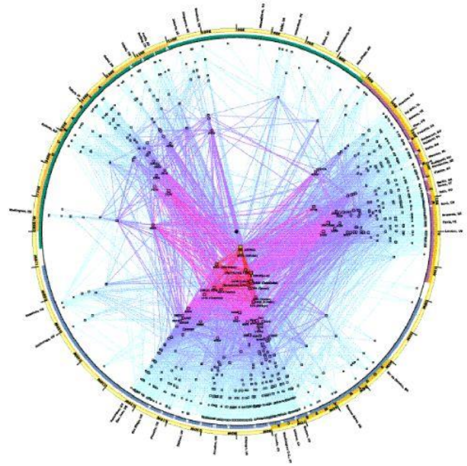
Network epidemics

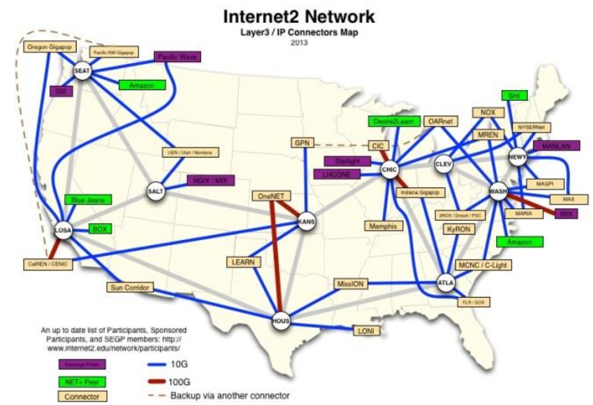


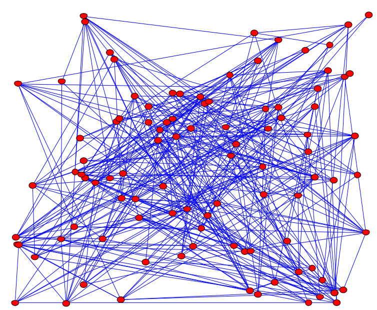
- ❑What is a “small world” network
  - ❑And how to search for information in it

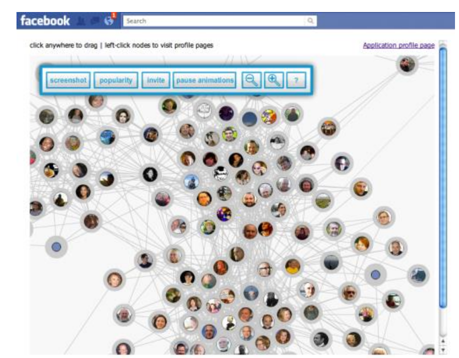
Network epidemics







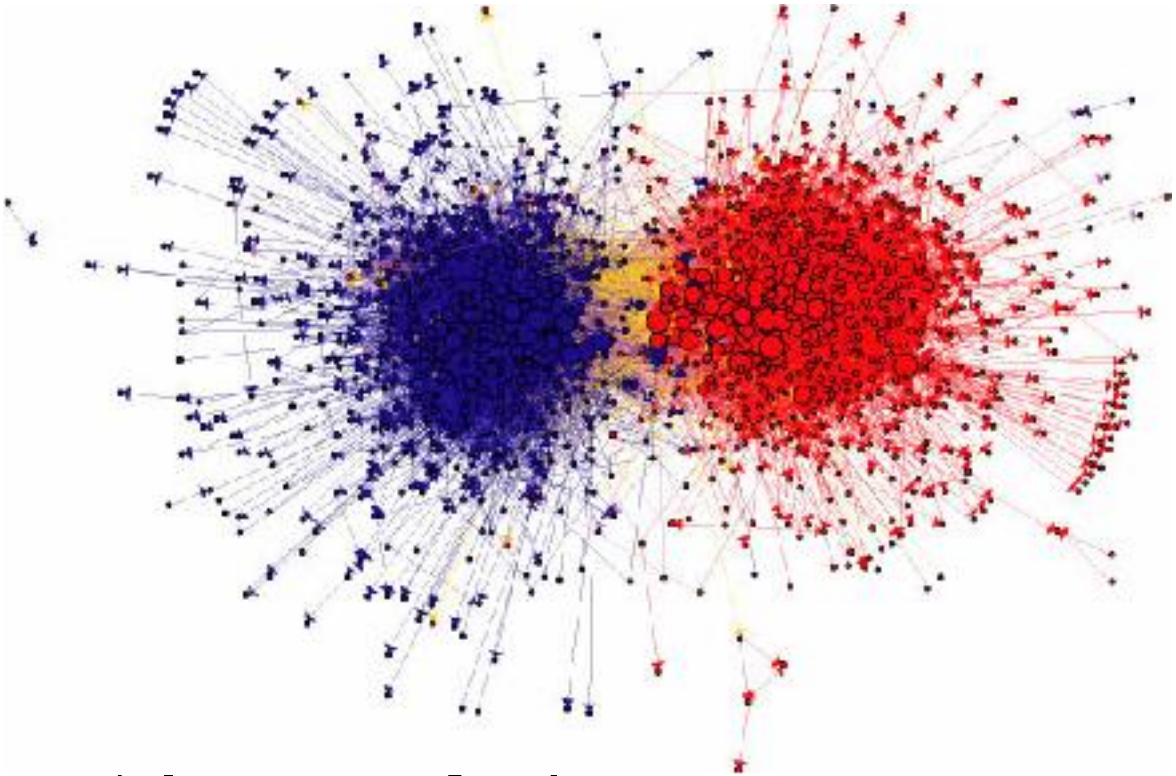




- ❑ Why are most networks “scale-free” (a.k.a. power-law)

  →martingales, coupling and “concentration inequalities”

Network epidemics



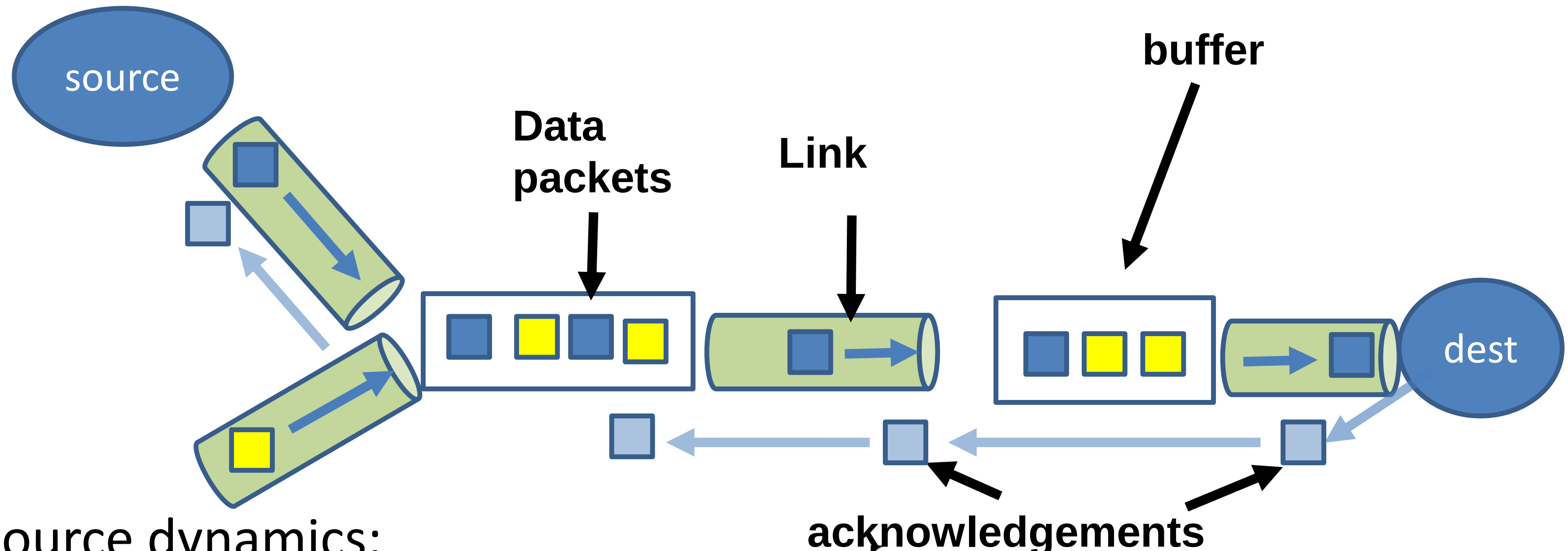
Political blogs:
Republican vs Democrats

- How to find community structure and recommend contacts in a social network
  - →spectra of random graphs and spectral methods

Network resource allocation: principles and algorithms

- Convex optimization model
- A “primal” algorithm
- Reverse-engineering TCP
- Lagrangian, duality and Lagrange multipliers
- A “dual” algorithm

TCP in one slide



Source dynamics:

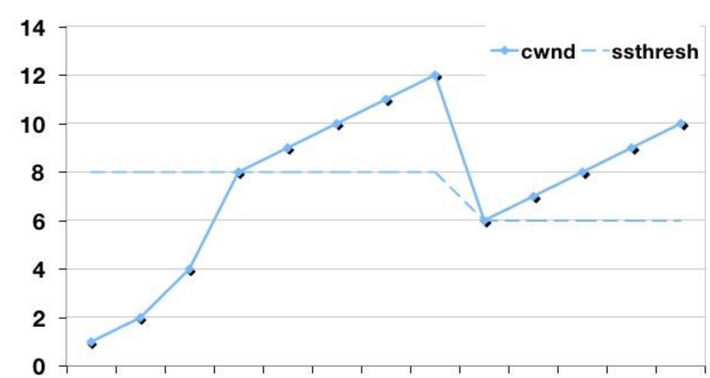
- Maintain Nb of (sent&not acked pkts)=cwnd (congestion window)
- Update cwnd

← cwnd+1/cwnd upon receipt of pkt ack

← cwnd/2 upon detection of pkt loss

"Congestion avoidance" alg introduced in 1993

After Internet congestion collapse



Outline

- Convex optimization model
- A "primal" algorithm
- Reverse-engineering TCP
- Lagrangian, duality and multipliers
- A "dual" algorithm

Network model

- Resources, or links, $\ell \in \mathcal{L}$, each with capacity $C_\ell > 0$
- Users, or transmissions, or flows, $s \in \mathcal{S}$
- User s uses same rate at all $\ell \in s$ ($s \leftrightarrow$ subset of $\mathcal{L}$)

Network model

- Resources, or links, $\ell \in \mathcal{L}$, each with capacity $C_\ell > 0$
- Users, or transmissions, or flows, $s \in \mathcal{S}$
- User s uses same rate at all $\ell \in s$ ($s \leftrightarrow$ subset of $\mathcal{L}$)

FEASIBLE RATES:
variables $x_s \geq 0$, $s \in \mathcal{S}$
such that $\forall \ell \in \mathcal{L}$, $\sum_{s \ni \ell} x_s \leq C_\ell$

Network model

- Resources, or links, $\ell \in \mathcal{L}$, each with capacity $C_\ell > 0$
- Users, or transmissions, or flows, $s \in \mathcal{S}$
- User s uses same rate at all $\ell \in s$ ($s \leftrightarrow$ subset of $\mathcal{L}$)

FEASIBLE RATES:
variables $x_s \geq 0$, $s \in \mathcal{S}$
such that $\forall \ell \in \mathcal{L}$, $\sum_{s \ni \ell} x_s \leq C_\ell$

POTENTIAL APPLICATIONS

- Links on single path from source to destination
- Links on tree of transmission from source to set of receivers

Allocation principles 1

- **max-min fairness**: feasible x^{mm} such that $\forall s \in \mathcal{S}, \exists \ell \in s$ with $\sum_{t \ni \ell} x_t^{mm} = C_\ell$ and $x_s^{mm} = \max_{t \ni \ell} x_t^{mm}$ ("no envy": each s can find competing t at least as poor as s)

Allocation principles 1

- **max-min fairness**: feasible x^{mm} such that $\forall s \in \mathcal{S}, \exists \ell \in s$ with $\sum_{t \ni \ell} x_t^{mm} = C_\ell$ and $x_s^{mm} = \max_{t \ni \ell} x_t^{mm}$ ("no envy": each s can find competing t at least as poor as s)
- **Proportional fairness**: feasible x^{pf} such that for all feasible y, $\sum_s \frac{y_s - x_s^{pf}}{x_s^{pf}} \leq 0$

Allocation principles 1

- **max-min fairness**: feasible x^{mm} such that
  $\forall s \in \mathcal{S}, \exists \ell \in s$ with $\sum_{t \ni \ell} x_t^{mm} = C_\ell$ and $x_s^{mm} = \max_{t \ni \ell} x_t^{mm}$
  ("no envy": each s can find competing t at least as poor as s)
- **Proportional fairness**: feasible x^{pf} such that
  for all feasible y, $\sum_s \frac{y_s - x_s^{pf}}{x_s^{pf}} \leq 0$

Alternative characterization:
Unique maximizer of $\sum_s \log(x_s)$ among feasible x

Allocation principles 1

Alternative characterization: Nash's bargaining solution

i.e. unique vector $\phi(\mathcal{C})$ in feasible convex set $\mathcal{C} \subset \mathbb{R}_+^{\mathcal{S}}$ s.t.

- Pareto efficiency: $\phi(\mathcal{C}) \leq x \in \mathcal{C} \Rightarrow x = \phi(\mathcal{C})$
- independence of irrelevant alternatives: $\phi(\mathcal{C}) \in \mathcal{C}' \subset \mathcal{C} \Rightarrow \phi(\mathcal{C}) = \phi(\mathcal{C}')$
- symmetry: $\mathcal{C}$ symmetric $\Rightarrow \phi(\mathcal{C})_i \equiv \phi(\mathcal{C})_1$
- scale invariance: for diagonal D with $D_{ii} \geq 0$, $\phi(D\mathcal{C}) = D\phi(\mathcal{C})$

Allocation principles 2

Network Utility Maximization x^*: solution of

$$\begin{array}{rc} \text{Max} & \sum_s U_s(x_s) \\ \text{Over} & x_s \geq 0 \\ \text{Such that} & \forall \ell, \sum_{s \ni \ell} x_s \leq C_\ell \end{array} \qquad (P)$$

for **concave, increasing** utility functions $U_s : \mathbb{R}_+ \to \mathbb{R}$

Allocation principles 2

Network Utility Maximization x^*: solution of

$$\begin{array}{rcc} \text{Max} & \sum_s U_s(x_s) & \\ \text{Over} & x_s \geq 0 & (P) \\ \text{Such that} & \forall \ell, \sum_{s \ni \ell} x_s \leq C_\ell & \end{array}$$

for **concave, increasing** utility functions $U_s : \mathbb{R}_+ \to \mathbb{R}$

$\Rightarrow$ A concave optimization program

Allocation principles 2

Network Utility Maximization x^*: solution of

$$\begin{array}{rcr} \text{Max} & \sum_s U_s(x_s) & \\ \text{Over} & x_s \geq 0 & (P) \\ \text{Such that} & \forall \ell, \sum_{s \ni \ell} x_s \leq C_\ell & \end{array}$$

for **concave, increasing** utility functions $U_s : \mathbb{R}_+ \to \mathbb{R}$

$\Rightarrow$ A concave optimization program

EXAMPLES

Proportional fair x^{pf}: $U_s = \log$

Network Utility Maximization x^*: solution of

$$\begin{array}{rcc} \text{Max} & \sum_s U_s(x_s) & \\ \text{Over} & x_s \geq 0 & (P) \\ \text{Such that} & \forall \ell, \sum_{s \ni \ell} x_s \leq C_\ell & \end{array}$$

for **concave, increasing** utility functions $U_s : \mathbb{R}_+ \to \mathbb{R}$

$\Rightarrow$ A concave optimization program

Examples

Proportional fair x^{pf}: $U_s = \log$

For $w, \alpha > 0$, (w, α)-fair $x = x(w, \alpha)$: $U_s(x_s) = w_s \frac{x_s^{1-\alpha}}{1-\alpha}$

Allocation principles 2

Network Utility Maximization x^*: solution of

$$\begin{array}{rcc} \text{Max} & \sum_s U_s(x_s) & \\ \text{Over} & x_s \geq 0 & (P) \\ \text{Such that} & \forall \ell, \sum_{s \ni \ell} x_s \leq C_\ell & \end{array}$$

for **concave, increasing** utility functions $U_s : \mathbb{R}_+ \to \mathbb{R}$

$\Rightarrow$ A concave optimization program

EXAMPLES

Proportional fair x^{pf}: $U_s = \log$

For $w, \alpha > 0$, (w, α)-fair $x = x(w, \alpha)$: $U_s(x_s) = w_s \frac{x_s^{1-\alpha}}{1-\alpha}$

[Exercise: $\lim_{\alpha \to 1} x(1, \alpha) = x^{pf}$ and $\lim_{\alpha \to +\infty} x(1, \alpha) = x^{mm}$]

Relaxed constraints and a "primal" algorithm

$$\text{Relaxed problem:} \quad \text{Max} \quad \sum_s U_s(x_s) - \sum_\ell C_\ell(y_\ell)$$

$$\text{Over} \quad x_s \geq 0 \qquad \text{(RP)}$$

$$\text{with} \quad y_\ell = \sum_{s \ni \ell} x_s$$

for concave increasing utility functions U_s and convex increasing cost functions C_ℓ

Relaxed constraints and a "primal" algorithm

$$\text{Relaxed problem:}\quad \text{Max} \quad \sum_s U_s(x_s) - \sum_\ell C_\ell(y_\ell)$$
$$\text{Over} \quad x_s \geq 0 \qquad \text{(RP)}$$
$$\text{with} \quad y_\ell = \sum_{s \ni \ell} x_s$$

for concave increasing utility functions U_s and convex increasing cost functions C_ℓ
primal algorithm: for U_s and C_ℓ differentiable, let

$$\frac{d}{dt} x_s = \kappa_s(x_s) \left(U_s'(x_s) - \sum_{\ell \in s} C_\ell'(y_\ell) \right) \qquad \text{"gradient ascent"}$$

Relaxed constraints and a "primal" algorithm

$$\text{Relaxed problem:} \quad \text{Max} \quad \sum_s U_s(x_s) - \sum_\ell C_\ell(y_\ell)$$
$$\text{Over} \quad x_s \geq 0 \qquad \text{(RP)}$$
$$\text{with} \quad y_\ell = \sum_{s \ni \ell} x_s$$

for concave increasing utility functions U_s and convex increasing cost functions C_ℓ

primal algorithm: for U_s and C_ℓ differentiable, let

$$\frac{d}{dt} x_s = \kappa_s(x_s) \left(U_s'(x_s) - \sum_{\ell \in s} C_\ell'(y_\ell) \right) \quad \text{"gradient ascent"}$$

$\rightarrow$ Implementable in a distributed fashion

Stability via Lyapunov functions

Criterion for convergence of ODE $\dot{x} = F(x)$ with trajectories in $O \subset \mathbb{R}^n$

Theorem

Assume F continuous on O, and $\exists V : O \to \mathbb{R}$ such that:

(i) V continuously differentiable

(ii) $\forall a \leq A$, $\{x \in O : V(x) \leq A\}$ and $\{x \in O : V(x) \in [a, A]\}$ either compact or empty

(iii) $\forall x \in O \setminus B$, $\nabla V(x) \cdot F(x) < 0$, where $B = \text{argmin}_{x \in O}\{V(x)\}$

Then $\lim_{t\to\infty} V(x(t)) = \inf_{x \in O} V(x)$, $\lim_{t\to\infty} d(x(t), B) = 0$.

If $B = \{x^*\}$ then $\lim_{t\to\infty} x(t) = x^*$.

Application to gradient ascent / descent dynamics

$$\frac{d}{dt}x_s = \kappa_s(x_s)\left(U_s'(x_s) - \sum_{\ell \in s} C_\ell'(y_\ell)\right)$$

Let $W(x) = \sum_s U_s(x_s) - \sum_\ell C_\ell(y_\ell)$ (system welfare)
and $V(x) = -W(x)$

Application to gradient ascent / descent dynamics

$$\frac{d}{dt}x_s = \kappa_s(x_s)\left(U_s'(x_s) - \sum_{\ell\in s} C_\ell'(y_\ell)\right)$$

Let $W(x) = \sum_s U_s(x_s) - \sum_\ell C_\ell(y_\ell)$ (system welfare)
and $V(x) = -W(x)$

Then: $\nabla V(x) \cdot F(x) = -\sum_s \kappa_s(x_s)\left[\frac{\partial}{\partial x_s} W(x)\right]^2$

Application to gradient ascent / descent dynamics

$$\frac{d}{dt}x_s = \kappa_s(x_s)\left(U_s'(x_s) - \sum_{\ell \in s} C_\ell'(y_\ell)\right)$$

Let $W(x) = \sum_s U_s(x_s) - \sum_\ell C_\ell(y_\ell)$ (system welfare)
and $V(x) = -W(x)$

Then: $\nabla V(x) \cdot F(x) = -\sum_s \kappa_s(x_s)\left[\frac{\partial}{\partial x_s} W(x)\right]^2$

Theorem

For U_s strictly concave with $U_s'(0^+) = +\infty$,
C_ℓ convex, continuously differentiable,
[$\Rightarrow$ strict concavity and continuous differentiability of W]
$\kappa_s > 0$, continuous [$\Rightarrow$continuity of F]
$\exists x_s > 0$ s.t. $U_s'(x_s) < \sum_{\ell \in s} C_\ell'(x_s)$
[$\Rightarrow$ Max of W achieved at single point $x^* \in O := (0,\infty)^{\mathcal{S}}$]
Then "primal" dynamics converge to unique maximizer x^* of W

Reverse engineering TCP

Approx. $x_s \approx cwnd_s / T_s$ where T_s: packet round-trip time

Reverse engineering TCP

Approx. $x_s \approx cwnd_s / T_s$ where T_s: packet round-trip time

Approx. $\frac{d}{dt} cwnd_s \approx x_s(1/cwnd_s) - p(s)[cwnd_s/2]$
where $p(s)$: packet loss probability along path of s

Reverse engineering TCP

Approx. $x_s \approx cwnd_s / T_s$ where T_s: packet round-trip time

Approx. $\frac{d}{dt} cwnd_s \approx x_s(1/cwnd_s) - p(s)[cwnd_s/2]$
where $p(s)$: packet loss probability along path of s

Approx. $p(s) \approx \sum_{\ell \in s} p_\ell(y_\ell)$ for link packet loss prob. $p_\ell(y)$
[e.g. $p_\ell(y) = \max(0, 1 - C_\ell/y)$]

Reverse engineering TCP

Approx. $x_s \approx cwnd_s / T_s$ where T_s: packet round-trip time

Approx. $\frac{d}{dt} cwnd_s \approx x_s(1/cwnd_s) - p(s)[cwnd_s/2]$
where $p(s)$: packet loss probability along path of s

Approx. $p(s) \approx \sum_{\ell \in s} p_\ell(y_\ell)$ for link packet loss prob. $p_\ell(y)$
[e.g. $p_\ell(y) = \max(0, 1 - C_\ell/y)$]

$$\Rightarrow \dot{x}_s = \left(\frac{x_s^2}{2}\right)\left[\frac{2}{(x_s T_s)^2} - \sum_{\ell \in s} p_\ell(y_\ell)\right]$$

Reverse engineering TCP

Approx. $x_s \approx cwnd_s / T_s$ where T_s: packet round-trip time

Approx. $\frac{d}{dt} cwnd_s \approx x_s (1/cwnd_s) - p(s)[cwnd_s/2]$
where $p(s)$: packet loss probability along path of s

Approx. $p(s) \approx \sum_{\ell \in s} p_\ell(y_\ell)$ for link packet loss prob. $p_\ell(y)$
[e.g. $p_\ell(y) = \max(0, 1 - C_\ell/y)$]

$$\Rightarrow \dot{x}_s = \left(\frac{x_s^2}{2}\right)\left[\frac{2}{(x_s T_s)^2} - \sum_{\ell \in s} p_\ell(y_\ell)\right]$$

TCP implicitly runs primal alg. with utility function:
$U_s(x) = w_s x^{1-\alpha}/(1-\alpha)$ with $\alpha = 2$, $w_s = 2/T_s^2$

Reverse engineering TCP

Approx. $x_s \approx cwnd_s/T_s$ where T_s: packet round-trip time

Approx. $\frac{d}{dt}cwnd_s \approx x_s(1/cwnd_s) - p(s)[cwnd_s/2]$
where $p(s)$: packet loss probability along path of s

Approx. $p(s) \approx \sum_{\ell \in s} p_\ell(y_\ell)$ for link packet loss prob. $p_\ell(y)$
[e.g. $p_\ell(y) = \max(0, 1 - C_\ell/y)$]

$$\Rightarrow \dot{x}_s = \left(\frac{x_s^2}{2}\right)\left[\frac{2}{(x_s T_s)^2} - \sum_{\ell \in s} p_\ell(y_\ell)\right]$$

TCP implicitly runs primal alg. with utility function:
$U_s(x) = w_s x^{1-\alpha}/(1-\alpha)$ with $\alpha = 2$, $w_s = 2/T_s^2$
$\rightarrow$ Leads to (w, α)-fairness with suitable parameters

Reverse engineering TCP

Approx. $x_s \approx cwnd_s/T_s$ where T_s: packet round-trip time

Approx. $\frac{d}{dt}cwnd_s \approx x_s(1/cwnd_s) - p(s)[cwnd_s/2]$
where $p(s)$: packet loss probability along path of s

Approx. $p(s) \approx \sum_{\ell \in s} p_\ell(y_\ell)$ for link packet loss prob. $p_\ell(y)$
[e.g. $p_\ell(y) = \max(0, 1 - C_\ell/y)$]

$$\Rightarrow \dot{x}_s = \left(\frac{x_s^2}{2}\right)\left[\frac{2}{(x_s T_s)^2} - \sum_{\ell \in s} p_\ell(y_\ell)\right]$$

TCP implicitly runs primal alg. with utility function:
$U_s(x) = w_s x^{1-\alpha}/(1-\alpha)$ with $\alpha = 2$, $w_s = 2/T_s^2$
$\rightarrow$ Leads to (w, α)-fairness with suitable parameters
Can tweak congestion avoidance alg. if want e.g. proportional fairness ($\alpha = 1$) instead

Convex optimization: Lagrangian, duality, multipliers

Generic convex optimization program
For convex set $\mathcal{C}^0$, convex functions J, $f_\ell : \mathcal{C}^0 \to \mathbb{R}$,

$$\begin{array}{rcc} \text{Min} & J(x) & \\ \text{Over} & x \in \mathcal{C}^0 & (P) \\ \text{Such that} & \forall \ell \in \mathcal{L}, f_\ell(x) \leq 0 & \end{array}$$

Convex optimization: Lagrangian, duality, multipliers

Generic convex optimization program
For convex set $\mathcal{C}^0$, convex functions J, $f_\ell : \mathcal{C}^0 \to \mathbb{R}$,

$$\begin{array}{rcc} \text{Min} & J(x) & \\ \text{Over} & x \in \mathcal{C}^0 & \quad (P) \\ \text{Such that} & \forall \ell \in \mathcal{L}, f_\ell(x) \leq 0 & \end{array}$$

Associated Lagrangian $L(x, \lambda) := J(x) + \sum_\ell \lambda_\ell f_\ell(x)$,
$x \in \mathcal{C}^0, \ \lambda \geq 0$

Convex optimization: Lagrangian, duality, multipliers

Generic convex optimization program
For convex set $\mathcal{C}^0$, convex functions J, $f_\ell : \mathcal{C}^0 \to \mathbb{R}$,

$$\begin{array}{rcc} \text{Min} & J(x) & \\ \text{Over} & x \in \mathcal{C}^0 & (P) \\ \text{Such that} & \forall \ell \in \mathcal{L}, f_\ell(x) \leq 0 & \end{array}$$

Associated Lagrangian $L(x,\lambda) := J(x) + \sum_\ell \lambda_\ell f_\ell(x)$,
$x \in \mathcal{C}^0, \ \lambda \geq 0$

λ: **Lagrange multipliers** of (P)'s constraints

Convex optimization: Lagrangian, duality, multipliers

Generic convex optimization program
For convex set $\mathcal{C}^0$, convex functions J, $f_\ell : \mathcal{C}^0 \to \mathbb{R}$,

$$\begin{array}{rcc} \text{Min} & J(x) & \\ \text{Over} & x \in \mathcal{C}^0 & (P) \\ \text{Such that} & \forall \ell \in \mathcal{L}, f_\ell(x) \leq 0 & \end{array}$$

Associated Lagrangian $L(x,\lambda) := J(x) + \sum_\ell \lambda_\ell f_\ell(x)$,
$x \in \mathcal{C}^0, \ \lambda \geq 0$

λ: **Lagrange multipliers** of (P)'s constraints

Dual problem (D): Max $D(\lambda)$ Over $\lambda \geq 0$
where $D(\lambda) := \inf_{x \in \mathcal{C}^0} L(x,\lambda)$

Kuhn-Tucker theorem and strong duality

Def: $\lambda^* \geq 0$ a Kuhn-Tucker vector iff $\forall x \in \mathcal{C}^0, L(x, \lambda^*) \geq J^*$ where J^*: optimal value of (P).

Kuhn-Tucker theorem and strong duality

Def: $\lambda^* \geq 0$ a Kuhn-Tucker vector iff $\forall x \in \mathcal{C}^0, L(x, \lambda^*) \geq J^*$ where J^*: optimal value of (P).
Remark: $J^* \geq D^*$ where D^* optimal value of (D)

Kuhn-Tucker theorem and strong duality

Def: $\lambda^* \geq 0$ a Kuhn-Tucker vector iff $\forall x \in \mathcal{C}^0, L(x, \lambda^*) \geq J^*$
where J^*: optimal value of (P).
Remark: $J^* \geq D^*$ where D^* optimal value of (D)

Theorem

Assume there exists λ^* a Kuhn-Tucker vector. Then
(i) λ^* solves (D), and $J^* = D^*$ (a.k.a. ***strong duality****)*
(ii) $x^* \in \mathcal{C}^0$ if optimal for (P) then achieves $\min_{x \in \mathcal{C}^0} L(x, \lambda^*)$
(iii) For $x^* int(\mathcal{C}^0)$ an optimum of (P) at which $\exists \nabla J, \nabla f_\ell$, then

$$\forall \ell, \lambda^*_\ell f_\ell(x^*) = 0 \quad \textbf{(complementarity)}$$
$$\nabla J(x^*) + \sum_\ell \lambda^*_\ell f_\ell(x^*) = 0 \quad \textbf{(stationarity)}$$

Reciprocally assume stationarity + complementarity
for some $\lambda^* \geq 0$ and some x^* feasible for (P),
Then λ^*: Kuhn-Tucker and x^* optimal for (P)

Sufficient conditions for applying Kuhn-Tucker

Lemma

Assume $J^* > -\infty$ *and* $\exists \hat{x} \in \mathcal{C}^0$ *such that* $\forall \ell,\ f_\ell(\hat{x}) < 0$.
Then a Kuhn-Tucker vector λ^* *exists.*

In practice: verify Lemma's conditions + existence of optimum $x^* \in \text{int}(\mathcal{C}^0)$ at which $\exists \nabla J,\ \nabla f_\ell$.
Then find x^* that verifies complementarity + stationarity (now guaranteed to exist)

Solving original problem: dual algorithm

Lagrangian: $L(x,\lambda) = \sum_s U_s(x_s) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} x_s]$

Solving original problem: dual algorithm

Lagrangian: $L(x,\lambda) = \sum_s U_s(x_s) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} x_s]$

Dual: $D(\lambda) = \sum_s U_s(g_s(\lambda^s)) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} g_s(\lambda^s)]$

where $\lambda^s := \sum_{\ell \in s} \lambda_s$ and $g_s := (U_s')^{-1}$

Solving original problem: dual algorithm

Lagrangian: $L(x,\lambda) = \sum_s U_s(x_s) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} x_s]$

Dual: $D(\lambda) = \sum_s U_s(g_s(\lambda^s)) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} g_s(\lambda^s)]$

where $\lambda^s := \sum_{\ell \in s} \lambda_s$ and $g_s := (U_s')^{-1}$

$\Rightarrow \frac{\partial}{\partial \lambda_s} D(\lambda) = C_\ell - \sum_{s \ni \ell} g_s(\lambda^s)$

Solving original problem: dual algorithm

Lagrangian: $L(x,\lambda) = \sum_s U_s(x_s) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} x_s]$

Dual: $D(\lambda) = \sum_s U_s(g_s(\lambda^s)) + \sum_\ell \lambda_\ell [C_\ell - \sum_{s \ni \ell} g_s(\lambda^s)]$

where $\lambda^s := \sum_{\ell \in s} \lambda_s$ and $g_s := (U_s')^{-1}$

$\Rightarrow \frac{\partial}{\partial \lambda_s} D(\lambda) = C_\ell - \sum_{s \ni \ell} g_s(\lambda^s)$

Dual algorithm:
$$x_s \equiv g_s(\lambda^s),$$
$$\dot{\lambda}_\ell = \kappa_\ell \left[\sum_{s \ni \ell} x_s - C_\ell \right]^+_{\lambda_\ell}$$

where $[a]_b^+ = a$ if $b > 0$, $\max(a, 0)$ if $b \le 0$

Solving original problem: dual algorithm

Theorem

Under suitable conditions
(U_s strictly concave, twice differentiable, $U_s'(0^+) = +\infty$, $U_s'(+\infty) = 0$)
Trajectories x_s of dual algorithm converge to unique maximizer x^* of primal problem.

Solving original problem: dual algorithm

Theorem

Under suitable conditions
(U_s strictly concave, twice differentiable, $U_s'(0^+) = +\infty$,
$U_s'(+\infty) = 0$)
Trajectories x_s of dual algorithm converge to unique maximizer x^* of primal problem.

[Proof: involved. Quasiproof: Lyapunov function argument]

Solving original problem: dual algorithm

Theorem

Under suitable conditions
(U_s strictly concave, twice differentiable, $U_s'(0^+) = +\infty$, $U_s'(+\infty) = 0$)
Trajectories x_s of dual algorithm converge to unique maximizer x^* of primal problem.

[Proof: involved. Quasiproof: Lyapunov function argument]

Potential implementation: multiplier dynamics $\equiv$ queue dynamics

Solving original problem: dual algorithm

Theorem

Under suitable conditions
(U_s strictly concave, twice differentiable, $U_s'(0^+) = +\infty$, $U_s'(+\infty) = 0$)
Trajectories x_s of dual algorithm converge to unique maximizer x^* of primal problem.

[Proof: involved. Quasiproof: Lyapunov function argument]

Potential implementation: multiplier dynamics $\equiv$ queue dynamics
$\Rightarrow$ Let λ^ℓ = queueing delay of packets and instantaneously let x_s to $g_s(\lambda^s)$

Solving original problem: dual algorithm

Theorem

Under suitable conditions
(U_s strictly concave, twice differentiable, $U_s'(0^+) = +\infty$,
$U_s'(+\infty) = 0$)
Trajectories x_s of dual algorithm converge to unique maximizer x^* of primal problem.

[Proof: involved. Quasiproof: Lyapunov function argument]

Potential implementation: multiplier dynamics $\equiv$ queue dynamics
$\Rightarrow$ Let λ^ℓ = queueing delay of packets and instantaneously let x_s to $g_s(\lambda^s)$
$\Rightarrow$ Principle underlying TCP-Vegas, an alternative to default TCP (TCP Reno)

Takeaway messages

- For unconstrained convex minimization, gradient descent converges to optimizer [Lyapunov stability]
- Admits distributed implementation in network optimization setting
- TCP implicitly achieves (w, α)-fair allocation by running gradient descent
- Kuhn-Tucker Theorem: Complementarity + Stationarity characterization of (P)'s optima
- Queue dynamics implicitly perform gradient descent for multipliers of constrained program

Takeaway messages

- For unconstrained convex minimization, gradient descent converges to optimizer [Lyapunov stability]
- Admits distributed implementation in network optimization setting
- TCP implicitly achieves (w, α)-fair allocation by running gradient descent
- Kuhn-Tucker Theorem: Complementarity + Stationarity characterization of (P)'s optima
- Queue dynamics implicitly perform gradient descent for multipliers of constrained program

Remaining question: How to discriminate between allocation objectives?