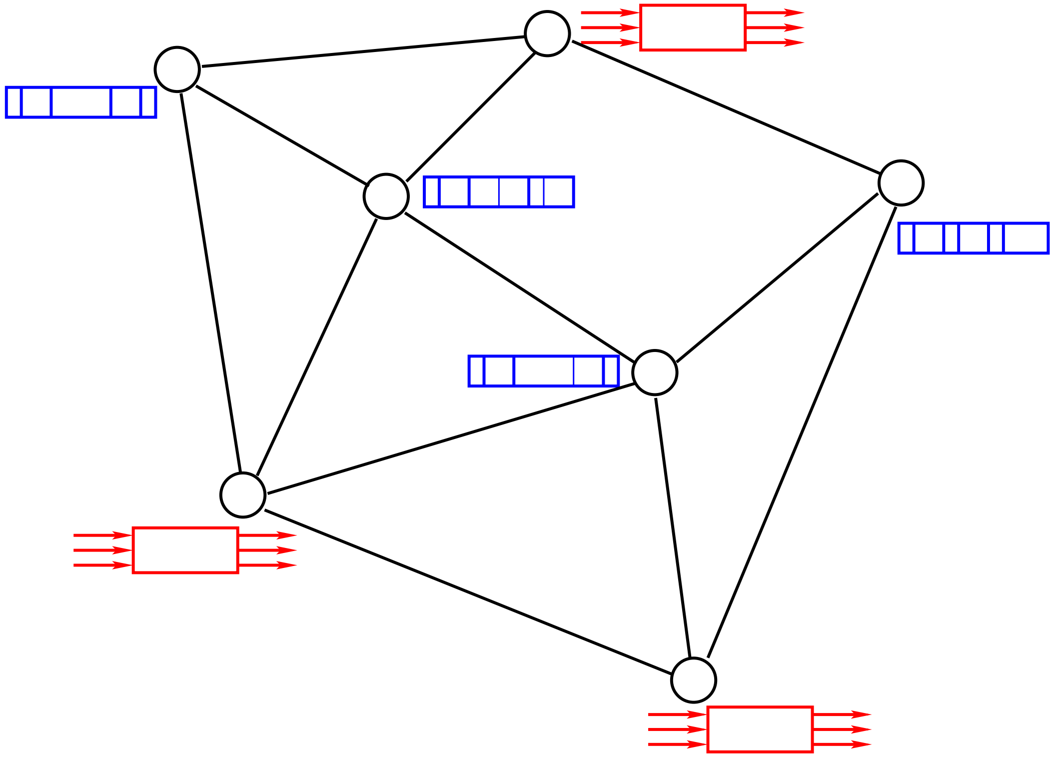
Approximation Algorithms for Network Problems

Susanne Albers
University of Freiburg
Germany

Large networks



Buffer management in switches

Online, competitive analysis

A., Schmidt STOC'04

Web caching, request reordering

Offline, approx. algorithms

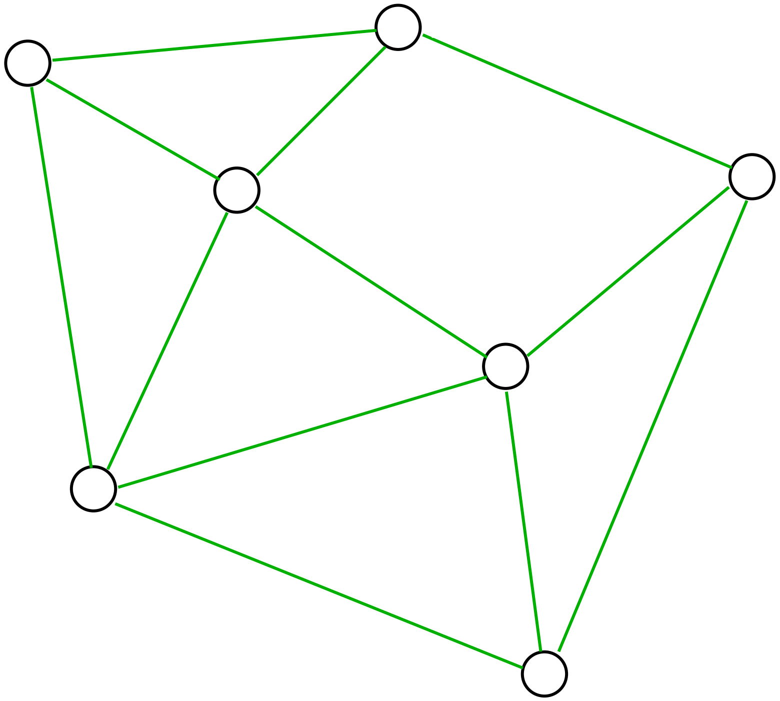
A. SPAA'04

Network creation game

Nash equilibria, price of anarchy

A. 05

Large networks



Buffer management in switches

Online, competitive analysis

A., Schmidt STOC'04

Web caching, request reordering

Offline, approx. algorithms

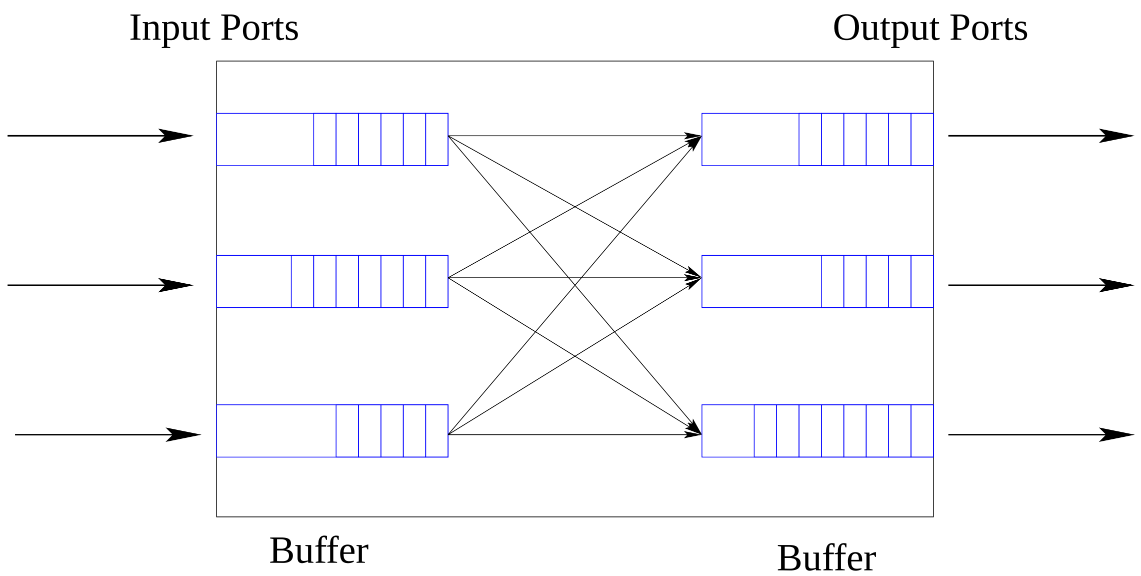
A. SPAA'04

Network creation game

Nash equilibria, price of anarchy

A. 05

Buffer management in switches

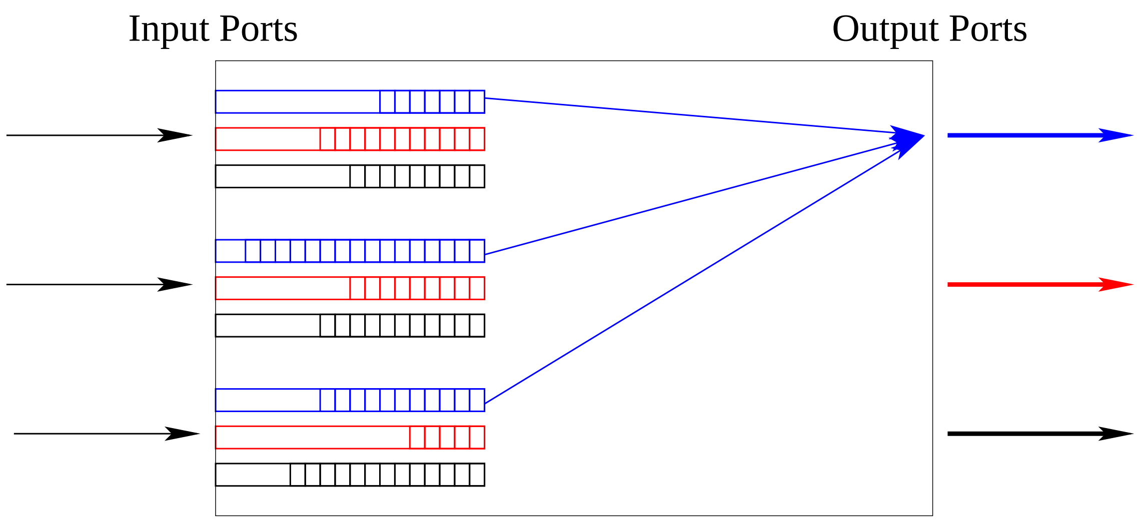


Switches forward data packets.

Buffers store packets temporarily if capacity available.

Goal: maximize throughput.

Virtual output queueing



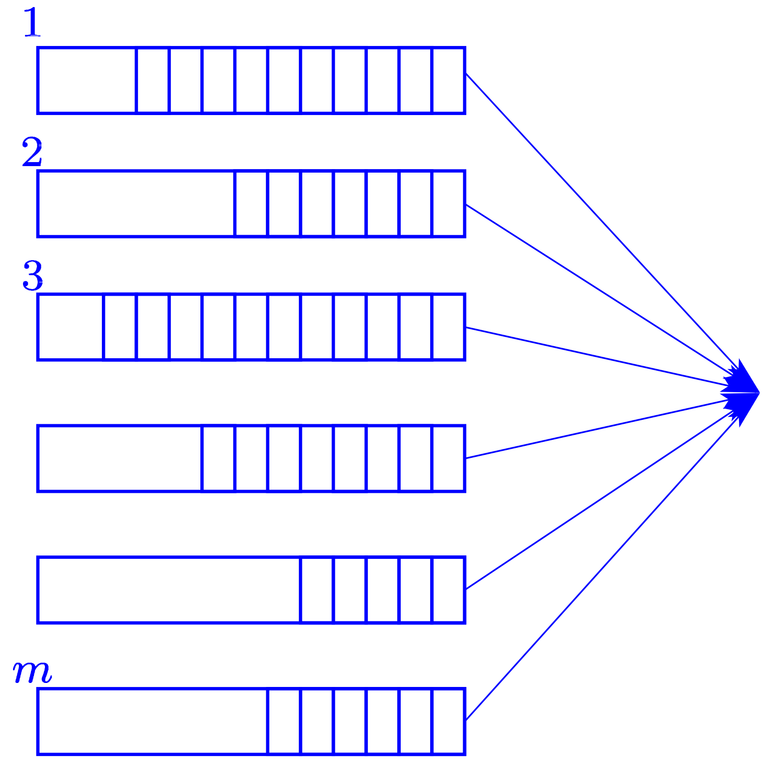
Each input port i maintains for each output port j a queue Q_{ij}.

Problem

m buffers, each of which can store
B pakets.

In each time step
– new packets arrive online
P_i: #packets in buffer i
N_i: #new packets at buffer i
paket loss: $\max\{N_i + P_i - B,\ 0\}$
– one buffer can send one paket
to the output

Goal: maximize #transferred pakets



Competitive analysis

Online problem

A: Online algorithm	OPT: Offline algorithm
$A(\sigma)$	$OPT(\sigma)$

A is c-competitive if $\exists a$ such that for all sequences σ

$$A(\sigma) \geq \frac{1}{c} \cdot OPT(\sigma) \; - a.$$

Previous results

- Every reasonable algorithm is 2-competitive.
- Randomized upper bound: $e/(e-1) \approx 1.58$
  Azar, Richter 2003
- Lower bounds
  Deterministic: 1.366
  Randomized: 1.46 ($B = 1$)
  Azar, Richter 2003
- Single buffer problems: pakets have values
  Upper bounds: 2, 1.75
  Kesselman et al. 2001; Bansal et al. 2004

Greedy algorithms

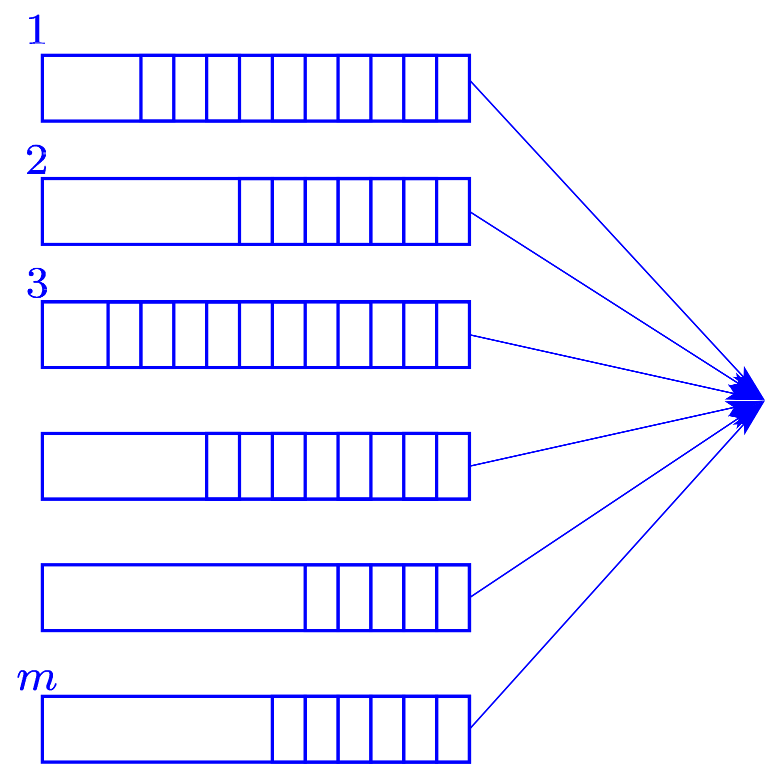
Greedy: Always serve a buffer currently storing a maximum number of packets.

Advantages:

– fast

– little extra memory

– best strategy to avoid packet loss



Our results

- Exact performance of all Greedy algorithms: 2-competitive
- New algorithm Semi-Greedy: $17/9 \approx 1.89$
  fast, little extra memory, serves full buffers
- Lower bounds (B arbitrary)
  Deterministic: $e/(e-1) \approx 1.58$
  Randomized: 1.46
- Extra resources: larger buffers, higher transmission rates
  Almost matching upper and lower bounds
- Optimal offline algorithm running in polynomial time

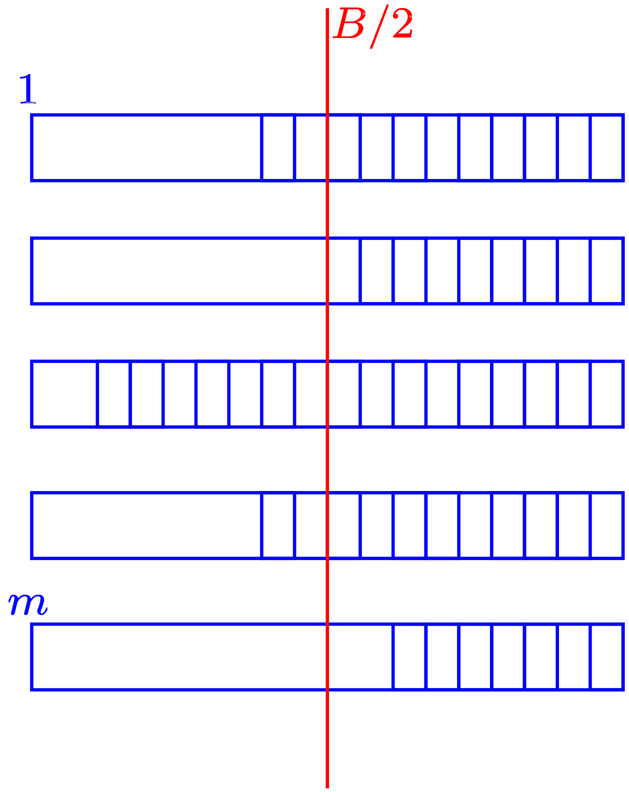
Semi-Greedy

In each time step execute the first applicable rule.

1. $\exists$ buffer with $> B/2$ packets
   $\rightarrow$ serve a buffer with
   max. number of packets

2. $\exists$ non-empty buffer that has never been full
   $\rightarrow$ amongst these, serve one with
   max. number of packets

3. Serve a buffer with max. number of packets



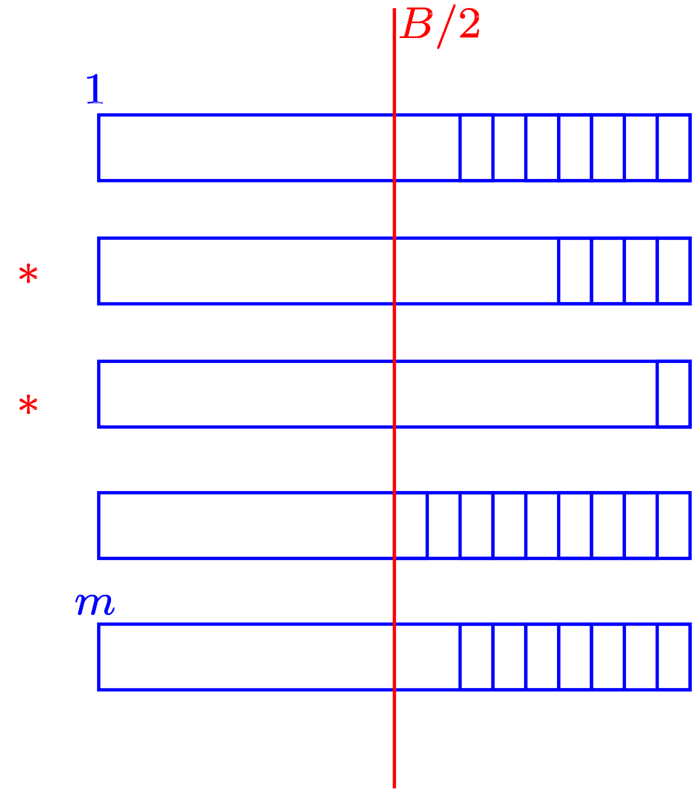
Semi-Greedy

In each time step execute the first applicable rule.

1. $\exists$ buffer with $> B/2$ packets
   $\rightarrow$ serve a buffer with
   max. number of packets

2. $\exists$ non-empty buffer that has never been full
   $\rightarrow$ amongst these, serve one with
   max. number of packets

3. Serve a buffer with max. number of packets



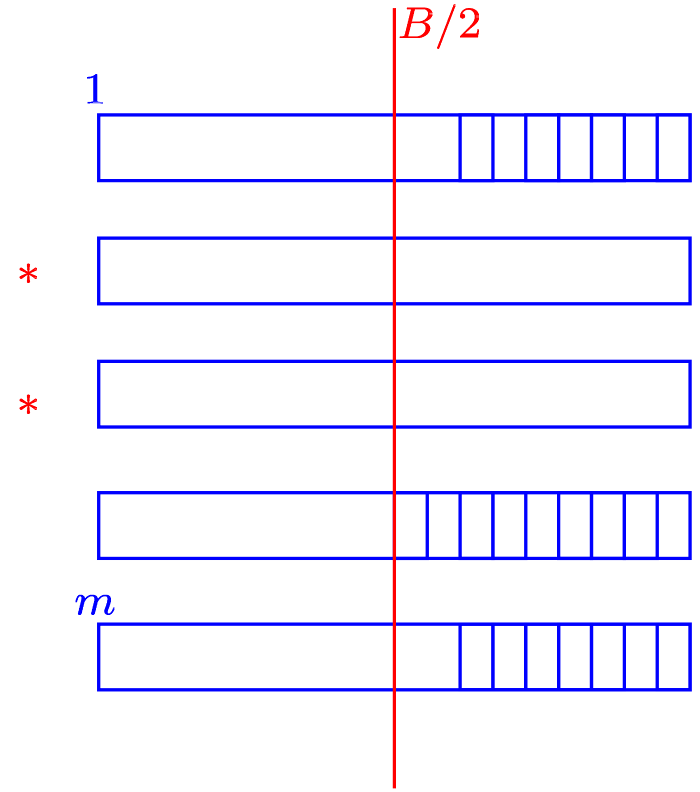
Semi-Greedy

In each time step execute the first applicable rule.

1. $\exists$ buffer with $> B/2$ packets
   $\rightarrow$ serve a buffer with max. number of packets

2. $\exists$ non-empty buffer that has never been full
   $\rightarrow$ amongst these, serve one with max. number of packets

3. Serve a buffer with max. number of packets



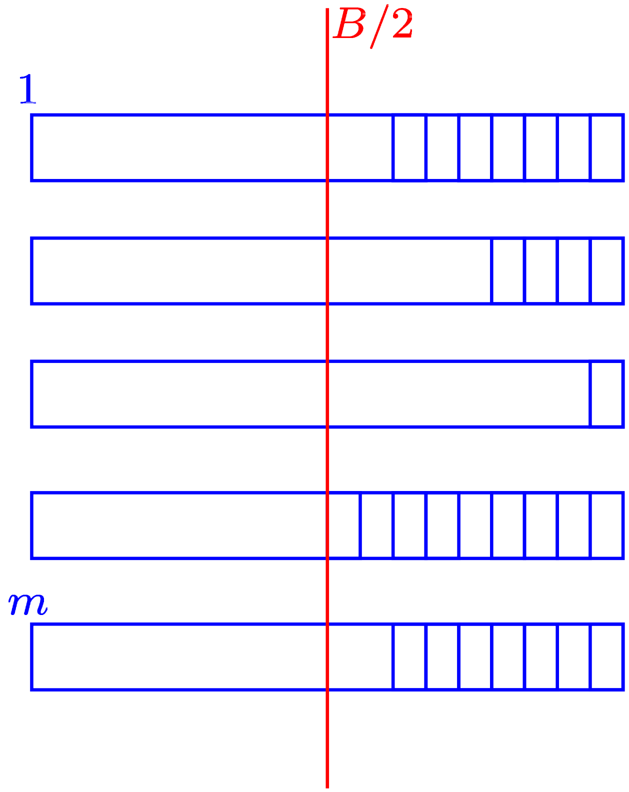
Semi-Greedy

In each time step execute the first applicable rule.

1. $\exists$ buffer with $> B/2$ packets
   $\rightarrow$ serve a buffer with max. number of packets

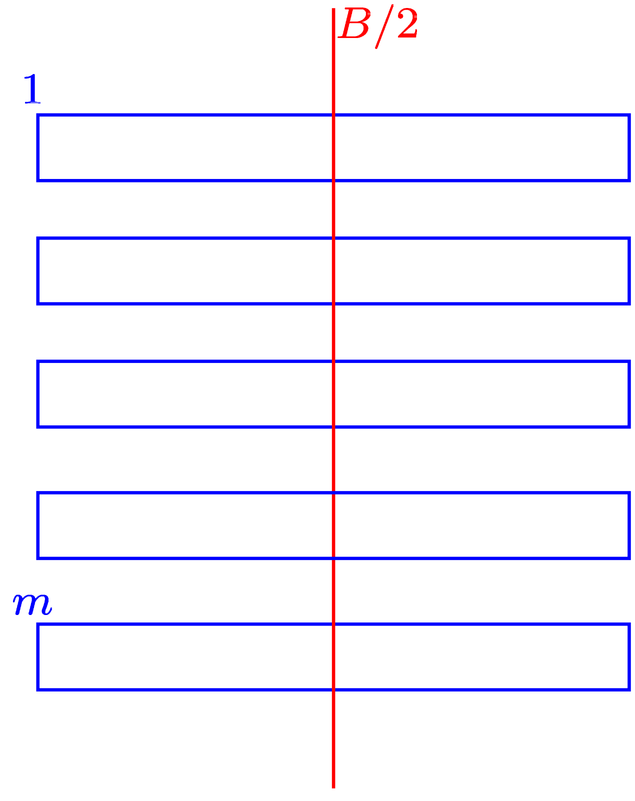
2. $\exists$ non-empty buffer that has never been full
   $\rightarrow$ amongst these, serve one with max. number of packets

3. Serve a buffer with max. number of packets



Semi-Greedy

Whenever all buffers are empty, the hitherto maximum load of each queue is set to 0.



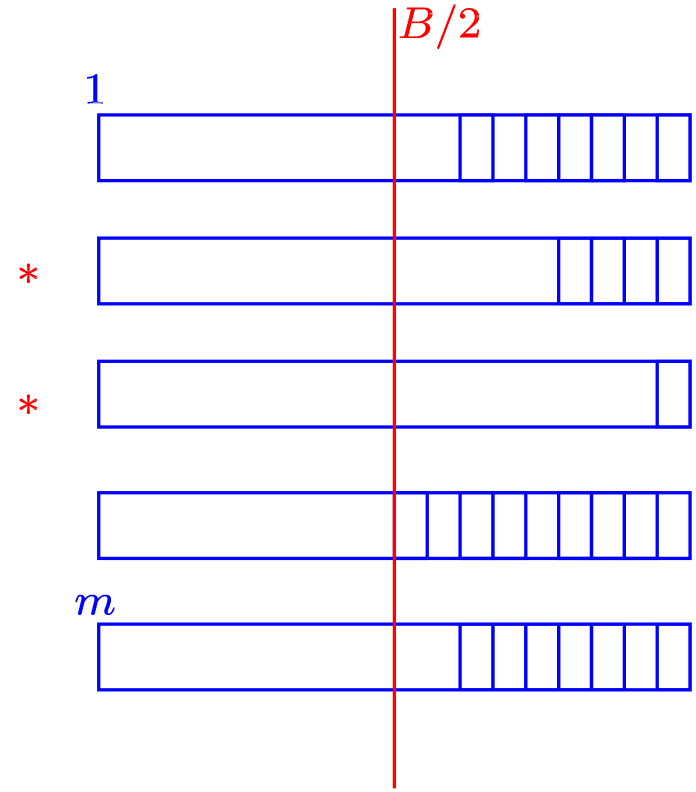
Semi-Greedy

In each time step execute the first applicable rule.

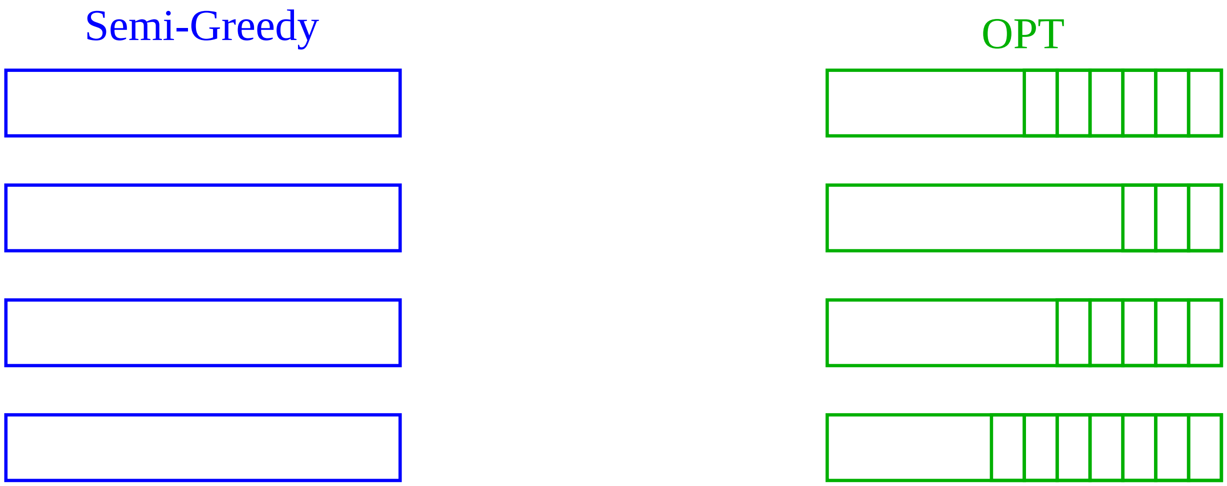
1. $\exists$ buffer with $> B/2$ packets
   $\rightarrow$ serve a buffer with max. number of packets

2. $\exists$ non-empty buffer that has never been full
   $\rightarrow$ amongst these, serve one with max. number of packets

3. Serve a buffer with max. number of packets



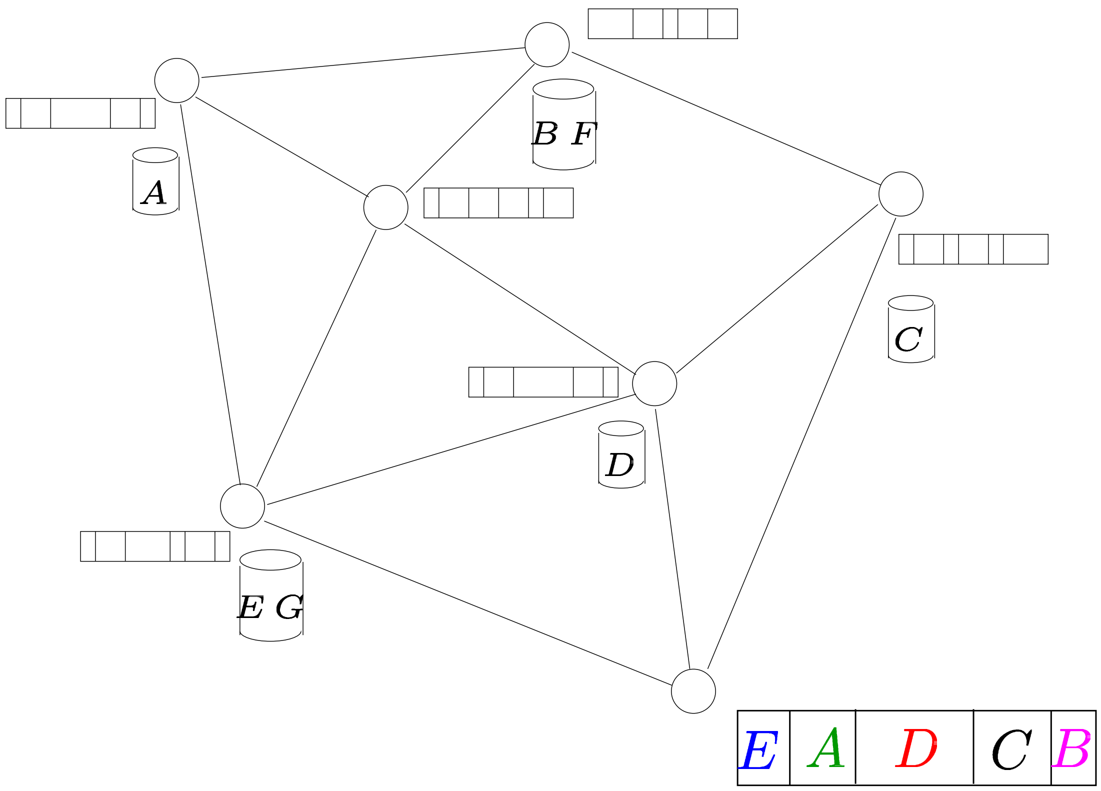
Analysis



Partition input into subsequences so that at the end of each subsequence Semi-Greedy's buffers are empty.

Compare: throughput Semi-Greedy / throughput OPT

Web caching



Documents are text files, images, html pages, . . .

Important properties:

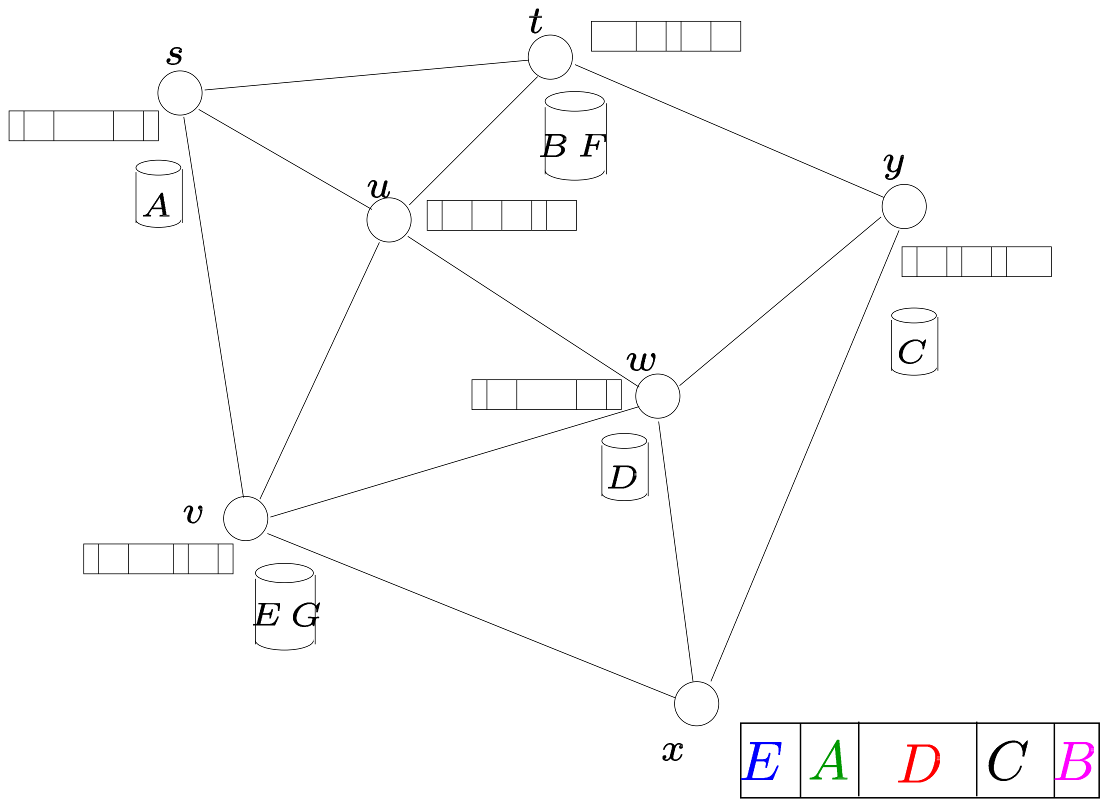
documents have different sizes and incur different costs

Web caching

Request: (x, D) x requests documents D

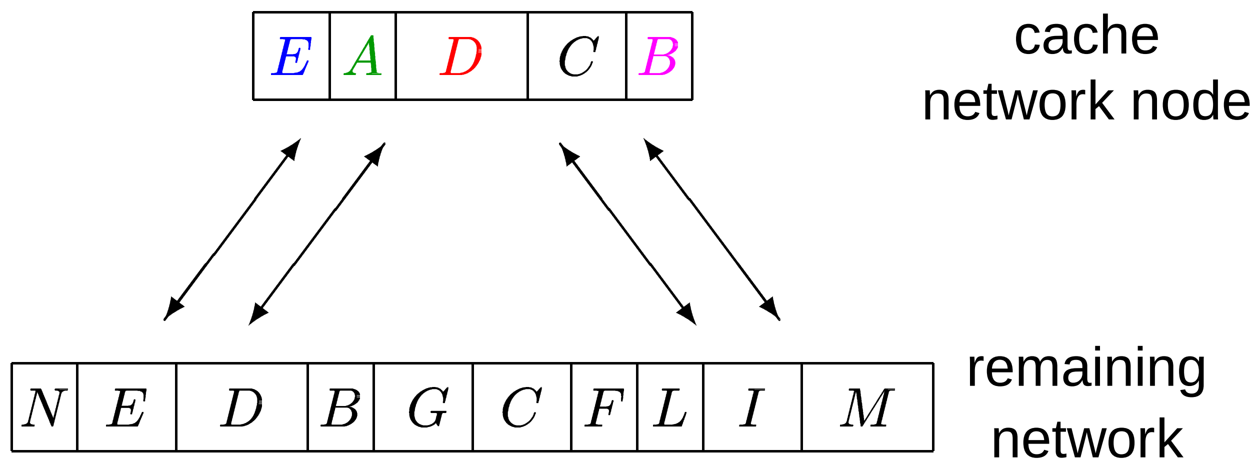
D not in x's cache: Cost(D)
Optional: loading D

Goal: Minimize total service cost



$$\sigma = (x, D)\ (y, E)\ (z, A)\ (v, C)\ (w, B)\ (x, F) \ldots$$

Web caching

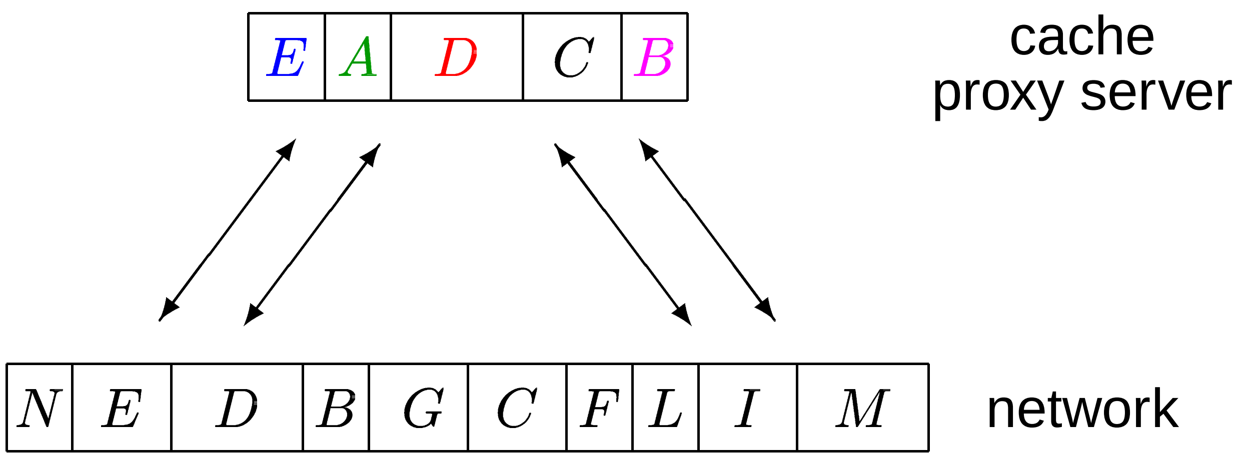


$$\sigma = D\ C\ A\ B\ C\ F\ E\ C\ N\ I\ L\ M\ B\ E\ A\ L\ I\ F$$

Goal: Serve a sequence of requests so that the total service cost at the node is minimized.

Request reordering

Proxy server: requests are independent



$$\sigma = D\ C\ A\ B\ C\ \underbrace{F\ E\ E\ A\ N}_{r}\ L\ M\ B\ E\ A\ L\ I\ F$$

$\sigma(j)$ may be served before $\sigma(i)$ if $j - i < r$ $\qquad r \in \mathbb{N}$

Advantage: improved cache hit rates

Feder, Motwani, Panigrahy, Zhu 2002

Cost models

Document D Size(D) Cost(D)

Uniform Model:

Cost(D) $=$ Size(D) $= 1$

Bit Model:

Cost(D) $=$ Size(D)

Fault Model:

Cost(D) $= 1$

General Model:

Cost(D) arbitrary

Previous results, reordering

Online

Uniform Model: $(K/s+2)$-competitive (deterministic)

Bit and Fault Models: $(K/s+3)$-competitive (deterministic)

Offline

General Model: Polynomial algorithm for cache size 1 if

r logarithmic in $|\sigma|$ or $\#$distinct documents is constant

K = size cache $\quad s$ = size smallest document

Feder, Motwani, Panigrahy, Seiden, van Stee, Zhu 2003

Our results

Online

General Model: optimal $(K/s+1)$-competitive alg. (deterministic)

Offline

	Approximation	Extra memory	$S = \max$ Size
Uniform Model:	2	$-$	
Bit Model:	$2+\epsilon$	$\frac{1}{(1+\epsilon/2)}S$	$\epsilon \geq 0$
Fault Model:	$2+\epsilon$	$(1+2/\epsilon)S$	
General Model:	8	$-$	

Approach: reduce problem to one of computing batched schedules.

Batched processing

$$\sigma = \underbrace{\sigma(1)\ \ldots\ \sigma(r)}_{B_1}\ \sigma(r+1)\ \ldots\ \sigma(2r)\ \sigma(2r+1)\ \ldots\ \sigma(3r)\ldots$$

Batch i

$$B_i = \sigma(ir+1)\ldots\sigma(ir+r)$$

Batched processing

$$\sigma = \sigma(1)\ \ldots\ \sigma(r)\ \underbrace{\sigma(r+1)\ \ldots\ \sigma(2r)}_{B_2}\ \sigma(2r+1)\ \ldots\ \sigma(3r)\ldots$$

Batch i

$$B_i = \sigma(ir+1)\ldots\sigma(ir+r)$$

Batched processing

$$\sigma = \sigma(1)\ \ldots\ \sigma(r)\ \ \sigma(r+1)\ \ldots\ \sigma(2r)\ \ \underbrace{\sigma(2r+1)\ \ldots\ \sigma(3r)}_{B_3}\ldots$$

Batch i

$$B_i = \sigma(ir+1)\ldots\sigma(ir+r)$$

Batched processing

Lemma: Suppose that A serves σ with cost C.
Then there exists A' that processes σ in batches and incurs a cost of at most $2C$.

Uniform Model

$$\sigma = \ldots\ldots \underbrace{\sigma(ir+1)\ \ldots\ \sigma(ir+r)}_{B_i}\ \ldots\ldots$$

Algorithm BMIN

1. Serve requests to documents in cache;
2. **while** $\exists D \in B_i$ with unserved requests **do**

   Serve requests to D;

   Determine E in cache whose next unserved request is farthest in future;

   if next unserved request to E is in a later batch than that to D **then**

      Load D by evicting E;

Uniform Model

Algorithm BMIN

1. Serve requests to documents in cache;
2. **while** $\exists D \in B_i$ with unserved requests **do**
   Serve requests to D;
   Determine E in cache whose next unserved request is farthest in future;
   if next unserved request to E is in a later batch than that to D **then**
   Load D by evicting E;

$$\sigma = \ldots\ldots\ \underbrace{A\ B\ A\ A\ D}_{B_i}\ \ldots\ldots\ \underbrace{A\ B\ B\ D\ D}_{B_j}\ \ldots\ldots \underbrace{B\ A\ B\ B\ E}_{B_k}$$

$$\begin{array}{|c|}\hline A \\ \hline B \\ \hline E \\ \hline \end{array}$$

Uniform Model

Algorithm BMIN

1. Serve requests to documents in cache;
2. **while** $\exists D \in B_i$ with unserved requests **do**

   Serve requests to D;

   Determine E in cache whose next unserved request is farthest in future;

   if next unserved request to E is in a later batch than that to D **then**

   Load D by evicting E;

$$\sigma = \ldots\ldots \underbrace{A\ B\ A\ A\ D}_{B_i} \ldots\ldots \underbrace{A\ B\ B\ D\ E}_{B_j} \ldots\ldots \underbrace{B\ A\ B\ B\ E}_{B_k}$$

$$\begin{array}{|c|}\hline A \\ \hline B \\ \hline E \\ \hline\end{array}$$

Uniform Model

Lemma: BMIN is optimal among algorithms processing request sequences in batches.

Theorem: BMIN achieves an approximation ratio of 2.

Approximations

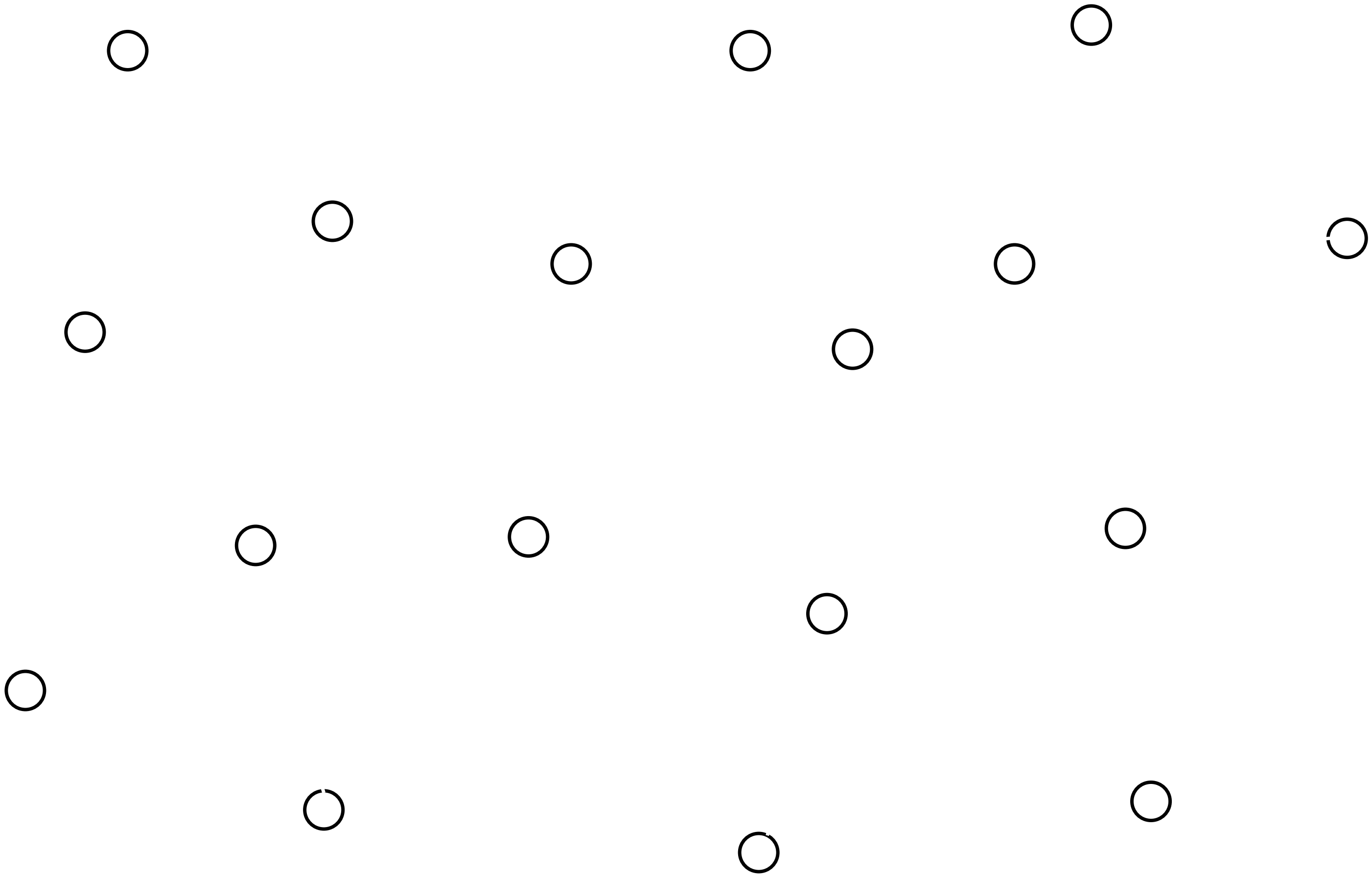
Construct schedules that serve σ in batches

Bit, Fault Models: Formulate problems as ILP.

General Model: Formulate problem as a loss minimization problem.

Bar-Noy, Bar-Yehuda, Freund, Naor, Schieber 2001

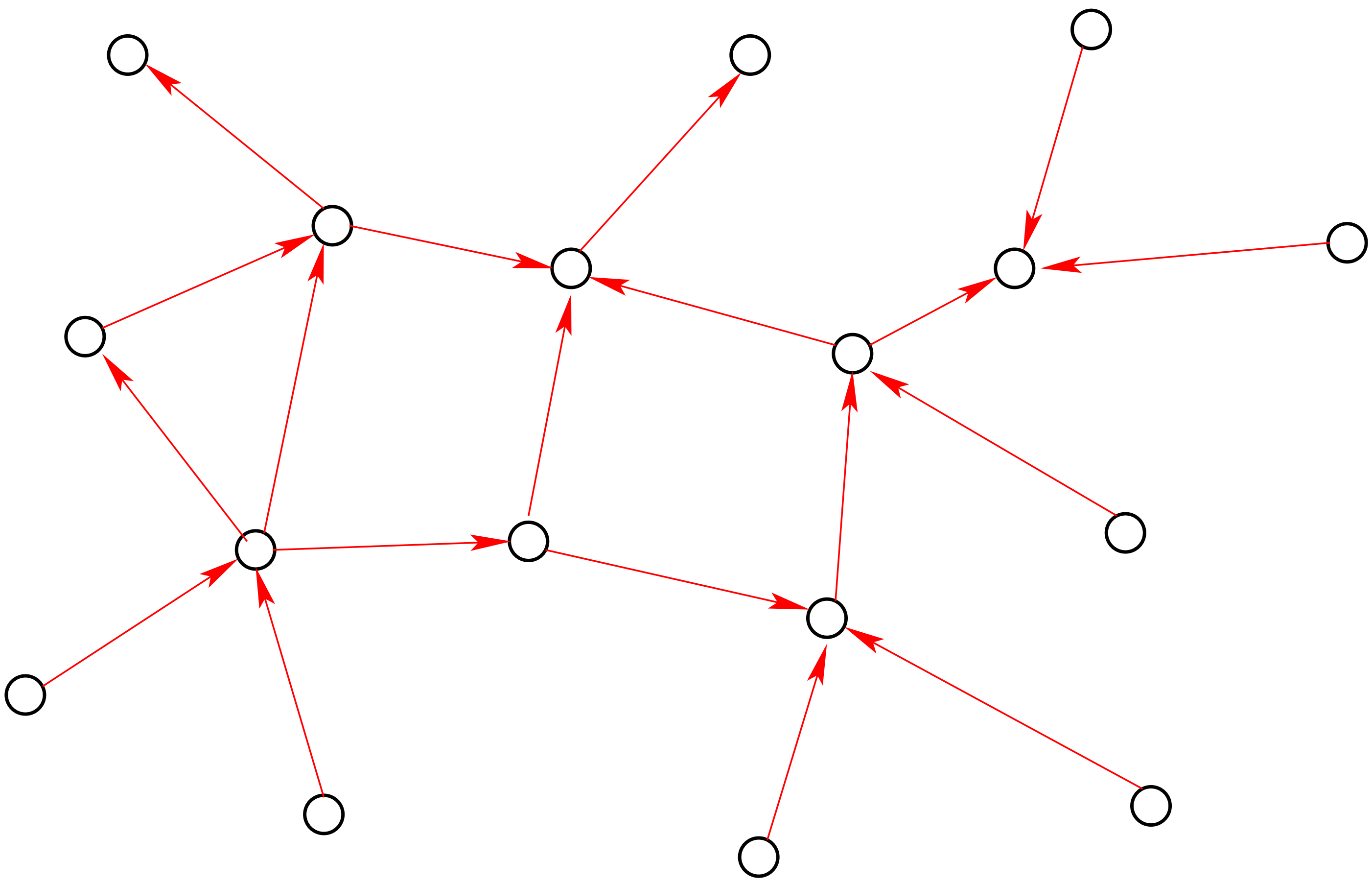
Network creation game



n agents have to build a connected network.

Fabrikant, Lutha, Maneva, Papadimitriou, Shenker PODC'03

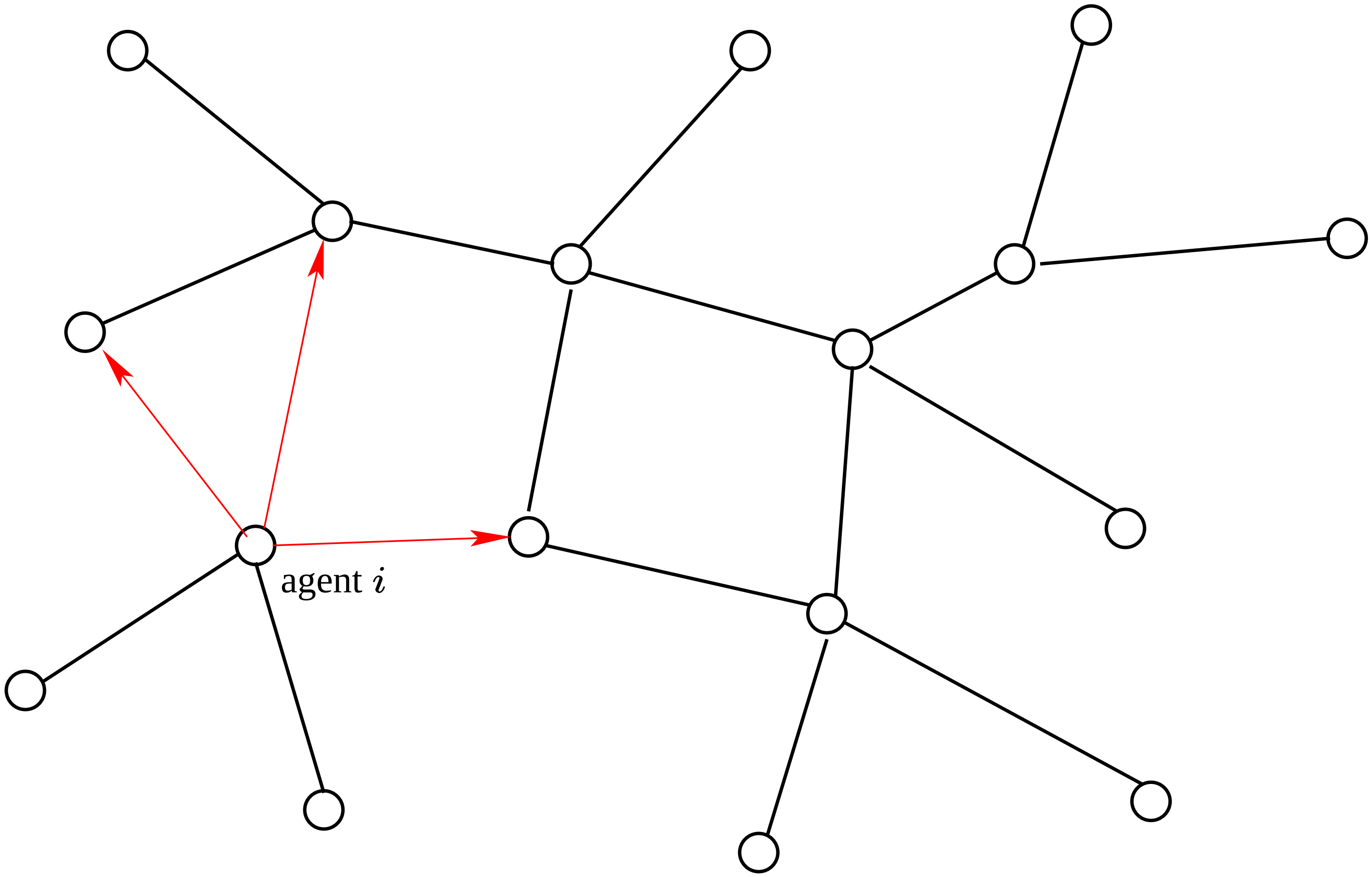
Network creation game



n agents have to build a connected network.

Fabrikant, Lutha, Maneva, Papadimitriou, Shenker PODC'03

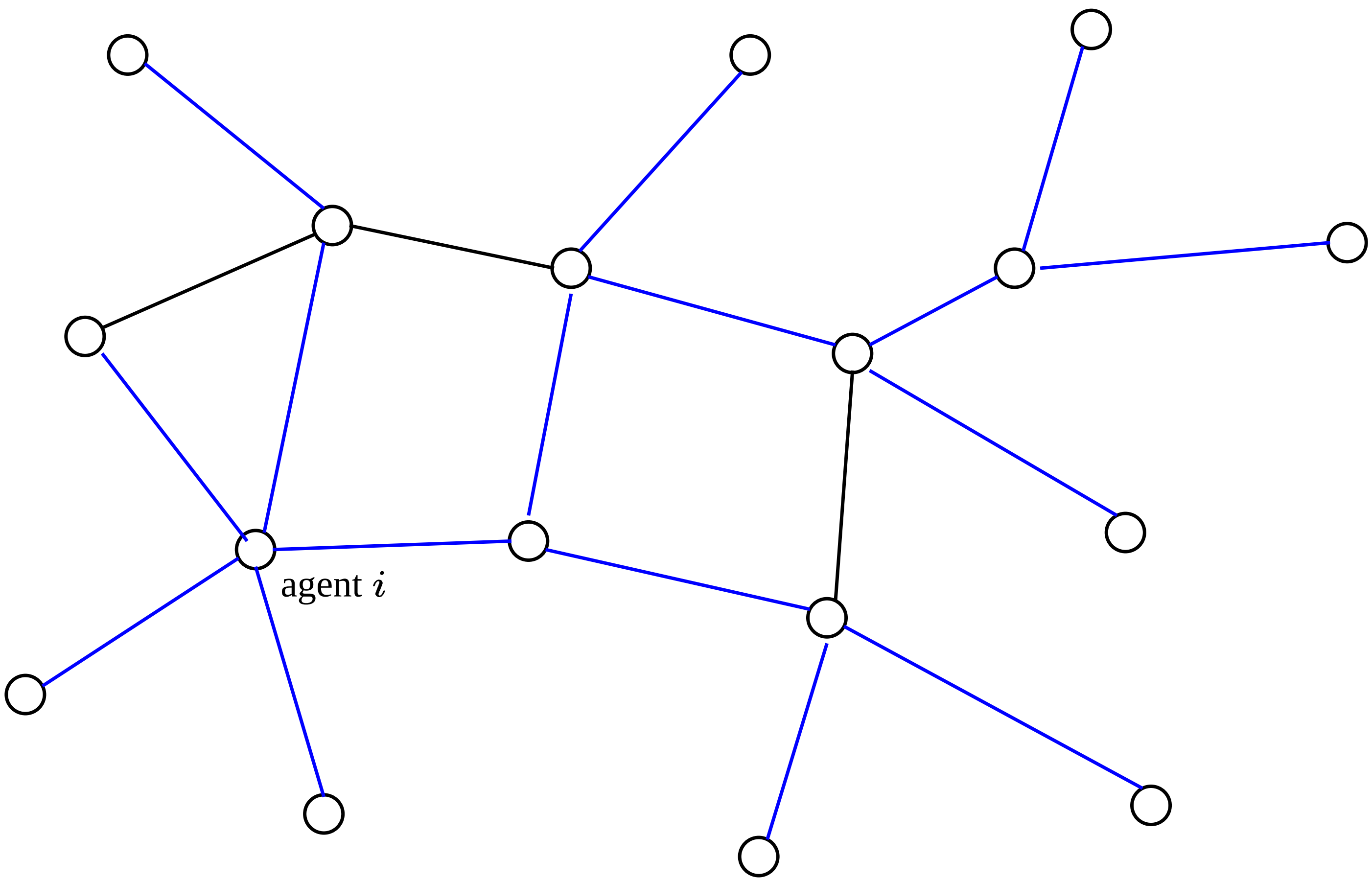
Network creation game



Cost of $\alpha \geq 0$ for each edge.

Fabrikant, Lutha, Maneva, Papadimitriou, Shenker PODC'03

Network creation game



Shortest path distance to agent j, for all $j \neq i$.

Fabrikant, Lutha, Maneva, Papadimitriou, Shenker PODC'03

Problem

n agents $\quad \alpha \geq 0$

$$\text{Cost}(\text{agent } i) = \alpha \ \#\text{edges built by agent } i + \sum_{j \neq i} \text{shortest path distance to agent } j$$

Nash equilibria

No agent can improve its cost if other agents keep their strategies.

Price of anarchy:

$$P = \max_{\text{Nash eq.}} \frac{\sum_i \text{Cost}(\text{agent } i)}{\text{Cost(OPT)}}$$

Koutsoupias, Papadimitriou ’99

Previous results

Fabrikant, Lutha, Maneva, Papadimitriou, Shenker PODC'03

$\alpha \leq 1, \quad \alpha > n^2$

P is constant

$1 < \alpha \leq n^2$

P is bounded by $\sqrt{\alpha}$

Tree-conjecture: $\exists\ C$ s.t. for $\alpha > C$ every Nash equilibrium is a tree.

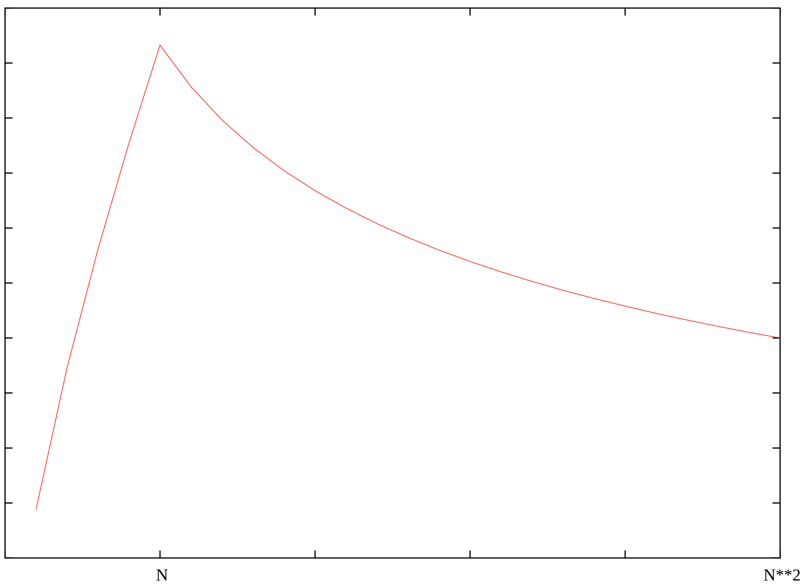
Our results

$\alpha > 0$: $P = O(1 + (\min\{\frac{\alpha^2}{n}, \frac{n^2}{\alpha}\})^{1/3})$

$\alpha \leq \sqrt{n}$: P is constant

$\sqrt{n} < \alpha \leq n$: P increasing, bounded by $n^{1/3}$

$n < \alpha$: P decreasing, constant for $\alpha \geq n^2$



Our results

Upper bounds can be extended to:

Weighted game: $t_{ij} =$ traffic sent from agent i to j

Cost sharing: agent can pay for a fraction of an edge

For any n and $n/4 \leq \alpha \leq n/3$

$\exists$ Nash equilibria that contain cycles.

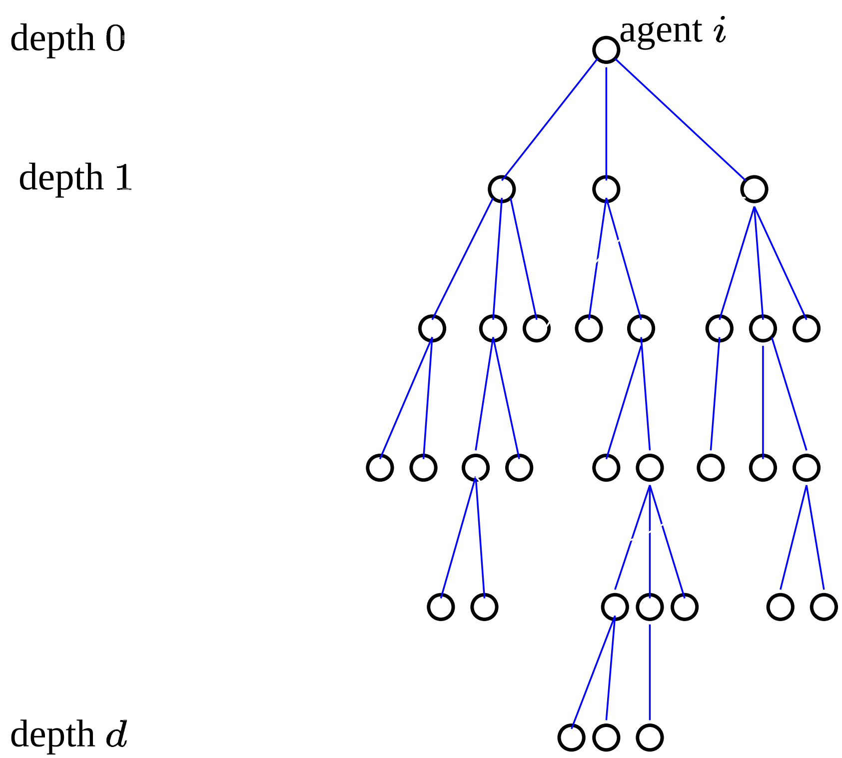
Transient: $\exists$ seq. of players' changes leading to non-equilibrium state.

Nash equilibrium representing a chordal graph is transient.

Upper bound

Nash equilibrium $N \quad G = (V, E)$

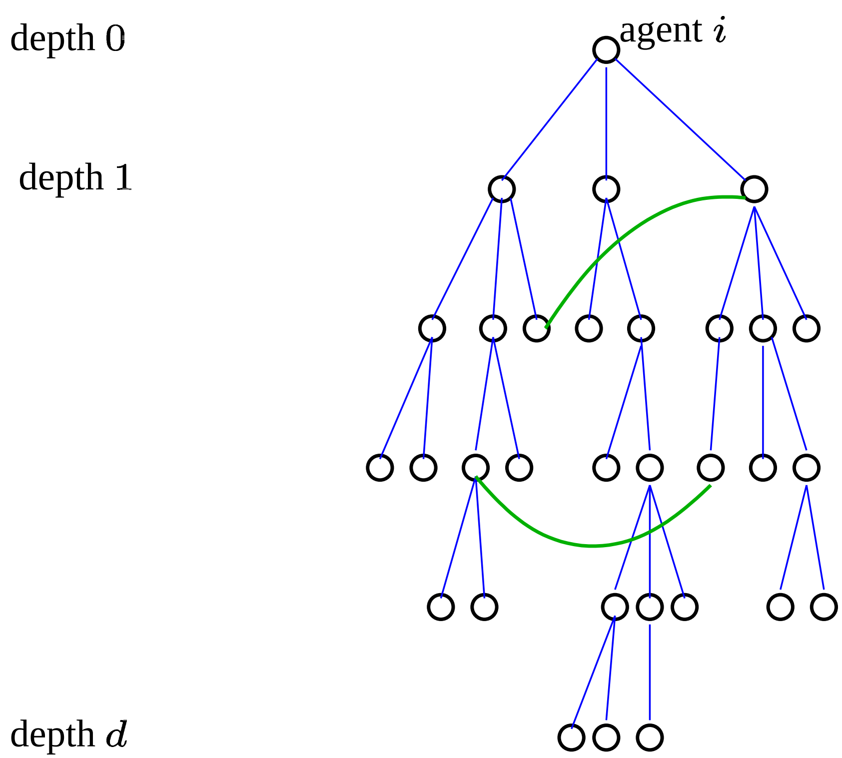
Shortest path tree rooted at agent i



Upper bound

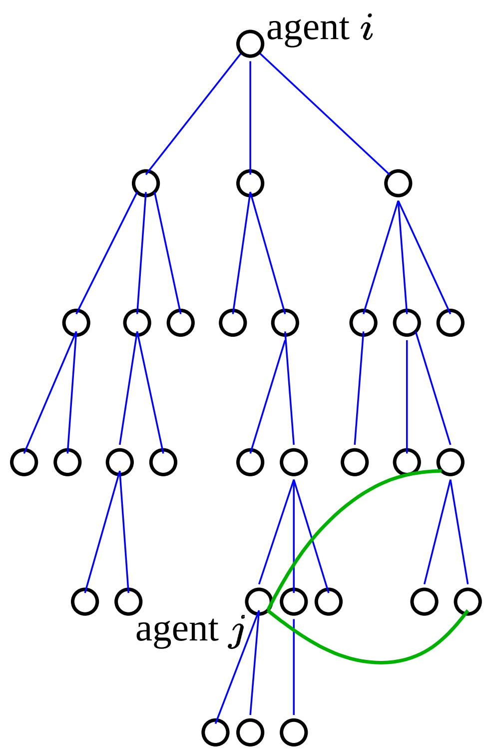
Nash equilibrium N $\quad G = (V, E)$

Shortest path tree rooted at agent i

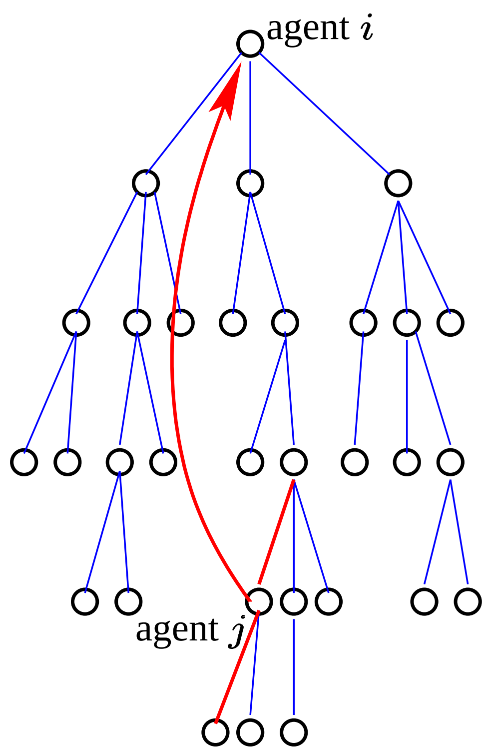


Cost(agent i) $\leq \alpha T_i + d(n-1)$ $\qquad$ $T_i =$ #tree edges built by agent i

Cost of agent j

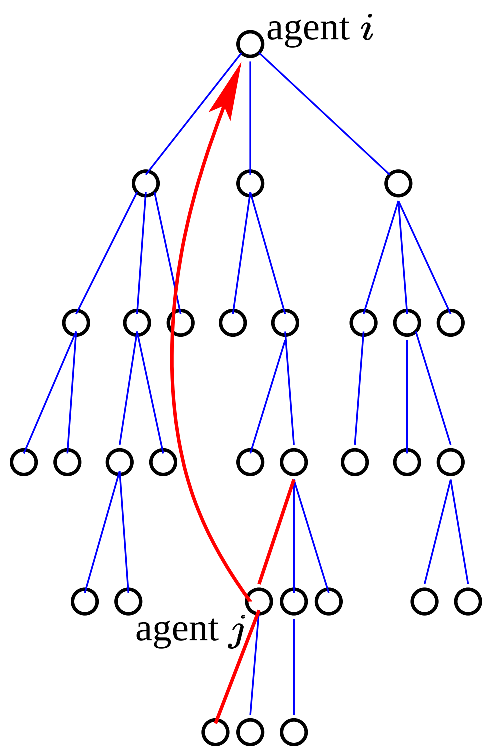


Cost of agent j



$\alpha T_j + \alpha$ + $(d+1)(n-1)$

Cost of agent j



Cost(agent j) $\leq$ $\alpha T_j + \alpha$ + $(d+1)(n-1)$

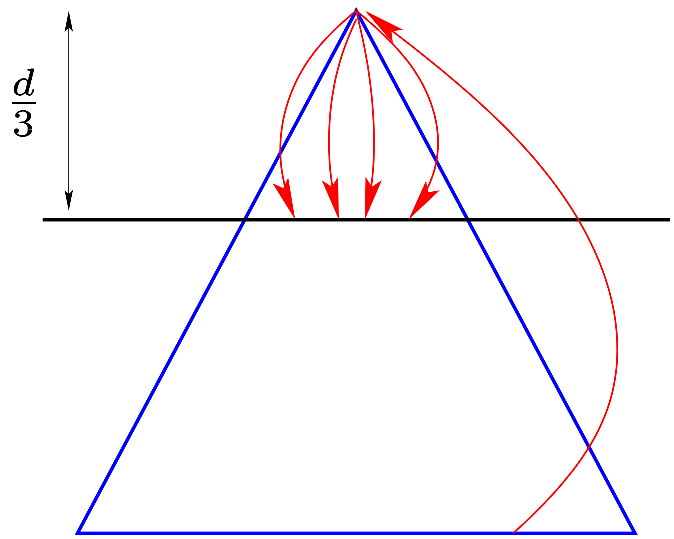
Cost Nash

$$\text{Cost(agent } i) \leq \alpha T_i + d(n-1)$$

$$\text{Cost(agent } j) \leq \alpha T_j + \alpha + (d+1)(n-1)$$

$$\text{Cost(Nash eq.)} \leq \alpha(n-1) + \alpha(n-1) + (d+1)(n-1)n$$

Analysis d



$$d \leq \frac{3\alpha}{n^c} \qquad\qquad c = \frac{1}{3}\left(\frac{\log \alpha}{\log n} + 1\right)$$

Price of anarchy

Cost(Nash eq.) $\leq 2\alpha(n-1) + \left(\frac{3\alpha}{n^c} + 1\right)(n-1)n$

Cost(OPT) $\geq \alpha(n-1) + n(n-1)$

Open problems

Buffer management:

Determine competitiveness of randomized algorithms.

Packets have limited lifeliness.

Web caching

Improve approximations guarantees.

Complexity in the Uniform, Fault Models.

Network creation

Settle price of anarchy of any α.

Study other network creation games.