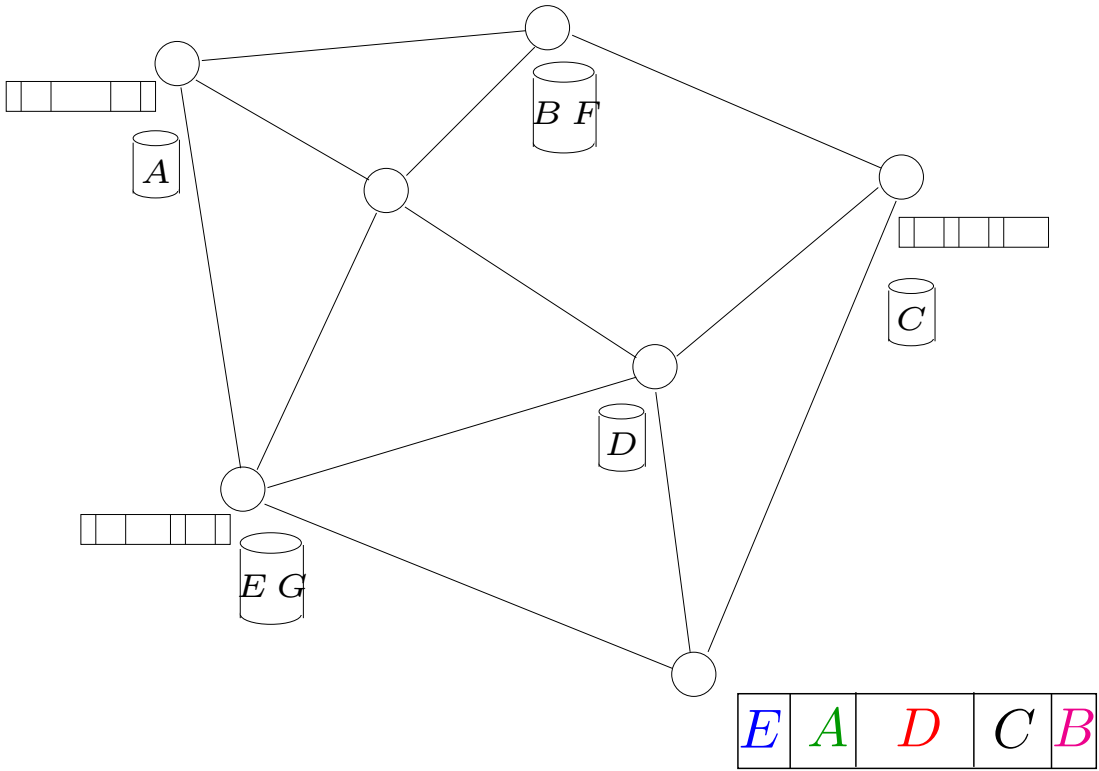


Caching Problems

Document & Connection Caching

Susanne Albers

Web caching



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

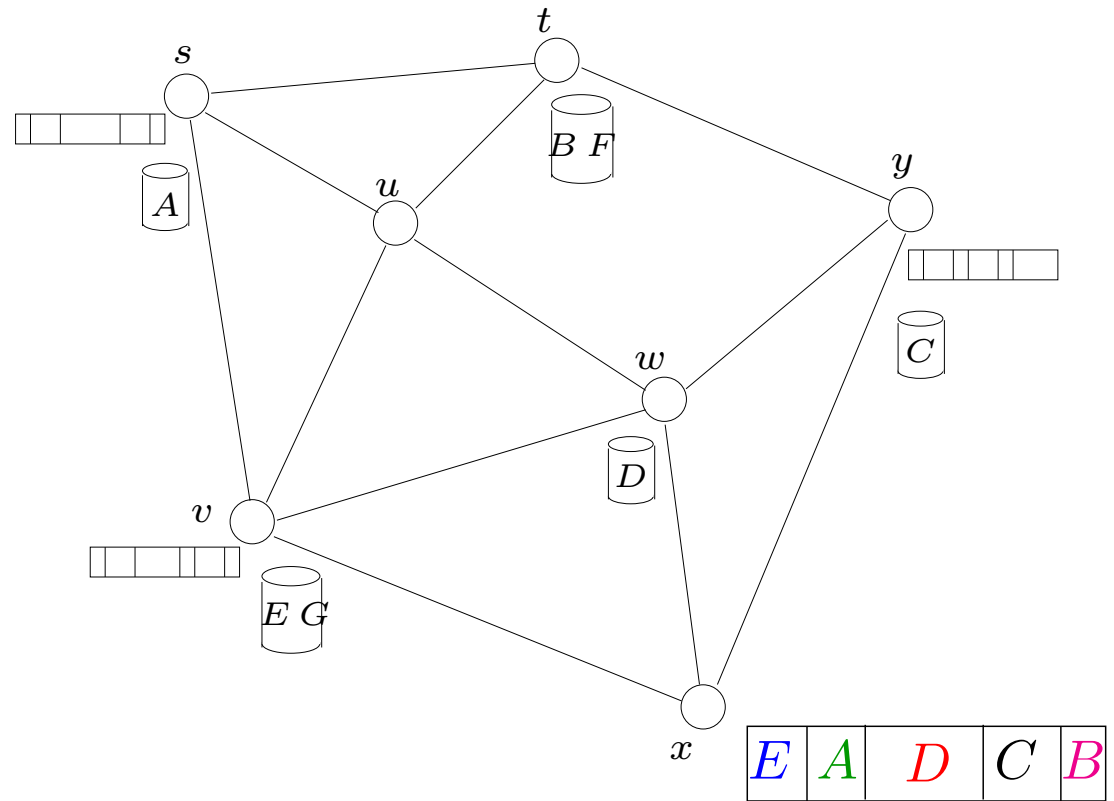
Important properties:
documents have
different sizes and incur
different costs

Web caching

Request: (x, D) x requests documents D

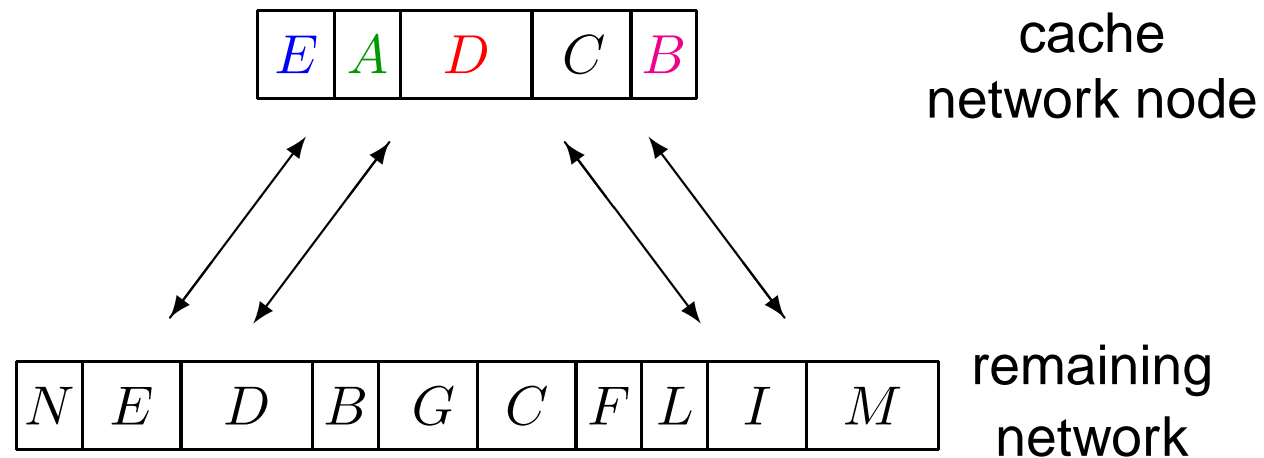
D not in x 's cache: Cost(D)

Goal: Minimize total service cost



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

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.

Cost models

Document D Size(D) Cost(D)

Bit Model:

$$\text{Cost}(D) = \text{Size}(D)$$

Fault Model:

$$\text{Cost}(D) = 1$$

General Model:

Cost(D) arbitrary

Results

k = size of the cache

Bit, Fault Models:

LRU is k -competitive

$O(\log^2 k)$ -competitive randomized alg.

General Model:

k -competitive (deterministic) alg. *Landlord*

Young 2002; Irani 2002

Proof technique

$$\sigma = \sigma(1) \sigma(2) \sigma(3) \dots \sigma(m)$$

$$A(\sigma)$$

$$OPT(\sigma)$$

Potential function Φ

$$\Phi(t) \geq 0 \quad \Phi(0) = 0$$

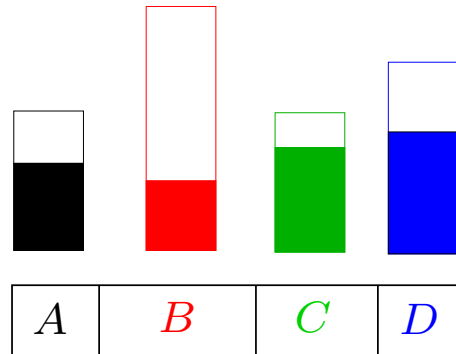
$$A(\sigma(t)) + \underbrace{\Phi(t) - \Phi(t-1)}_{\Delta\Phi} \leq c \cdot OPT(\sigma(t))$$

$$\Rightarrow \sum_{t=1}^m A(\sigma(t)) + \Phi(m) - \Phi(0) \leq c \cdot \sum_{t=1}^m OPT(\sigma(t))$$

$$\Leftrightarrow A(\sigma) \leq c \cdot OPT(\sigma) - \Phi(m)$$

A is c -competitive.

Landlord



D in cache: $\text{Credit}(D)$

if E is requested, E not in cache **then**

repeat

$\delta := \min_{D \in \text{cache}} \text{Credit}(D) / \text{Size}(D);$

For each D in cache:

$\text{Credit}(D) := \text{Credit}(D) - \delta \cdot \text{Size}(D);$

Evict D with $\text{Credit}(D) = 0$

until there is room to load E ;

$\text{Credit}(E) := \text{Cost}(E);$

Analysis

$$\Phi = (k - 1) \sum_{D \in S_{LL}} \text{Credit}(D) + k \sum_{D \in S_{OPT}} (\text{Cost}(D) - \text{Credit}(D))$$

1. **OPT** loads document E at cost $\text{Cost}(E) \implies \Delta\Phi \leq k \cdot \text{Cost}(E)$
2. **LL** loads document at cost $\text{Cost}(E) \implies \Delta\Phi \leq -\text{Cost}(E)$
3. At all other times $\Delta\Phi \leq 0$

1.

a) **OPT** evicts D :

$$\Delta\Phi \leq 0$$

b) **OPT** loads E :

$$\text{Loading cost} = \text{Cost}(E), \Delta\Phi \leq k\text{Cost}(E)$$

Analysis, step 2

a) LL decreases $\text{Credit}(D)$ for all $D \in S_{LL}$

$$\Delta\Phi \leq -\delta((k-1)\text{Size}(S_{LL}) - k \cdot \text{Size}(S_{OPT} \cap S_{LL}))$$

$$\text{Size}(S_{LL}) \geq k - \text{Size}(E) + 1$$

$$\text{Size}(S_{OPT} \cap S_{LL}) \leq k - \text{Size}(E) \text{ since } E \in S_{OPT}$$

$$\begin{aligned} \Delta\Phi &\leq -\delta((k-1)(k - \text{Size}(E) + 1) - k(k - \text{Size}(E))) \\ &\leq -\delta(-(k - \text{Size}(E) + 1) + k) \leq 0 \end{aligned}$$

b) LL evicts D

$$\Delta\Phi = 0$$

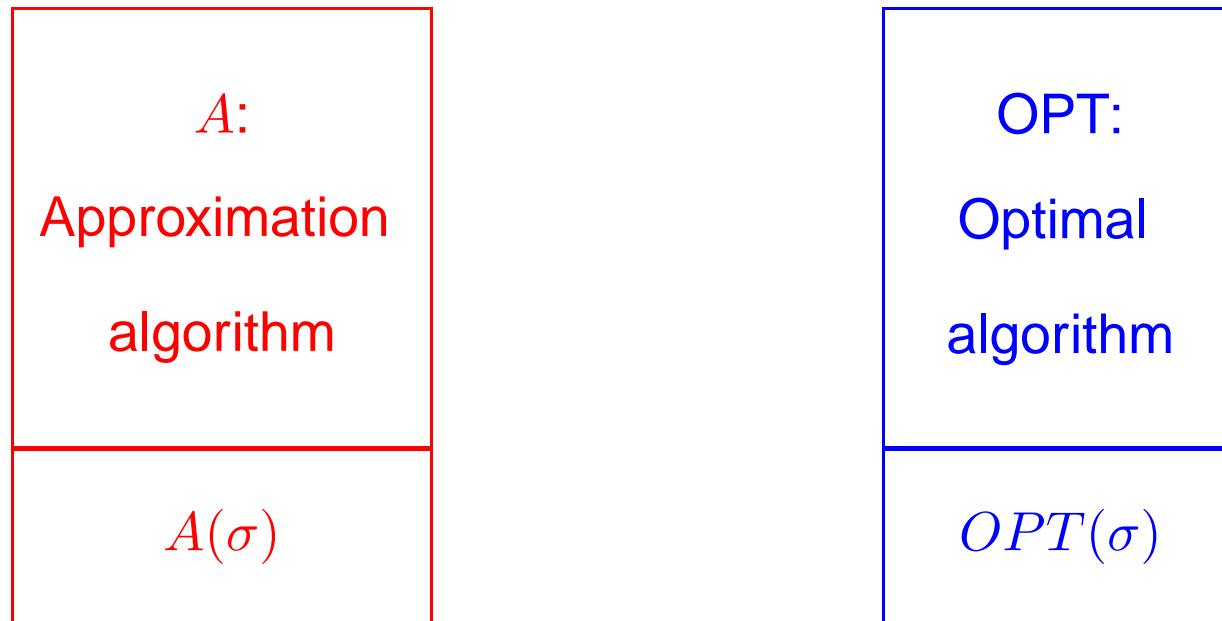
c) LL loads E , $\text{Credit}(E) := \text{Cost}(E)$

$$\text{Loading cost} = \text{Cost}(E)$$

$$\Delta\Phi \leq (k-1)\text{Cost}(E) - k \cdot \text{Cost}(E) = -\text{Cost}(E)$$

Approximations

Offline problem



A achieves approximation ratio of c if for all sequences σ

$$A(\sigma) \leq c \cdot OPT(\sigma).$$

Results

$$\epsilon > 0$$

Approximation ratio

Memory

$$c_1$$

$$k + c_2 S$$

Bit Model:

$$1 + \epsilon$$

$$1/(1 + \epsilon) \quad \epsilon \geq 0$$

Fault Model:

$$1 + \epsilon$$

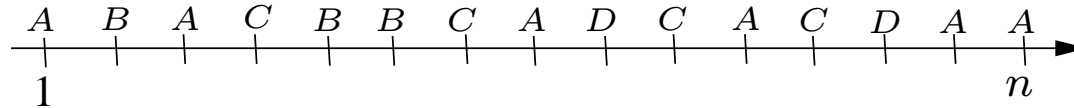
$$1 + 1/\epsilon$$

General Model:

$$4$$

Albers, Arora, Khanna 2000; Bar-Noy et al. 2001

Linear program



$$x_{D,t} = \begin{cases} 1 & D \text{ in cache at time } t \\ 0 & D \text{ not in cache at time } t \end{cases}$$

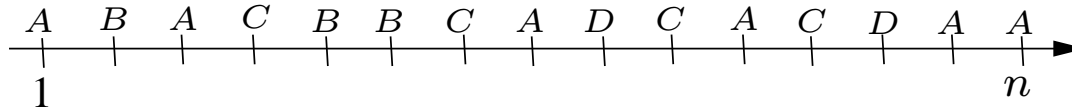
Missing documents are only loaded at the **next request**.

D_t = document requested at time t

Objective function

$$\min \sum_{t=1}^n \text{Cost}(D_t)(1 - x_{D_t,t-1})$$

Linear program



$$x_{D,t} = \begin{cases} 1 & D \text{ in cache at time } t \\ 0 & D \text{ not in cache at time } t \end{cases}$$

D_t = document requested at time t

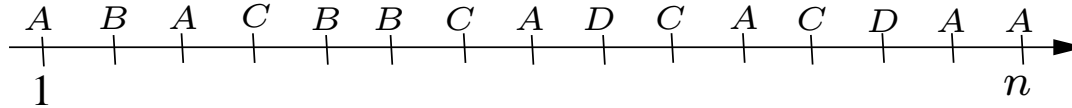
Objective function

$$\min \sum_{t=1}^n \text{Cost}(D_t)(1 - x_{D_t,t-1})$$

Constraint: cache capacity may not be violated

$$\sum_D \text{Size}(D)x_{D,t} \leq k \quad \forall t$$

Linear program



$$x_{D,t} = \begin{cases} 1 & D \text{ in cache at time } t \\ 0 & D \text{ not in cache at time } t \end{cases}$$

D_t = document requested at time t

Objective function

$$\min \sum_{t=1}^n \text{Cost}(D_t)(1 - x_{D_t,t-1})$$

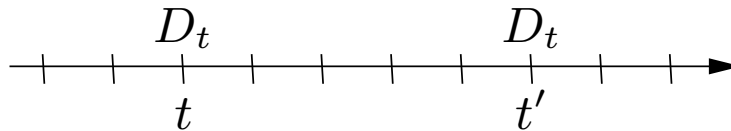
Constraint: requested document must be in cache

$$x_{D_t,t} = 1 \quad \forall t$$

Linear program

Objective function

$$\min \sum_{t=1}^n \text{Cost}(D_t)(1 - x_{D_t,t-1})$$



Extent to which D_t is in cache does not change in (t, t') .

$$I_t = \{t + 1, \dots, t' - 1\}$$

Constraint:

$$x_{D_t,t} = x_{D_t,s} \quad \forall t, s \in I_t$$

Linear program

minimize $\sum_{t=1}^n \text{Cost}(D_t)(1 - x_{D_t,t-1})$

s. t.

$$\sum_D \text{Size}(D)x_{D,t} \leq k \quad \forall t$$

$$x_{D_t,t} = 1 \quad \forall t$$

$$x_{D_t,t} = x_{D_t,s} \quad \forall t, s \in I_t$$

$$x_{D,t} \in \{0, 1\} \quad \forall D, t$$

Linear program

minimize $\sum_{t=1}^n \text{Cost}(D_t)(1 - x_{D_t,t-1})$

s. t.

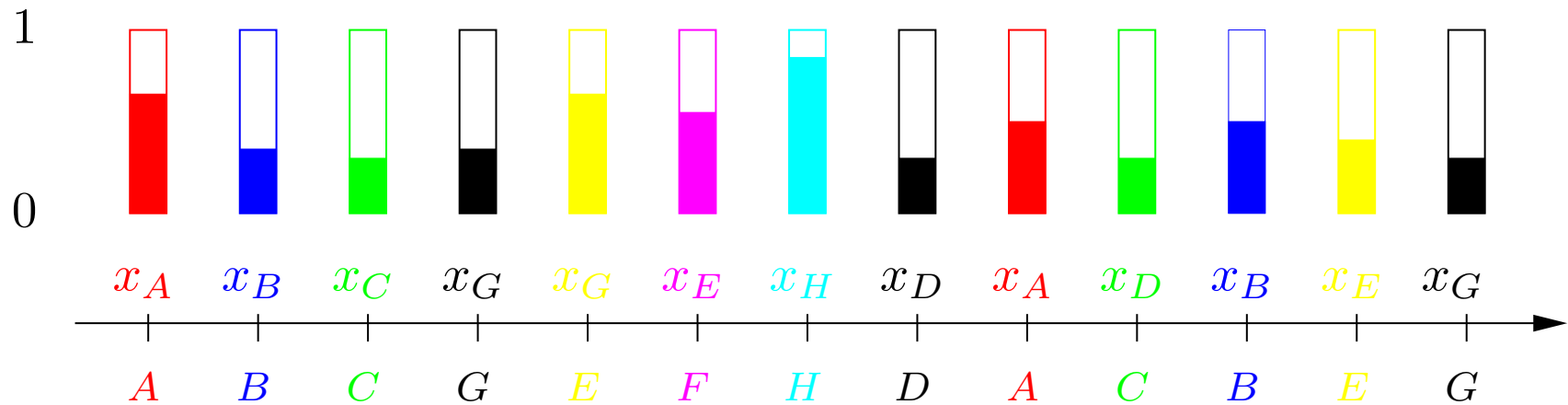
$$\sum_D \text{Size}(D)x_{D,t} \leq k \quad \forall t$$

$$x_{D_t,t} = 1 \quad \forall t$$

$$x_{D_t,t} = x_{D_t,s} \quad \forall t, s \in I_t$$

$$x_{D,t} \in [0, 1] \quad \forall D, t$$

Rounding algorithm



extra space rounded-up documents \leq freed up space rounded-down documents

$$\text{Cost}(\text{rounded solution}) \leq c \cdot \text{Cost}(\text{optimal solution})$$

Connection caching

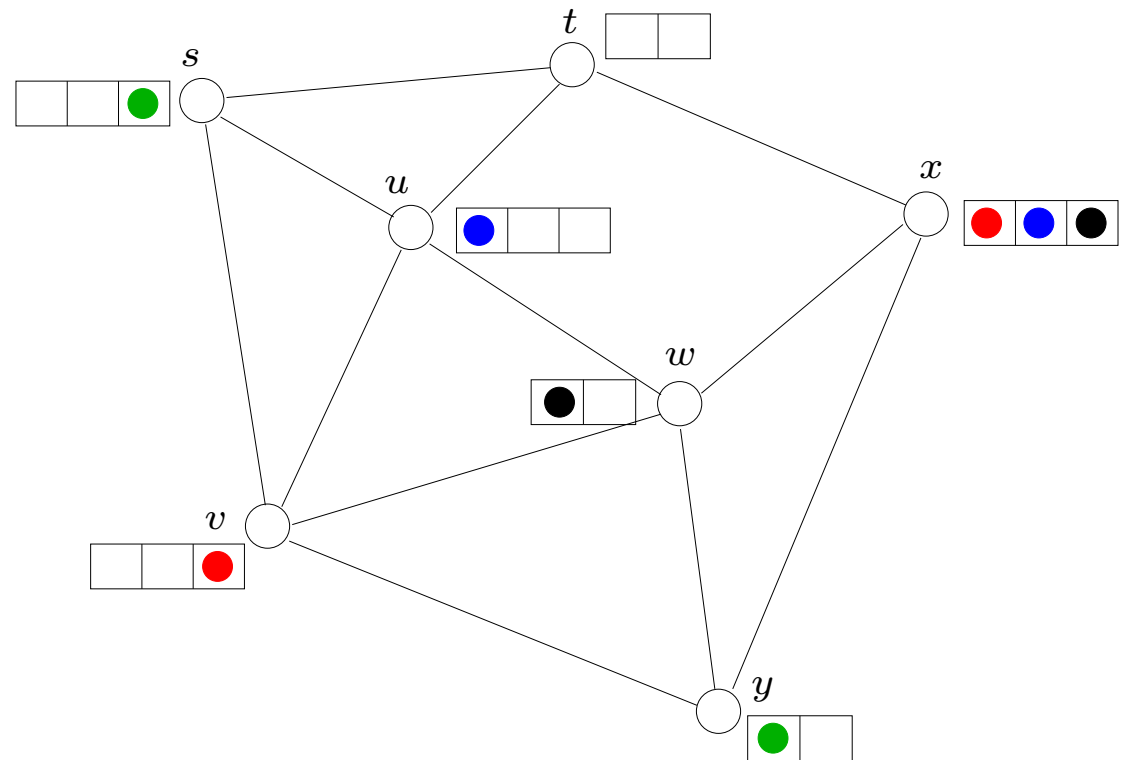
HTTP: TCP connections

HTTP/1.0 opens/closes connection for each request.

HTTP/1.1 allows

persistent connections.

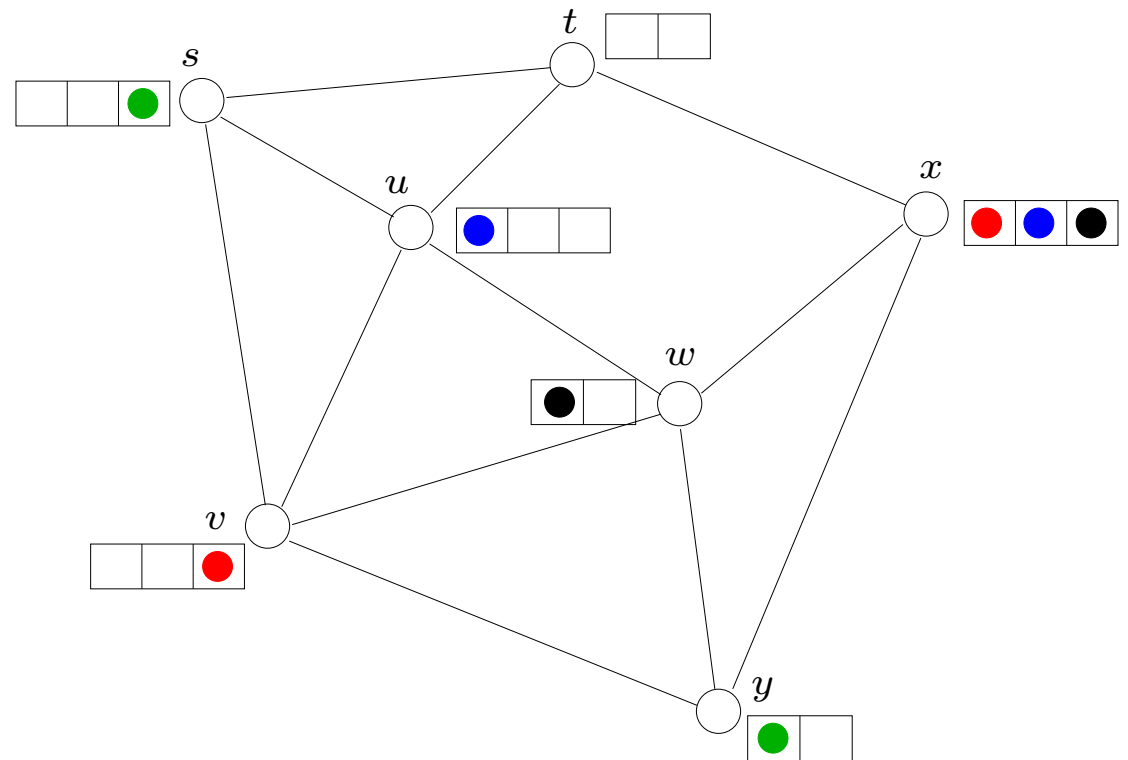
- CPU time saved
- Pipelining of HTTP requests
- Congestion reduced



Connection caching

Important properties:

- Open connection must be **cached at both endpoints**
- Connections may incur **varying establishment costs**



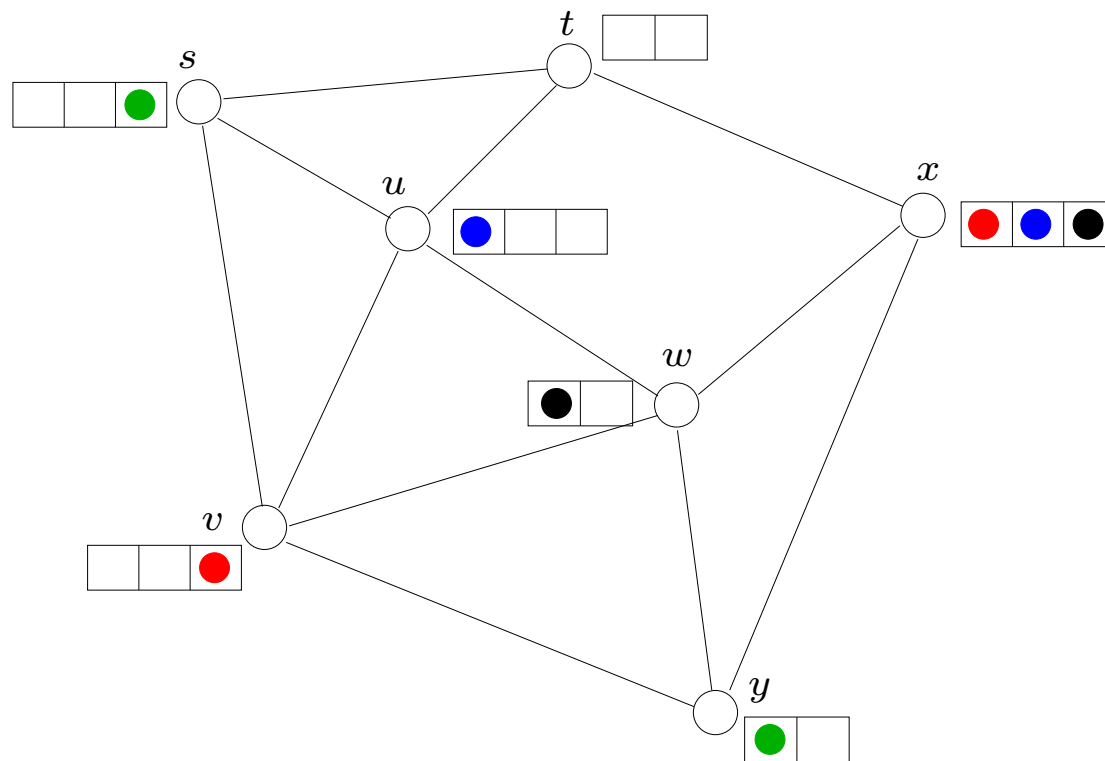
Problem definition

Request: TCP connection

$$C = (x, y)$$

Establishment of C : $\text{Cost}(C)$

Goal: Minimize total connection establishment cost



$$\sigma = (x, u) (y, s) (x, v) (x, u) (x, w) (x, t) \dots$$

Landlord

C : $0 \leq \text{Credit}(C) \leq \text{Cost}(C)$

$C = (x, y)$ to be established

Node x :

Close C' with min. Credit.

Reduce Credits of other connections by $\text{Credit}(C')$.

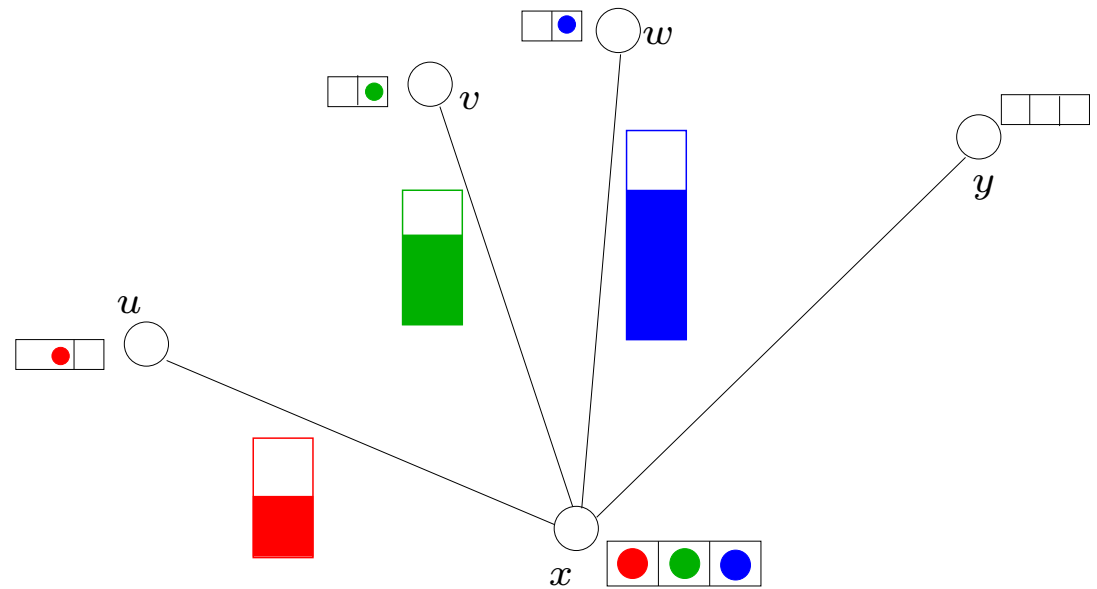
Analogous at y .

Open $C = (x, y)$.

$\text{Credit}(C) := \text{Cost}(C)$.

Thm: Landlord is k -competitive.

$k = \max.$ #open connect. at any node



Landlord

C : $0 \leq \text{Credit}(C) \leq \text{Cost}(C)$

$C = (x, y)$ to be established

Node x :

Close C' with min. Credit.

Reduce Credits of other connections by $\text{Credit}(C')$.

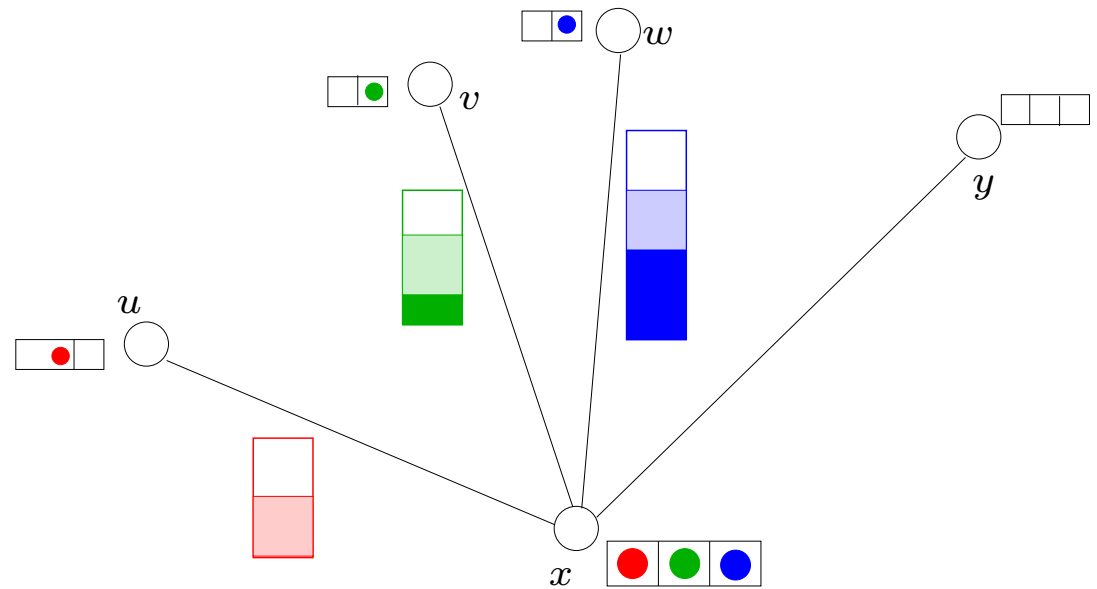
Analogous at y .

Open $C = (x, y)$.

$\text{Credit}(C) := \text{Cost}(C)$.

Thm: Landlord is k -competitive.

$k = \max.$ #open connect. at any node



Reduced communication

Alg. Landlord ($0 < \epsilon \leq 1$):

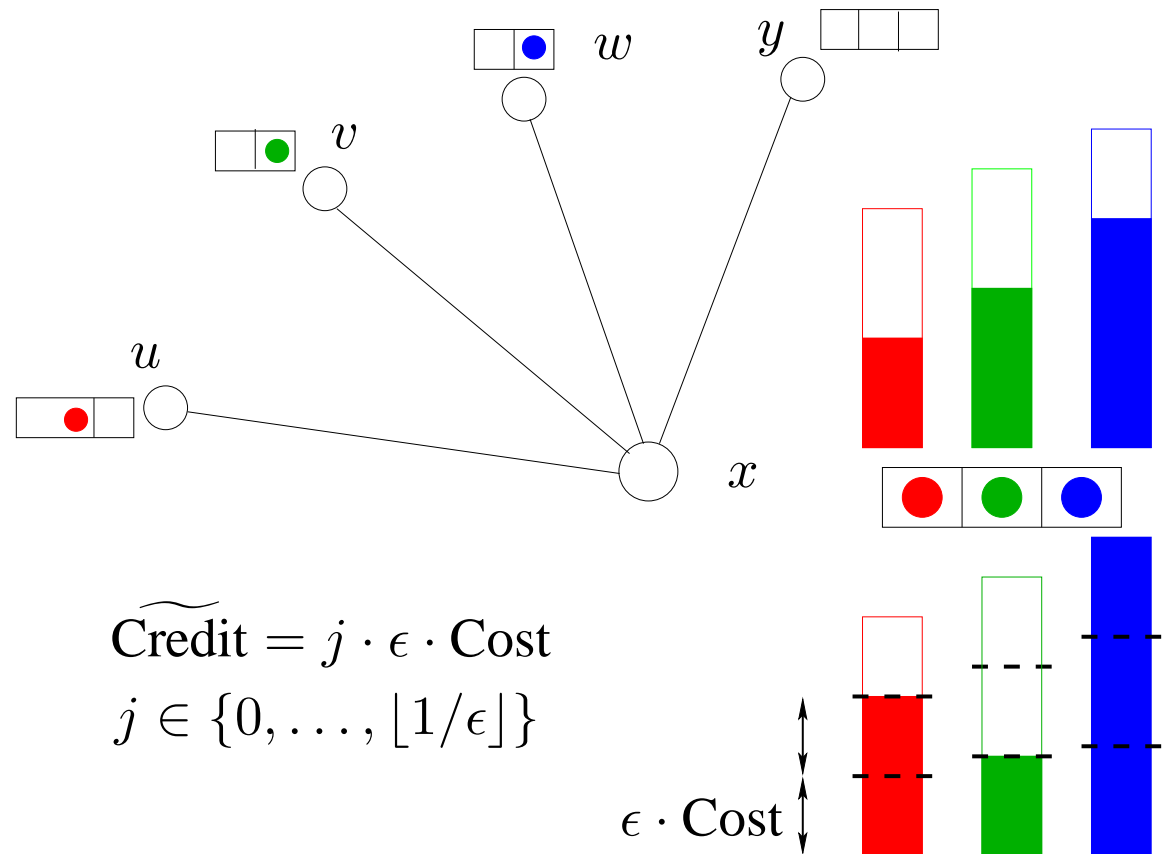
Node x stores for $C = (x, x')$

$\text{Credit}(C, x)$, $\widetilde{\text{Credit}}(C, x')$

Close C when

$\text{Credit}(C, x) + \widetilde{\text{Credit}}(C, x')$
 $\leq \text{Cost}(C)$

Thm: LL is $(1 + \epsilon)k$ -competitive
 $\lceil 1/\epsilon \rceil - 1$ extra bits per open con-
 nection.



Randomized algorithm

Algorithm Harmonic

$C = (x, y)$ to be established

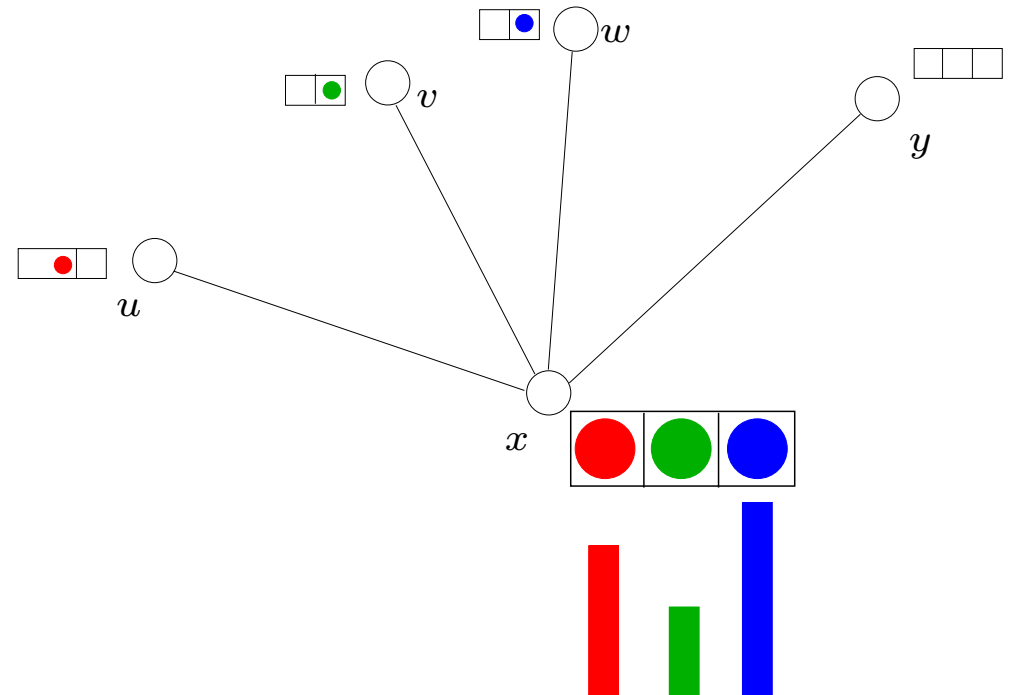
At x close C' with probability

$$\frac{1}{\text{Cost}(C')} \cdot \frac{1}{C_x}$$

$$C_x = \sum_{D \text{ cached at } x} \frac{1}{\text{Cost}(D)}$$

Analogous at y .

Thm: Harmonic is k -competitive



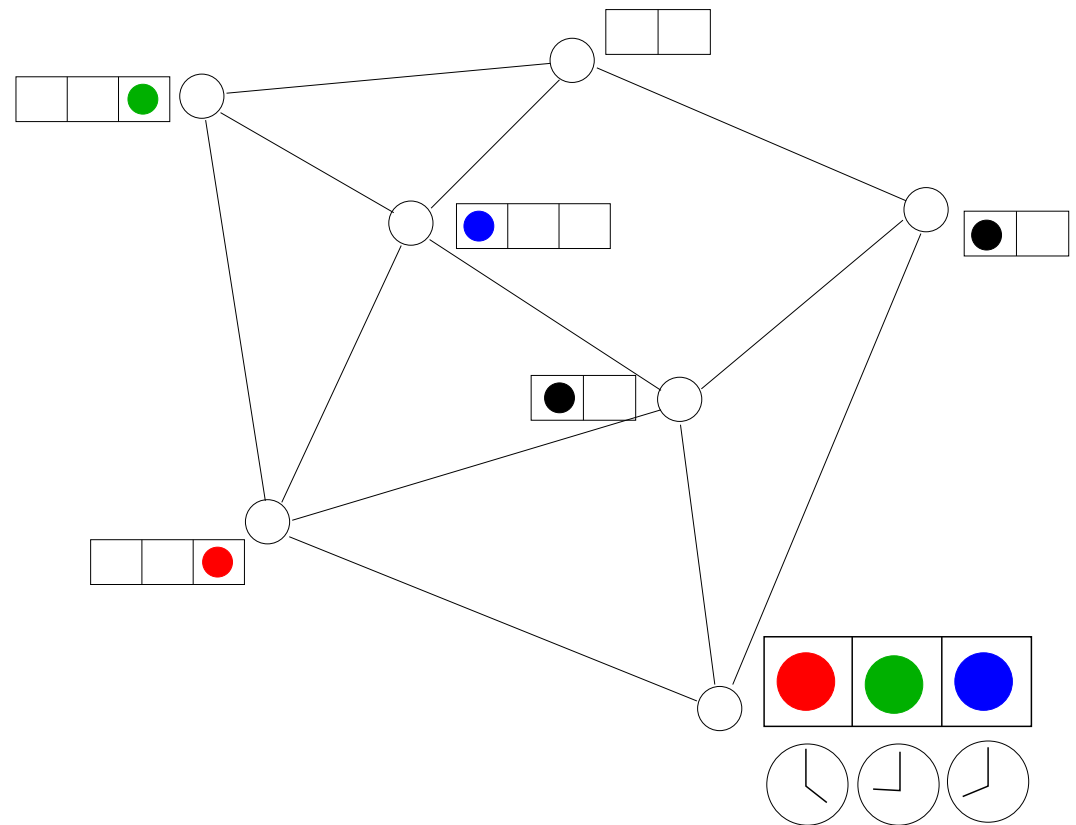
Time-out values

C not used for n_C time units,
 C expires

Modified Landlord

$C = (x, y)$ to be established
Remove **expired connections**
at x and y .

Thm: Mod. LL k -competitive



TCP Acknowledgement

Multicast Acknowledgement

Susanne Albers

TCP acknowledgement

Input: $I = a_1, a_2, \dots, a_n$

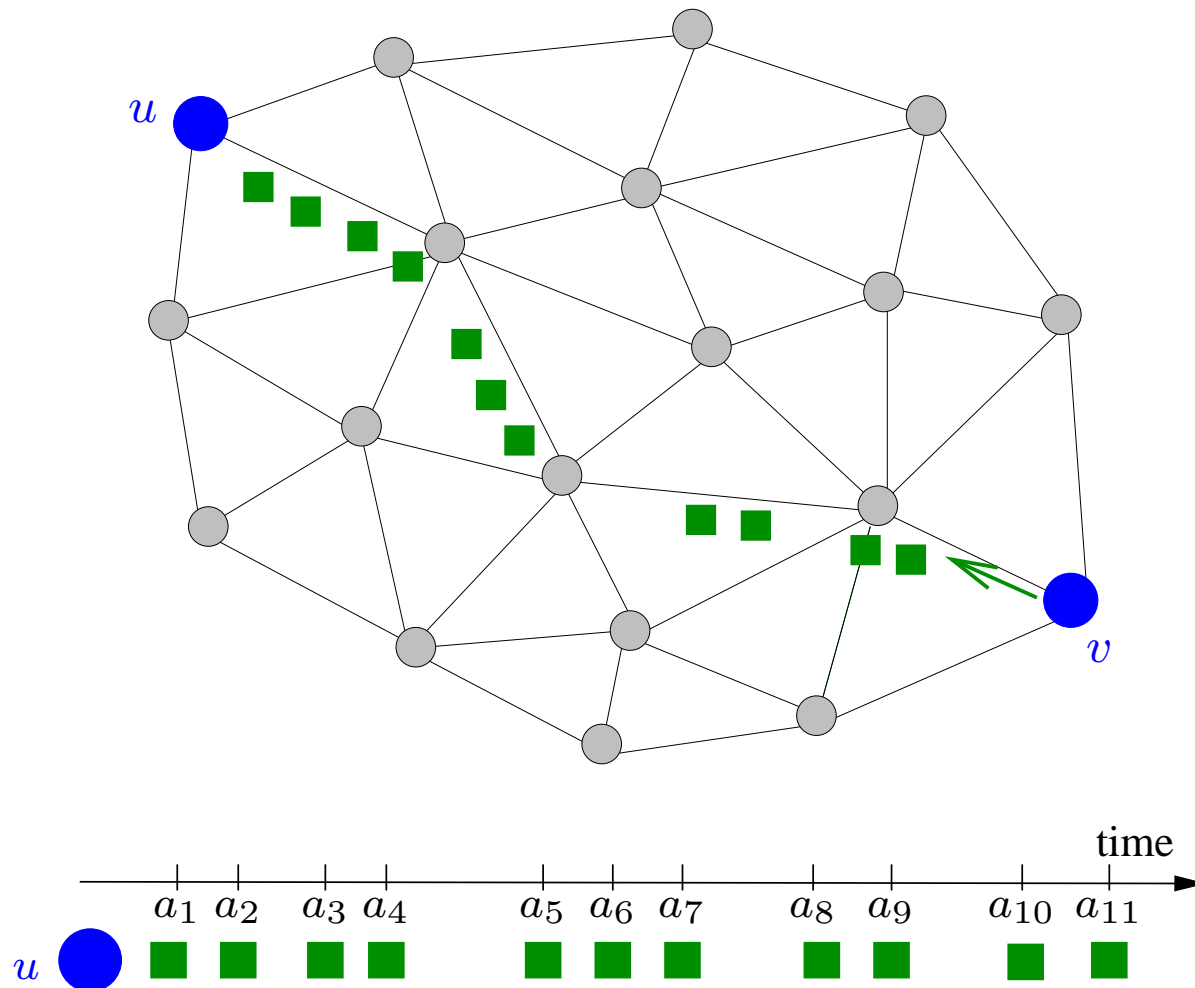
a_i = arrival time of i -th packet

Problem: Determine when
acknowledgements are sent

Delayed acknowledgement:

- Overhead for sending/receiving acknowledgements is reduced
- Network congestion is reduced
- Latency is added to connection

TCP connection betw. u and v .



TCP acknowledgement

Input: $I = a_1, a_2, \dots, a_n$

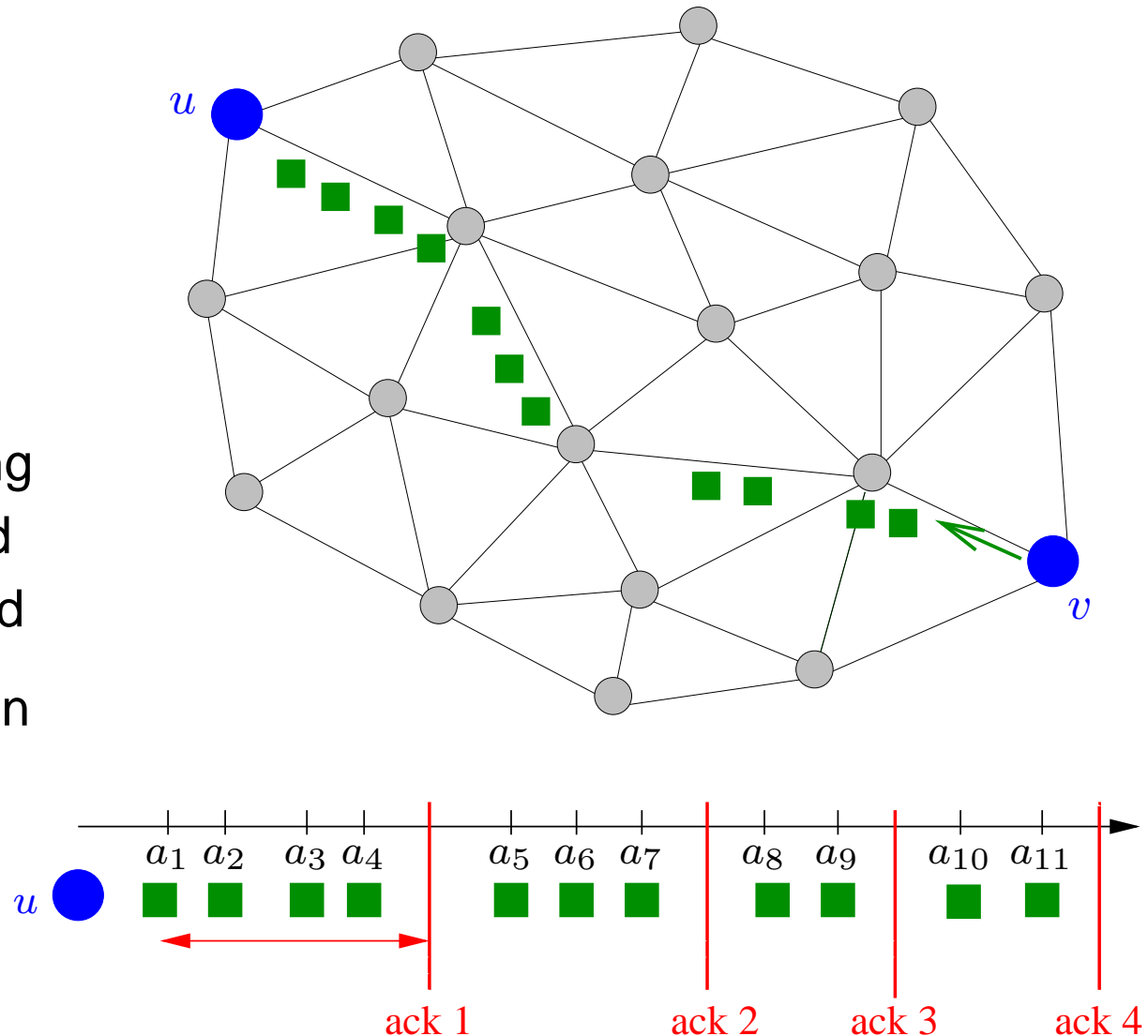
a_i = arrival time of i -th packet

Problem: Determine when **acknowledgements** are sent

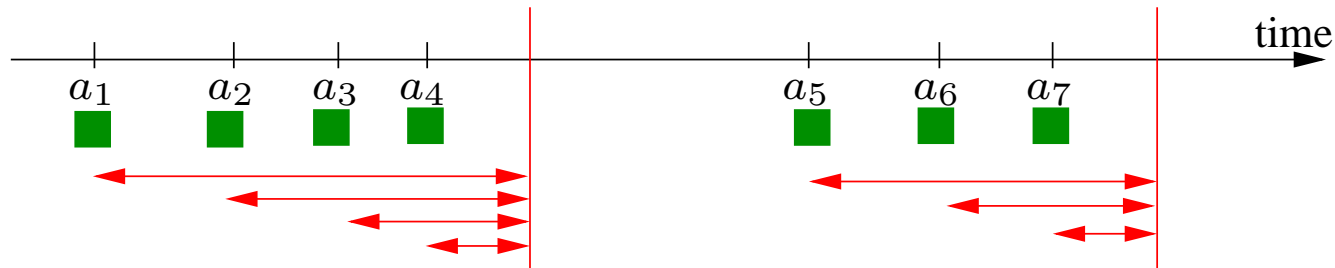
Delayed acknowledgement:

- Overhead for sending/receiving acknowledgements is reduced
- Network congestion is reduced
- Latency is added to connection

TCP connection betw. u and v .



Total delay



$\min f$

$$\# \text{ack} + \sum_i (t_i - a_i)$$

$t_i = \text{ack time packet } i$

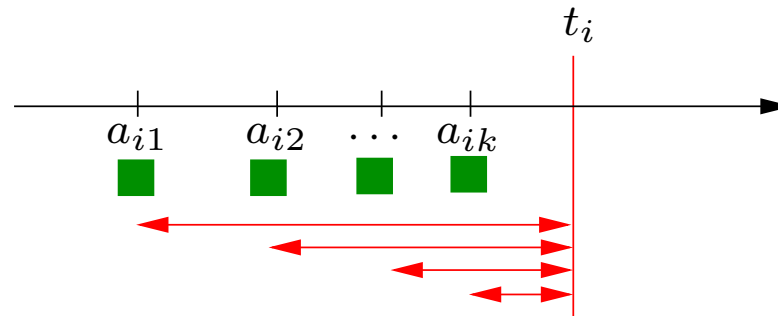
Thm: Greedy is 2-competitive.

Thm: No det. alg. better than 2-competitive.

Thm: Rand. alg. achieve competitiveness of $e/(e-1)$.

Dooly, Goldman and Scott 2001; Noga, Seiden, Woeginger, 2001;
Karlin Kenyon Randall 2001

Total delay



Algorithm Greedy: Send ack when total delay of outstanding packets is **equal to cost of ack**, i.e. equal to 1.

Thm: *Greedy* is 2-competitive.

Analysis: Each ack_i contributes 2 to f . Let a_{i1}, \dots, a_{ik} be acknow. packets.

If OPT acknowledges within $[a_{i1}, t_i]$, then its acknowledgement cost is 1.

Otherwise delay cost within $[a_{i1}, t_i]$ is 1.

Maximum delay

Avoid long delays

TCP used for

- interactive data transfer: long delays are noticeable to users
- bulk data transfer: long delays incur overhead at endpoints

$$f = \#\text{ack} + \max_i (t_i - a_i)$$

$$f_p = \#\text{ack} + \max_i (t_i - a_i)^p \quad \text{for some } p \geq 1$$

$$\min f, f_p$$

Albers, Bals 2003

Maximum delay

- $f = \# \text{ack} + \max_i (t_i - a_i)$

Deterministic: competitive ratio of

$$\frac{\pi^2}{6} = \sum_{j \geq 1} 1/j^2 \approx 1.6449$$

- $f_p = \# \text{ack} + \max_i (t_i - a_i)^p$

Deterministic: competitive ratio of

$$1 + \sum_{s=1}^{p+1} (-1)^{p+1-s} \zeta(s)$$

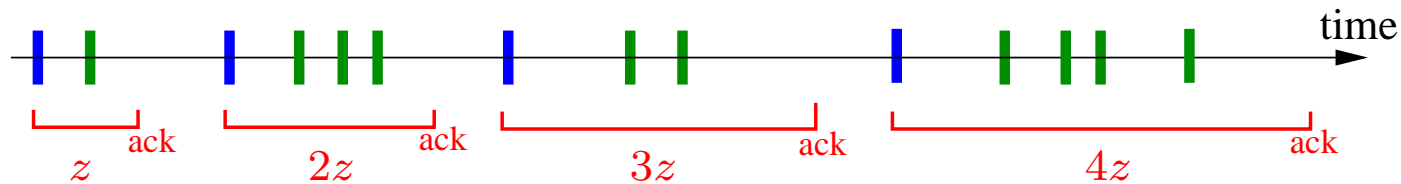
$$\zeta(s) = \sum_{j \geq 1} 1/j^s \quad \zeta(1) = 1$$

Riemann's zeta function

Competitive ratios

p	c_p
1	1.6449
2	1.5571
3	1.5252
4	1.5117
5	1.5056
6	1.5027
7	1.5013
8	1.5007
9	1.5003
10	1.5002

Maximum delay



Algorithm Linear-Delay(z): $z > 0$

Maximum delay in i -th acknowledgement is iz .

Thm: Competitive ratio of Linear-Delay(z) is
 $\max\{1 + z, (1 + z)/(2 + z - \pi^2/6)\}$.

Cor: $z = \pi^2/6 - 1 \Rightarrow c = \pi^2/6$

Multicast acknowledgement

$T = (V, E, w)$ in-tree

n packets p_1, \dots, p_n

p_i : a_i arrival time

v_i release node

Ack. & packets may be aggregated.

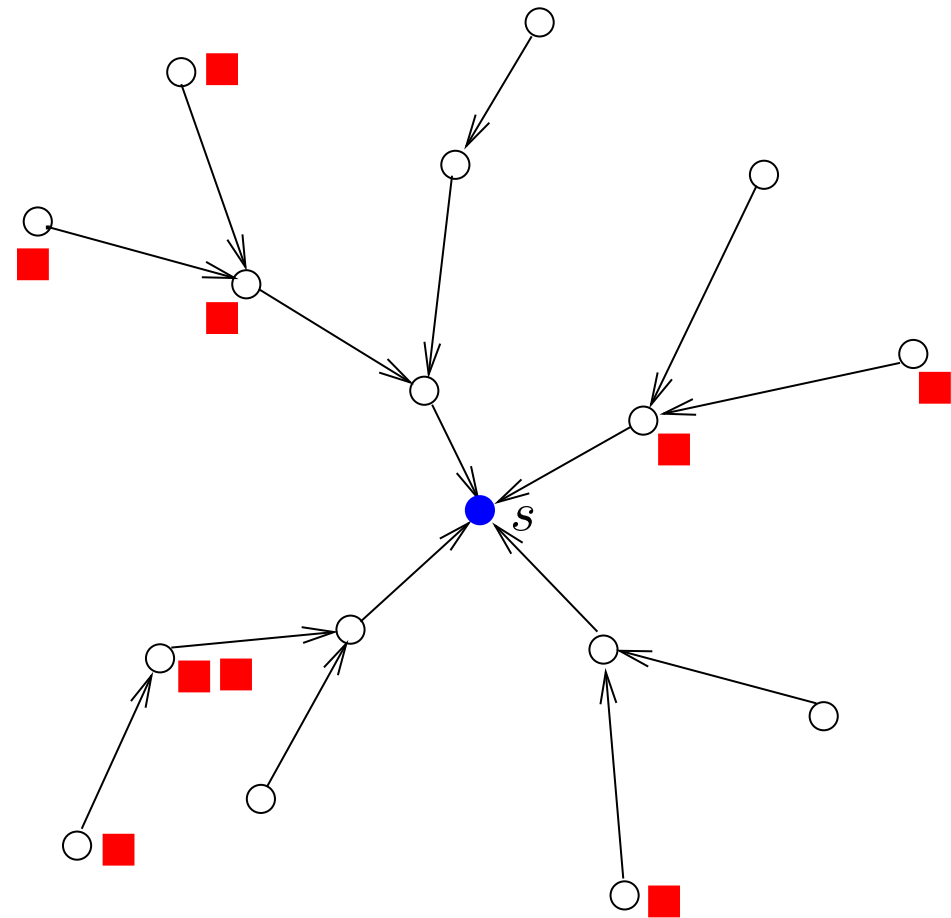
Sending set of packets along

e costs $w(e)$.

No propagation delay.

Goal:

$$\text{Min. } \sum_{e \in E} n_e w(e) + \sum_{i=1}^n (t_i - a_i)$$



Communication models

- **Asynchronous Model:** All nodes act in **isolation**; there is no way to coordinate.
- **Synchronous Model:** There is **global clock** allowing the network nodes to synchronize actions.
- **Full Information Model:** At any time each node has **complete knowledge** of the state of the network. Future packet arrivals are unknown.

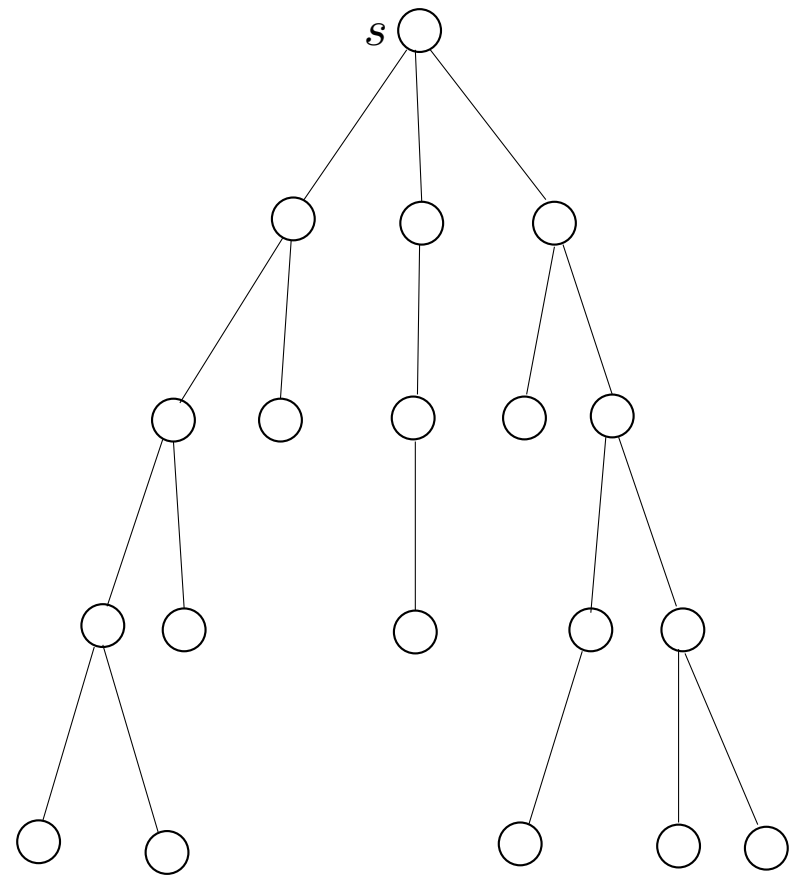
Asynchronous model

$\mathcal{S}_T = \{ \text{subtrees } T' \text{ of } T \text{ rooted at } r \}$

$\mathcal{P}(T') = \{ \text{paths } \pi \text{ starting in } T', \text{ ending at } r \}$

$$\chi(T) = \max_{T' \in \mathcal{S}_T} \frac{\sum_{\pi \in \mathcal{P}(T')} w(\pi)}{w(T')}.$$

Brito, Koutsoupias, Vaya 2004



Asynchronous model

Algorithm Wait: Node v works as follows.

- Packets **released at v** are sent to the sink when the total delay is $w(\pi_v)/\chi(T)$.
- Packets arriving from **descendants** are forwarded **without delay**.

Thm: *Wait* is $(\chi(T) + 1)$ -competitive.

Thm: Any det. alg. has competitive ratio **at least $\chi(T)$** .

Synchronous model

Algorithm Greedy1: Node v ; time t

- Set of packets $P(v, t)$ is sent to parent u along e when $\text{delay}(P(v, t)) = w(e)$.
- At u reduce $\text{delay}(P(v, t))$ by $w(e)$.

Thm: *Greedy1* is $O(h \log w(T))$ -competitive.

Thm: Any oblivious. alg. has competitive ratio at least $O(\sqrt{h})$.

Oblivious alg. makes decisions based on local information.

h = height of T

Khanna, Naor, Raz 2002

Full information

Algorithm Greedy2: At any time t

- Identify **max.** $T' \in \mathcal{S}_T$ for which **total delay** of packets waiting in T' is at least $w(T')$. Deliver to the root.

Thm: *Greedy2* is $O(\log w(T))$ -competitive.

Khanna, Naor, Raz 2002

Centralized routing

Thm: $\Theta(\log n)$ -competitive.

Aspnes et al. JACM 1997

Oblivious routing

Path selected for a requested is independent of other requests.

Thm: $O(\log^3 n)$ -competitive.

Räcke 2002

Thm: $O(\log^2 n \log \log n)$ -competitive.

Harrelson, Hildrum, Rao 2003