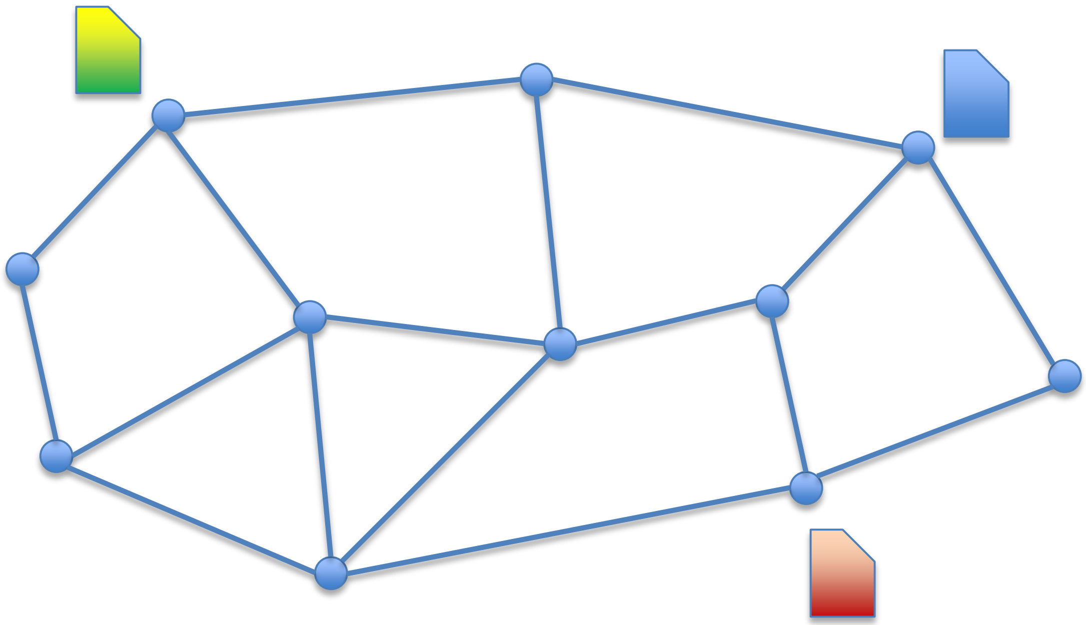
Adaptive Caching Algorithms with Optimality Guarantees for NDN Networks

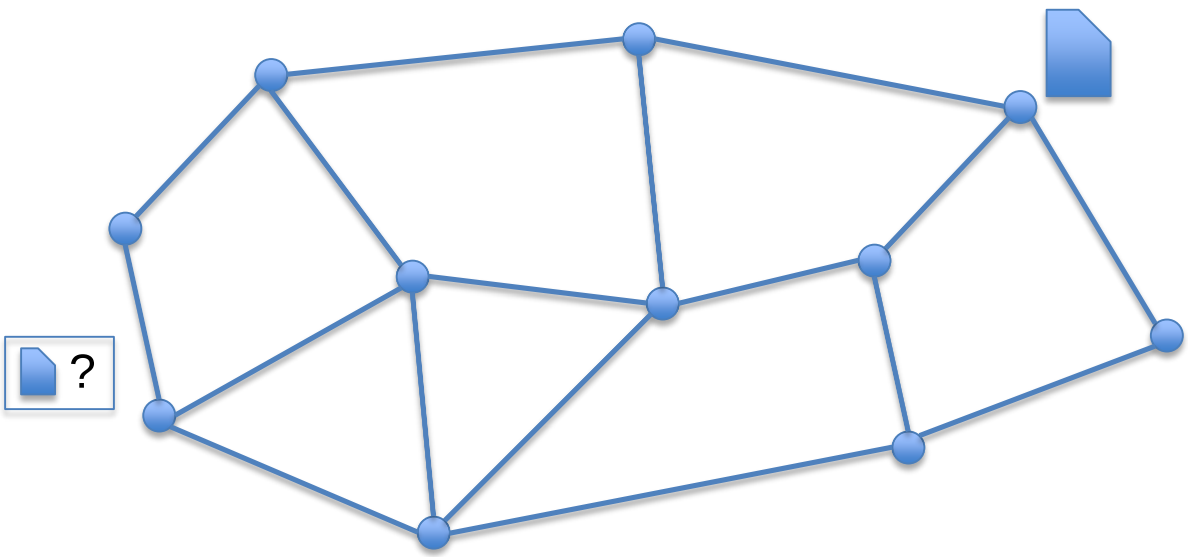
Stratis Ioannidis and **Edmund Yeh**

A Caching Network



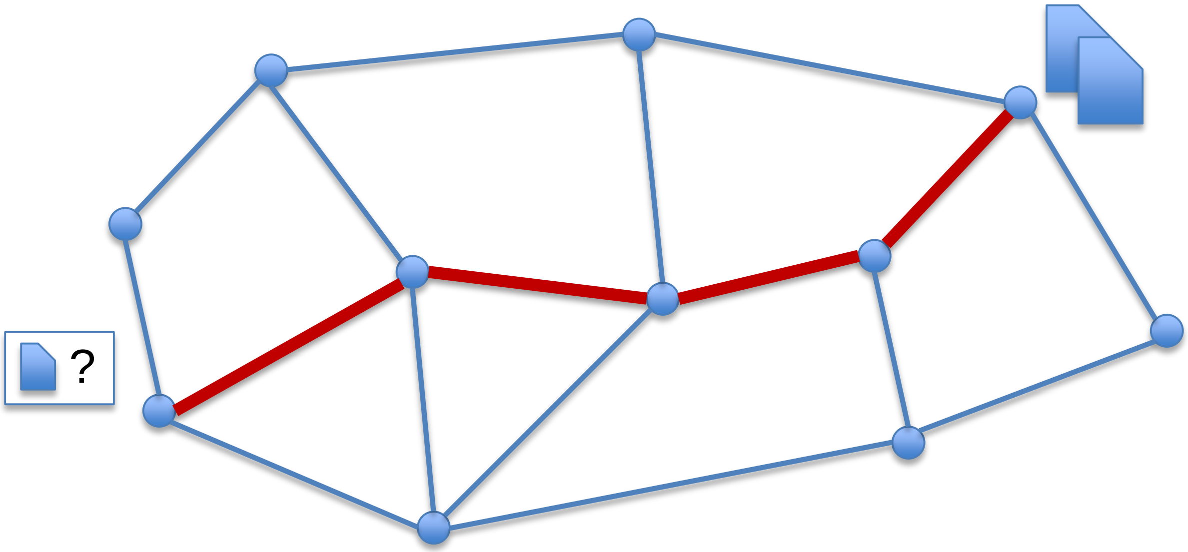
Nodes in the network store **content items** (e.g., files, file chunks)

A Caching Network



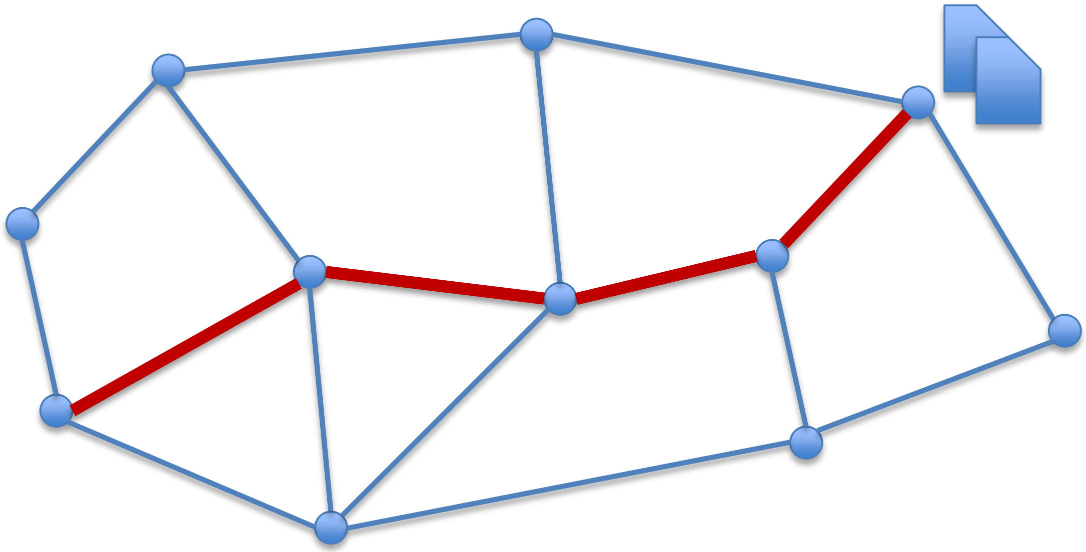
Nodes generate **requests** for content items

A Caching Network



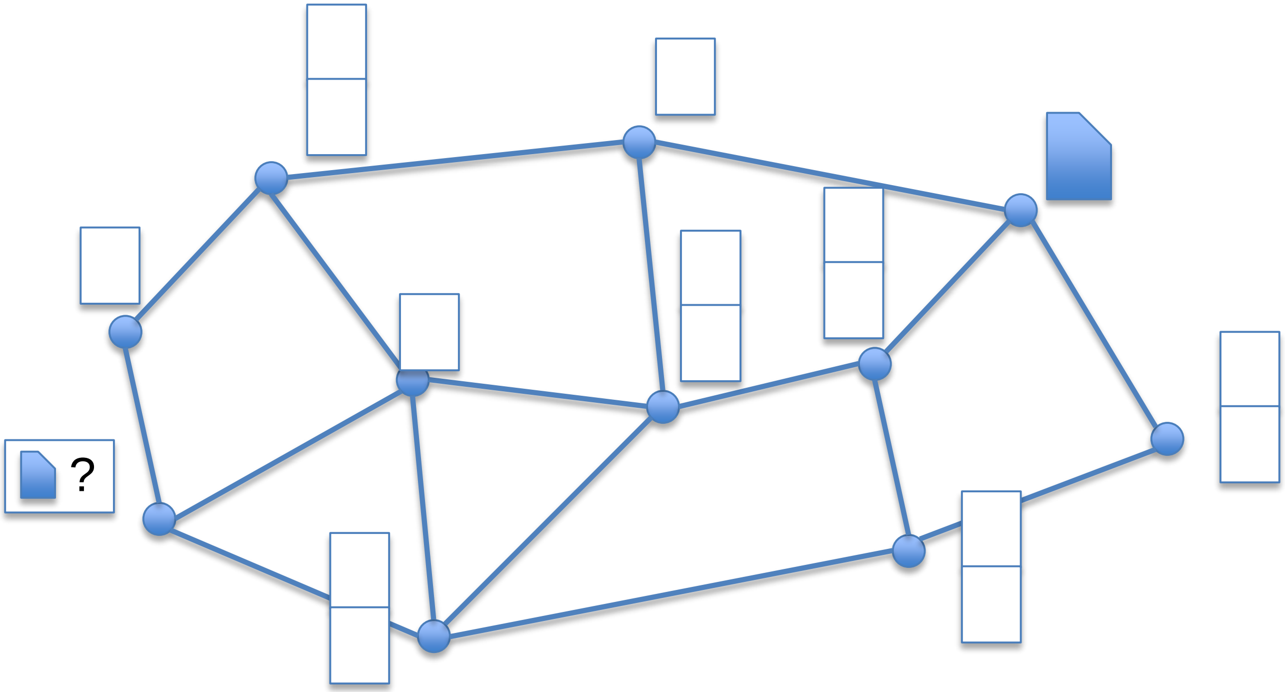
Requests are routed towards a content source

A Caching Network



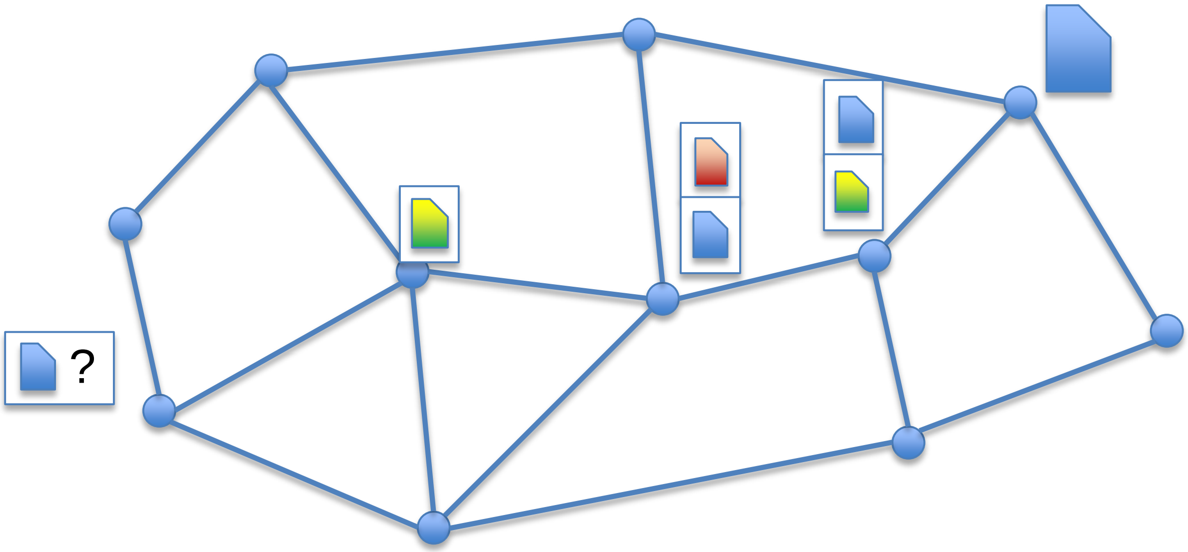
Responses routed over **reverse** path

A Caching Network



Nodes have **caches** with finite capacities

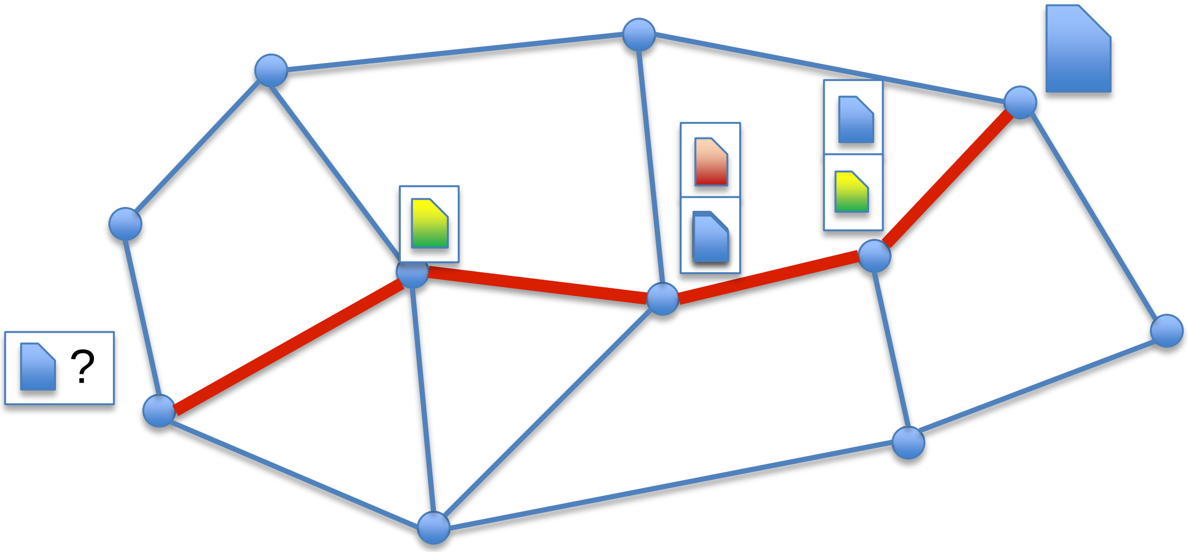
A Caching Network



Nodes have **caches** with finite capacities

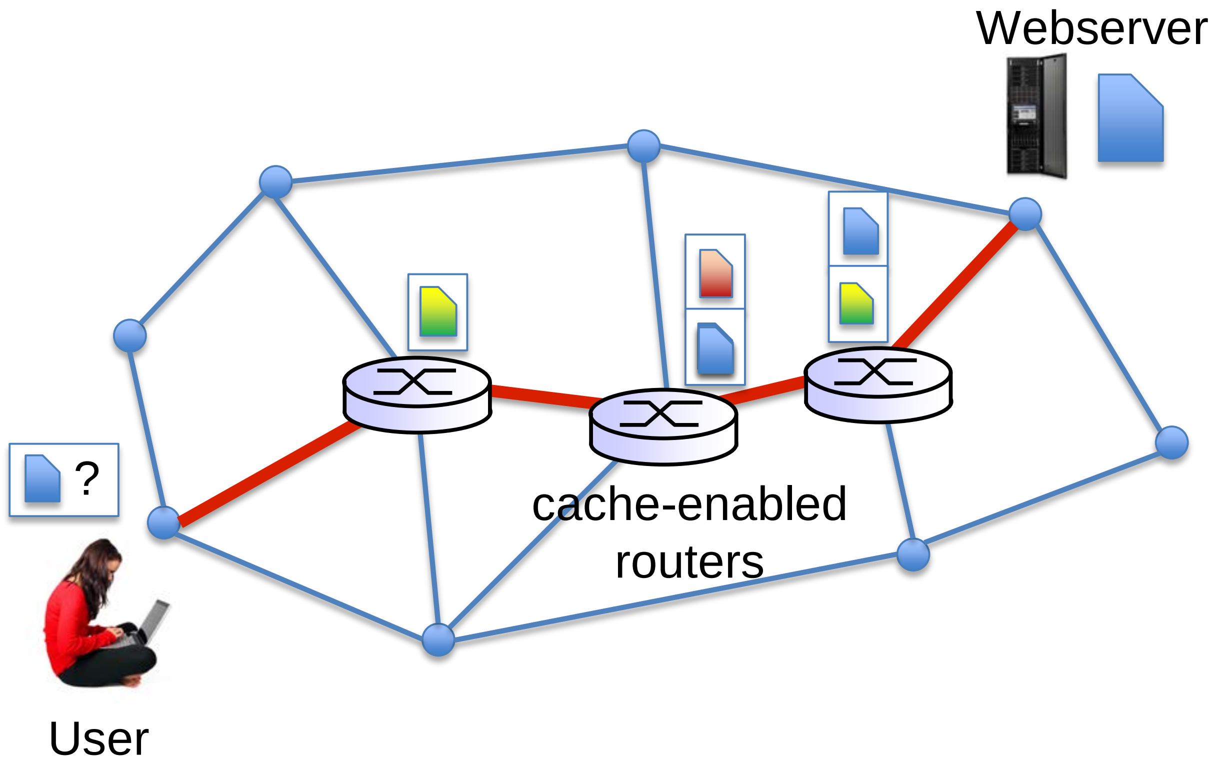
Northeastern

A Caching Network

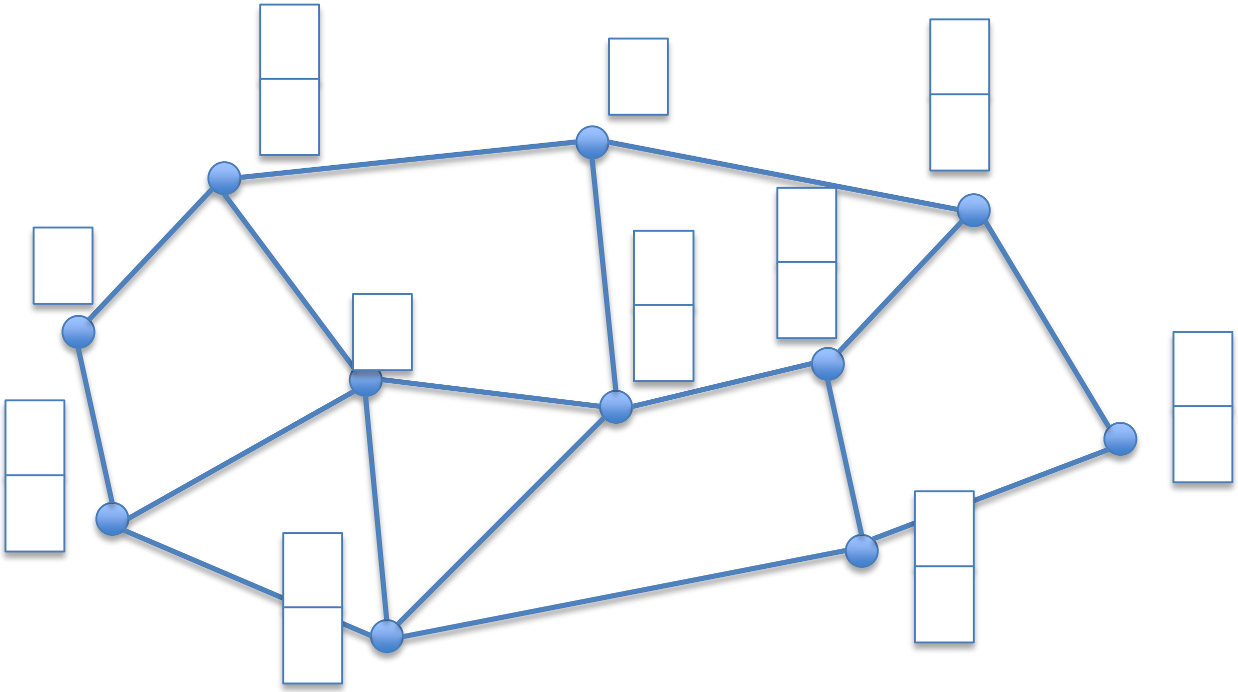


Requests terminate early upon a **cache hit**

Example: Named Data Networks



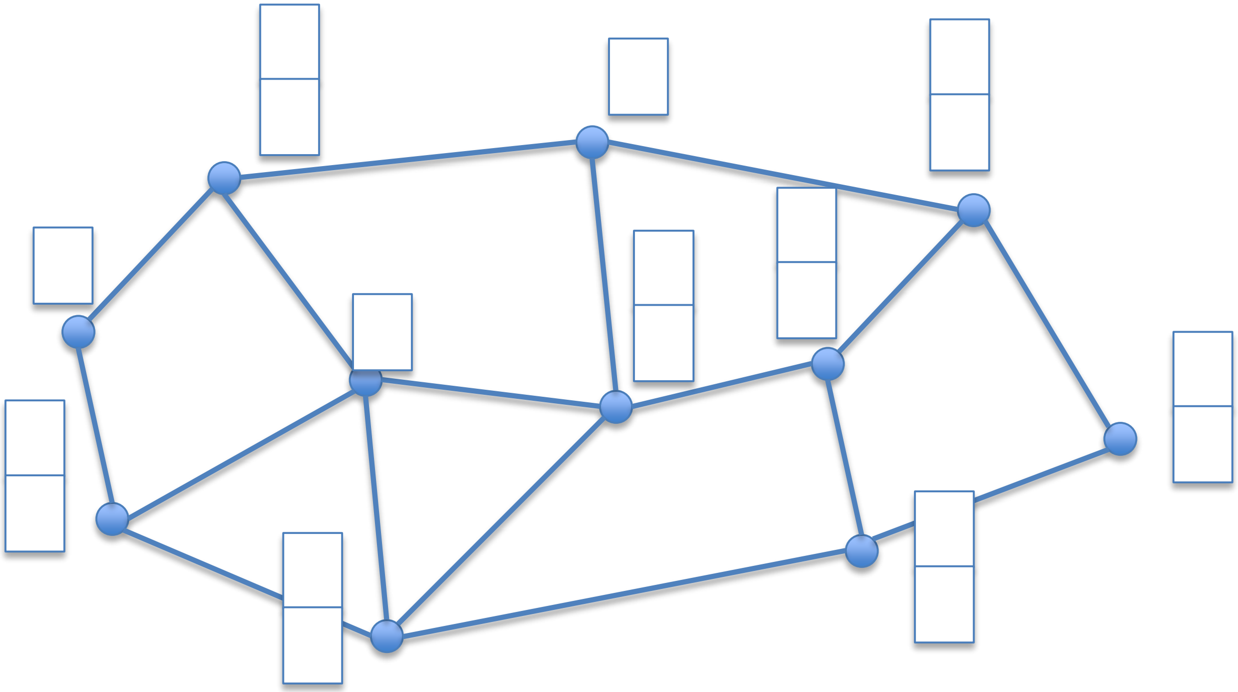
Optimal Content Allocation



Q: How should items be allocated to caches so that **routing costs are minimized**?

Northeastern

Optimal Content Allocation

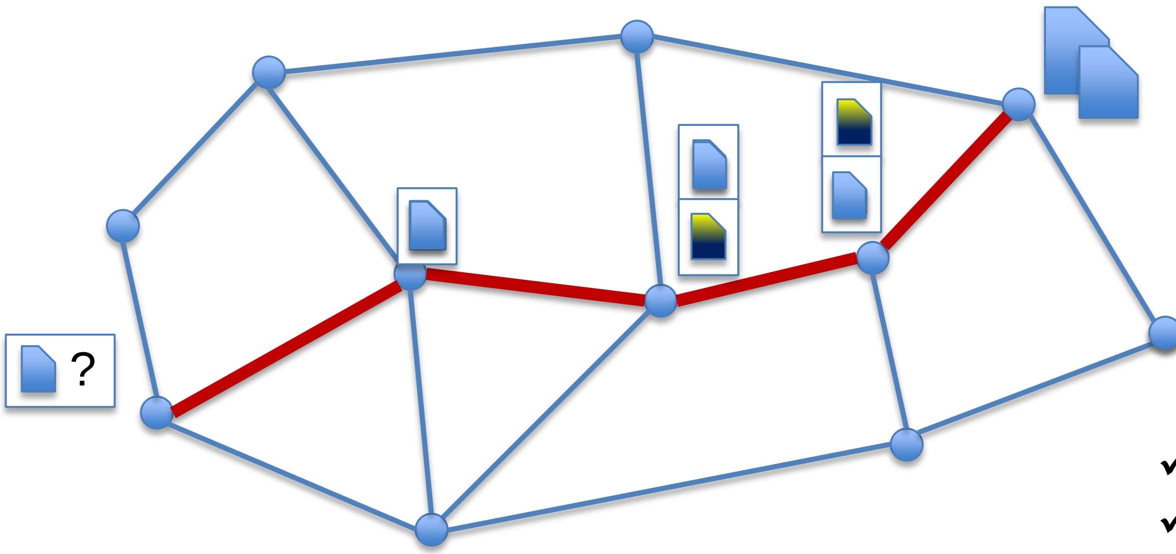


Challenge: Caching algorithm should be

- ❑ **adaptive**, and
- ❑ **distributed.**

A Simple Algorithm: Path-Replication

[Cohen and Shenker 2002]
[Jacobson et al. 2009]



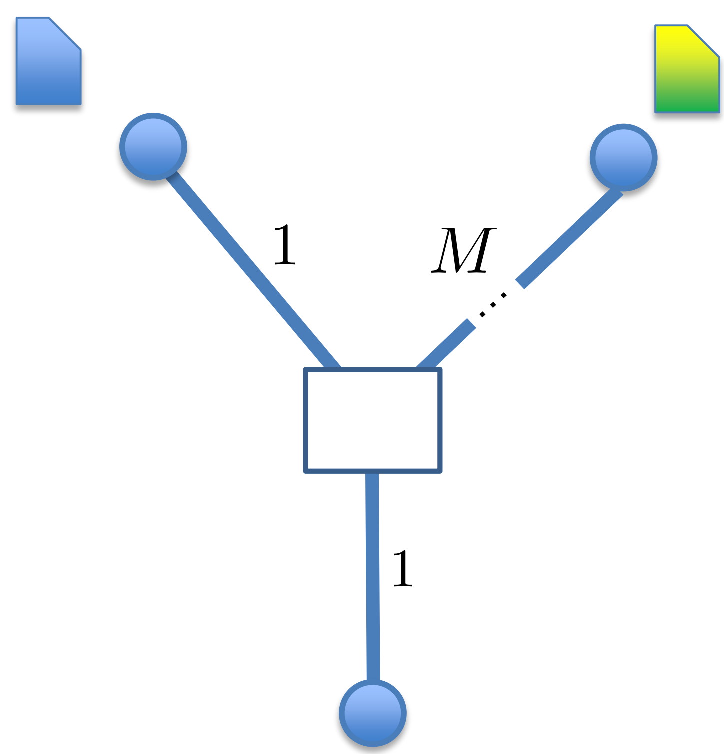
- ✓ Distributed
- ✓ Adaptive
- ✓ Popular!

- ❑ Cache item on every node in the reverse path
- ❑ Evict using a simple policy, e.g., LRU, LFU, FIFO etc.

But…

Path Replication combined with traditional eviction policies (LRU, LFU, FIFO, etc.) is **arbitrarily suboptimal.**

Path Replication + LRU is Arbitrarily Suboptimal



$\lambda_{?} = \lambda_{?} = 0.5$ requests per sec

Cost when caching :

$$0.5 \times 1 + 0.5 \times 2 = \mathbf{1.5}$$

Cost of PR+LRU:

$$0.25 \times (M+1) + 0.25 \times 1 + \\ + 0.25 \times 2 + 0.25 \times 1 = \mathbf{0.25}M\mathbf{+1.25}$$

- ❑ When M is large, PR+LRU is **arbitrarily suboptimal!**
- ❑ True for any strategy (LRU,LFU,FIFO,RR) that **ignores upstream costs**

Our Contributions

- Formal statement of offline problem
  - NP-Hard [Shanmugam et al. IT 2013]

- Path Replication +LRU, LFU, FIFO, etc. is arbitrarily suboptimal

- **Distributed**, **adaptive** algorithm, within a **constant approximation** from optimal offline allocation

- Path Replication+novel eviction policy
  - Great performance under 20+ network topologies

Overview

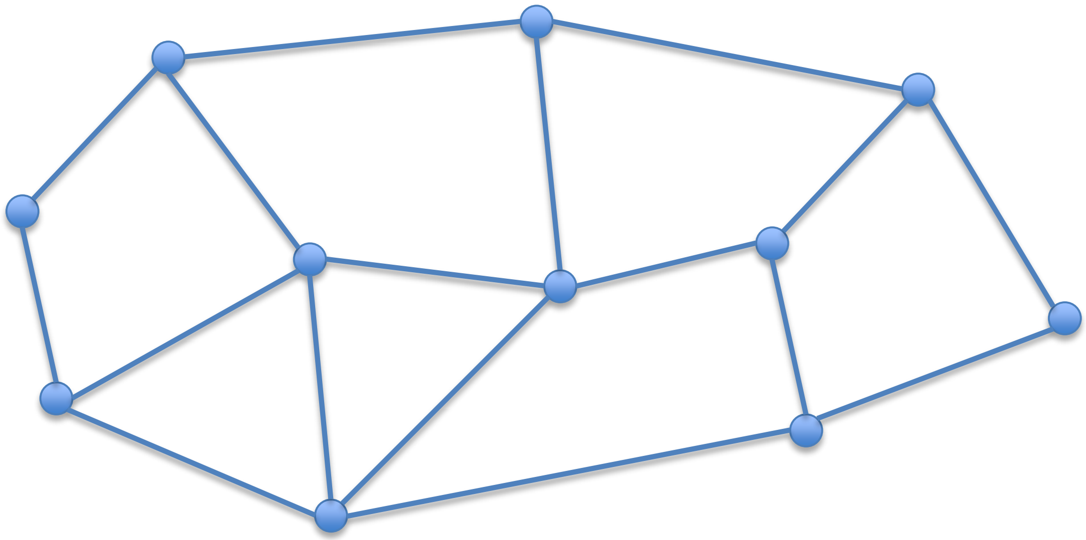
- ❑Problem Formulation
- ❑Distributed Adaptive Algorithms
- ❑Evaluation

Overview

- Problem Formulation
- Distributed Adaptive Algorithms
- Evaluation

Model: Network

$G(V, E)$

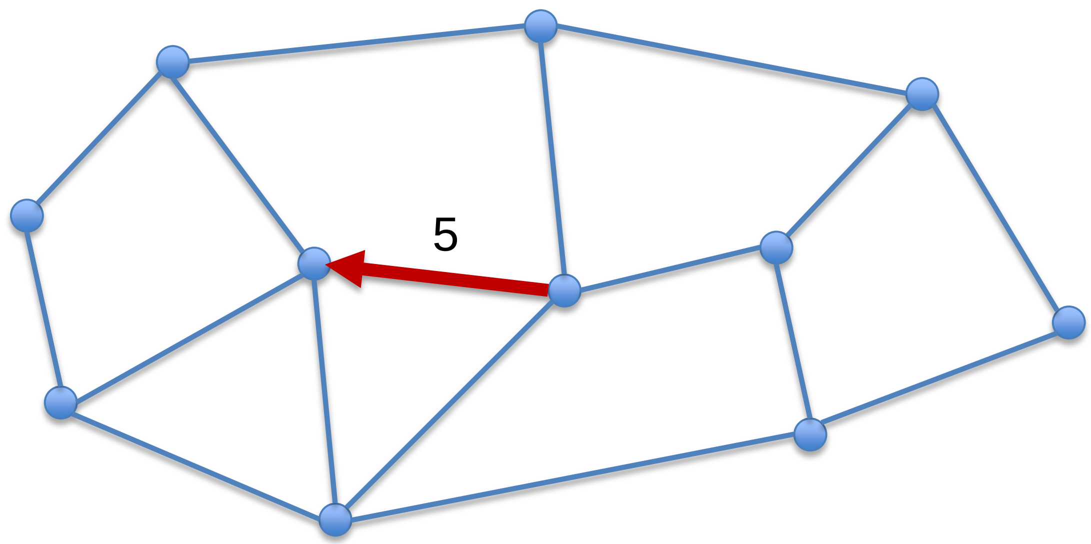


Network represented as a **directed**, **bi-directional** graph $G(V, E)$

Model: Edge Costs

$G(V, E)$

Edge costs: $w_{uv}, (u, v) \in E$



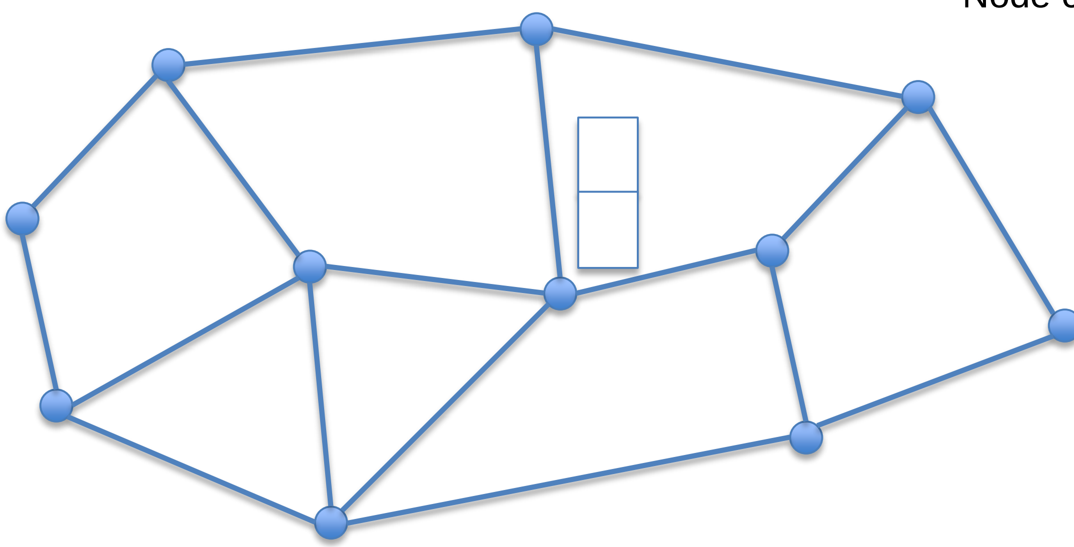
Each edge $(u, v) \in E$ has a **cost/weight** w_{uv}

Model: Node Caches

$G(V, E)$

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$



Node $v \in V$ has a cache with capacity $c_v \in \mathbb{N}$

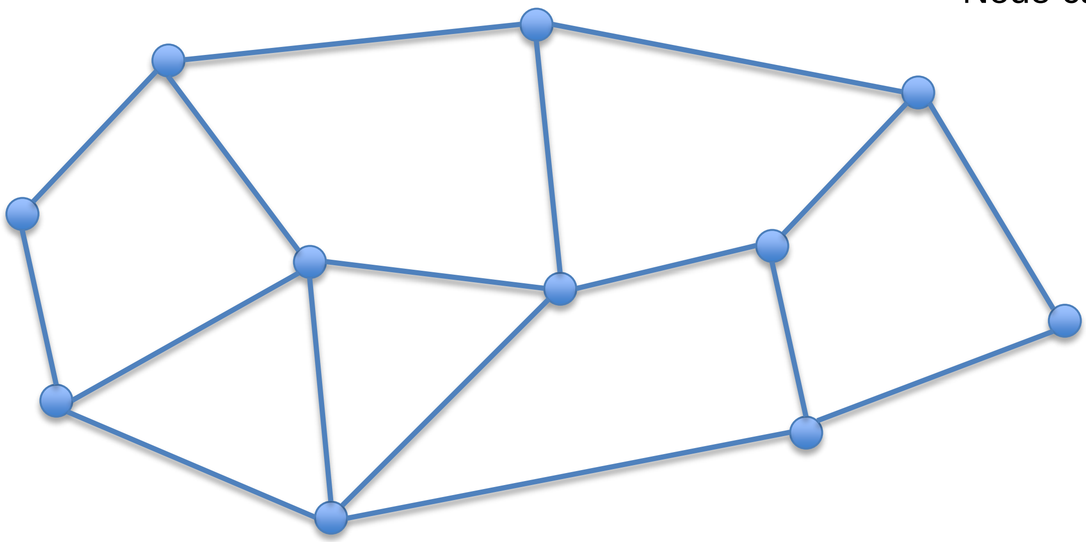
Model: Cache Contents

$G(V, E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$



Items stored and requested form the **item catalog** $\mathcal{C}$

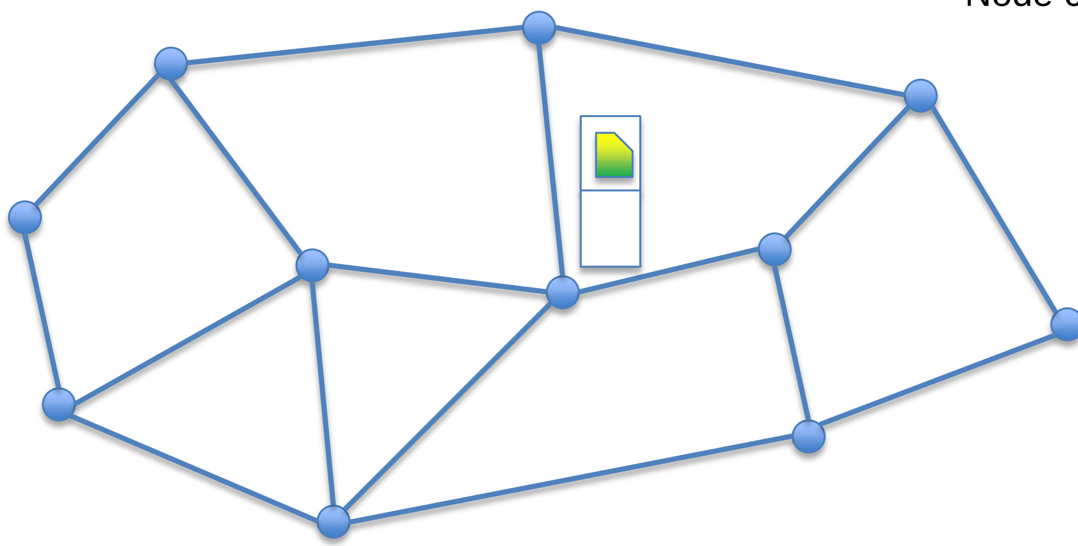
Model: Cache Contents

$G(V,E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$



For $v \in V$ and $i \in \mathcal{C}$, let

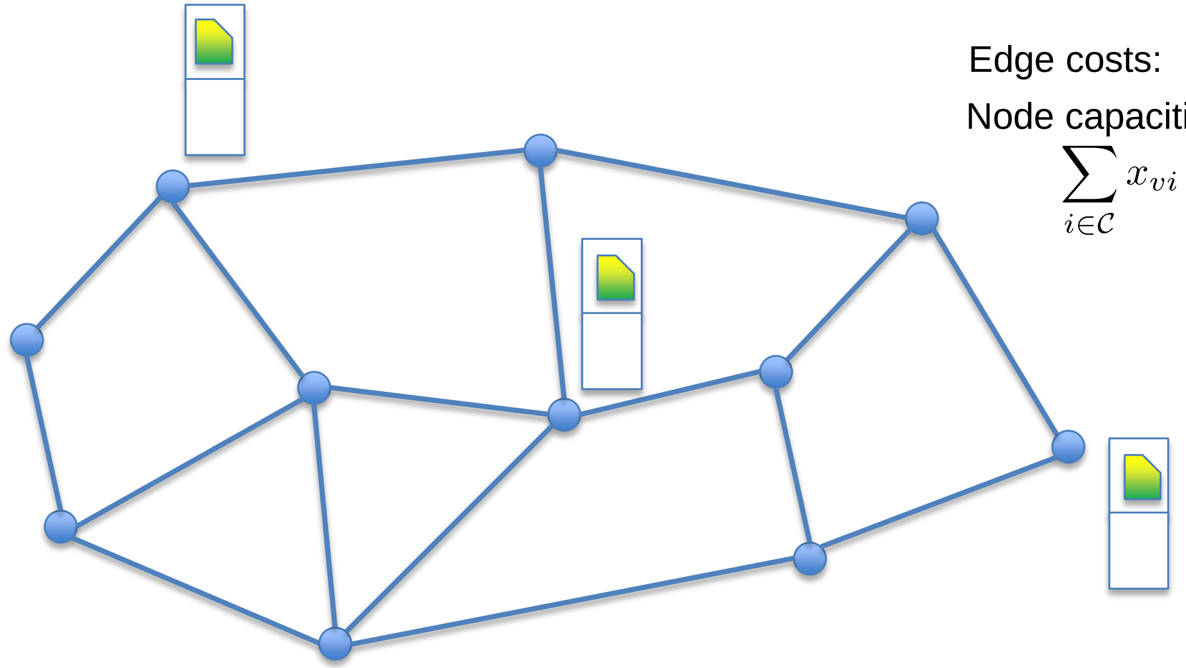
$$x_{vi} = \begin{cases} 1, & \text{if } v \text{ stores } i \\ 0, & \text{o.w.} \end{cases}$$

Then, for all $v \in V$, $\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$

Model: Designated Sources

$G(V, E)$

$\mathcal{C} = \{ \quad \}$



Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

For each and $i \in \mathcal{C}$, there exists a set of nodes $S_i \subset V$ (the **designated sources** of i) that **permanently store** i.

I.e., if $v \in S_i$ then $x_{vi} = 1$

Model: Demand

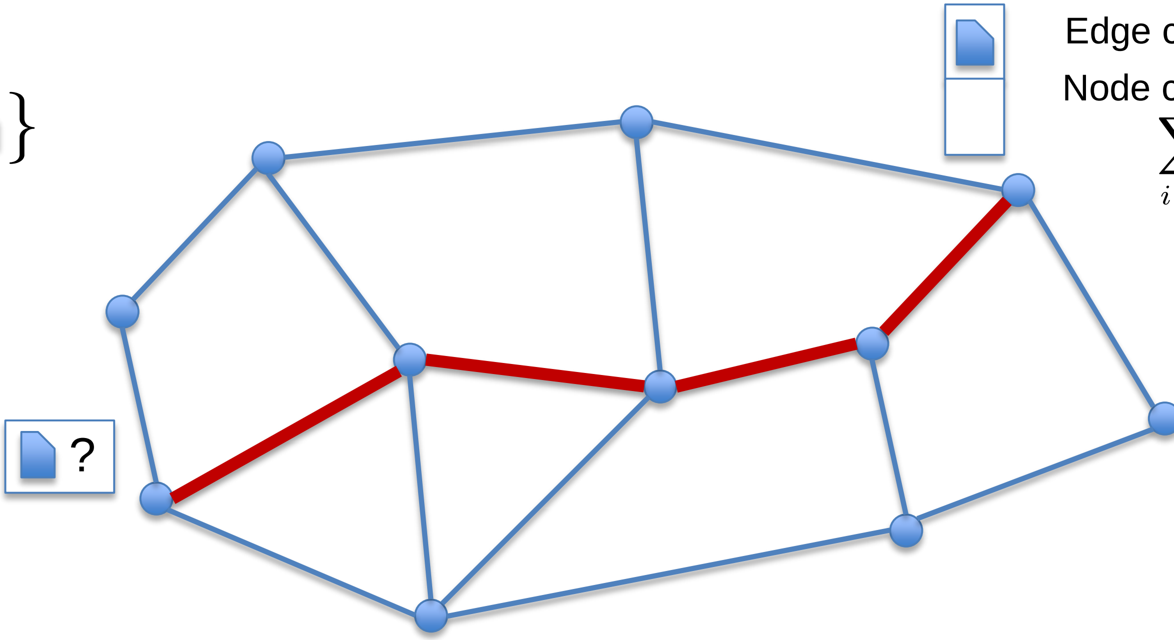
$G(V,E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$



Requests are always satisfied!

A **request** is a pair (i,p) such that:

- i is an item in $\mathcal{C}$
- $p = \{p_1, \ldots, p_K\}$ is a simple path in G such that $p_K \in S_i$.

Model: Demand

$G(V,E)$

$\mathcal{C}=\{$  $\}$

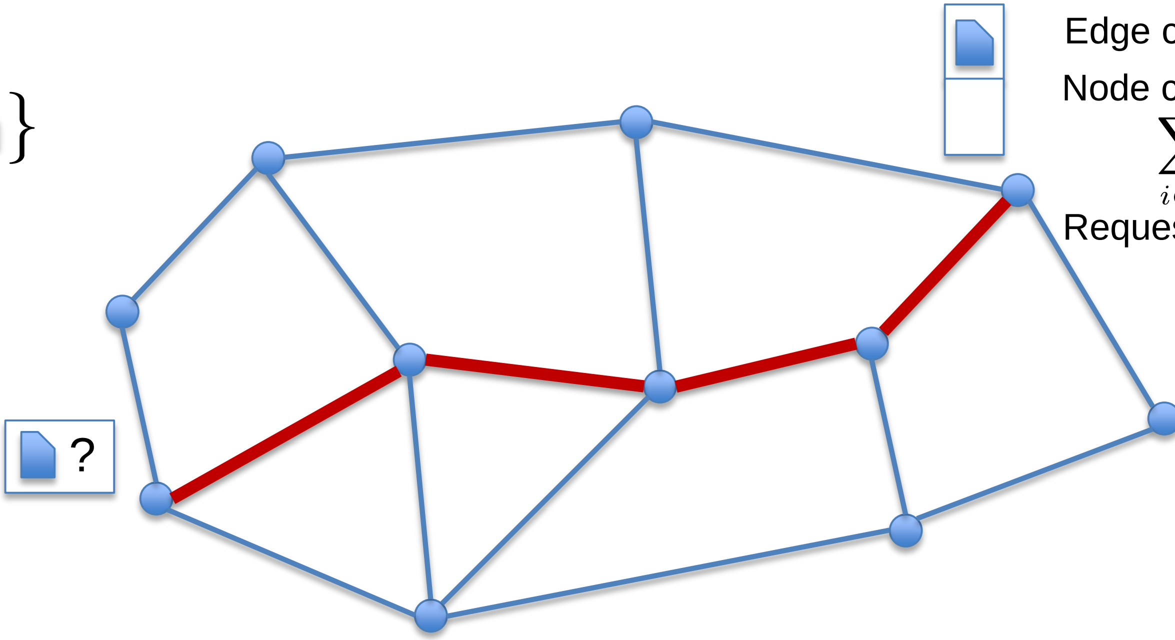
$\mathcal{R}$: demand

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$



Demand $\mathcal{R}$: set of all requests (i,p)

Request arrival process is Poisson with rate $\lambda_{(i,p)}$

Northeastern

Model: Routing Costs & Caching Gain

$G(V, E)$

$\mathcal{C} = \{ \quad \}$

$\mathcal{R}$: demand

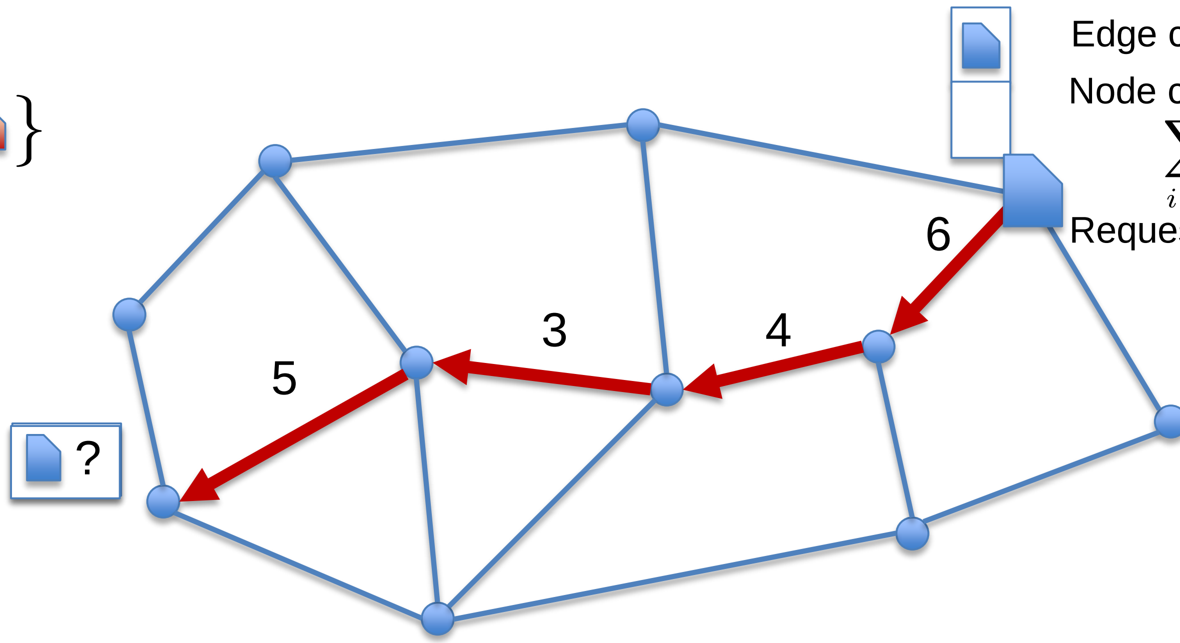
Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i, p) \in \mathcal{R}$

Request (i, p)



Worst case routing cost: $\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}$

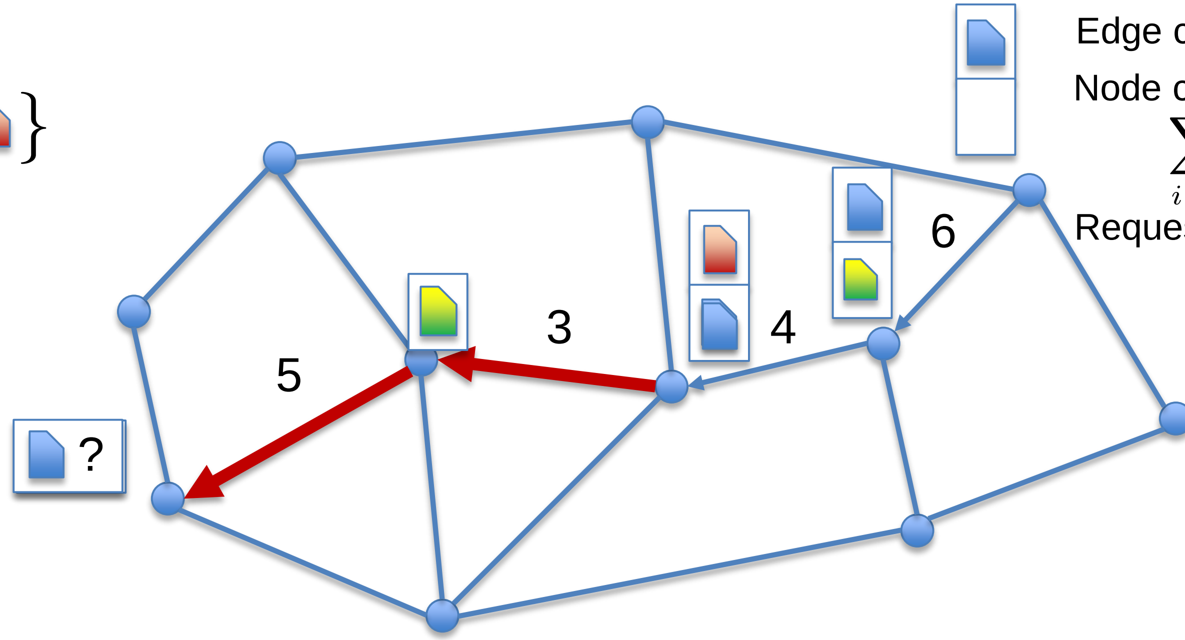
Northeastern

Model: Routing Costs & Caching Gain

$G(V, E)$

$\mathcal{C} = \{$  $\}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i, p) \in \mathcal{R}$

Worst case routing cost: $\sum_{k=1}^{|p|-1} w_{p_{k+1} p_k}$

Cost due to intermediate caching: $\sum_{k=1}^{|p|-1} w_{p_{k+1} p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'} i})$

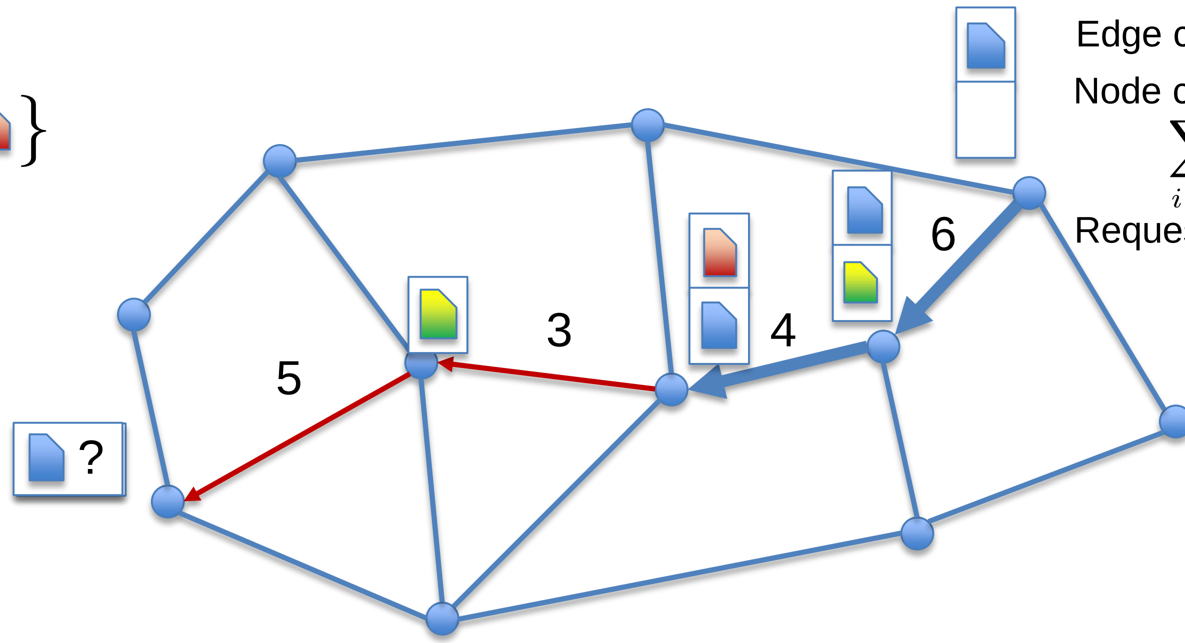
Model: Routing Costs & Caching Gain

$G(V, E)$

$\mathcal{C} = \{$  $\}$

$\mathcal{R}$: demand

Request (i,p)



Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$

Worst case routing cost: $\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}$

Cost due to intermediate caching: $\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'}i})$

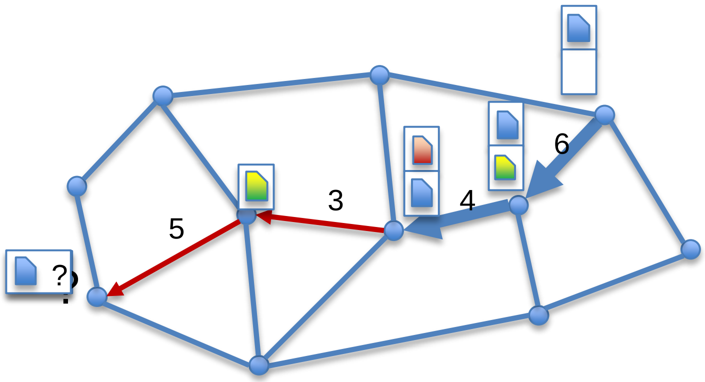
Caching Gain: $\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - x_{p_{k'}i})\right)$

Caching Gain Maximization

$G(V, E)$

$\mathcal{C} = \{$  $\}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$

Caching Gain:

$\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - x_{p_{k'}i})\right)$

The global **allocation strategy** is the binary $|V| \times |\mathcal{C}|$ matrix

$$X = [x_{vi}]_{v \in V, i \in \mathcal{C}}$$

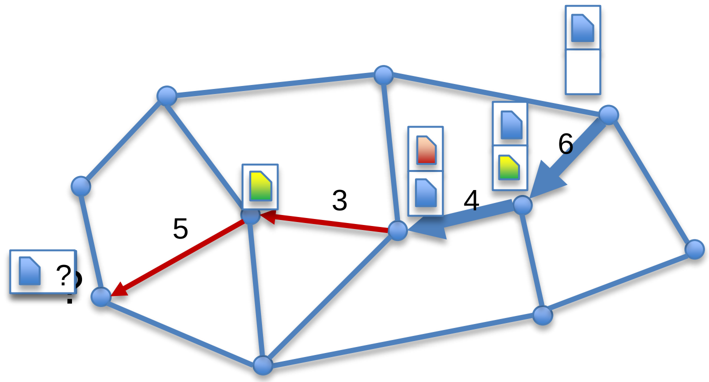
 Northeastern

Caching Gain Maximization

$G(V, E)$

$\mathcal{C} = \{\ \}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$

Caching Gain:

$\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}\left(1 - \prod_{k'=1}^{k}(1 - x_{p_{k'}i})\right)$

MaxCG

Maximize: $$F(X) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}\left(1 - \prod_{k'=1}^{k}(1 - x_{p_{k'}i})\right)$$

Subject to:

$\sum_{i \in \mathcal{C}} x_{vi} = c_v,$ for all $v \in V$

$x_{vi} = 1,$ for all $i \in \mathcal{C}$ and $v \in S_i$

$x_{vi} \in \{0, 1\},$ for all $v \in V$ and $i \in \mathcal{C}$

Offline Problem

Maximize: $$F(X) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}\Big(1-\prod_{k'=1}^{k}(1-x_{p_{k'}i})\Big)$$

Subject to:

$$\sum_{i\in\mathcal{C}} x_{vi} = c_v, \quad \text{for all } v \in V$$

$$x_{vi} = 1, \quad \text{for all } i \in \mathcal{C} \text{ and } v \in S_i$$

$$x_{vi} \in \{0,1\}, \quad \text{for all } v \in V \text{ and } i \in \mathcal{C}$$

Shanmugam, Golrezaei, Dimakis, Molisch, and Caire. *Femtocaching: Wireless Content Delivery Through Distributed Caching Helpers*. IT, 2013

- NP-hard
- **Submodula**r objective, **matroid constraints**
  - Greedy algorithm gives ½-approximation ratio
- 1-1/e ratio can be achieved through **pipage rounding** method [Ageev and Sviridenko, J. of Comb. Opt., 2004]

Pipage Rounding [Ageev & Sviridenko 2004]

Maximize: $F(X) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}\left(1 - \prod_{k'=1}^{k}(1 - x_{p_{k'}i})\right)$

Subject to: $\sum_{i\in\mathcal{C}} x_{vi} = c_v,$ for all $v \in V$

$x_{vi} = 1,$ for all $i \in \mathcal{C}$ and $v \in S_i$

$x_{vi} \in \{0,1\},$ for all $v \in V$ and $i \in \mathcal{C}$

Pipage Rounding [Ageev & Sviridenko 2004]

Maximize: $$F(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k}(1 - y_{p_{k'}i})\right)$$

Expected CG

Subject to:

$$\sum_{i\in\mathcal{C}} y_{vi} = c_v, \quad \text{for all } v \in V$$

$$y_{vi} = 1, \quad \text{for all } i \in \mathcal{C} \text{ and } v \in S_i$$

$$\boxed{y_{vi} \in [0,1]}, \quad \text{for all } v \in V \text{ and } i \in \mathcal{C}$$

Satisfied *in expectation*

Think:

- $y_{vi} = P(x_{vi} = 1)$
- All x_{vi} are **independent Bernoulli random variables.**

Pipage Rounding [Ageev & Sviridenko 2004]

Maximize: $$F(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k}(1 - y_{p_{k'}i})\right)$$

Subject to: $$\sum_{i\in\mathcal{C}} y_{vi} = c_v, \quad \text{for all } v \in V$$

$$y_{vi} = 1, \quad \text{for all } i \in \mathcal{C} \text{ and } v \in S_i$$

$$y_{vi} \in [0,1], \quad \text{for all } v \in V \text{ and } i \in \mathcal{C}$$

- **Key idea:** There exists a **concave** function $L(Y)$ such that

$$(1 - \frac{1}{e})L(Y) \leq F(Y) \leq L(Y)$$

Pipage Rounding [Ageev & Sviridenko 2004]

Maximize: $$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \textstyle\sum_{k'=1}^{k} y_{p'_k i}\}$$

Subject to:

$\sum_{i\in\mathcal{C}} y_{vi} = c_v,$ for all $v \in V$

$y_{vi} = 1,$ for all $i \in \mathcal{C}$ and $v \in S_i$

$y_{vi} \in [0,1]$, for all $v \in V$ and $i \in \mathcal{C}$

OFFLINE
CENTRALIZED

- **Key idea:** There exists a **concave** function $L(Y)$ such that

$$(1-\frac{1}{e})L(Y) \le F(Y) \le L(Y)$$

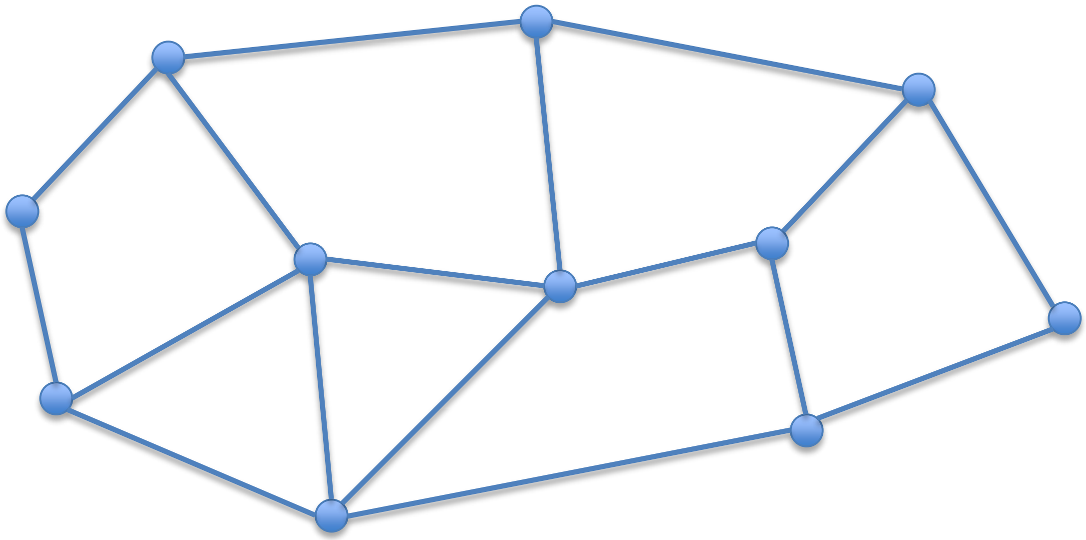
- **Algorithm Sketch**: Maximize $L(Y)$; round solution to obtain discrete solution X.

Overview

- ❑Problem Formulation
- ❑Distributed Adaptive Algorithms
- ❑Evaluation

Projected Gradient Ascent

$\mathcal{C} = \{$  $\}$

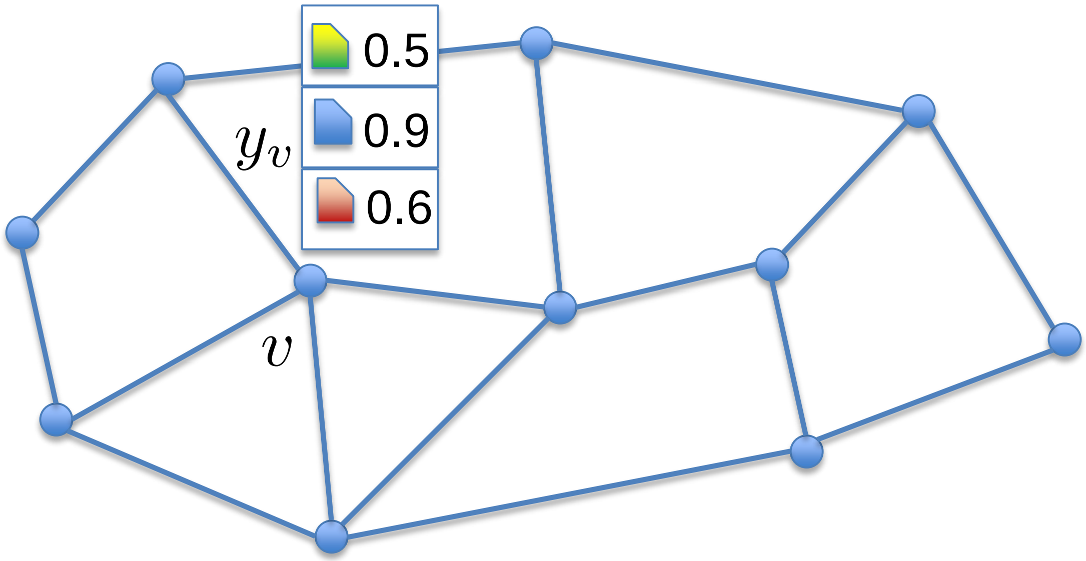


Time is divided into **slots**

Northeastern

Projected Gradient Ascent

$\mathcal{C}=\{$  $\}$ $Y = [y_v]_{v\in V}$

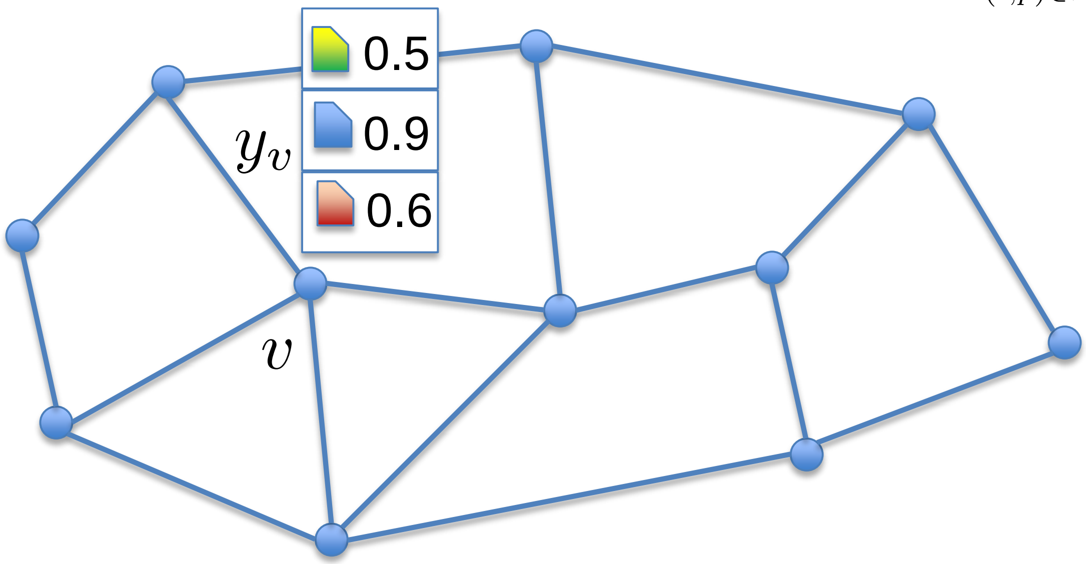


Each node $v \in V$ keeps track of its own marginal distribution $y_v \in [0,1]^{|\mathcal{C}|}$

Projected Gradient Ascent

$\mathcal{C} = \{$ [three content icons] $\}$ $\quad Y = [y_v]_{v \in V}$

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \sum_{k'=1}^{k} y_{p'_k i}\}$$

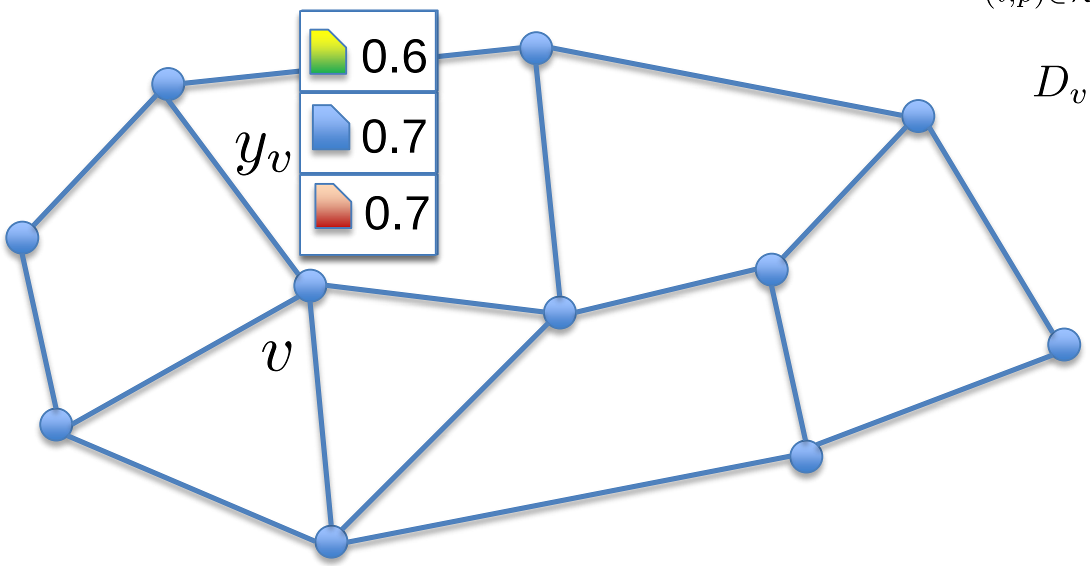


During a slot, v estimates $\nabla_{y_v} L(Y)$ by collecting measurements **through passing packets**.

Projected Gradient Ascent

$\mathcal{C}=\{\quad\}$ $\quad Y=[y_v]_{v\in V}$

$$L(Y)=\sum_{(i,p)\in\mathcal{R}}\lambda_{(i,p)}\sum_{k=1}^{|p|-1}w_{p_{k+1}p_k}\min\{1,\sum_{k'=1}^{k}y_{p'_k i}\}$$



$$D_v=\{y_v,\ \text{s.t.}\ \sum_{i\in\mathcal{C}}y_{vi}=c_v,y_{vi}\in[0,1]\}$$

At the conclusion of the k-th slot, v updates its marginals through:

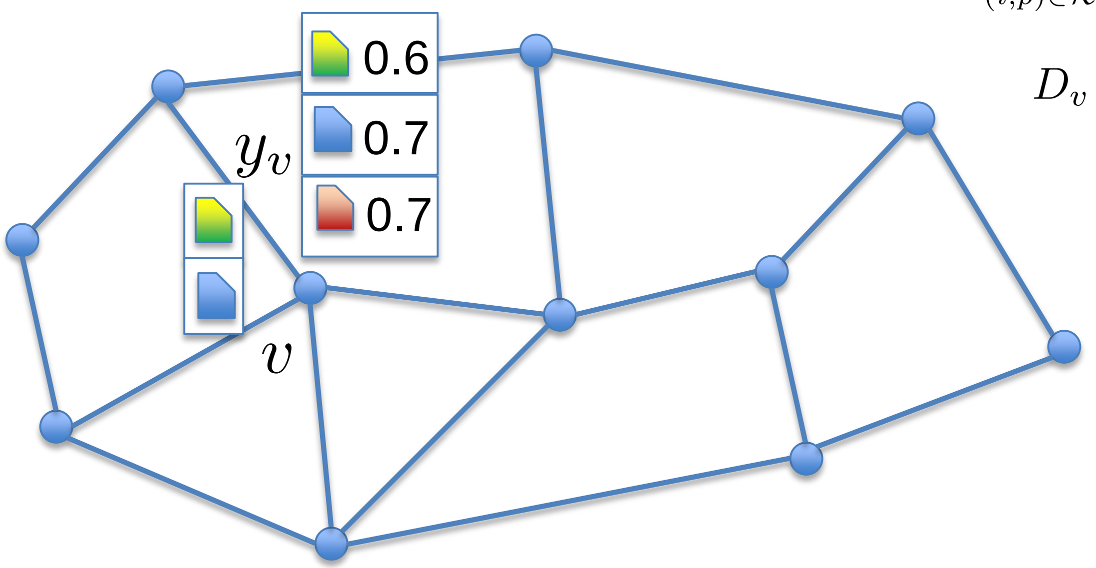
$$y_v\leftarrow\Pi_{D_v}(y_v+\gamma_k\nabla_{y_v}L(Y))$$

Projected Gradient Ascent

$\mathcal{C}=\{$  $\}$ $\quad Y = [y_v]_{v \in V}$

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \sum_{k'=1}^{k} y_{p'_k i}\}$$

$$D_v = \{y_v, \text{ s.t. } \sum_{i\in\mathcal{C}} y_{vi} = c_v, y_{vi} \in [0,1]\}$$



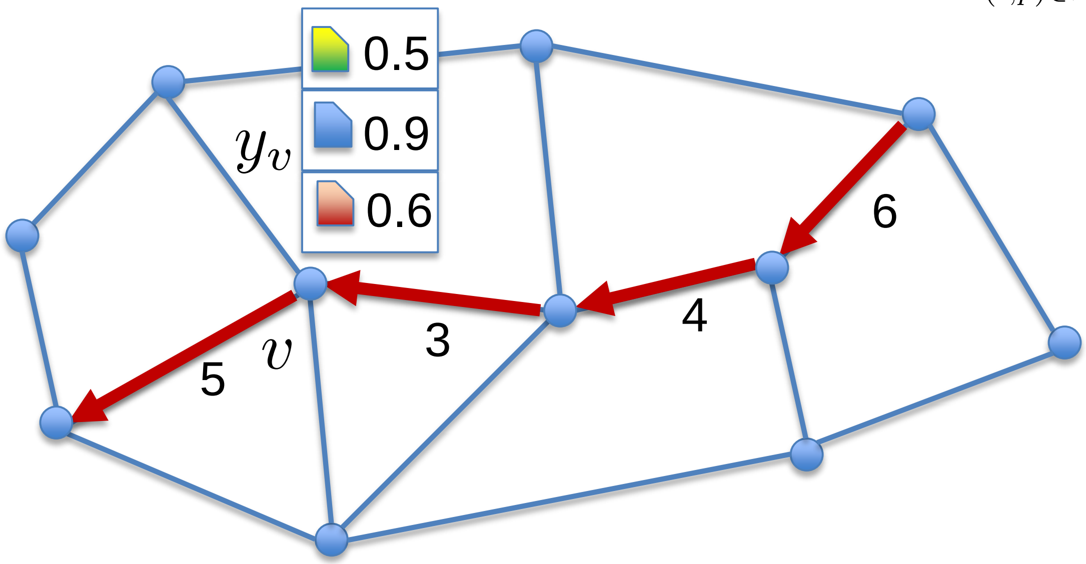
After updating y_v, node v places c_v **random** items in its cache, **independently of other nodes**, so that:

$$P(x_{vi} = 1) = y_{vi} \,, \qquad \text{for all } i \in \mathcal{C}$$

Gradient Estimation

$\mathcal{C} = \{$  $\}$ $\quad Y = [y_v]_{v \in V}$

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \sum_{k'=1}^{k} y_{p'_k i}\}$$

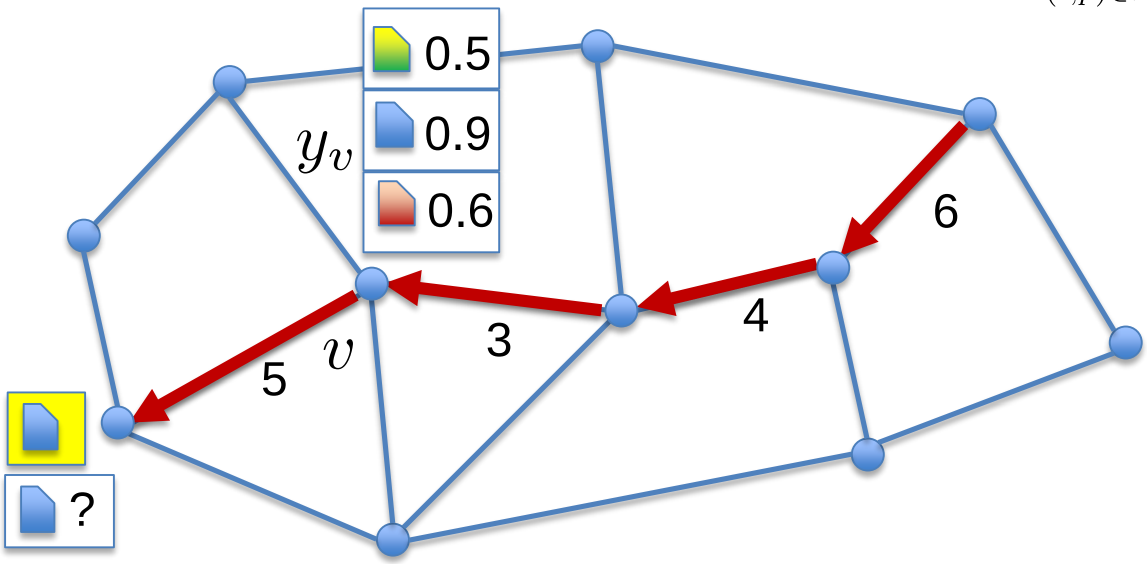


How can v *estimate* $\nabla_{y_v} L(Y)$ *in a distributed fashion?*

Gradient Estimation

$\mathcal{C} = \{$  $\}$ $\quad Y = [y_v]_{v \in V}$

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \textstyle\sum_{k'=1}^{k} y_{p'_k i}\}$$



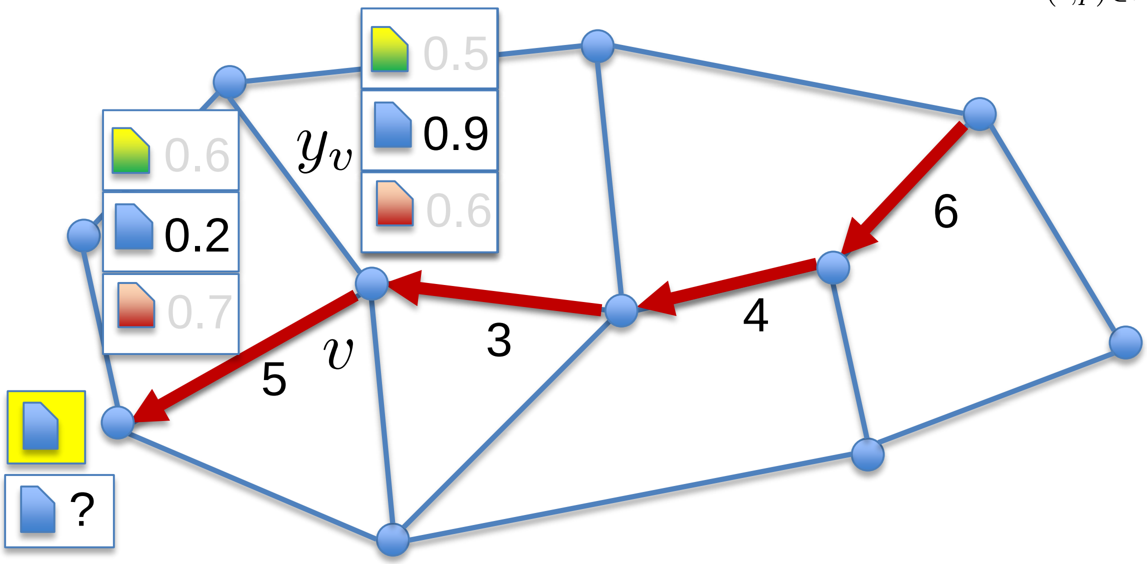
When request (i, p) is generated, create a new **control message**

Northeastern

Gradient Estimation

$\mathcal{C}=\{$  $\}$ $Y = [y_v]_{v\in V}$

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \textstyle\sum_{k'=1}^{k} y_{p'_k i}\}$$



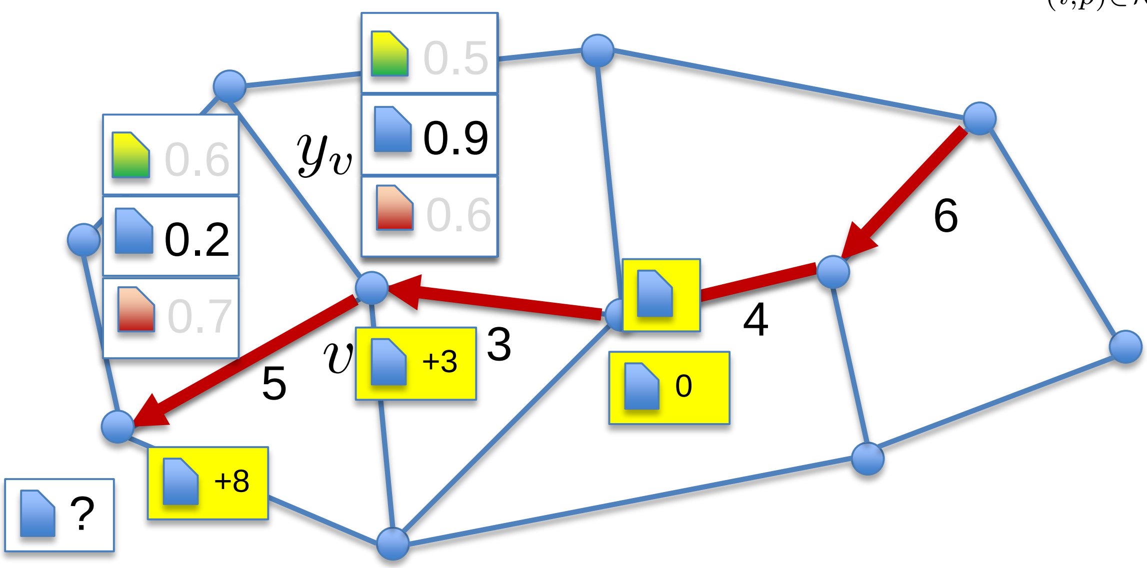
Forward control message over path p until: $\sum_{k'=1}^{k} y_{p_{k'} i} > 1$

Northeastern

Gradient Estimation

$\mathcal{C}=\{\quad\}$ $\quad Y=[y_v]_{v\in V}$

$$L(Y)=\sum_{(i,p)\in\mathcal{R}}\lambda_{(i,p)}\sum_{k=1}^{|p|-1}w_{p_{k+1}p_k}\min\{1,\sum_{k'=1}^{k}y_{p'_k i}\}$$



Forward until: $\sum_{k'=1}^{k} y_{p_{k'} i} > 1$

Send control message over reverse path, collecting **sum of edge costs.**

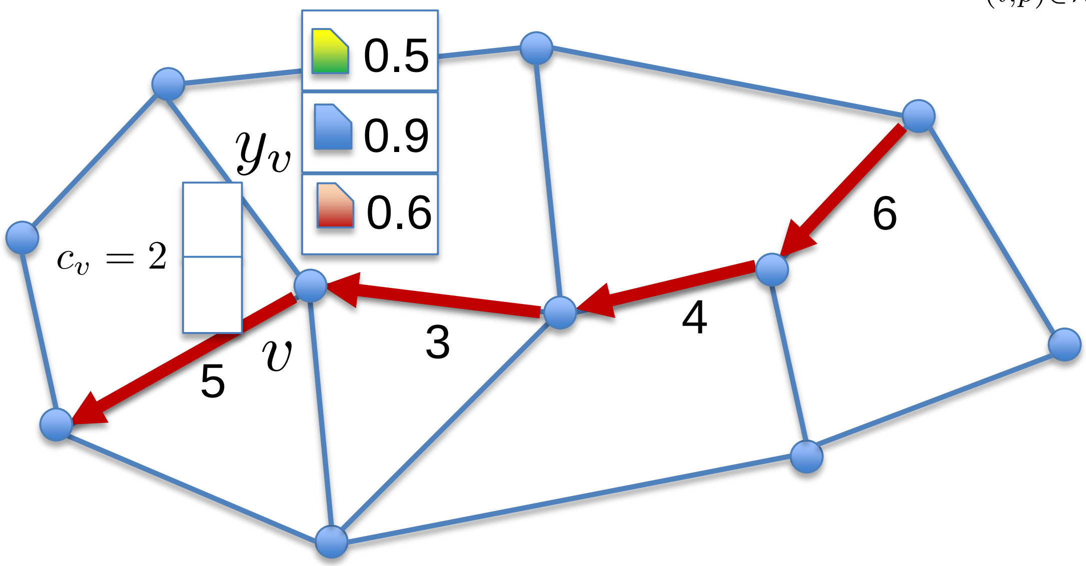
Each node on reverse path, **sniffs upstream costs**, and **maintains average** per item $i\in\mathcal{C}$.

Average at end of slot is estimate of $\dfrac{\partial L(Y)}{\partial y_{vi}}$

Randomized Placement

$\mathcal{C} = \{$ ■ ■ ■ $\}$ $\quad Y = [y_v]_{v \in V}$

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\{1, \sum_{k'=1}^{k} y_{p'_k i}\}$$

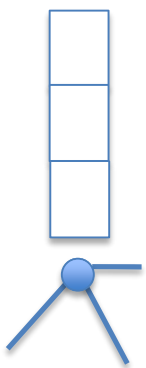


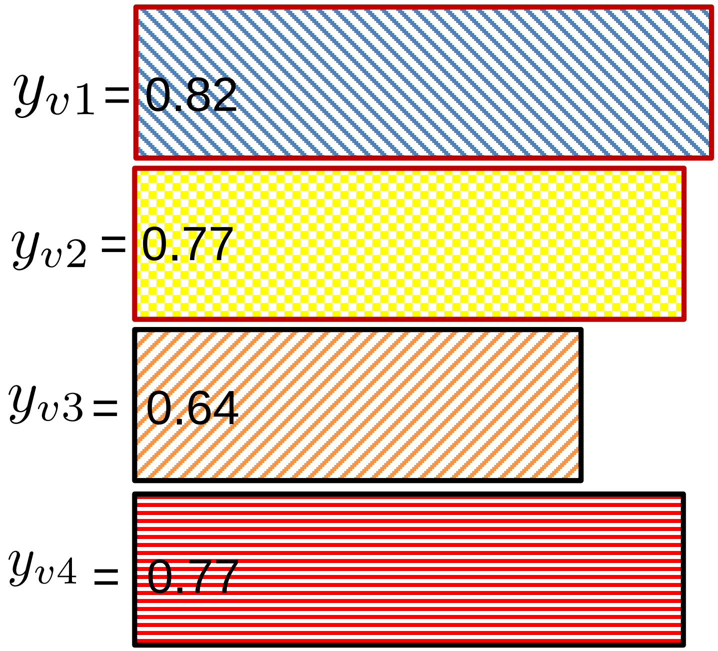
*How can v place **exactly** c_v items in its cache, so that marginals are satisfied?*

Randomized Placement

$$\mathcal{C} = \{1,2,3,4\}$$

$$c_v = 3$$





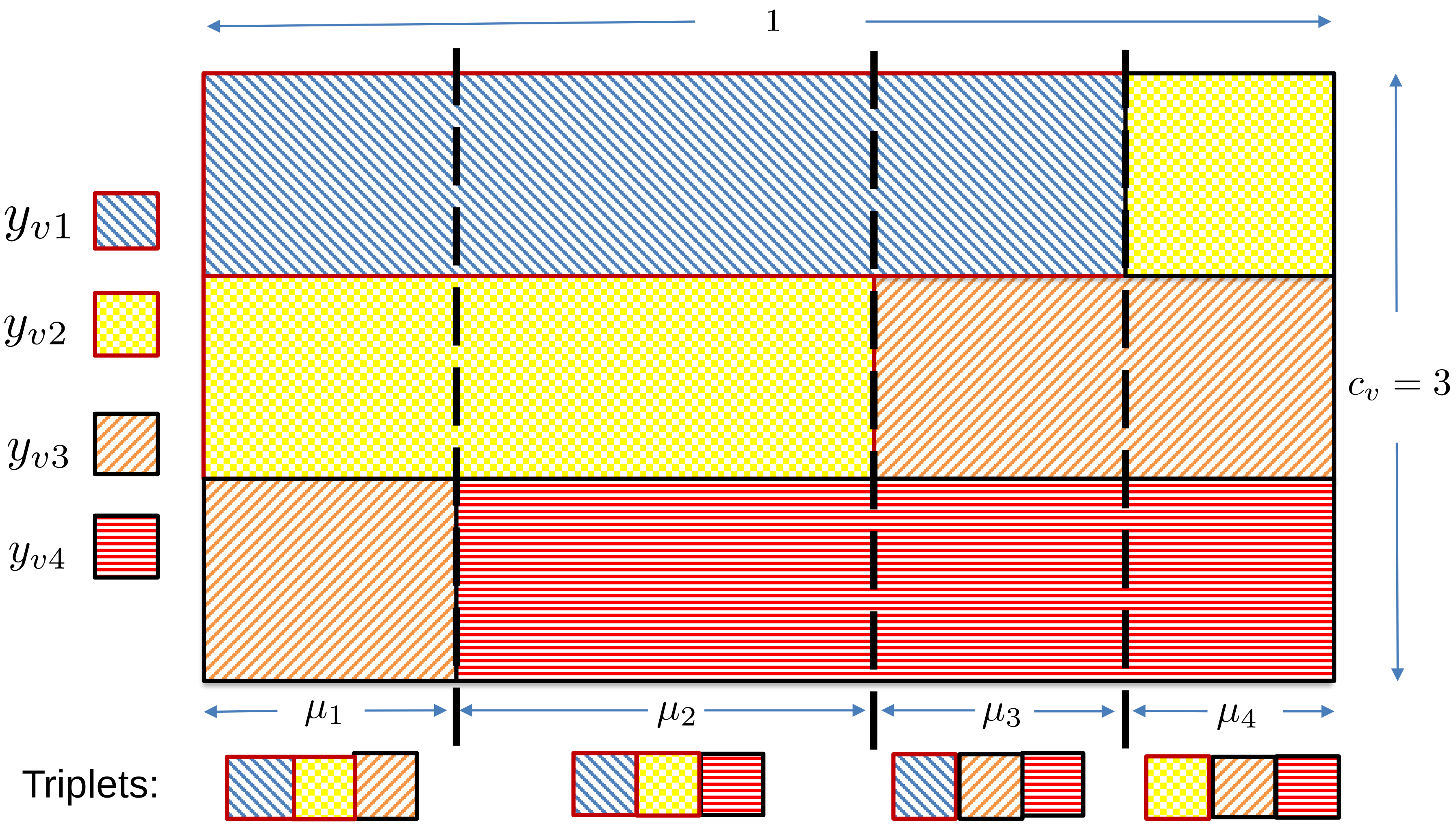
$$\sum_{i\in\mathcal{C}} y_{vi} = 3$$

Suppose that I give you a $y_v \in [0,1]^{|\mathcal{C}|}$ such that $\sum_{i\in\mathcal{C}} y_{vi} = c_v$.

Is there a way to select **exactly** c_v **items at random**, so that the probability that item i is selected is y_{vi}?

Northeastern

Randomized Placement: Sketch of Algorithm



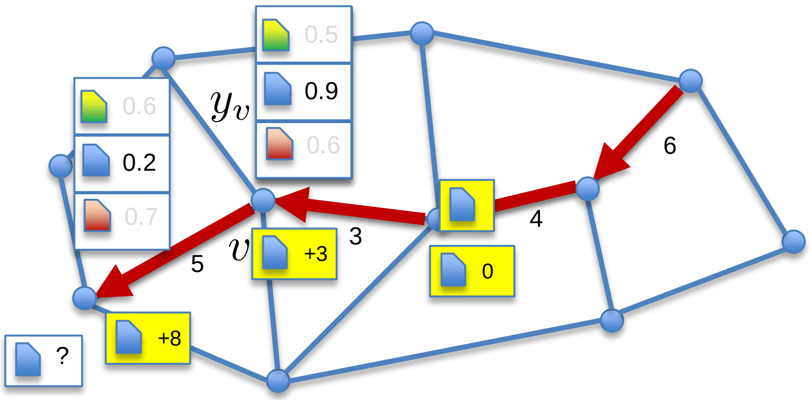
Convergence

Theorem: For $\gamma_k = 1/\sqrt{k}$, Projected Gradient Ascent leads to an allocation X_k such that

$$\lim_{k\to\infty} \mathbb{E}[F(X_k)] \geq (1-\frac{1}{e})F(X^*)$$

where X^* an optimal solution to the offline problem.

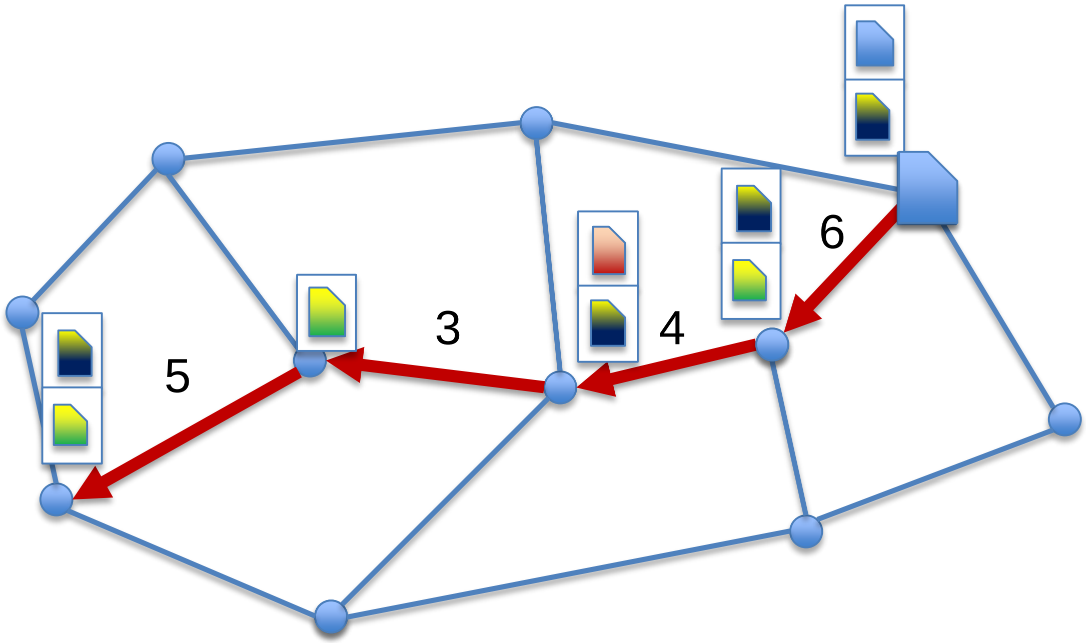
Projected Gradient Ascent (vs. Path Replication)



- ✓ Distributed
- ✓ Adaptive
- ✓ Constant Approximation to Optimal

- ✗ Overhead for control traffic
- ✗ Overhead to retrieve content at end of timeslot
- ✗ Not so simple…

Path-Replication + Greedy Eviction Policy

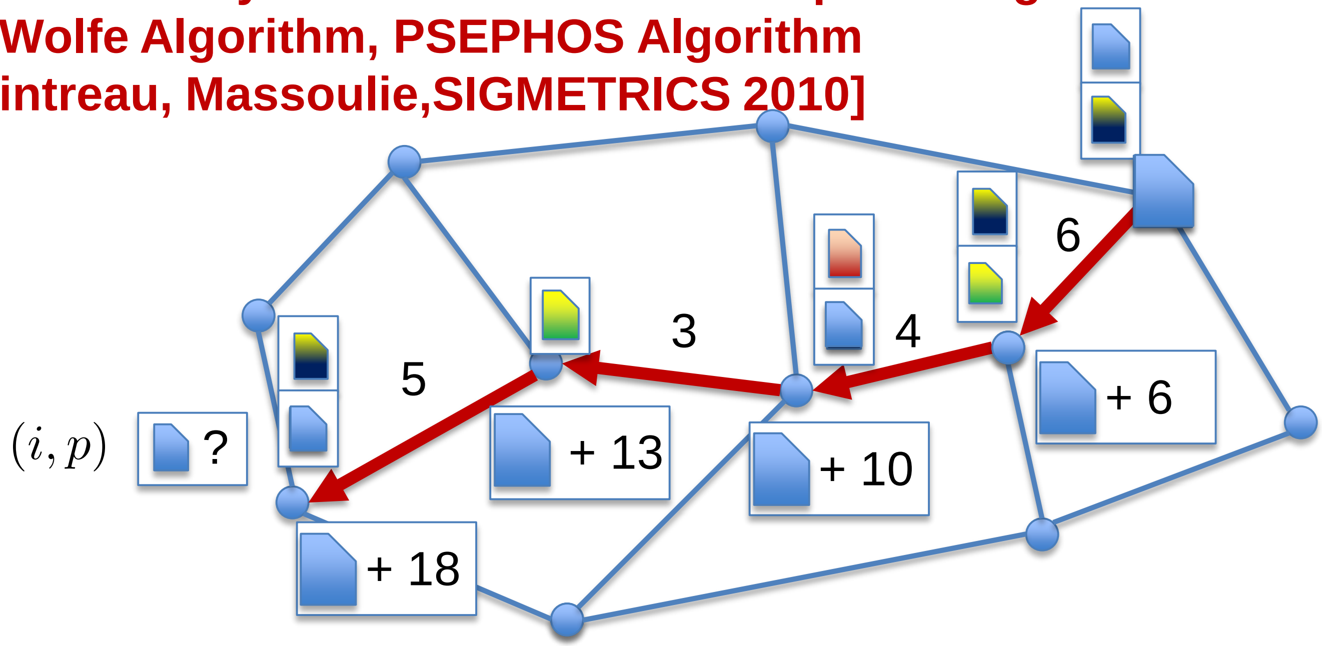


- Each node v maintains an estimate for the (sub)gradient $\partial y_v L(X)$

- At any point in time, v caches "top" c_v items, with highest gradients $\dfrac{\partial L(X)}{\partial y_{vi}}$

Path-Replication + Greedy Eviction

Intuition: Greedily cache item with best "upstream gain"
Frank-Wolfe Algorithm, PSEPHOS Algorithm
[I.,Chaintreau, Massoulie,SIGMETRICS 2010]

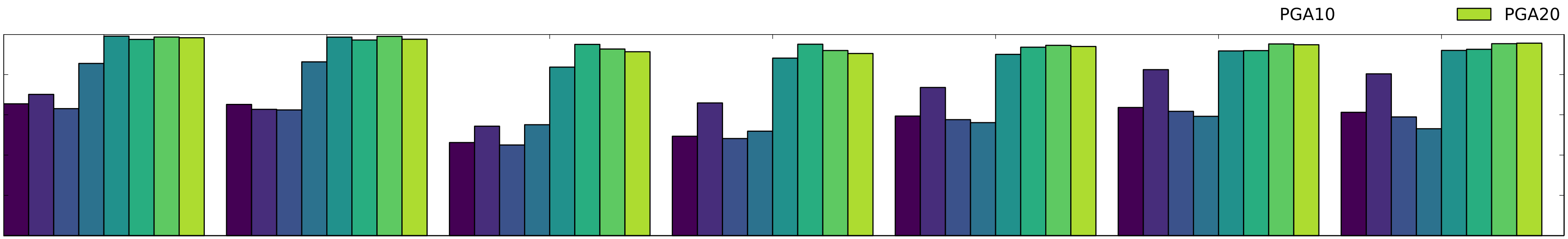


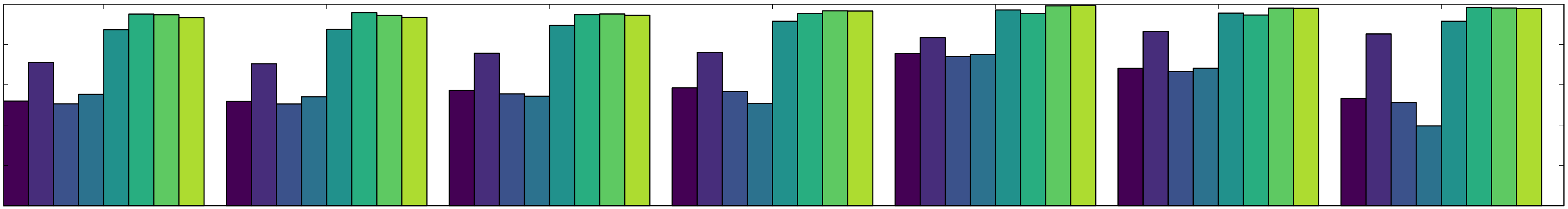
- A response carrying the item i **adds weights on the reverse path**, and reports them to intermediate nodes.
- Weights are used to update estimate of $\frac{\partial L(X)}{\partial y_{vi}}$.
- **Greedy Eviction**: if i becomes one of the top c_v items, evict item with smallest gradient, and cache i.

Overview

- ❑Problem Formulation
- ❑Distributed Algorithms
- ❑Evaluation

Multiple Topologies





y-axis: ratio to offline solution

Open Questions

- Joint caching & routing
- PR+Greedy Eviction guarantees
- Delay vs. Throughput Optimality
- Broader resource management applications

Reference

S. Ioannidis and E. Yeh, “Adaptive Caching Networks with Optimality Guarantees.” Proc. ACM SIGMETRICS, Antibes Juan-les-Pins, June 14-18, 2016, pp. 113-124.

Northeastern

Thank you!