Order Optimal Information Spreading Using Algebraic Gossip

Chen Avin, Michael Borokhovich, Keren Censor-Hillel, Zvi Lotker

Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Israel

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA

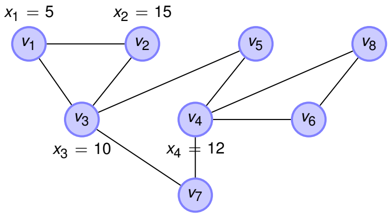
Israeli Networking Day 2011

(To be presented in PODC11)

Motivation

- Wireless (sensor) networks and peer-to-peer networks need efficient algorithms for information dissemination.
- In such networks, there is no central management entity, thus local, distributed algorithms are needed.
- Network Coding with gossip algorithms (a.k.a. Algebraic Gossip) will help us to achieve faster information dissemination.
- We look at: k nodes want to disseminate their value to all other nodes in the network.

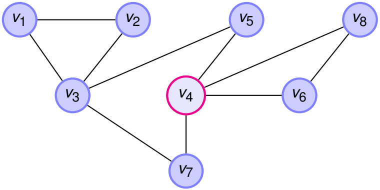
Information Spreading - The k-Dissemination Problem



- A network represented by a graph $G(V, E)$. $V = \{v_1, v_2, \ldots, v_n\}$
- $k \leq n$ values $\{x_1, x_2, \ldots, x_k\}$ need to be distributed to all nodes
- A node knows only its neighbors
- Limited messages size

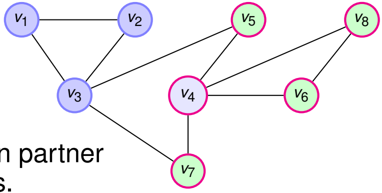
Gossip Algorithm – The Way Information is Spread

In each round **every node** takes a **gossip action**



Gossip Algorithm – The Way Information is Spread

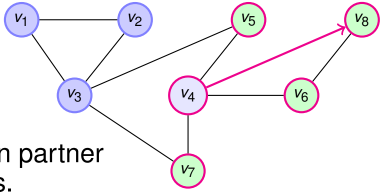
In each round **every node** takes a **gossip action**



- Gossip Algorithm:
  1. Determines a communication partner (randomly) among neighbors.
     - Uniform gossip.
     - Non uniform gossip.

Gossip Algorithm – The Way Information is Spread

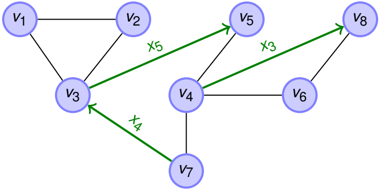
In each round **every node** takes a **gossip action**



- Gossip Algorithm:
  1. Determines a communication partner (randomly) among neighbors.
     - Uniform gossip.
     - Non uniform gossip.
  2. Determines how the message is sent.
     - **PUSH** – a message is sent to the partner.
     - **PULL** – a message is sent from the partner.
     - **EXCHANGE** - PUSH and PULL.

Algebraic Gossip is Based on Random Linear Network Coding

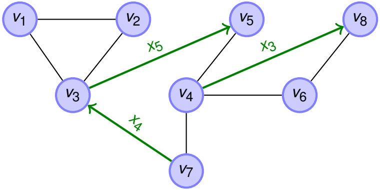
instead of sending randomly chosen values...



every message – a single value: msg $= x_i$

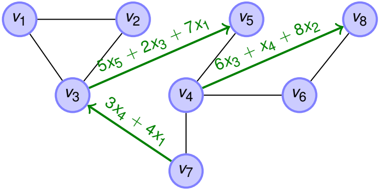
Algebraic Gossip is Based on Random Linear Network Coding

instead of sending randomly chosen values...



every message – a single value: msg $= x_i$

nodes send random linear combinations

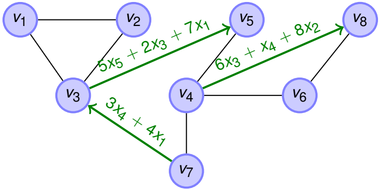


every message – linear equation: msg $= \sum a_i x_i$

- Nodes (routers) can manipulate packets.
- All operations are in a field $\mathcal{F}$ so messages size is (almost) the same in both cases.

Algebraic Gossip is Based on Random Linear Network Coding

nodes send random linear combinations



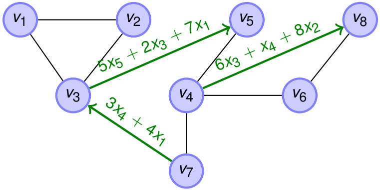
every message – linear equation

linear equations are stored in a matrix form:

$$\begin{bmatrix} 4 & 3 & 7 & 6 \\ 2 & 0 & 0 & 7 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 22 \\ 45 \\ 78 \\ 30 \end{bmatrix}$$

Algebraic Gossip is Based on Random Linear Network Coding

nodes send random linear combinations



every message – linear equation

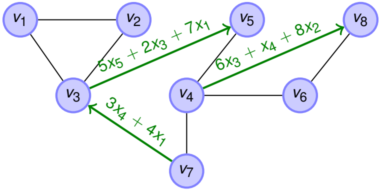
linear equations are stored in a matrix form:

$$\begin{bmatrix} 4 & 3 & 7 & 6 \\ 2 & 0 & 0 & 7 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 22 \\ 45 \\ 78 \\ 30 \end{bmatrix}$$

once a node has rank k – it finishes

Algebraic Gossip is Based on Random Linear Network Coding

nodes send random linear combinations



every message – linear equation

linear equations are stored in a matrix form:

$$\begin{bmatrix} 4 & 3 & 7 & 6 \\ 2 & 0 & 0 & 7 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 22 \\ 45 \\ 78 \\ 30 \end{bmatrix}$$

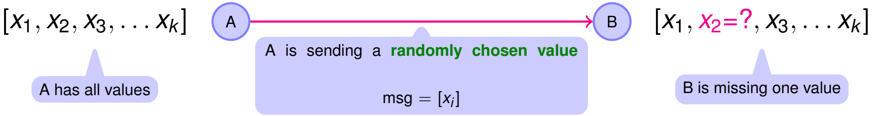
once a node has rank k – it finishes

only **helpful** messages are stored

messages that increase the rank of the matrix

So, Why Algebraic Gossip is Faster?

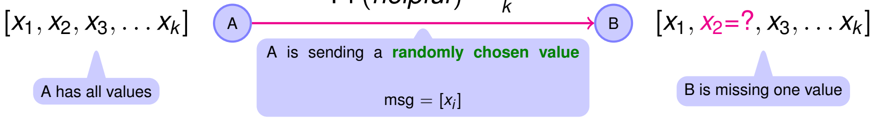
Without Algebraic Gossip (Random Message Selection)



So, Why Algebraic Gossip is Faster?

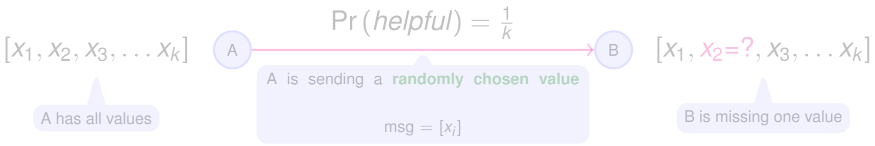
Without Algebraic Gossip (Random Message Selection)

$$\Pr(helpful) = \frac{1}{k}$$

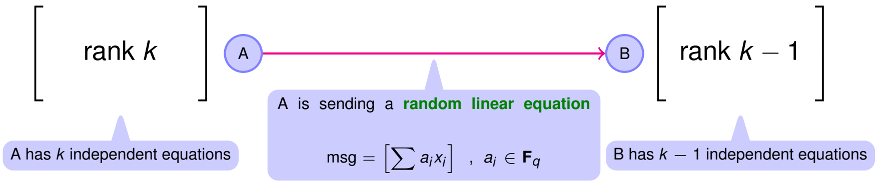


So, Why Algebraic Gossip is Faster?

Without Algebraic Gossip (Random Message Selection)



With Algebraic Gossip (Random Linear Equations)



So, Why Algebraic Gossip is Faster?

Without Algebraic Gossip (Random Message Selection)

$$\Pr(helpful) = \frac{1}{k}$$

$[x_1, x_2, x_3, \dots x_k]$ A → B $[x_1, x_2{=}?, x_3, \dots x_k]$

A has all values

A is sending a **randomly chosen value**

msg $= [x_i]$

B is missing one value

With Algebraic Gossip (Random Linear Equations)

$$\Pr(helpful) = 1 - \frac{1}{q}$$

$\begin{bmatrix} \text{rank } k \end{bmatrix}$ A → B $\begin{bmatrix} \text{rank } k-1 \end{bmatrix}$

A has k independent equations

A is sending a **random linear equation**

msg $= \left[\sum a_i x_i\right]$, $a_i \in \mathbf{F}_q$

B has $k-1$ independent equations

So, Why Algebraic Gossip is Faster?

Without Algebraic Gossip (Random Message Selection)

$$\Pr(\textit{helpful}) = \frac{1}{k}$$

$[x_1, x_2, x_3, \ldots x_k]$ A → B $[x_1, x_2{=}?, x_3, \ldots x_k]$

A has all values

A is sending a **randomly chosen value**

msg $= [x_i]$

B is missing one value

With Algebraic Gossip (Random Linear Equations)

$$\Pr(\textit{helpful}) = 1 - \frac{1}{q}$$

[rank k] A → B [rank $k-1$]

A has k independent equations

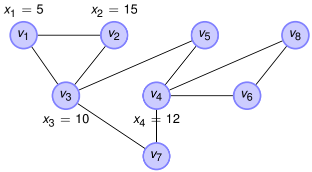
A is sending a **random linear equation**

msg $= \left[\sum a_i x_i\right]$, $a_i \in \mathbf{F}_q$

B has $k-1$ independent equations

$$\Pr(\textit{helpful}) = \frac{q^k - q^{k-1}}{q^k} = 1 - \frac{1}{q}$$

Research Goal – Optimal Protocol for k-Dissemination Problem



1. Analyze uniform algebraic gossip for k-dissemination
   - Is it optimal?
   - For which graphs?
2. Study non-uniform gossip to achieve optimal k-dissemination

Related Work on Algebraic Gossip

- Trivial lower bound – $\Omega(k)$, kn messages needed to be delivered so k rounds.

Related Work on Algebraic Gossip

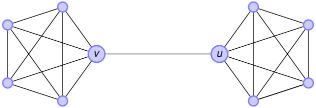
- Trivial lower bound – $\Omega(k)$, kn messages needed to be delivered so k rounds.
- [Deb et al., 2006] – (almost) Tight bound for the complete graph. $\Theta(k)$, for $k \gg \ln^3 n$. (push/pull)

Related Work on Algebraic Gossip

- Trivial lower bound – $\Omega(k)$, kn messages needed to be delivered so k rounds.
- [Deb et al., 2006] – (almost) Tight bound for the complete graph. $\Theta(k)$, for $k \gg \ln^3 n$. (push/pull)
- [Mosk-Aoyama and Shah, 2006] – n-dissemination. Upper bound for **arbitrary graphs**, based on *conductance* measure. The bound is not tight. For complete graph: $O(n \log n)$, for Ring: $O(n^2)$.

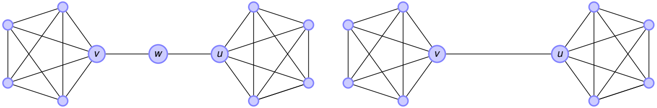
Related Work on Algebraic Gossip

- Trivial lower bound – $\Omega(k)$, kn messages needed to be delivered so k rounds.
- [Deb et al., 2006] – (almost) Tight bound for the complete graph. $\Theta(k)$, for $k \gg \ln^3 n$. (push/pull)
- [Mosk-Aoyama and Shah, 2006] – n-dissemination. Upper bound for **arbitrary graphs**, based on *conductance* measure. The bound is not tight. For complete graph: $O(n \log n)$, for Ring: $O(n^2)$.
- [BAL. ISIT10] – n-dissemination. Upper bound for any graph: $O(\Delta n)$. Tight bound of $\Theta(n)$ for *constant degree graphs*. Worst case graph for algebraic gossip (barbell): $\Omega(n^2)$.



Related Work on Algebraic Gossip

- Trivial lower bound – $\Omega(k)$, kn messages needed to be delivered so k rounds.
- [Deb et al., 2006] – (almost) Tight bound for the complete graph. $\Theta(k)$, for $k \gg \ln^3 n$. (push/pull)
- [Mosk-Aoyama and Shah, 2006] – n-dissemination. Upper bound for **arbitrary graphs**, based on *conductance* measure. The bound is not tight. For complete graph: $O(n \log n)$, for Ring: $O(n^2)$.
- [BAL. ISIT10] – n-dissemination. Upper bound for any graph: $O(\Delta n)$. Tight bound of $\Theta(n)$ for *constant degree graphs*. Worst case graph for algebraic gossip (barbell): $\Omega(n^2)$.
- Open question: What graph property capture the stopping time?



Related Work on Algebraic Gossip

- [Haeupler. STOC11] – k-dissemination. Conductance and expansion based arguments. Two parameters: γ and λ.
  - Tight bound for the case $k = \Omega(n)$: $\Theta(n/\gamma)$
  - For $k < o(n)$: $O(k/\gamma + \log^2 n/\lambda)$. The bound is not tight for e.g., line: $O(k + n \log^2 n)$, grid: $O(k + \sqrt{n} \log^2 n)$, binary tree: $O(k + n \log^2 n)$
  - Gave also results for dynamic networks

1st Result: k-Dissemination With Uniform Algebraic Gossip

Theorem 1

For any graph with n nodes, diameter D, and maximum degree Δ, stopping time of **uniform** algebraic gossip is $O(\Delta(k + \log n + D))$ with high probability.

1st Result: k-Dissemination With Uniform Algebraic Gossip

Theorem 1

For any graph with n nodes, diameter D, and maximum degree Δ, stopping time of **uniform** algebraic gossip is $O(\Delta(k + \log n + D))$ with high probability.

Corollary 1

For any graph with n nodes and with **constant** maximum degree, stopping time of **uniform** algebraic gossip is $\Theta(k + D)$ in the **synchronous** time model.

1st Result: k-Dissemination With Uniform Algebraic Gossip

Theorem 1

For any graph with n nodes, diameter D, and maximum degree Δ, stopping time of **uniform** algebraic gossip is $O(\Delta(k + \log n + D))$ with high probability.

Corollary 1

For any graph with n nodes and with **constant** maximum degree, stopping time of **uniform** algebraic gossip is $\Theta(k + D)$ in the **synchronous** time model.

- Tight for e.g., Line, Cycle, Grids, Binary Trees, etc.

1st Result: k-Dissemination With Uniform Algebraic Gossip

Theorem 1

For any graph with n nodes, diameter D, and maximum degree Δ, stopping time of **uniform** algebraic gossip is $O(\Delta(k + \log n + D))$ with high probability.

Corollary 1

For any graph with n nodes and with **constant** maximum degree, stopping time of **uniform** algebraic gossip is $\Theta(k + D)$ in the **synchronous** time model.

- Tight for e.g., Line, Cycle, Grids, Binary Trees, etc.
- The result holds for any gossip variation: Push, Pull, Exchange.

1st Result: k-Dissemination With Uniform Algebraic Gossip

Theorem 1

For any graph with n nodes, diameter D, and maximum degree Δ, stopping time of **uniform** algebraic gossip is $O(\Delta(k + \log n + D))$ with high probability.

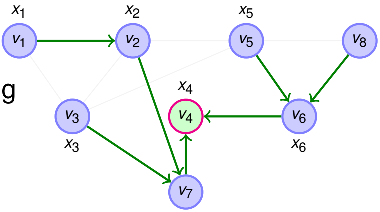
Corollary 1

For any graph with n nodes and with **constant** maximum degree, stopping time of **uniform** algebraic gossip is $\Theta(k + D)$ in the **synchronous** time model.

- Tight for e.g., Line, Cycle, Grids, Binary Trees, etc.
- The result holds for any gossip variation: Push, Pull, Exchange.
- When does the uniform algebraic gossip perform bad? e.g., $\Omega(kn)$ for a barbell graph.

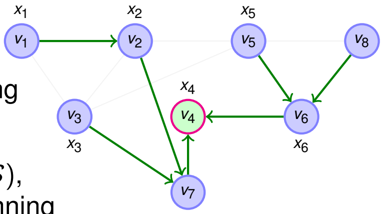
2nd Result: k-Dissemination With TAG

- Construct a spanning tree of the graph using some gossip spanning tree protocol $\mathcal{S}$.

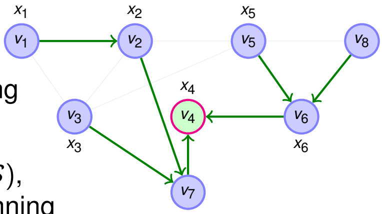


2nd Result: k-Dissemination With TAG

- Construct a spanning tree of the graph using some gossip spanning tree protocol $\mathcal{S}$.
  - The stopping time of $\mathcal{S}$ is $t(\mathcal{S})$, and the diameter of the spanning tree is $d(\mathcal{S})$.

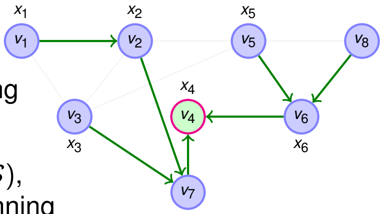


2nd Result: k-Dissemination With TAG



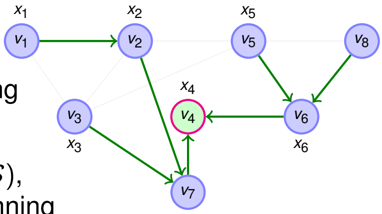
- Construct a spanning tree of the graph using some gossip spanning tree protocol $\mathcal{S}$.
  - The stopping time of $\mathcal{S}$ is $t(\mathcal{S})$, and the diameter of the spanning tree is $d(\mathcal{S})$.
  - Spanning tree can be constructed e.g., by a randomize broadcast protocol, or more sophisticated method.

2nd Result: k-Dissemination With TAG



- Construct a spanning tree of the graph using some gossip spanning tree protocol $\mathcal{S}$.
    - The stopping time of $\mathcal{S}$ is $t(\mathcal{S})$, and the diameter of the spanning tree is $d(\mathcal{S})$.
    - Spanning tree can be constructed e.g., by a randomize broadcast protocol, or more sophisticated method.
- Once the tree is constructed, every node knows its parent.

2nd Result: k-Dissemination With TAG



- Construct a spanning tree of the graph using some gossip spanning tree protocol $\mathcal{S}$.
  - The stopping time of $\mathcal{S}$ is $t(\mathcal{S})$, and the diameter of the spanning tree is $d(\mathcal{S})$.
  - Spanning tree can be constructed e.g., by a randomize broadcast protocol, or more sophisticated method.
- Once the tree is constructed, every node knows its parent.
- Perform algebraic gossip, where every node uses a single communication partner – its parent.
  Notice, we have here **non-uniform** algebraic gossip.

2nd Result: k-Dissemination With TAG

Theorem 2

For any graph with n nodes,
stopping time of TAG protocol is $O(k + \log n + d(\mathcal{S}) + t(\mathcal{S}))$
with high probability, where:
$t(\mathcal{S})$ – stopping time of the gossip spanning tree protocol $\mathcal{S}$.
$d(\mathcal{S})$ – diameter of the spanning tree created by $\mathcal{S}$.

2nd Result: *k*-Dissemination With TAG

Theorem 2

For any graph with n nodes,
stopping time of TAG protocol is $O(k + \log n + d(\mathcal{S}) + t(\mathcal{S}))$ with high probability, where:
$t(\mathcal{S})$ – stopping time of the gossip spanning tree protocol $\mathcal{S}$.
$d(\mathcal{S})$ – diameter of the spanning tree created by $\mathcal{S}$.

Corollary 2

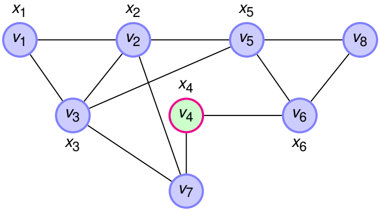
For any graph with n nodes, and for $k = \Omega(k)$,
TAG protocol is **order optimal** for k-dissemination task, i.e., the stopping time is $\Theta(n)$ with high probability.

Proof Overview

k-dissemination with **uniform** algebraic gossip

Proof Overview – Converting a Graph to a System of Queues

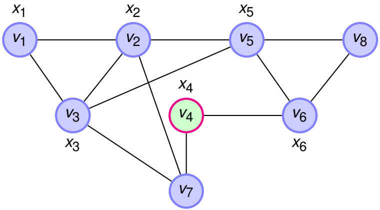
initial graph with an arbitrary node v_4



when v_4 will finish the protocol?

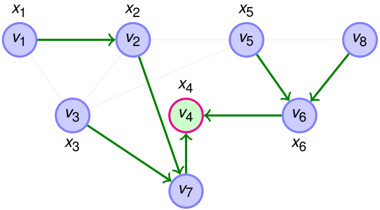
Proof Overview – Converting a Graph to a System of Queues

initial graph with an arbitrary node v_4



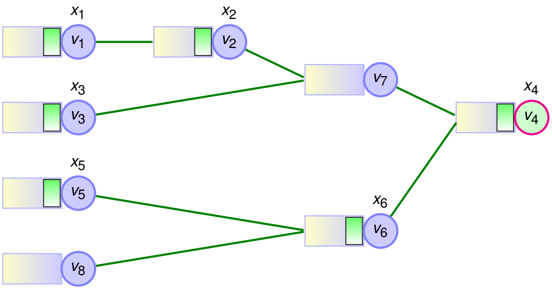
when v_4 will finish the protocol?

BFS spanning tree rooted at v_4



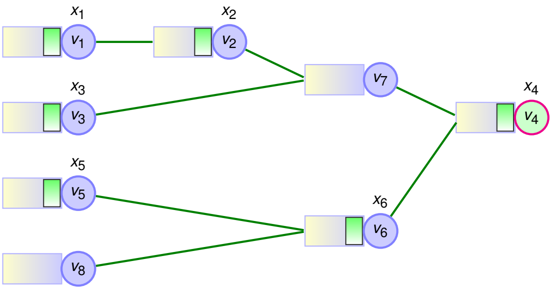
we **ignore** the messages coming from other edges

Proof Overview – Converting a Graph to a System of Queues



customers are **helpful** messages

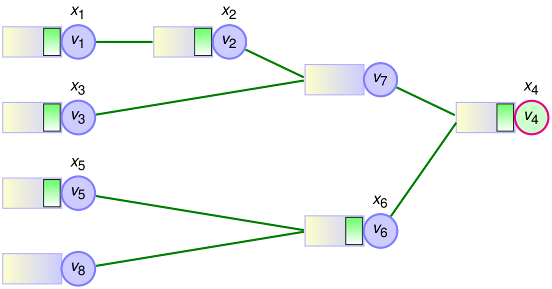
Proof Overview – Converting a Graph to a System of Queues



customers are **helpful** messages

initially, some nodes have **helpful** messages

Proof Overview – Converting a Graph to a System of Queues

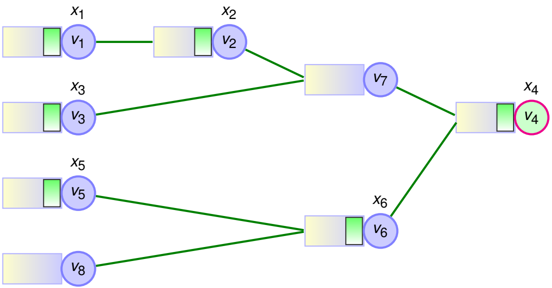


customers are **helpful** messages

initially, some nodes have **helpful** messages

customer arriving at some node, increases its rank by 1

Proof Overview – Converting a Graph to a System of Queues



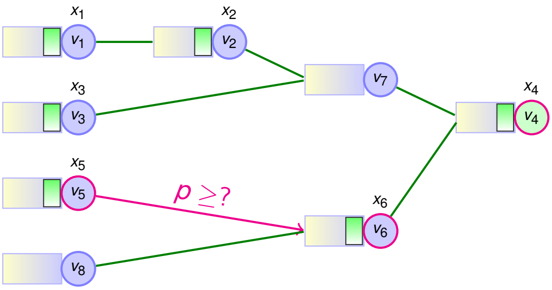
customers are **helpful** messages

initially, some nodes have **helpful** messages

customer arriving at some node, increases its rank by 1

once v_4 receives k **helpful** messages it finishes

Proof Overview – Converting a Graph to a System of Queues



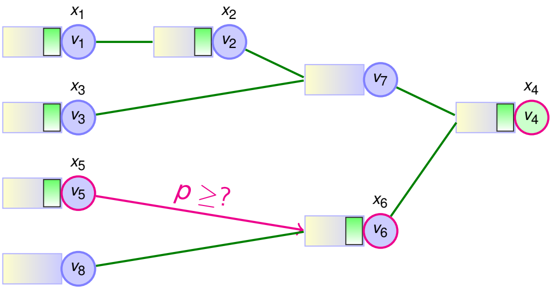
customers are **helpful** messages

initially, some nodes have **helpful** messages

customer arriving at some node, increases its rank by 1

once v_4 receives k **helpful** messages it finishes

Proof Overview – Converting a Graph to a System of Queues



customers are **helpful** messages

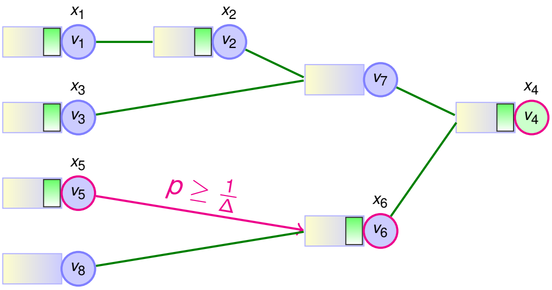
initially, some nodes have **helpful** messages

customer arriving at some node, increases its rank by 1

once v_4 receives k **helpful** messages it finishes

in a given **round**, v_5 wakes up exactly once

Proof Overview – Converting a Graph to a System of Queues



customers are **helpful** messages

initially, some nodes have **helpful** messages

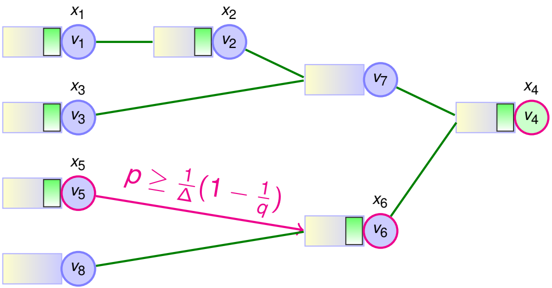
customer arriving at some node, increases its rank by 1

once v_4 receives k **helpful** messages it finishes

in a given **round**, v_5 wakes up exactly once

v_5 chooses v_6 as a partner w.p. $\geq \frac{1}{\Delta}$

Proof Overview – Converting a Graph to a System of Queues



customers are **helpful** messages

initially, some nodes have **helpful** messages

customer arriving at some node, increases its rank by 1

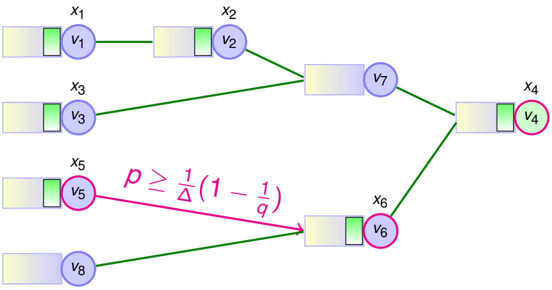
once v_4 receives k **helpful** messages it finishes

in a given **round**, v_5 wakes up exactly once

v_5 chooses v_6 as a partner w.p. $\geq \frac{1}{\Delta}$

the message will be **helpful** w.p. $\geq (1 - \frac{1}{q})$

Proof Overview – Converting a Graph to a System of Queues



customers are **helpful** messages

initially, some nodes have **helpful** messages

customer arriving at some node, increases its rank by 1

once v_4 receives k **helpful** messages it finishes

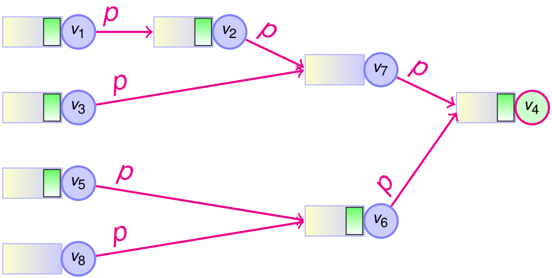
in a given **round**, v_5 wakes up exactly once

v_5 chooses v_6 as a partner w.p. $\geq \frac{1}{\Delta}$

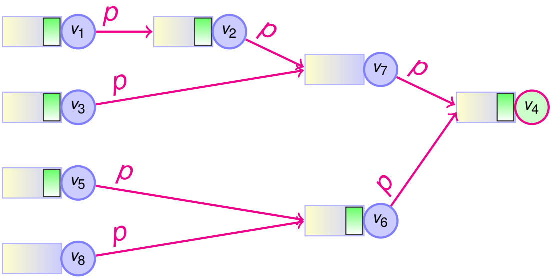
the message will be **helpful** w.p. $\geq (1 - \frac{1}{q})$

service time is geometrically distributed with p

Proof Overview – Exponential Servers Instead of Geometric

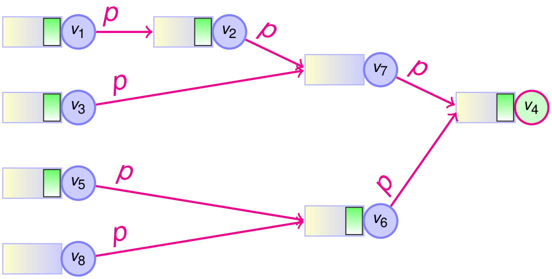


Proof Overview – Exponential Servers Instead of Geometric



If $X \sim \text{Geom}(p)$, and $Y \sim \text{Exp}(p)$, then: $\Pr(Y > t) \geq \Pr(X > t)$

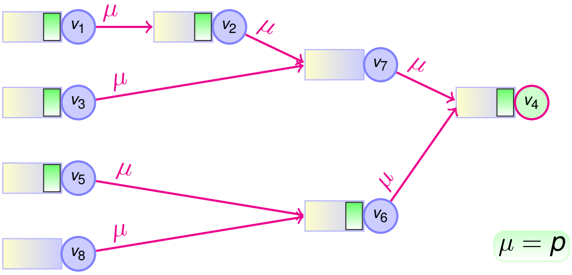
Proof Overview – Exponential Servers Instead of Geometric



If $X \sim \text{Geom}(p)$, and $Y \sim \text{Exp}(p)$, then: $\Pr(Y > t) \geq \Pr(X > t)$

so, exponential server is slower than geometric

Proof Overview – Exponential Servers Instead of Geometric



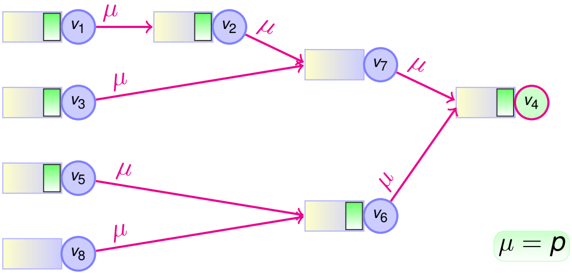
If $X \sim$ Geom(p), and $Y \sim$ Exp(p), then: $\Pr(Y > t) \geq \Pr(X > t)$

so, exponential server is slower than geometric

we replace servers, thus increasing the stopping time

Line is Slower Than Tree

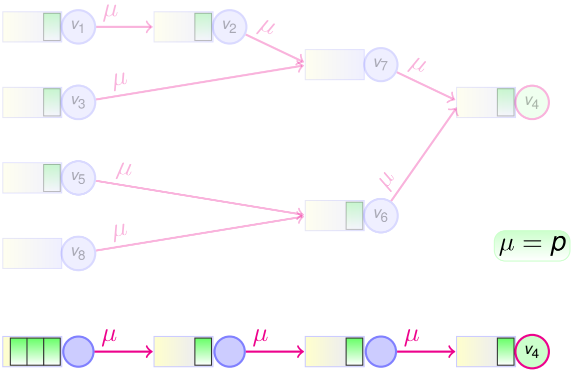
When does the last customer leave the system?



Line is Slower Than Tree

When does the last customer leave the system?

Reduce tree to line

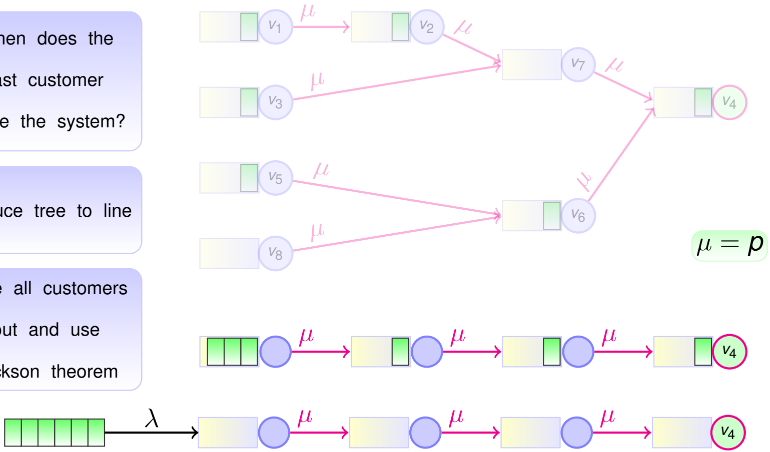


Line is Slower Than Tree

When does the last customer leave the system?

Reduce tree to line

Take all customers out and use Jackson theorem



Line of D Queues

- **Jackson's Theorem:** If utilization at every queue is less than 1, the equilibrium state distribution of number of customers in each queue exists and it is given by:

For state $(k_1, k_2, \ldots, k_n)$, and utilization $\rho_i = \frac{\lambda_i}{\mu_i}$: $\pi(k_1, k_2, \ldots, k_n) = \prod_{i=1}^{n} \rho_i^{k_i}(1 - \rho_i)$.

Line of D Queues

- **Jackson's Theorem:** If utilization at every queue is less than 1, the equilibrium state distribution of number of customers in each queue exists and it is given by:

$$\text{For state } (k_1, k_2, \ldots, k_n), \text{ and utilization } \rho_i = \frac{\lambda_i}{\mu_i}: \pi(k_1, k_2, \ldots, k_n) = \prod_{i=1}^{n} \rho_i^{k_i}(1-\rho_i).$$

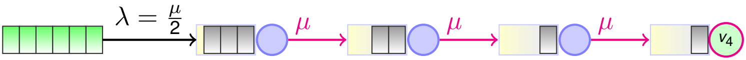
- By setting $\lambda = \mu/2$, we obtain: $\rho < 1$.

Line of D Queues

- **Jackson's Theorem:** If utilization at every queue is less than 1, the equilibrium state distribution of number of customers in each queue exists and it is given by:

$$\text{For state } (k_1, k_2, \ldots, k_n)\text{, and utilization } \rho_i = \frac{\lambda_i}{\mu_i} : \pi(k_1, k_2, \ldots, k_n) = \prod_{i=1}^{n} \rho_i^{k_i}(1 - \rho_i).$$

- By setting $\lambda = \mu/2$, we obtain: $\rho < 1$.
- We add **dummy** customers according to stationary distribution. So, the real **customers** see stationary distribution.

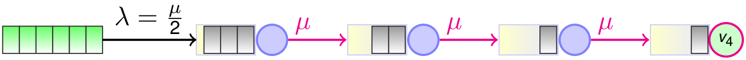


Line of D Queues

- **Jackson's Theorem:** If utilization at every queue is less than 1, the equilibrium state distribution of number of customers in each queue exists and it is given by:

$$\text{For state } (k_1, k_2, \ldots, k_n) \text{, and utilization } \rho_i = \frac{\lambda_i}{\mu_i}: \pi(k_1, k_2, \ldots, k_n) = \prod_{i=1}^{n} \rho_i^{k_i}(1 - \rho_i).$$

- By setting $\lambda = \mu/2$, we obtain: $\rho < 1$.
- We add **dummy** customers according to stationary distribution. So, the real **customers** see stationary distribution.
- Time by which all the customers enter the system is $O((k + \log n)/\mu)$, since $\lambda = \mu/2$, and $\log n$ is needed for high probability.

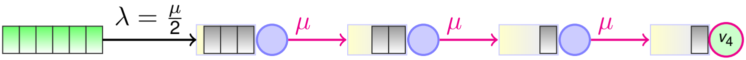


Line of D Queues

- **Jackson's Theorem:** If utilization at every queue is less than 1, the equilibrium state distribution of number of customers in each queue exists and it is given by:

$$\text{For state } (k_1, k_2, \ldots, k_n), \text{ and utilization } \rho_i = \frac{\lambda_i}{\mu_i}: \pi(k_1, k_2, \ldots, k_n) = \prod_{i=1}^{n} \rho_i^{k_i}(1 - \rho_i).$$

- By setting $\lambda = \mu/2$, we obtain: $\rho < 1$.
- We add **dummy** customers according to stationary distribution. So, the real **customers** see stationary distribution.
- Time by which all the customers enter the system is $O((k + \log n)/\mu)$, since $\lambda = \mu/2$, and $\log n$ is needed for high probability.
- Time to cross one MM1 queue in the stationary state is exponentially distributed with $\mu - \lambda = \mu/2$.

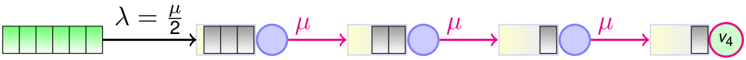


Line of D Queues

- **Jackson's Theorem:** If utilization at every queue is less than 1, the equilibrium state distribution of number of customers in each queue exists and it is given by:

$$\text{For state } (k_1, k_2, \ldots, k_n), \text{ and utilization } \rho_i = \frac{\lambda_i}{\mu_i}: \pi(k_1, k_2, \ldots, k_n) = \prod_{i=1}^{n} \rho_i^{k_i}(1-\rho_i).$$

- By setting $\lambda = \mu/2$, we obtain: $\rho < 1$.
- We add **dummy** customers according to stationary distribution. So, the real **customers** see stationary distribution.
- Time by which all the customers enter the system is $O((k + \log n)/\mu)$, since $\lambda = \mu/2$, and $\log n$ is needed for high probability.
- Time to cross one MM1 queue in the stationary state is exponentially distributed with $\mu - \lambda = \mu/2$.
- So, the time needed to cross D MM1 queues is $O((d + \log n)/\mu)$, where $\log n$ is needed for high probability.



Last customer leaves after: $O((k + \log n + D)/\mu) = O(\Delta(k + \log n + D))$ rounds

Proof Overview

k-dissemination with TAG
Tree-Based Algebraic Gossip

TAG – Tree Based Algebraic Gossip Protocol

- The proof is also based on analyzing a tree network of queues
- The uniform gossip bound is

$$O(\Delta(k + \log n + D))$$

- The TAG based bound is:

$$O(t(\mathcal{S}) + k + \log n + d(\mathcal{S}))$$

- In the synchronous time model $t(\mathcal{B}) \geq d(\mathcal{B})$. ($\mathcal{B}$ is a broadcast that builds a tree)
- For *Round Robin* Broadcast, $t(\mathcal{B}_{\mathcal{RR}}) = O(n)$ so for $k = \Omega(n)$ the stopping time of TAG is $\Theta(n)$

Summary

- When does uniform algebraic gossip is order optimal?
    - In graphs with constant maximum degree
    - But not only, e.g., complete graph. so when exactly?

Summary

- When does uniform algebraic gossip is order optimal?
  - In graphs with constant maximum degree
  - But not only, e.g., complete graph. so when exactly?
- When does TAG is order optimal?
  - When $k = \Omega(n)$
  - In graphs with large weak conductance (see paper)
  - When else? What spanning algorithm to use?

Summary

- When does uniform algebraic gossip is order optimal?
  - In graphs with constant maximum degree
  - But not only, e.g., complete graph. so when exactly?
- When does TAG is order optimal?
  - When $k = \Omega(n)$
  - In graphs with large weak conductance (see paper)
  - When else? What spanning algorithm to use?
- Thank you!

Deb, S., Médard, M., and Choute, C. (2006).
Algebraic gossip: a network coding approach to optimal multiple rumor mongering.
IEEE Transactions on Information Theory, 52(6):2486–2507.

Mosk-Aoyama, D. and Shah, D. (2006).
Information dissemination via network coding.
In *ISIT*, pages 1748–1752.