# Distributed Optimization for Machine Learning

## Lecture 12 - Gossip and Push-sum Protocols


Tianyi Chen

School of Electrical and Computer Engineering
Cornell Tech, Cornell University


October 6, 2025

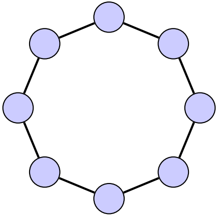
# Last-lecture: In-class average consensus game

**Each student:**

- Receives a random integer between 1-10.
- Writes it on a piece of paper (your $x_i(0)$).

**Network topologies:**

- **Circle:** Talk to your left and right neighbor.

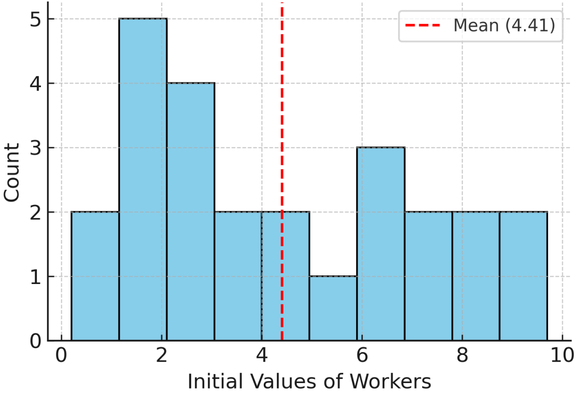


Ring Graph

**Goal:** After 4-5 **synchronous rounds** of the consensus game, everyone's number should approach the same value.

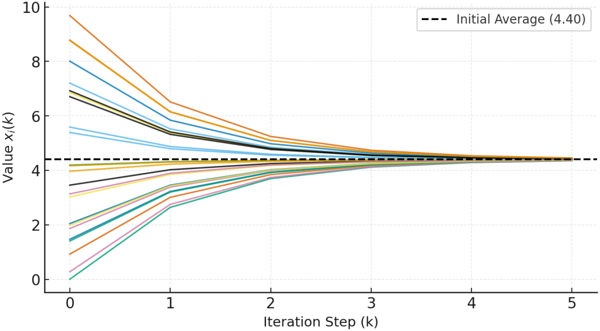
# Results: Initial values of the whole class

- Histogram of the workers' initial values (randomly between 0-10, mean $\approx 4.4$)



# Results: Achieve consensus among the class average

- Reach average consensus after 5 rounds of synchronous communications



# Today: In-class gossip consensus game

**Each student:**

- Receives a random integer between 1-10.
- Writes it on a piece of paper (your $x_i(0)$).

**Network topologies:**

- In each round, students randomly select **one student** as a neighbor.
- Those two students exchange and update to their **average**.

**Goal:** After several **asynchronous rounds** of the pairwise consensus game, everyone's number should approach the same value.

# Discussion and takeaways

**Compare both parts:**

- How many rounds happened today?
- Which converged faster – synchronous or gossip?
- Which felt more realistic to how devices communicate?
- What if some nodes were disconnected?

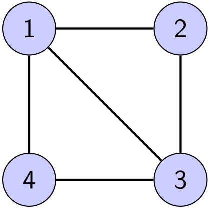
**Local communication $\rightarrow$ Global consensus!**

# Table of Contents

# Review: Graph description of a network

- **Setup:** A network of $N$ nodes connected by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

- **Goal:** Each node $i$ has a copy of the average $\frac{1}{N}\sum_{i=1}^{N} x_i(0)$.




- **Node set** $\mathcal{V}$:

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

- **Edge set** $\mathcal{E}$:

$$\mathcal{E} = \{(1,2), (1,3), (1,4), (2,3), (3,4)\}$$

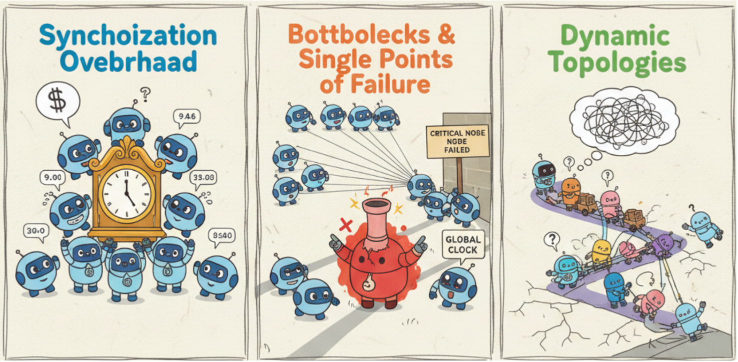
- **Average consensus protocol**:

$$x_i(k+1) = \sum_{\{j:(i,j)\in\mathcal{E}\}} w_{ij} x_j(k)$$

# Problem with synchronous consensus

Average consensus has several limitations:

- Synchronization overhead: global clock is challenging and expensive.
- Bottlenecks and single points of failure
- Lack of robustness to dynamic topologies



# Moving beyond synchronous updates


**Gossip:** A **randomized**, and **asynchronous** average consensus protocol.

# Randomized gossip protocol

- Realize consensus using only local, asynchronous communication.
- Converge to the global average "in expectation" and with similar rates as synchronous consensus.

## Broadcast Gossip Algorithms for Consensus

Tuncer Can Aysal, *Member, IEEE*, Mehmet Ercan Yildiz, *Student Member, IEEE*, Anand D. Sarwate, *Member, IEEE*, and Anna Scaglione, *Senior Member, IEEE*

***Abstract*—Motivated by applications to wireless sensor, peer-to-peer, and ad hoc networks, we study distributed broadcasting algorithms for exchanging information and computing in an arbitrarily connected network of nodes. Specifically, we study a broadcasting-based gossiping algorithm to compute the (possibly weighted) average of the initial measurements of the nodes at every node in the network. We show that the *broadcast gossip algorithm* converges al-**

all nodes to eventually agree on a parameter [6]. The work in [13] provided the theoretical explanation for behavior observed in these reported simulation studies. This paper focuses on a prototypical example of agreement in asynchronous networked systems, namely, the randomized average consensus problem in a wireless broadcast communication network.

# Pairwise randomized gossip

- Consider the basic gossip protocol: only two nodes become active.

**Pairwise randomized gossip:**

1. At time $k$, a single edge $(i, j) \in \mathcal{E}$ is chosen **uniformly at random**.
2. Nodes $i$ and $j$ exchange and update their states:

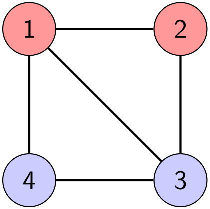
$$\text{For } \ell \in \{i, j\} : \quad x_\ell(k+1) = \frac{1}{2} x_i(k) + \frac{1}{2} x_j(k)$$

All other nodes remain inactive: $x_m(k+1) = x_m(k)$ for $m \notin \{i, j\}$.

- Weight matrices are important. **What is the weight matrix here?**

# Weight matrix of pairwise gossip

- The weight matrix $\mathbf{W}(k)$ is **time-varying** and **random**, e.g.,




$$\mathbf{W}(k) = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- $\mathbf{W}(k)$ is **doubly stochastic** ($\mathbf{1}^T\mathbf{W}(k) = \mathbf{1}^T$ and $\mathbf{W}(k)\mathbf{1} = \mathbf{1}$).

# Random weight matrix

- At each time, $\mathbf{W}(k)$ is selected **randomly** from a set of matrices,

$$\mathbf{W}^{(1,2)} = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{W}^{(1,3)} = \begin{bmatrix} 1/2 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{W}^{(1,4)} = \begin{bmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/2 & 0 & 0 & 1/2 \end{bmatrix} \qquad \mathbf{W}^{(2,3)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{W}^{(3,4)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}$$

# Why does randomized pairwise gossip ensure consensus?

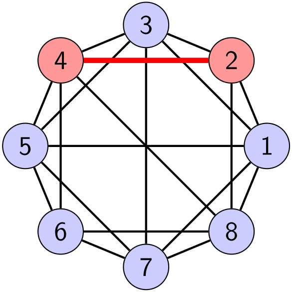
- If we choose $\mathbf{W}(k)$ uniformly from all 5 pairwise gossip matrices,

$$\mathbb{E}[\mathbf{W}(k)] = \bar{\mathbf{W}} = \frac{1}{|\mathcal{E}|}\left(\sum_{(i,j)\in\mathcal{E}} \mathbf{W}^{(i,j)}\right) = \begin{bmatrix} 7/10 & 1/10 & 1/10 & 1/10 \\ 1/10 & 8/10 & 1/10 & 0 \\ 1/10 & 1/10 & 7/10 & 1/10 \\ 1/10 & 0 & 1/10 & 8/10 \end{bmatrix}$$

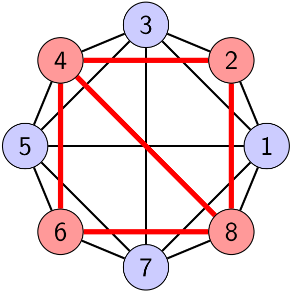
- $\bar{\mathbf{W}}$ is **doubly stochastic**.
- **Reach consensus "in expectation"!**
- Of course, activating more nodes could be beneficial...

# “Denser” activation

- Pairwise randomized gossip protocol: only two nodes become active.
- If resources allow, we can activate a subset of nodes $\mathcal{S}(k) \subseteq \mathcal{V}$.



Pairwise Gossip: activate 2 and 4



Set-averaging Gossip:
$\mathcal{S}(k) = \{2, 4, 6, 8\}$

# Set-averaging randomized gossip

## Set-Averaging Gossip Protocol

At each round $k$,

1. Randomly activate a subset of nodes $\mathcal{U}(k) \subseteq \mathcal{V}$.
2. Construct a doubly stochastic weight matrix $\mathbf{W}(k)$
3. All nodes in $\mathcal{U}(k)$ exchange and update their states.

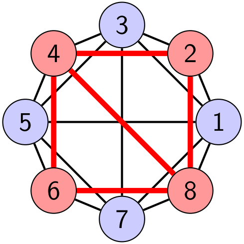
$$x_i(k+1) = \begin{cases} \sum_{j \in \mathcal{U}(k)} w_{ij}(k) x_j(k), & \text{if } i \in \mathcal{U}(k) \\ x_i(k), & \text{if } i \notin \mathcal{U}(k) \end{cases}$$

- **Set-averaging randomized gossip** enables faster information mixing and more flexible protocol design.
- Larger activated sets can accelerate consensus speed.

# Weight matrix of set-averaging randomized gossip

- Set-averaging randomized gossip has a time-varying and randomized weight matrix.

$$\mathbf{W}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/4 & 0 & 0 & 0 & 1/4 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 0 & 1/2 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 \end{bmatrix}$$




Set-averaging Gossip:
$\mathcal{S}(k) = \{2, 4, 6, 8\}$

# Does the randomized algorithm ensure average consensus?

- **Update equation:** The state vector $\mathbf{x}(k) \in \mathbb{R}^n$ evolves via a product of random matrices:

$$\mathbf{x}(k+1) = \mathbf{W}(k)\mathbf{x}(k) = \prod_{j=0}^{k} \mathbf{W}(j)\mathbf{x}(0)$$

- $\mathbf{W}(k)$ are randomly chosen from a set of doubly stochastic matrices.

- **Consensus guarantee? Speed?**

# Table of Contents

# Review: Convergence rate of consensus

- Recall the average consensus algorithm (constant weight matrix)

$$\mathbf{x}(k+1) = \mathbf{W}\mathbf{x}(k) = \mathbf{W}^k\mathbf{x}(0) \quad \text{where} \quad \mathbf{x}(0) = \mathbf{z} \in \mathbb{R}^N$$

**Theorem 1 (Convergence rate of average consensus)**

If $\mathbf{W}$ is doubly stochastic, it holds for the average consensus protocol that

$$\left\| \mathbf{x}(k) - \frac{\mathbf{1}\mathbf{1}^T\mathbf{z}}{N} \right\| \leq \rho^k \|\mathbf{z}\|,$$

where $\rho = \max_{i\geq 2} |\lambda_i(\mathbf{W})| < 1$.

**Q:** Does randomized gossip protocol converge in the same rate?

# Expected evolution of randomized gossip

- Gossip protocol generates a random sequence $\{\mathbf{x}(k)\}$

$$\mathbf{x}(k+1) = \mathbf{W}(k)\mathbf{x}(k) = \prod_{j=0}^{k} \mathbf{W}(j)\mathbf{x}(0)$$

- It is natural to consider its **expected behavior**, e.g.,

$$\mathbb{E}[\mathbf{W}(j)] = \bar{\mathbf{W}} = \frac{1}{|\mathcal{E}|}\left(\sum_{(i,j)\in\mathcal{E}} \mathbf{W}^{(i,j)}\right) = \begin{bmatrix} 1/2 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/2 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/2 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/2 \end{bmatrix}$$

# Expected evolution of randomized gossip

- Consider an independent and identically distributed (i.i.d.) sequence $\mathbf{W}(k)$. Define $\bar{\mathbf{W}} = \mathbb{E}[\mathbf{W}(k)]$:

$$\mathbb{E}[\mathbf{x}(k+1)] = \mathbb{E}\left[\prod_{j=0}^{k} \mathbf{W}(j)\right] \mathbf{x}(0) = (\bar{\mathbf{W}})^{k+1} \mathbf{x}(0)$$

- **That's crucial** - it iterates like weight matrix is constant.

# Convergence metric for randomized gossip

- Review: we used the consensus error in average consensus:

$$\mathbf{e}(k) = \mathbf{x}(k) - \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{x}(0)$$

- **Convergence metric:** norm of the error vector

$$\|\mathbf{e}(k)\| = \left\|\mathbf{x}(k) - \frac{\mathbf{1}\mathbf{1}^T\mathbf{x}(0)}{N}\right\|$$

- $\|\mathbf{e}(k)\|$ **is a random variable!**

# Convergence metric

- For randomized protocols, we often consider the expected error:

$$\mathbb{E}\left[\|\mathbf{e}(k)\|^2\right]$$

- By variance decomposition:

$$\mathbb{E}\left[\|\mathbf{e}(k)\|^2\right] = \|\mathbb{E}[\mathbf{e}(k)]\|^2 + \mathrm{Var}(\mathbf{e}(k))$$

- $\mathbb{E}\left[\|\mathbf{e}(k)\|^2\right] \to 0$ implies both **expectation** and **variance** of the consensus error vanish.

# Convergence rate of gossip

**Theorem 2 (Convergence rate of gossip)**

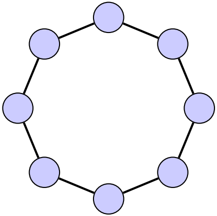
If $\bar{\mathbf{W}}$ is doubly stochastic, it holds for the gossip protocol that

$$\mathbb{E}\left[\left\|\mathbf{x}(k) - \frac{\mathbf{1}\mathbf{1}^T\mathbf{z}}{N}\right\|^2\right] \le (\bar{\rho})^{2k}\|\mathbf{z}\|^2,$$

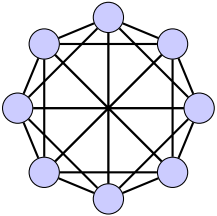
where $\bar{\rho} = \max_{i\ge 2}|\lambda_i(\bar{\mathbf{W}})| < 1$.

**Has the same order as average consensus.**
**The conclusion from last lecture can be applied here!**

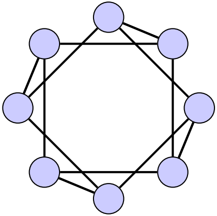
# Various graph topologies



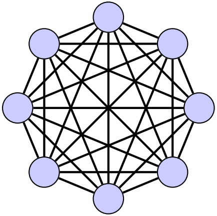
Ring Graph: $T(\epsilon) \sim \mathcal{O}\left(N^2\right)$



Expander Graph: $T(\epsilon) \sim \mathcal{O}(\log(N))$



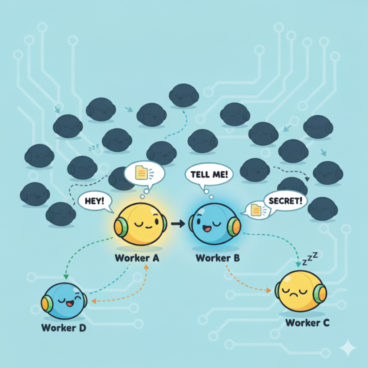
Torus Graph: $T(\epsilon) \sim \mathcal{O}(N)$



Complete Graph: $T(\epsilon) \sim \mathcal{O}(1)$

# Summary of randomized gossip protocol

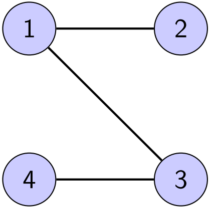
- **Principle:** In each time step, a small subset of nodes (often just two) communicate and update their states.
- **Benefit:** High fault tolerance and robustness to network failures or communication delays; similar consensus rates in expectation.



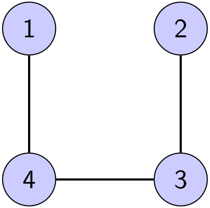
# Table of Contents

# Dynamic network model

- Randomized gossip protocols induce network **dynamics**, since active set varies randomly at each round.
- **Setup:** $N$ nodes connected by a varying graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$.




$\mathcal{E}(k) = \{(1,2),(1,3),(3,4)\}$




$\mathcal{E}(k) = \{(1,4),(2,3),(3,4)\}$

# Applications of dynamic networks

- **Dynamic networks are common:** network topology often changes over time due to node mobility, failures, or intermittent connectivity.


Figure: Example of dynamic networks: Internet of Things (IoT).

# Challenge: Construct doubly stochastic weight matrices

- Graph topology changes $\Rightarrow$ weight matrix $\mathbf{W}(k)$ changes over time.

## Set-averaging gossip protocol

At each round $k$,

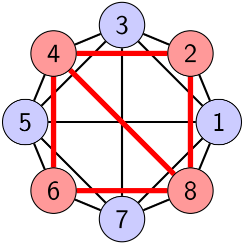
1. Randomly activate a subset of nodes $\mathcal{U}(k) \subseteq \mathcal{V}$.
2. Construct a doubly stochastic weight matrix $\mathbf{W}(k)$
3. All nodes in $\mathcal{U}(k)$ exchange and update their states.

$$x_i(k+1) = \begin{cases} \sum_{j \in \mathcal{U}(k)} w_{ij}(k) x_j(k), & \text{if } i \in \mathcal{U}(k) \\ x_i(k), & \text{if } i \notin \mathcal{U}(k) \end{cases}$$

- Constructing $\mathbf{W}(k)$ requires global info. (e.g., graph Laplacian).

# Challenge: Doubly stochastic weight matrices (cont.)

- Constructing $\mathbf{W}(k)$ at each round is **computationally expensive** and **not scalable**.
- **Q:** Can we compute the average without constructing $\mathbf{W}(k)$?




Set-averaging Gossip: $\mathcal{S}(k) = \{2, 4, 6, 8\}$

# Consensus without doubly stochastic weights

We will introduce a **physically inspired** gossip protocol:

**Gossip-Based Computation of Aggregate Information**

David Kempe,* Alin Dobra, and Johannes Gehrke†
Department of Computer Science, Cornell University
Ithaca, NY 14853, USA
{kempe,dobra,johannes}@cs.cornell.edu

**Abstract**

*Over the last decade, we have seen a revolution in connectivity between computers, and a resulting paradigm shift from centralized to highly distributed systems. With massive scale also comes massive instability, as node and link fail-*

our physical environment with sensor networks consisting of hundreds of thousands of small sensor nodes [24, 28, 35]. Applications for such large-scale distributed systems have three salient properties that distinguish them from traditional centralized or small-scale distributed systems.

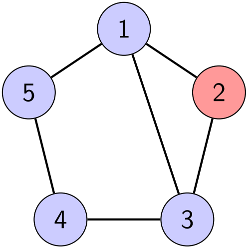
First, the dynamics of large-scale distributed systems are

- **Information diffusion is analogous to mass diffusion.**

# Rethinking gossip from the "transfer of mass" perspective

- **Example:** a non-doubly stochastic matrix

$$\mathbf{W}(k) = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 0 & 1/3 & 1/3 \end{bmatrix}$$




  - Interprete $x_i(k)$ as **"node $i$ possesses material of mass $x_i(k)$"**.

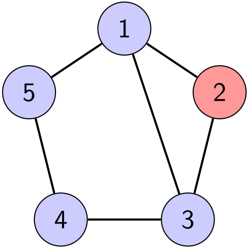
  - Node 1:

$$x_1(k+1) = \frac{1}{4} \cdot x_1(k) + \frac{1}{4} \cdot x_2(k) + \frac{1}{4} \cdot x_3(k) + 0 \cdot x_4(k) + \frac{1}{4} \cdot x_5(k)$$

  - $w_{12} = 1/4$ means **node 2 sends** $1/4$ **of its mass to node 1**.

# Rethinking the doubly stochastic requirement

$$\mathbf{W}(k) = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 0 & 1/3 & 1/3 \end{bmatrix}$$



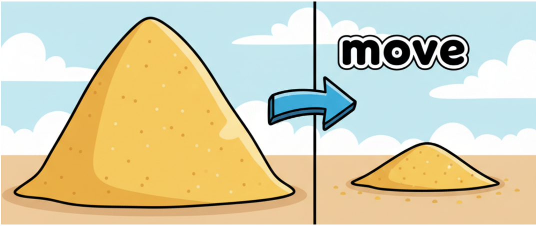

- Node 2 sends $1/4$ of its value to node 1 and 3, leaves $1/3$ to itself

$$w_{12} + w_{22} + w_{32} + w_{42} + w_{52} = 1/4 + 1/3 + 1/4 + 0 + 0 = 5/6 < 1$$

- **1/6 of $x_2(k)$ is lost!**

# Rethinking the doubly stochastic requirement (con't)

- In the previous example, $j = 2$, $\sum_{i=1}^{N} w_{ij} < 1$: **mass is lost!**




- Column stochastic (but not necessarily row stochastic) weight matrix $\mathbf{W}(k)$ ensures **mass conservation**!

# Interpret communication as mass diffusion

- Consider a column stochastic weight matrix:

$$\mathbf{W}(k) = \begin{bmatrix} 1/4 & 1/3 & 1/4 & 0 & 1/4 \\ 1/4 & 1/3 & 1/4 & 0 & 0 \\ 1/4 & 1/3 & 1/4 & 1/4 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 0 & 1/3 & 1/4 \end{bmatrix}$$

- Node 2 sends $1/3$ of its mass to node 1, 3, and leaves $1/3$ to itself.
- No mass is lost!

# Doubly stochastic matrix ensures conservation law

- **Conservation of mass** holds in the whole network

$$\frac{1}{N}\sum_{i=1}^{N} x_i(k+1) = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{N} w_{ij}\right)x_j(k) = \frac{1}{N}\sum_{i=1}^{N} x_i(k)$$

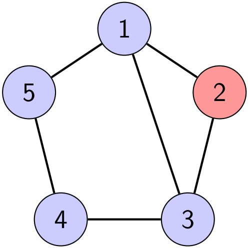
$$= \cdots = \frac{1}{N}\sum_{i=1}^{N} x_i(0) \quad \text{(total mass)}$$

- **Goal:** estimate how much mass is in the network.

# Mass reallocation via diffusion: The push-sum protocol

- At each round $k$, a subset of nodes $\mathcal{U}(k) \subseteq \mathcal{V}$ is randomly activated.
- Each node $i \in \mathcal{U}(k)$ **evenly separates** $x_i(k)$ **into** $(d_i + 1)$ **parts** and sends it to its neighbors, where $d_i$ is the number of $i$'s neighbors.

$$\mathbf{W}(k) = \begin{bmatrix} 1/4 & 1/3 & 1/4 & 0 & 1/3 \\ 1/4 & 1/3 & 1/4 & 0 & 0 \\ 1/4 & 1/3 & 1/4 & 1/3 & 0 \\ 0 & 0 & 1/4 & 1/3 & 1/3 \\ 1/4 & 0 & 0 & 1/3 & 1/3 \end{bmatrix}$$




- Column stochastic, **not row stochastic**. No consensus among $x_i(k)$!

# Recover the average from stationary distribution

- Each state $x_i(k)$ converges to

$$\lim_{k\to\infty} x_i(k) = \left(\sum_{i=1}^{N} x_i(0)\right)\pi_i$$

- If node $i$ knows $\pi_i$, it can recover the average:

$$\frac{1}{N}\sum_{i=1}^{N} x_i(0) = \frac{1}{N\pi_i}\lim_{k\to\infty} x_i(k)$$

- **How to find the distribution $\pi_i$?**

# Compute the stationary distribution

- **How to find $\pi_i$?** Leverage the mass conservation property again.
- Each node $i$ maintains a state $s_i(k)$, initialized as $s_i(0) = 1/N$.
- Each node diffuses the $s_i(k)$ as $x_i(k)$ and it converges to

$$\lim_{k\to\infty} s_i(k) = \left(\sum_{i=1}^{N} s_i(0)\right)\pi_i = \pi_i$$

- $\pi_i$ **can be found by the same push-sum diffusion as well!**

# Push-sum gossip protocol

At each round $k$,

1. Randomly activate a subset of nodes $\mathcal{U}(k) \subseteq \mathcal{V}$.
2. All nodes in $\mathcal{U}(k)$ compute number of $j$'s active neighbors.

$$d_j(k) = |\{i : (i,j) \in \mathcal{E} \text{ and } i,j \in \mathcal{U}(k)\}|$$

3. All nodes in $\mathcal{U}(k)$ evenly push a part of their state to their neighbors

$$x_i(k+1) = \begin{cases} \sum_{j \in \mathcal{U}(k)} x_j(k)/(d_j(k)+1), & \text{if } i \in \mathcal{U}(k) \\ x_i(k), & \text{if } i \notin \mathcal{U}(k) \end{cases}$$

$$s_i(k+1) = \begin{cases} \sum_{j \in \mathcal{U}(k)} s_j(k)/(d_j(k)+1), & \text{if } i \in \mathcal{U}(k) \\ s_i(k), & \text{if } i \notin \mathcal{U}(k) \end{cases}.$$

**Return:** $\frac{x_i(k)}{N \cdot s_i(k)}$ **as the estimate of the average.**

# Column stochastic weight matrix of push-sum

- The number of $j$'s active neighbors is defined as

$$d_j(k) = |\{i : (i, j) \in \mathcal{E} \text{ and } i, j \in \mathcal{U}(k)\}|$$

- The weight matrix $\mathbf{W}(k)$ of push-sum gossip protocol is defined as

$$w_{ij}(k) = \begin{cases} 1/(d_j(k) + 1), & \text{if } (i, j) \in \mathcal{E} \text{ and } i, j \in \mathcal{U}(k) \\ 1/(d_j(k) + 1), & \text{if } j = i, \\ 0, & \text{otherwise} \end{cases}$$

**Lemma 2**

The dynamic weight matrix $\mathbf{W}(k)$ is a column-stochastic matrix.

# Proof: Column stochastic weight matrix of push-sum

- By construction, we have

$$w_{ij}(k) = \begin{cases} 1/(d_j(k)+1), & \text{if } (i,j) \in \mathcal{E} \text{ and } i,j \in \mathcal{U}(k) \\ 1/(d_j(k)+1), & \text{if } j = i, \\ 0, & \text{otherwise} \end{cases}$$

- By definition, for all $j \in \mathcal{V}$ and $k \geq 0$, we have

$$\begin{aligned} \sum_{i=1}^{N} w_{ij}(k) &= \frac{1}{d_j(k)+1} \times d_j(k) + \frac{1}{d_j(k)+1} \times 1 + 0 \\ &= 1 \end{aligned}$$

- $\mathbf{W}(k)$ is column stochastic for each $k$.

# Recap and fine-tuning

- What we have talked about today?
  - $\Rightarrow$ **Randomized gossip** avoids costly global clock synchronization.
  - $\Rightarrow$ **Randomized gossip** achieves the same consensus rate as average.
  - $\Rightarrow$ **Push-sum** achieves this without doubly stochastic matrices.

Welcome anonymous survey!