# Asymptotics, asynchrony, and asymmetry in distributed consensus

Anand D. Sarwate

Information Theory and Applications Center
University of California, San Diego

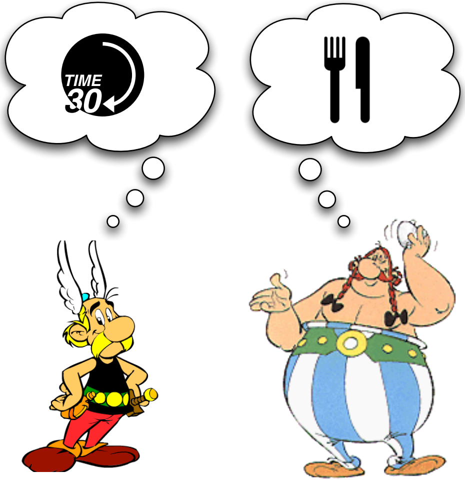
9 March 2011

Joint work with Alex G. Dimakis, Tuncer Can Aysal, Mehmet Ercan Yildiz, Martin Wainwright, and Anna Scaglione, and Tara Javidi

# Rapprochement, consensus, accord

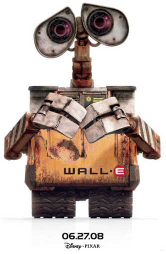
# Rapprochement, consensus, accord

# Rapprochement, consensus, accord



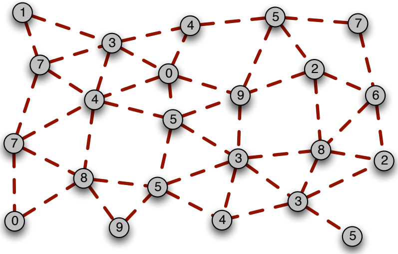
# Rapprochement, consensus, accord

# Consensus is an important task

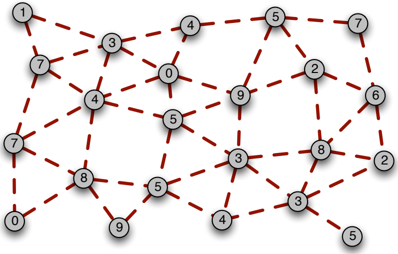



- Calibration
- Dissemination
- Coordination

# Abstracting the task

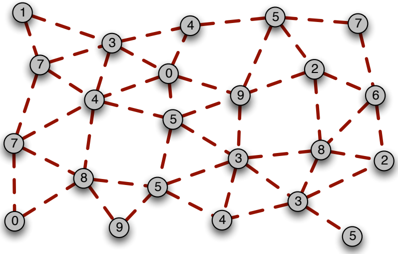


# Abstracting the task



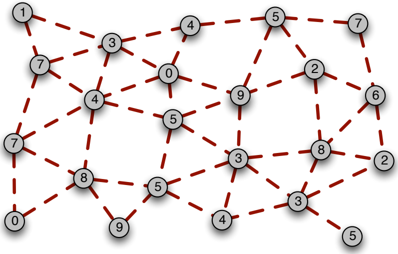

- Network of agents, each with an observation

# Abstracting the task




- Network of agents, each with an observation
- *Communicate locally* – exchange messages about observations

# Abstracting the task




- Network of agents, each with an observation
- *Communicate locally* – exchange messages about observations
- *Compute locally* – estimate a function of all values

# There are many aspects to consider

# There are many aspects to consider

- What are observations?
    - continuous or discrete?
    - scalar or vector?

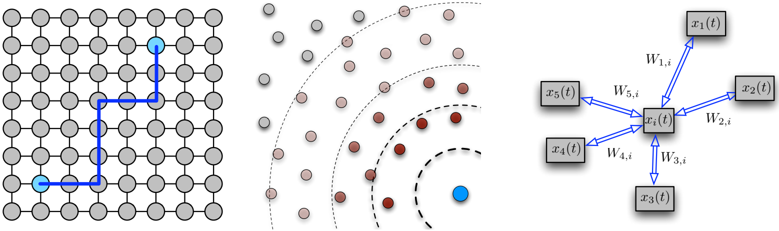
# There are many aspects to consider

- What are observations?
  - continuous or discrete?
  - scalar or vector?
- How can we communicate?
  - point-to-point or broadcast?
  - low resolution or high resolution?

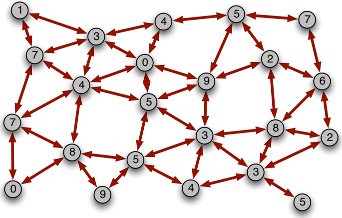
# There are many aspects to consider

- What are observations?
  - continuous or discrete?
  - scalar or vector?
- How can we communicate?
  - point-to-point or broadcast?
  - low resolution or high resolution?
- What do we compute?
  - averages
  - medians, quantiles
  - convex optimization

# The goal(s) for today

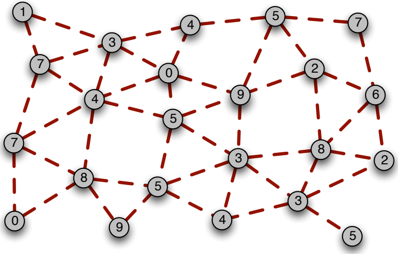



1. The basic mathematical model for consensus
2. Routing and mobility can speed up convergence
3. Broadcasting can trade off accuracy for speed
4. The discreet charm of discrete messages

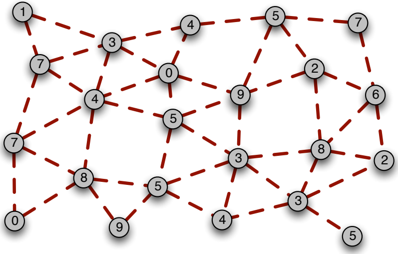


# Building a mathematical model

# The data model

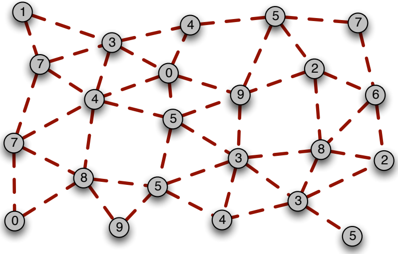


# The data model



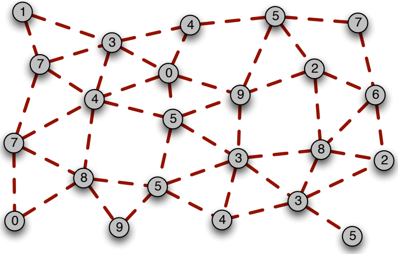

- Set of $n$ agents

# The data model



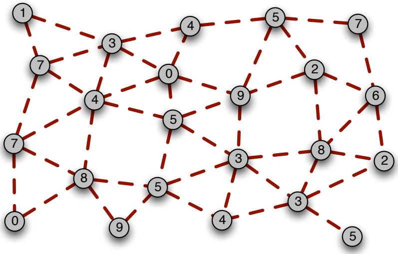

- Set of $n$ agents
- Agent $i$ observes initial value $x_i(0) \in \mathbb{R}$ for $i = 1, 2 \ldots n$

# The data model

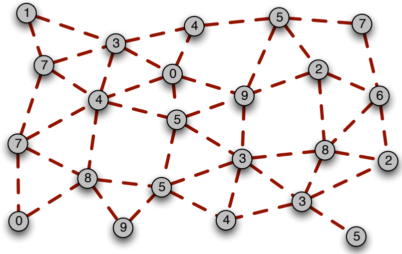


- Set of $n$ agents
- Agent $i$ observes initial value $x_i(0) \in \mathbb{R}$ for $i = 1, 2 \ldots n$
- Assume data is bounded : $x_i(0) \in [0, 10]$, for example

# The communication graph

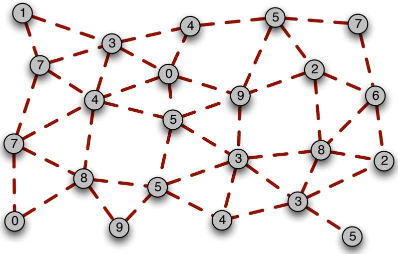


# The communication graph



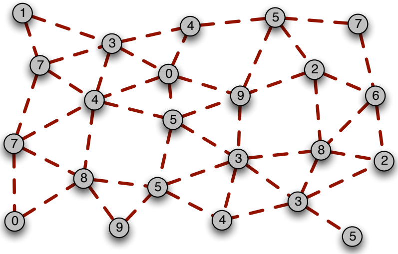

- Agents are arranged in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

# The communication graph




- Agents are arranged in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
- Agents $i$ can communicate with $j$ if there is an edge $(i, j)$ (e.g. $j \in \mathcal{N}_i$).

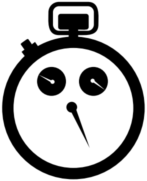
# The communication graph




- Agents are arranged in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
- Agents $i$ can communicate with $j$ if there is an edge $(i, j)$ (e.g. $j \in \mathcal{N}_i$).
- Bidirectional communication : agents *exchange* messages.

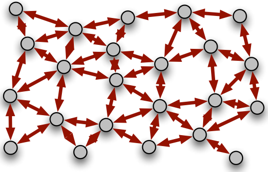
# Constraints on the communication

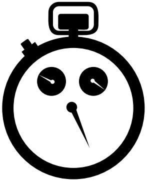
# Constraints on the communication



- Time is slotted : only one transmission per slot.

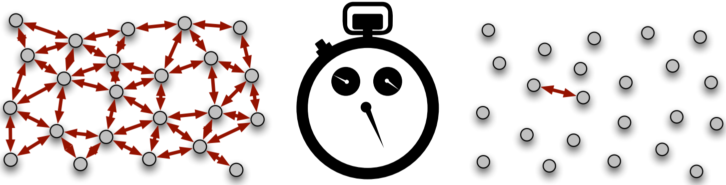
# Constraints on the communication





- Time is slotted : only one transmission per slot.
- Synchronous : use many edges, then update.

# Constraints on the communication



- Time is slotted : only one transmission per slot.
- Synchronous : use many edges, then update.
- Asynchronous : edges chosen randomly in each slot.

# Measuring performance

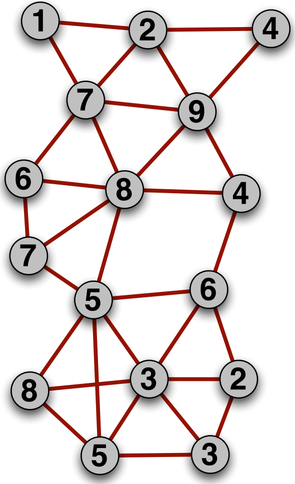
The goal is to pass messages between agents such that they can estimate the average of the initial observations:

$$\mathbf{x}(t) \to \left( \sum_i x_i(0) \right) \cdot \mathbf{1}$$

**Averaging time** $T_{\text{ave}}(n, \epsilon)$ is time when $\mathbf{x}(t)$ is within $\epsilon$ of the average:

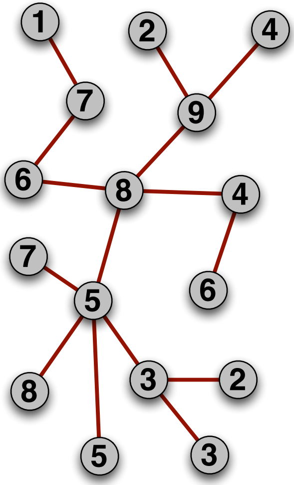
$$T_{\text{ave}}(n, \epsilon) = \sup_{\mathbf{x}(0)} \inf_t \left\{ \mathbb{P}_{Alg} \left( \frac{\|\mathbf{x}(t) - x_{\text{ave}} \cdot \mathbf{1}\|}{\|\mathbf{x}(0)\|} \geq \epsilon \right) \leq \epsilon \right\}$$

# A centralized solution




Simple centralized algorithm:

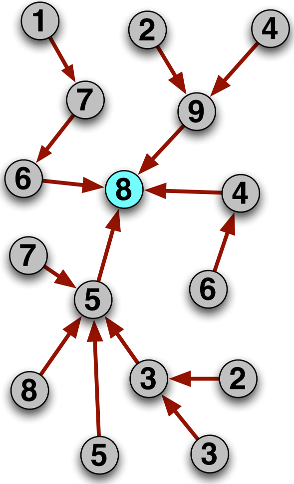
# A centralized solution




Simple centralized algorithm:

1. Build a spanning tree

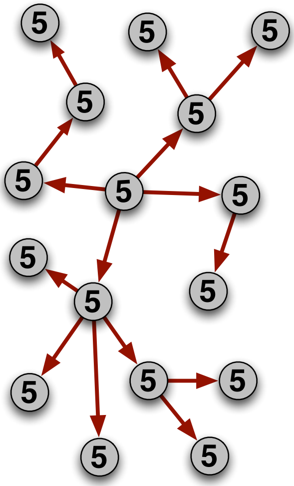
# A centralized solution




Simple centralized algorithm:

1. Build a spanning tree
2. Gather all the values at root

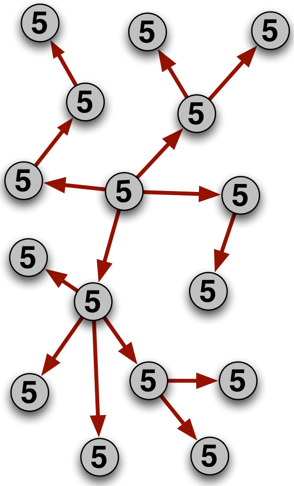
# A centralized solution




Simple centralized algorithm:

1. Build a spanning tree
2. Gather all the values at root
3. Compute and disseminate average

# A centralized solution




Simple centralized algorithm:

1. Build a spanning tree
2. Gather all the values at root
3. Compute and disseminate average

**Pro:** requires $\Theta(n)$ messages

**Con:** completely centralized

# Distributed synchronous consensus

Suppose each agent linearly combines itself and its neighbors:

$$x_i(t+1) = W_{ii} x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij} x_j(t)$$

$$\sum_j W_{ij} = 1 \qquad \forall i$$

$$W_{ij} = W_{ji}$$

Synchronous algorithm where the update after each slot is given by:

$$\mathbf{x}(t+1) = W \mathbf{x}(t)$$

where $W$ is a **doubly stochastic** matrix.

# A simple result

**Theorem**

*For synchronous consensus with update matrix $W$,*

$$T_{\text{ave}}(n, \epsilon) = \Theta\left(|\mathcal{E}| \cdot T_{\text{relax}}(W) \cdot \log \epsilon^{-1}\right)$$

*where $T_{\text{relax}}(W)$ is the* **relaxation time** *of the matrix $W$:*

$$T_{\text{relax}}(W) = \frac{1}{1 - \lambda_2(W)} \ .$$

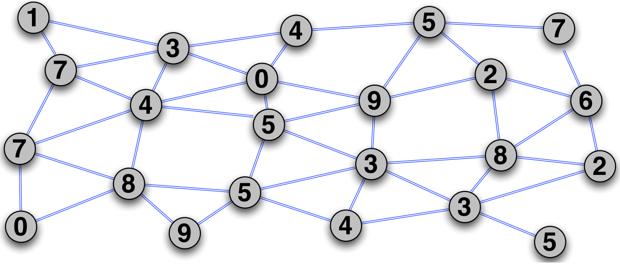
Proof : $W$ is the transition matrix of a Markov chain – consensus is the convergence of the chain to its stationary distribution.

# A theme with variations

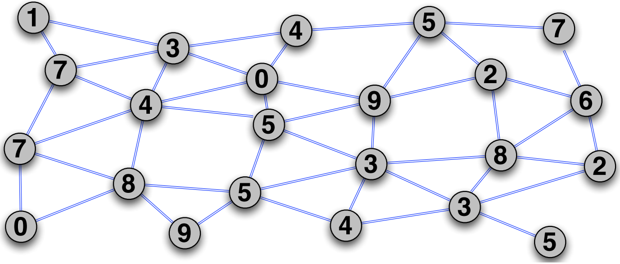
Survey article by Dimakis et al. in *Proc. IEEE*.

- **Synchronous** DeGroot (1974), Tsitsiklis (1984)
- **Time-varying topologies** Chatterjee-Seneta (1977), Tsitsiklis et al. (1986), Jadbabaie et al. (2003), Ren-Beard (2005), Gao-Cheng (2006), Fagnani-Zampieri (2008)
- **Asynchronous** Boyd et al. (2006)
- **Quantization** Kashyap et al. (2007), Nedic et al. (2009), Yildiz-Scaglione (2008), Aysal et. al (2009), Kar-Moura (2010), Carli et al. (2010), Lavaie-Murray (2010)
- **Discrete values** Benezit et al. (2010)
- **Many others!**

# Asynchronous updates = "gossip"

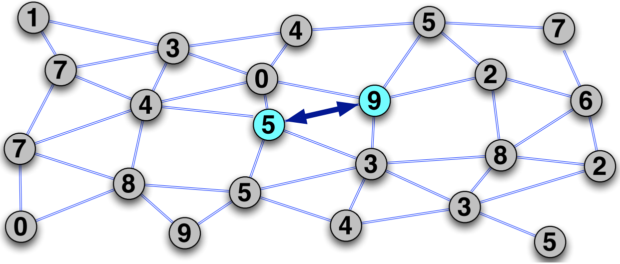


# Asynchronous updates = "gossip"



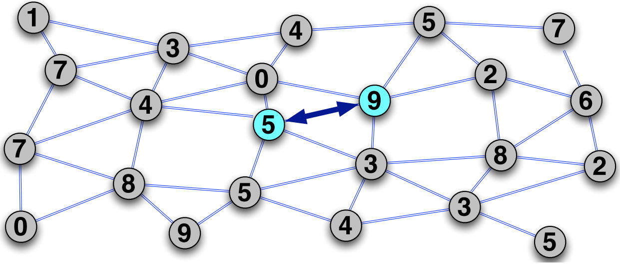

- Node $i$ wakes up at random, chooses neighbor $j$ at random.

# Asynchronous updates = "gossip"



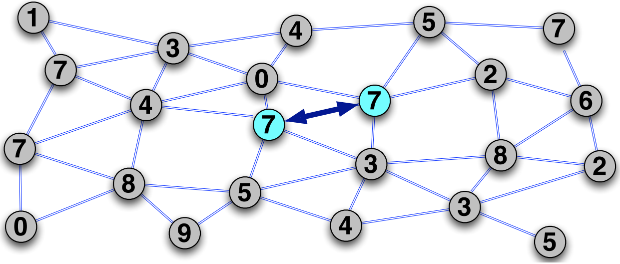

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.

# Asynchronous updates = "gossip"



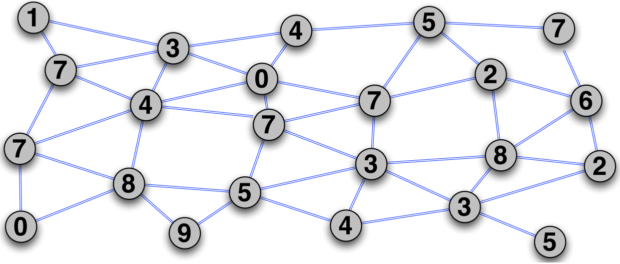

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"



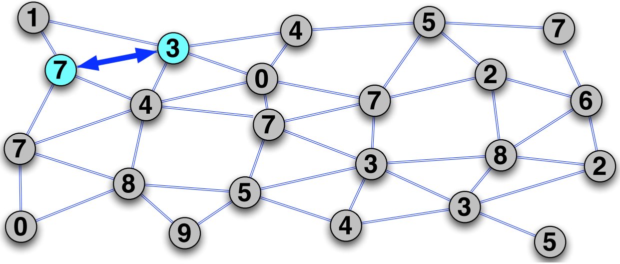

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"



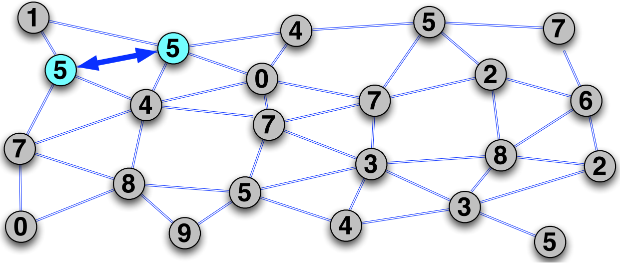

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"



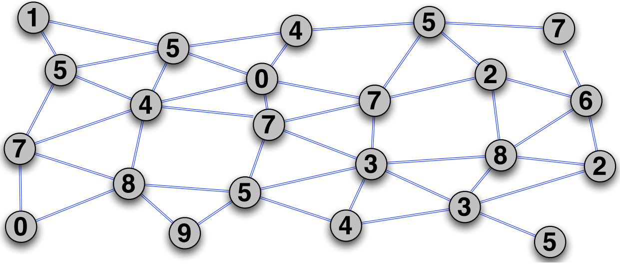

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"



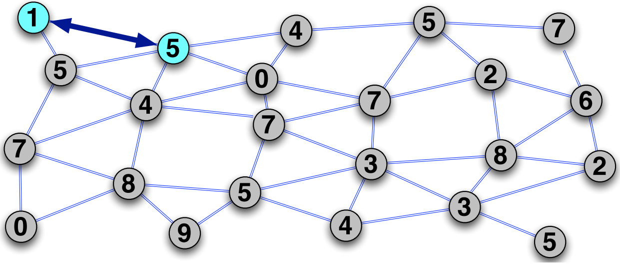

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"



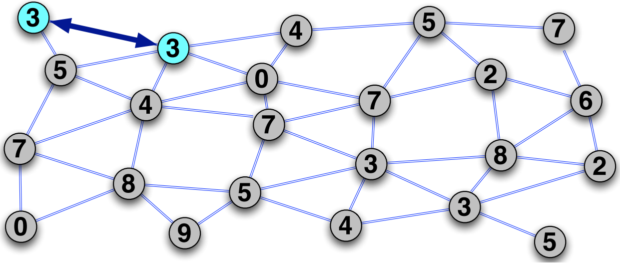

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"



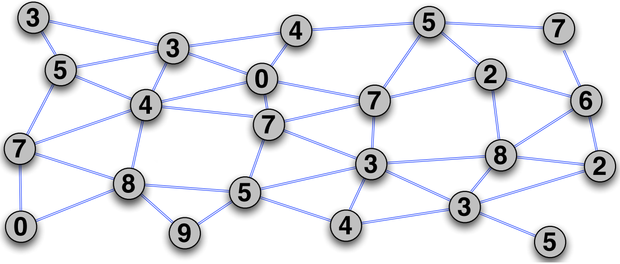

- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = "gossip"




- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Asynchronous updates = “gossip”




- Node $i$ wakes up at random, chooses neighbor $j$ at random.
- Nodes $i$ and $j$ exchange $x_i(t)$ and $x_j(t)$ and compute average.
- Set $x_i(t+1) = x_j(t+1) = \frac{1}{2}(x_i(t) + x_j(t))$.

# Gossip uses random linear updates

At each time a random pair $(i, j) \in \mathcal{E}$ averages:

$$x_i(t+1) = x_j(t+1) = \frac{x_i(t) + x_j(t)}{2}.$$

Each update is linear : $\mathbf{x}(t+1) = W^{(i,j)}(t)\mathbf{x}(t)$.

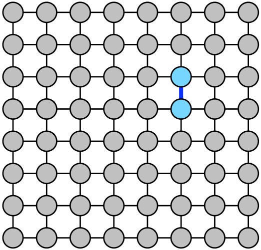
**Theorem**

*Let $\bar{W} = \mathbb{E}[W^{(i,j)}]$ over the edge selection process. Then*

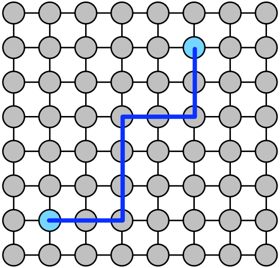
$$T_{\text{ave}}(n, \epsilon) = \Theta\left(T_{\text{relax}}(\bar{W}) \cdot \log \epsilon^{-1}\right)$$

# The implication for big graphs

For the grid with uniform selection, gossip takes $\Theta(n^2)$ transmissions!



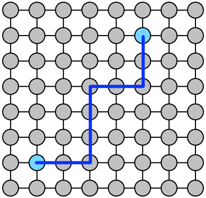
Selecting edges at random is inefficient! Local exchange is inefficient!



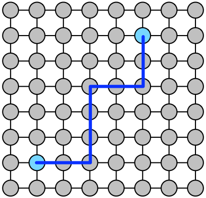
# Network properties can accelerate convergence

Joint work with Alex Dimakis and Martin Wainwright

# Geographic gossip with routing

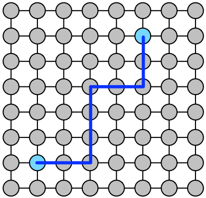


# Geographic gossip with routing



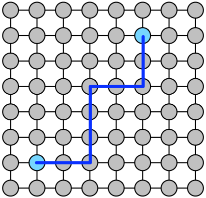
- Assume that packets can be routed between any two nodes.

# Geographic gossip with routing



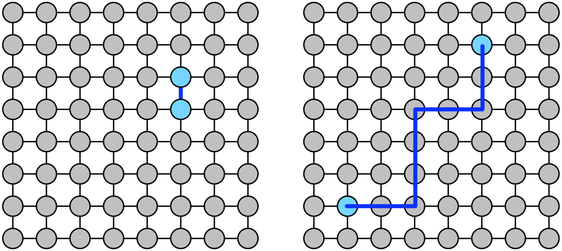
- Assume that packets can be routed between any two nodes.
- Now select “neighbor” uniformly from all nodes and route message.

# Geographic gossip with routing



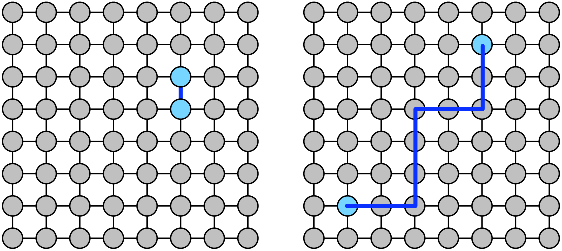
- Assume that packets can be routed between any two nodes.
- Now select "neighbor" uniformly from all nodes and route message.
- "Effective graph" is now the complete graph.

# Example : the grid



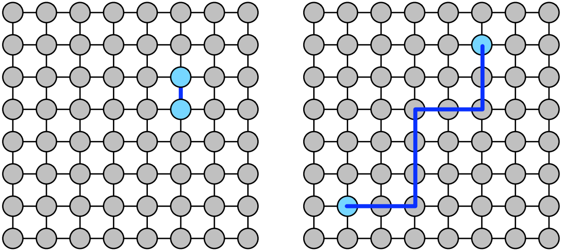
| algorithm | $T_{\mathrm{relax}}(\bar{W})$ |
|---|---|

# Example : the grid



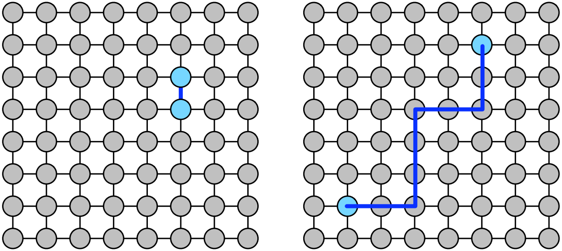
| algorithm | $T_{\text{relax}}(\bar{W})$ |
|---|---|
| **Local** | $\Theta(n^2)$ |

# Example : the grid



| algorithm | $T_{\text{relax}}(\bar{W})$ |
|---|---|
| **Local** | $\Theta(n^2)$ |
| **With routing** | $\Theta(n)$ |

# Example : the grid



| algorithm | $T_{\mathrm{relax}}(\bar{W})$ |
|---|---|
| **Local** | $\Theta(n^2)$ |
| **With routing** | $\Theta(n)$ |

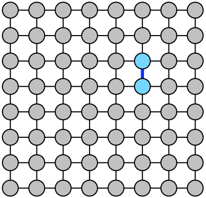
This is unfair, since routing costs in number of hops.

# One-hop transmissions to reach consensus

Count number of hops (power) to get within $\epsilon$ of the average:

| algorithm | one-hop transmission |
| --- | --- |

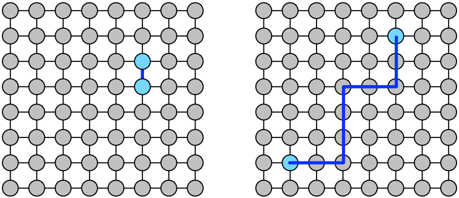
# One-hop transmissions to reach consensus



Count number of hops (power) to get within $\epsilon$ of the average:

| algorithm | one-hop transmission | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |

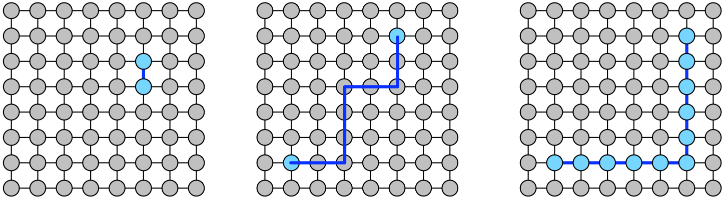
# One-hop transmissions to reach consensus



Count number of hops (power) to get within $\epsilon$ of the average:

| algorithm | one-hop transmission | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |
| **With routing** | $\Theta(n^{3/2})$ | Dimakis,Sarwate, Wainwright |

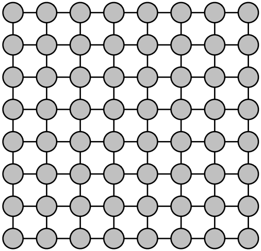
# One-hop transmissions to reach consensus



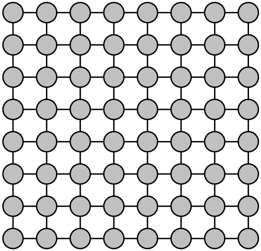
Count number of hops (power) to get within $\epsilon$ of the average:

| algorithm | one-hop transmission | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |
| **With routing** | $\Theta(n^{3/2})$ | Dimakis,Sarwate, Wainwright |
| **Average on the way** | $\Theta(n)$ | Benezit et al. |

# Gossip with mobility

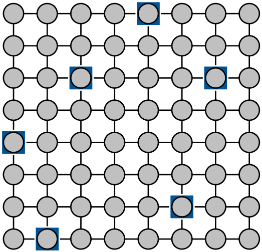


# Gossip with mobility



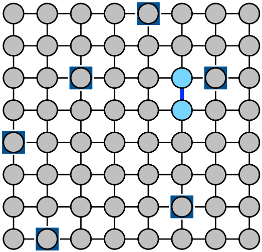
- Start with a grid of static nodes.

# Gossip with mobility



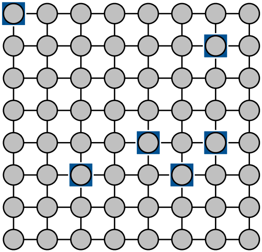
- Start with a grid of static nodes.
- Add $m$ *fully mobile nodes*.

# Gossip with mobility



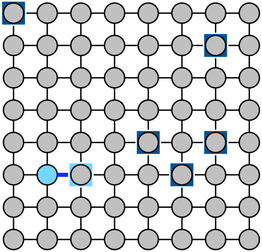
- Start with a grid of static nodes.
- Add $m$ *fully mobile nodes*.

# Gossip with mobility



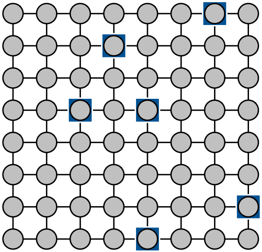
- Start with a grid of static nodes.
- Add $m$ *fully mobile nodes*.
- At each time, $m$ mobile nodes choose new locations uniformly at random.

# Gossip with mobility



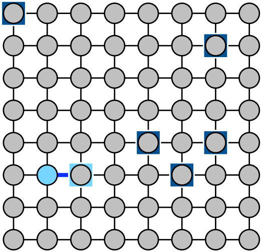
- Start with a grid of static nodes.
- Add $m$ *fully mobile nodes*.
- At each time, $m$ mobile nodes choose new locations uniformly at random.

# Gossip with mobility

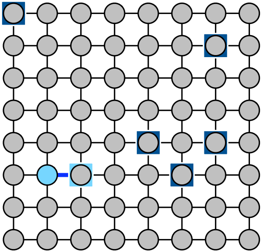


- Start with a grid of static nodes.
- Add $m$ *fully mobile nodes*.
- At each time, $m$ mobile nodes choose new locations uniformly at random.

# Gossip with mobility

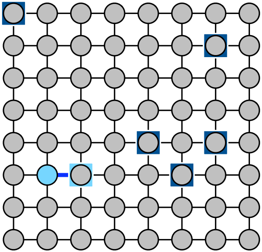


# Gossip with mobility



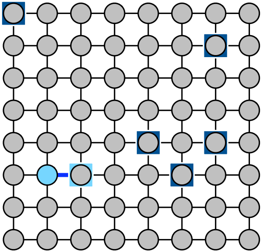
- Same local transmission model.

# Gossip with mobility



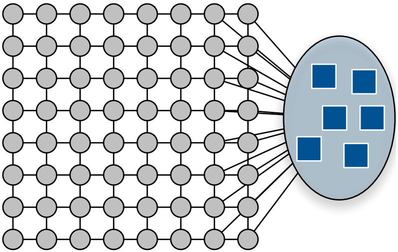
- Same local transmission model.
- Mobile nodes reduce effective diameter to 2.

# Gossip with mobility

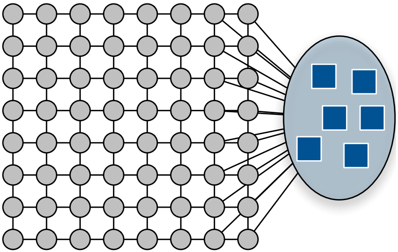


- Same local transmission model.
- Mobile nodes reduce effective diameter to 2.
- Mobile nodes are accessed rarely.

# Lower bounds on $T_{\mathrm{relax}}(\bar{W})$

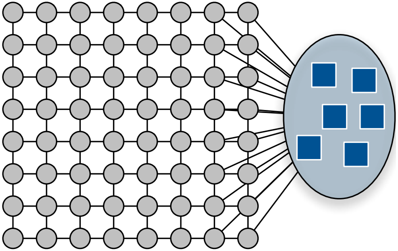


# Lower bounds on $T_{\mathrm{relax}}(\bar{W})$



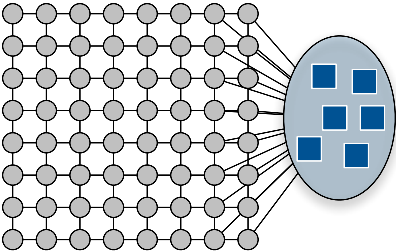
- Merge all mobile nodes into a "super node."

# Lower bounds on $T_{\text{relax}}(\bar{W})$



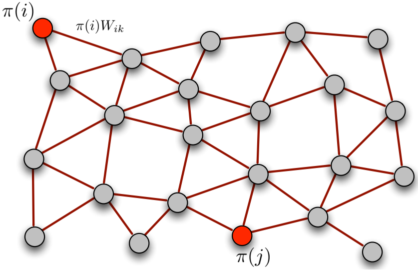
- Merge all mobile nodes into a "super node."
- $T_{\text{relax}}$ for induced chain $\leq T_{\text{relax}}$ for original chain.

# Lower bounds on $T_{\mathrm{relax}}(\bar{W})$



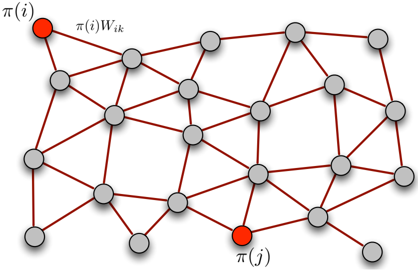
- Merge all mobile nodes into a "super node."
- $T_{\mathrm{relax}}$ for induced chain $\leq T_{\mathrm{relax}}$ for original chain.
- At most a $m$-factor improvement.

# Upper bounds on $T_{\mathrm{relax}}(\bar{W})$




Use a "flow" argument and the Poincaré inequality:

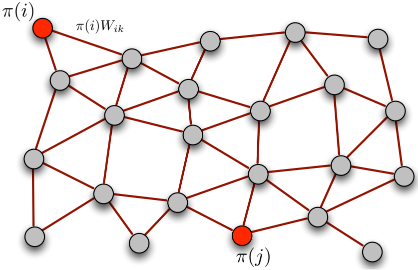
# Upper bounds on $T_{\text{relax}}(\bar{W})$




Use a "flow" argument and the Poincaré inequality:

- Demands $D_{ij} = \pi(i)\pi(j) = n^{-2}$ between each pair of nodes.

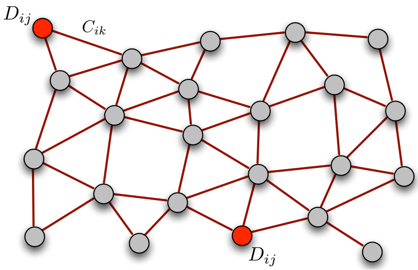
# Upper bounds on $T_{\mathrm{relax}}(\bar{W})$




Use a "flow" argument and the Poincaré inequality:

- Demands $D_{ij} = \pi(i)\pi(j) = n^{-2}$ between each pair of nodes.
- Capacity $C_{ik} = \pi(i)\bar{W}_{ik} = n^{-1}\bar{W}_{ik}$ between each edge.

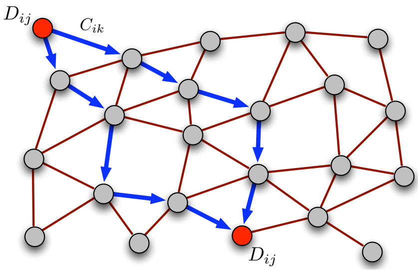
# Upper bounds on $T_{\mathrm{relax}}(\bar{W})$




Use a “flow” argument and the Poincaré inequality:

- Demands $D_{ij} = \pi(i)\pi(j) = n^{-2}$ between each pair of nodes.
- Capacity $C_{ik} = \pi(i)\bar{W}_{ik} = n^{-1}\bar{W}_{ik}$ between each edge.

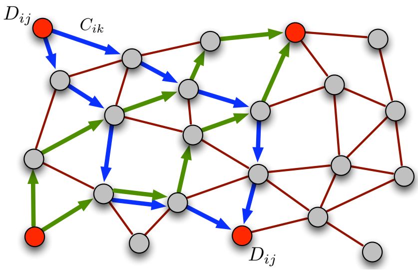
# Upper bounds on $T_{\mathrm{relax}}(\bar{W})$




Use a "flow" argument and the Poincaré inequality:

- Demands $D_{ij} = \pi(i)\pi(j) = n^{-2}$ between each pair of nodes.
- Capacity $C_{ik} = \pi(i)\bar{W}_{ik} = n^{-1}\bar{W}_{ik}$ between each edge.
- Route flows $i \to j$ to minimize *overload* on each edge.

# Upper bounds on $T_{\text{relax}}(\bar{W})$




Use a "flow" argument and the Poincaré inequality:

- Demands $D_{ij} = \pi(i)\pi(j) = n^{-2}$ between each pair of nodes.
- Capacity $C_{ik} = \pi(i)\bar{W}_{ik} = n^{-1}\bar{W}_{ik}$ between each edge.
- Route flows $i \to j$ to minimize *overload* on each edge.

# Network effects on convergence

| algorithm | transmissions | |
|---|---|---|

# Network effects on convergence

| algorithm | transmissions | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |

# Network effects on convergence

| algorithm | transmissions | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |
| **With routing** | $\Theta(n^{3/2})$ | Dimakis-Sarwate-Wainwright |

# Network effects on convergence

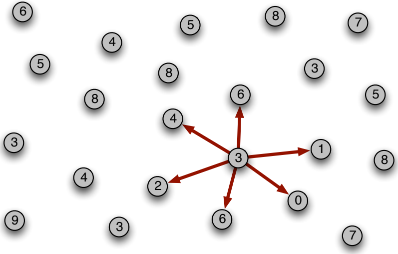
| algorithm | transmissions | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |
| **With routing** | $\Theta(n^{3/2})$ | Dimakis-Sarwate-Wainwright |
| **Average on the way** | $\Theta(n)$ | Benezit et al. |

# Network effects on convergence

| algorithm | transmissions | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |
| **With routing** | $\Theta(n^{3/2})$ | Dimakis-Sarwate-Wainwright |
| **Average on the way** | $\Theta(n)$ | Benezit et al. |
| **Add $m$ mobile** | $\Theta\left(\frac{n^2}{m}\right)$ | Sarwate-Dimakis |

# Network effects on convergence

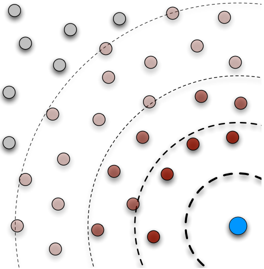
| algorithm | transmissions | |
|---|---|---|
| **Local** | $\Theta(n^2)$ | Boyd et al. |
| **With routing** | $\Theta(n^{3/2})$ | Dimakis-Sarwate-Wainwright |
| **Average on the way** | $\Theta(n)$ | Benezit et al. |
| **Add $m$ mobile** | $\Theta\left(\frac{n^2}{m}\right)$ | Sarwate-Dimakis |
| $k$**-local** | $O\left(\frac{n^2}{k^2}\right)$ | Sarwate-Dimakis |



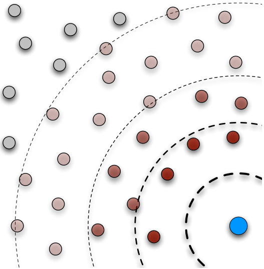

# Asymmetric gossip using broadcasting

Joint work with T.C. Aysal, M.E. Yildiz and A. Scaglione

# Wireless is inherently broadcast

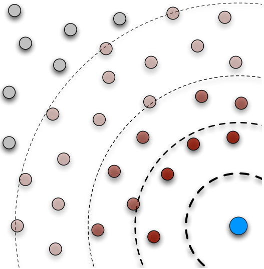


# Wireless is inherently broadcast



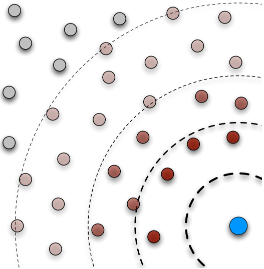
- In a wireless network, all neighbors can hear a transmission.

# Wireless is inherently broadcast



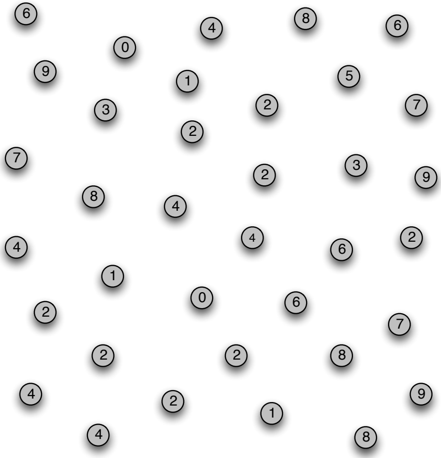
- In a wireless network, all neighbors can hear a transmission.
- Can perform multiple computations per slot.

# Wireless is inherently broadcast

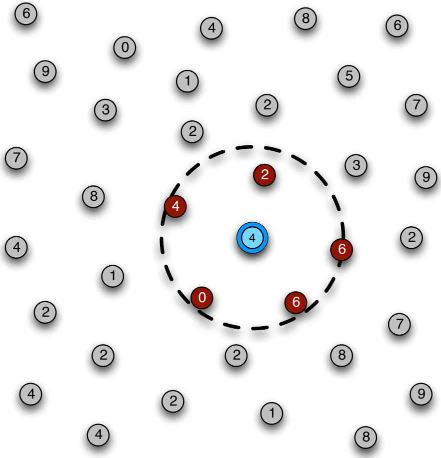


- In a wireless network, all neighbors can hear a transmission.
- Can perform multiple computations per slot.
- When graph is well-connected, can get performance gains.

# Gossip in one direction

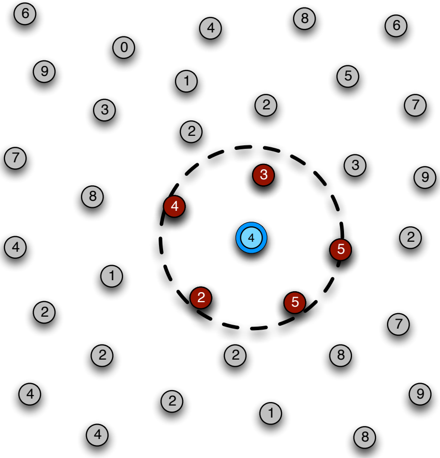


# Gossip in one direction



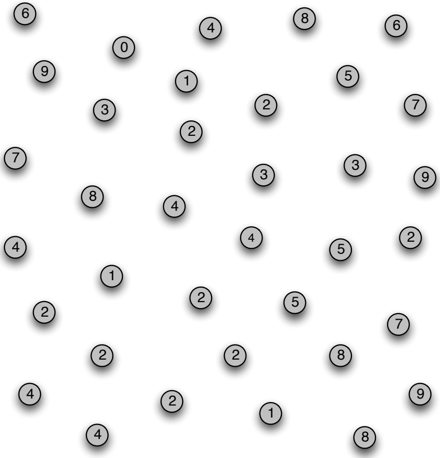
- All neighbors $j \in \mathcal{N}_i$ of node $i$ can hear transmission.

# Gossip in one direction



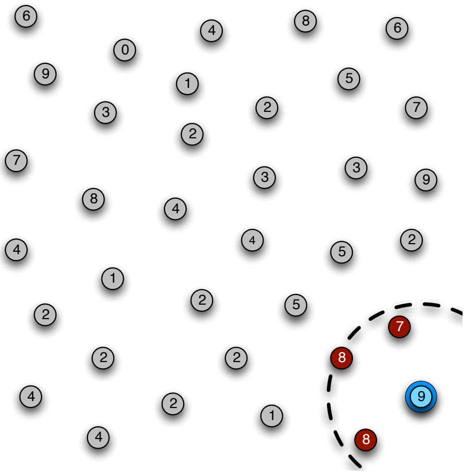

- All neighbors $j \in \mathcal{N}_i$ of node $i$ can hear transmission.
- Can do a **simultaneous update** $x_j(t+1) = \gamma x_j(t) + (1-\gamma)x_i(t)$.

# Gossip in one direction



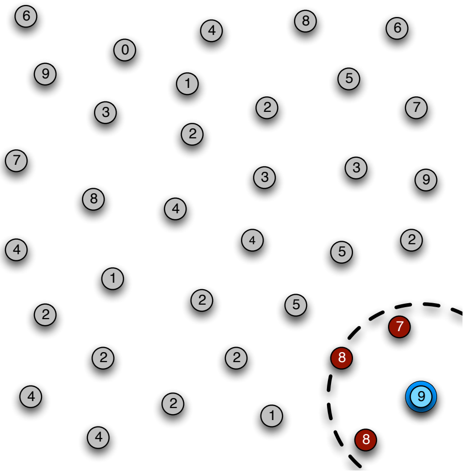

- All neighbors $j \in \mathcal{N}_i$ of node $i$ can hear transmission.
- Can do a **simultaneous update** $x_j(t+1) = \gamma x_j(t) + (1-\gamma)x_i(t)$.

# Gossip in one direction



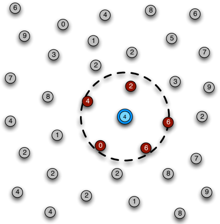

- All neighbors $j \in \mathcal{N}_i$ of node $i$ can hear transmission.
- Can do a **simultaneous update** $x_j(t+1) = \gamma x_j(t) + (1-\gamma)x_i(t)$.

# Gossip in one direction



- All neighbors $j \in \mathcal{N}_i$ of node $i$ can hear transmission.
- Can do a **simultaneous update** $x_j(t+1) = \gamma x_j(t) + (1-\gamma) x_i(t)$.
- No information **exchange** – can get **consensus** (agreement) but not the **true average**.

# Analyzing the broadcast gossip algorithm

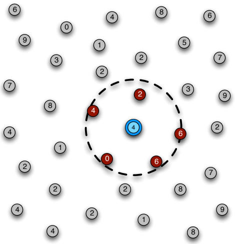


Again, update given by a matrix multiplication:

$$\mathbf{x}(T) = \left( \prod_{t=1}^{T} W^{(i_t)} \right) \mathbf{x}(0)$$

For all $t$ we have $W^{(i_t)}\mathbf{1} = \mathbf{1}$, so consensus is *stable*.

# Benefits and challenges of broadcast




- No coordination to exchange data.
- Exploits potential long-range connections from shadowing/fading.
- No convergence to true average, but to *consensus*.
- Important to control the MSE of the consensus.

# Main results

Algorithm reaches consensus almost surely:

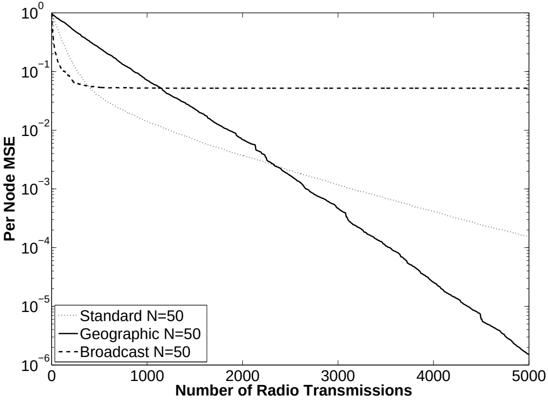
$$\mathbb{P}\left(\lim_{t\to\infty} \mathbf{x}(t) = c\mathbf{1}\right) = 1.$$

The expected consensus value is the true average:

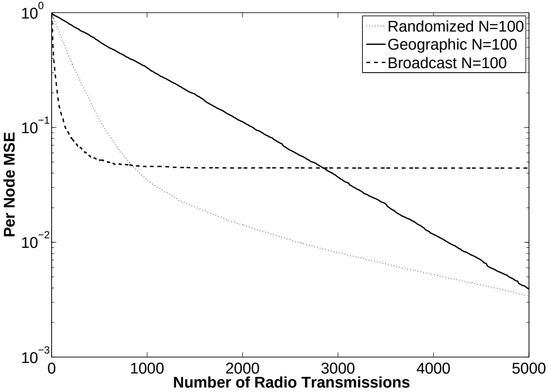
$$\mathbb{E}[c] = \bar{x}$$

Moreover, there is a closed form for the limiting MSE.

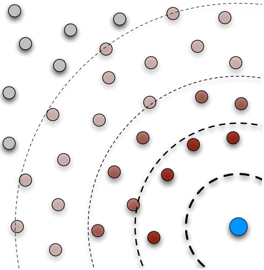
# Simulations : MSE



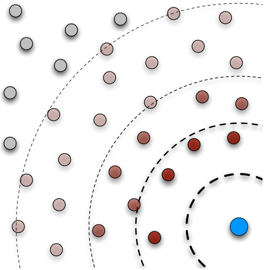
# Simulations : MSE



# Extensions

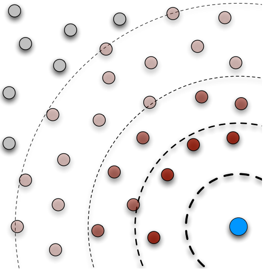


# Extensions



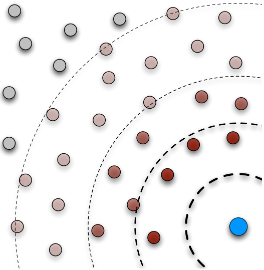
- Can look at effect of the wireless medium as well.

# Extensions



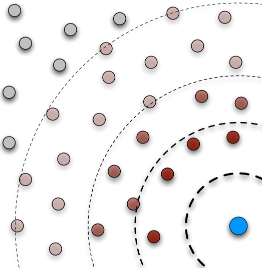
- Can look at effect of the wireless medium as well.
- Fading allows long-distance connections.

# Extensions

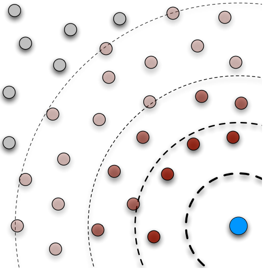


- Can look at effect of the wireless medium as well.
- Fading allows long-distance connections.
- Initial results suggest significant improvement when path loss is not too severe.

# Implications

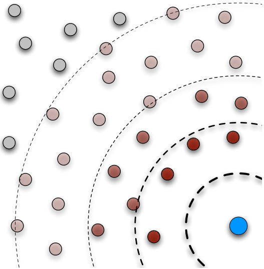


# Implications



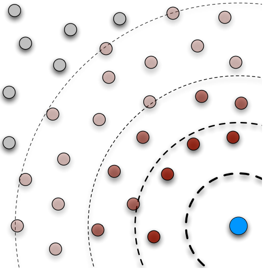
- Broadcasting is simpler than standard gossip – no exchange.

# Implications

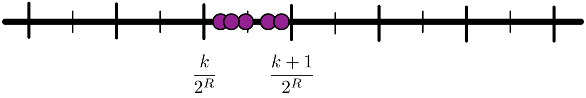


- Broadcasting is simpler than standard gossip – no exchange.
- More robust to packet drops which may occur in wireless.

# Implications



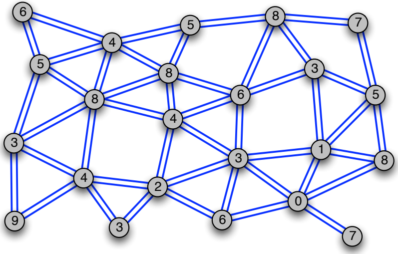
- Broadcasting is simpler than standard gossip – no exchange.
- More robust to packet drops which may occur in wireless.
- Faster convergence in small-to-medium networks.




# Reaching consensus discretely

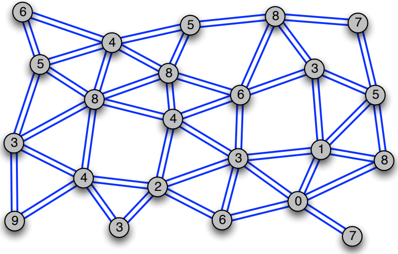
Joint work with Tara Javidi

# Typical assumptions are unrealistic




Existing work doesn't "look practical":

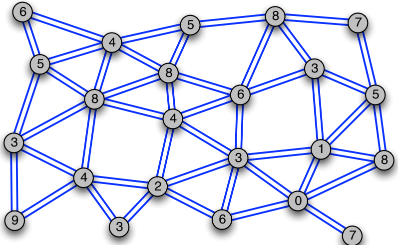
# Typical assumptions are unrealistic




Existing work doesn't "look practical":

- Transmit and receive real numbers

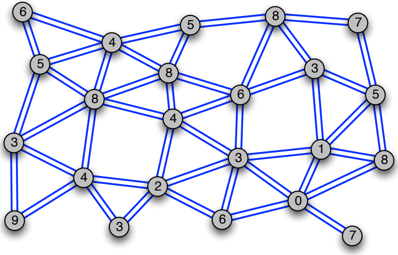
# Typical assumptions are unrealistic




Existing work doesn't "look practical":

- Transmit and receive real numbers
- Consensus is the only goal of the network

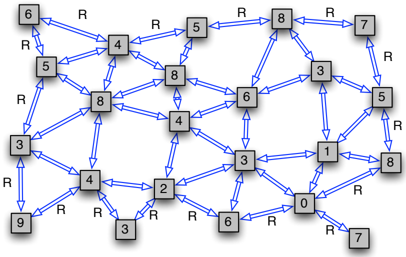
# Typical assumptions are unrealistic



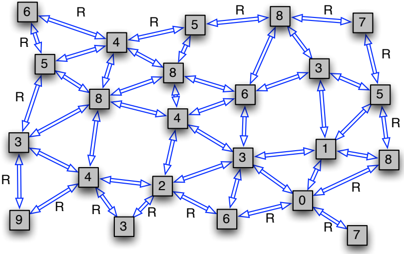

Existing work doesn't "look practical":

- Transmit and receive real numbers
- Consensus is the only goal of the network
- Asymptotics and universality

# Synchronous quantized communication

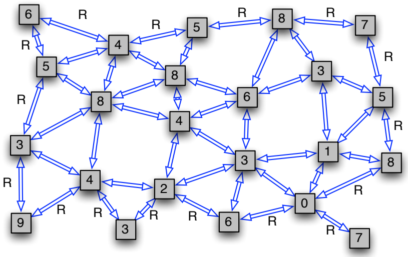


# Synchronous quantized communication



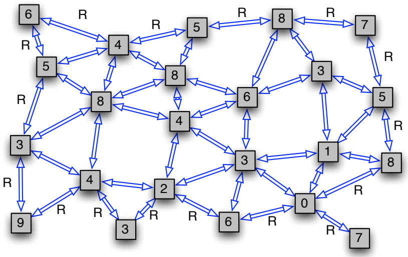

- At each time $t$ all neighbors $(i, j)$ exchange quantized values $\hat{x}_j(t)$.

# Synchronous quantized communication



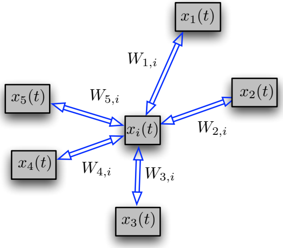
- At each time $t$ all neighbors $(i, j)$ exchange quantized values $\hat{x}_j(t)$.
- Messages $i \rightarrow j$ and $j \rightarrow i$ must take no more than $R$ bits.

# Synchronous quantized communication




- At each time $t$ all neighbors $(i, j)$ exchange quantized values $\hat{x}_j(t)$.
- Messages $i \rightarrow j$ and $j \rightarrow i$ must take no more than $R$ bits.
- Update $x_i(t+1)$ as a function of $x_i(t)$ and messages $\{\hat{x}_j(t)\}$.

# A simple protocol



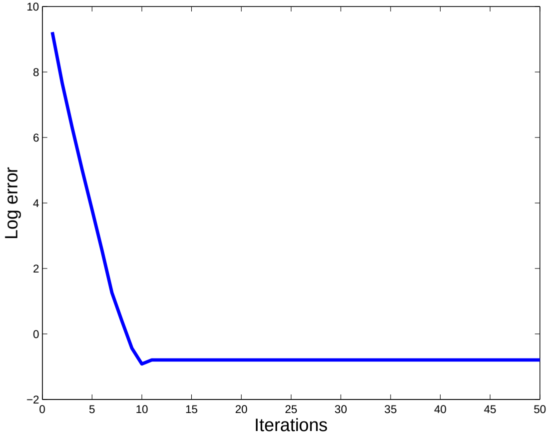

$$x_i(t+1) = (x_i(t) - \hat{x}_i(t)) + \sum_{j \in \mathcal{N}_i \cup \{i\}} W_{ij} \hat{x}_j(t).$$

- Quantization error plus weighted sum of messages
- Iterations preserve sum $\sum_i x_i(t)$

# So how well does it work?

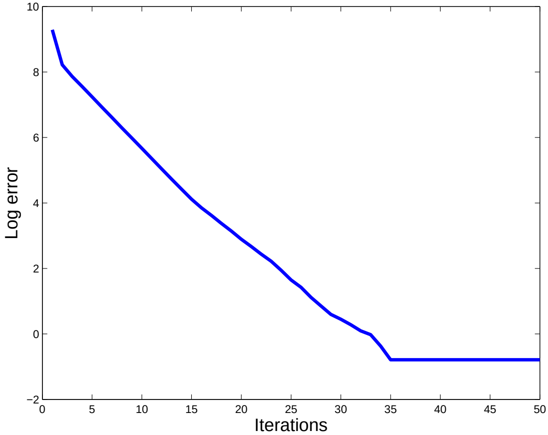
# So how well does it work?

## Random topology, 49 nodes, good connectivity

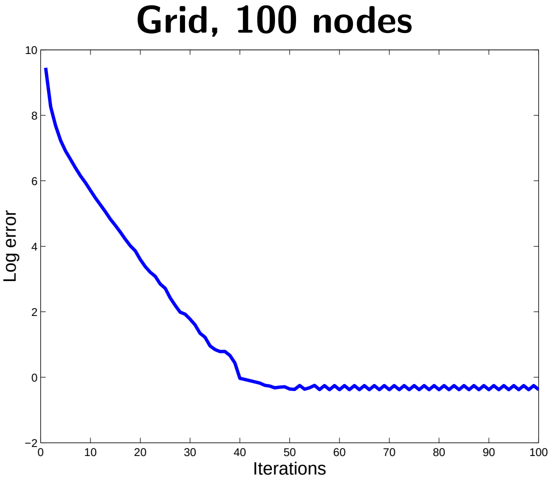


# So how well does it work?

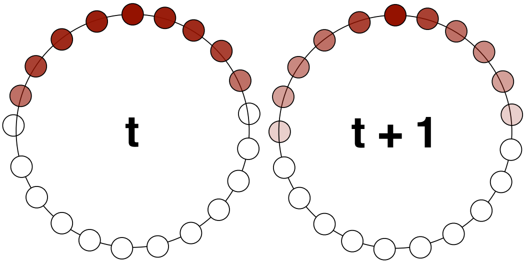
## Random topology, 49 nodes, poor connectivity



# So how well does it work?

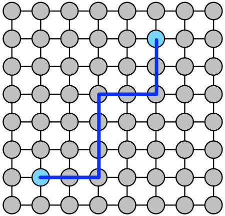


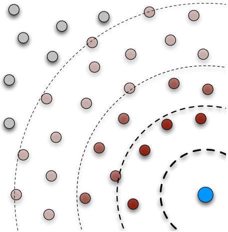
# Observations

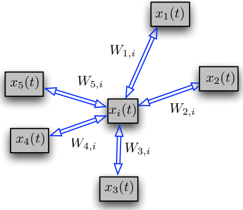



- Quantization is important for practical applications.
- Average consensus to within reasonable resolution can be fast.
- Overhead can be reduced by piggybacking on existing traffic.

# Conclusions

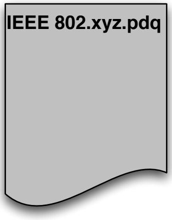


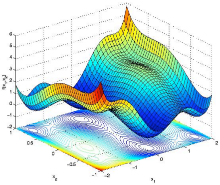


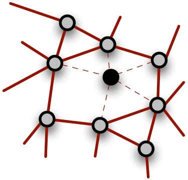



- Algorithm can use network resources to accelerate convergence.
- Reaching consensus may be faster than computing averages.
- Lower-resolution averages can be fast and require less overhead.

# Some challenges for the future





- Implementing consensus in protocols for applications.
- Extending to other distributed computation problems.
- Quantifying robustness in rate, connectivity, etc.

# Thank you!