Comparisson of State-of-the-Art Consensus Algorithms

Richard Heusdens

Introduction

Unlike the Telephone network or the Internet, many of the next generation networks are not engineered for the purpose of providing efficient communication between various networked entities

Examples of such networks are

- Sensor networks
- Peer-to-peer networks
- Mobile networks of vehicles
- Social networks
- · · ·

Introduction

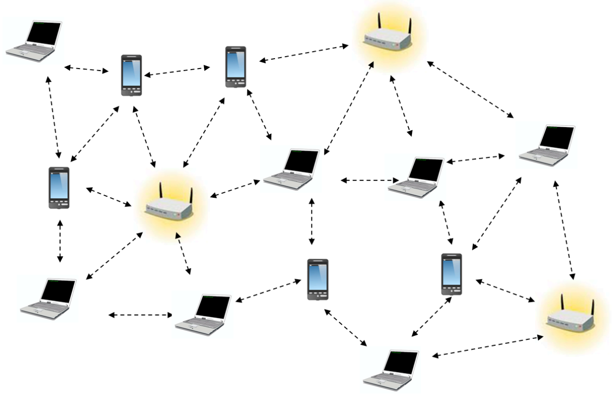
Modern networks lack infrastructure; they exhibit unpredictable dynamics and they face stringent resource constraints

Algorithms operating within them need to be

- Extremely simple
- Distributed
- Robust against networks dynamics
- Efficient in resource utilization

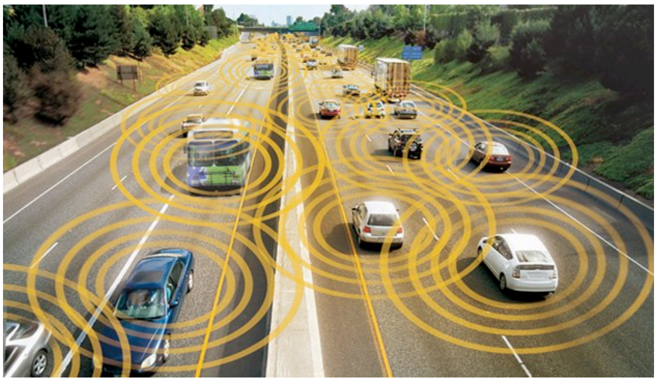
Introduction

Peer-to-peer network



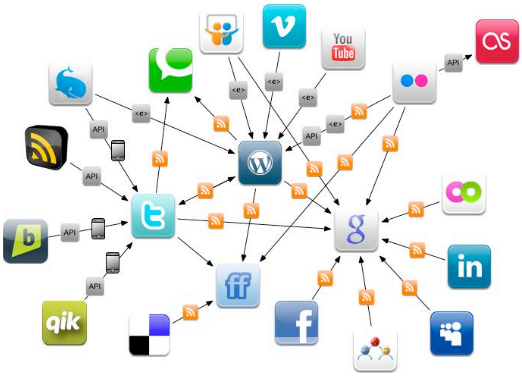
Introduction

Mobile network of vehicles



Introduction

Social network



Introduction

- Distributed consensus: distributed averaging (2003), randomized gossip (2006), geographic gossip (2008), weighted gossip (2010), greedy gossip with eavesdropping (2010), broadcast gossip (2011)
- Convex optimization: dual ascent (mid-1960's), method of multipliers (ADMM, late-1960's), alternating direction method of multipliers (1976), distributed subgradient methods (2009), primal-dual method of multipliers (PDMM, 2014)
- Probabilistic inference: (loopy) belief propagation (min-sum or max-product algorithm, 1982), approximate inference (linear programming relaxation, 2010)

Outline

- Distributed averaging
  - Synchronous
  - Asynchronous
- Gossip algorithms:
  - Randomized gossip
  - Geographic gossip
  - Gossip with eavesdropping
  - Sum-weight averaging
  - Broadcast weighted gossip

Distributed averaging

Each node i holds an initial scalar value $x_i(0) \in \mathbb{R}$, and $x(0) = (x_1(0), \ldots, x_n(0))^T$ denotes the vector of the initial node values on the network.

- The network gives the allowed communication between nodes: two nodes i and j can communicate with each other if and only if they are neighbors, i.e., if and only if $(i,j) \in E$
- We are interested in computing the average of the initial values

$$x_{\text{ave}} = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$$

Distributed averaging

We consider *linear iterations* where each iteration k is of the form

$$x_i(k) = W_{ii}x_i(k-1) + \sum_{j\in\mathcal{N}(i)} W_{ij}x_j(k-1), \quad i \in V$$

Setting $W_{ij} = 0$ for $j \notin \mathcal{N}(i) \cup i$, the iterations can be expressed in vector form as

$$x(k) = Wx(k-1)$$

We want to choose the weight matrix W so that for any initial vector $x(0)$, $x(k)$ converges to the vector of averages $x_{\text{ave}}\mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^n$ is the vector of all ones.

Distributed averaging

We will consider the behavior of

$$x(k) = W^k x(0)$$

If $x(k)$ must converge to the vector of averages given by

$$x_{\text{ave}}\mathbf{1} = \frac{\mathbf{1}\mathbf{1}^T}{n} x(0)$$

for every initial condition $x(0)$, we must have that

$$\lim_{k\to\infty} W^k = \frac{\mathbf{1}\mathbf{1}^T}{n}$$

Distributed averaging

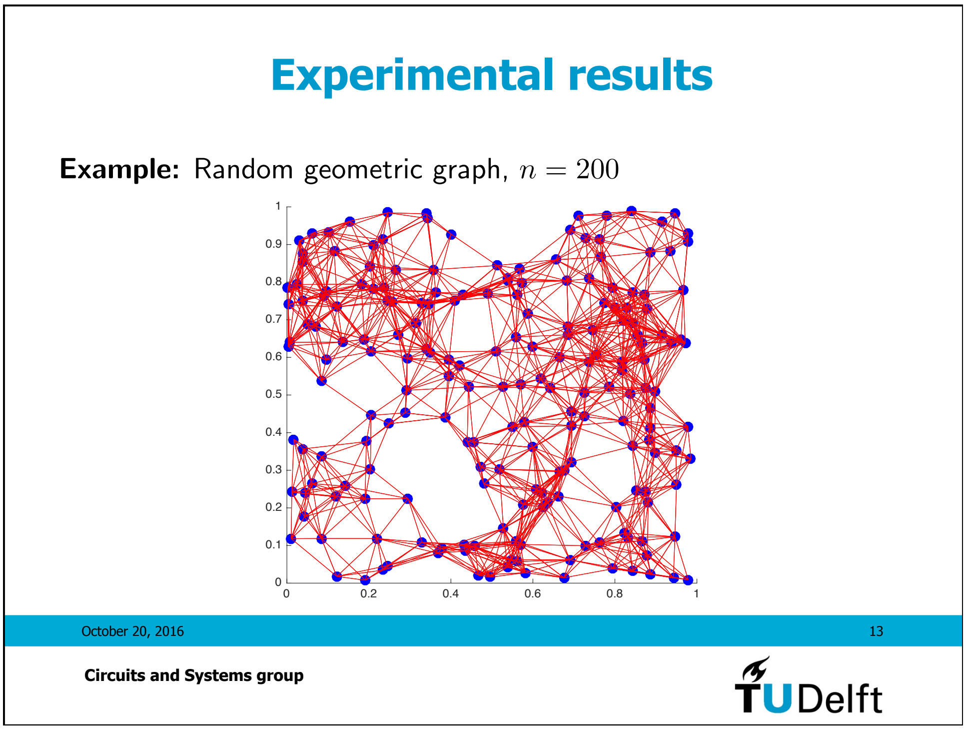
We have the following necessary and sufficient conditions for

$$\lim_{k\to\infty} W^k = \frac{\mathbf{1}\mathbf{1}^T}{n}$$

to hold:

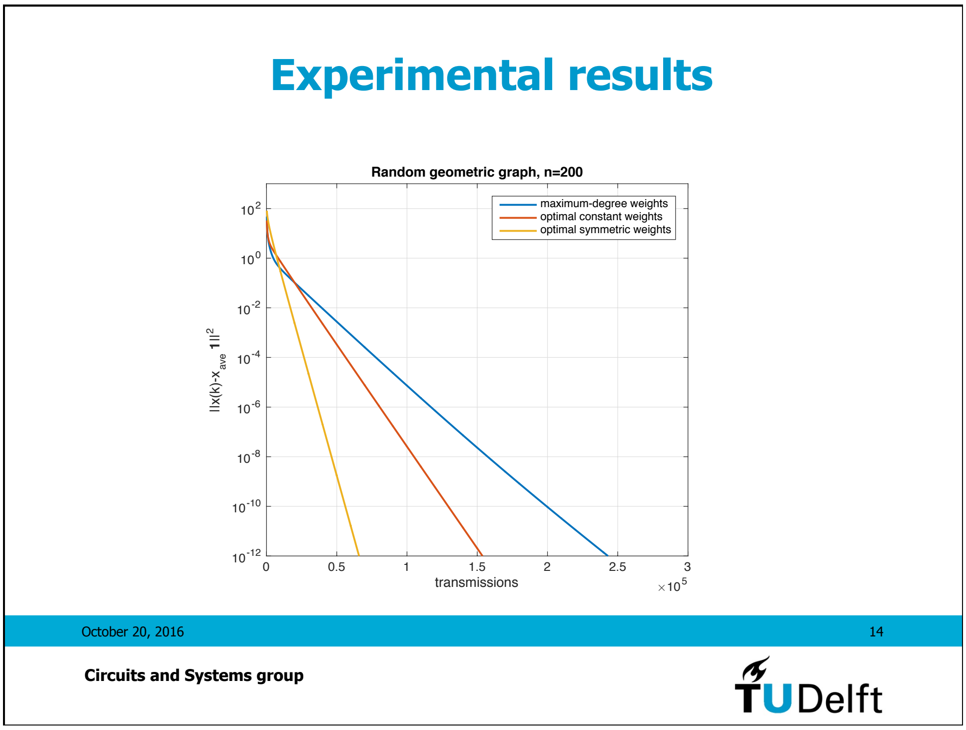
- $\mathbf{1}^T W = \mathbf{1}^T$
- $W\mathbf{1} = \mathbf{1}$
- $\rho\left(W - \mathbf{1}\mathbf{1}^T/n\right) < 1$

where $\rho(\cdot)$ denotes the spectral radius



Experimental results

Example: Random geometric graph, $n = 200$



Experimental results

Random geometric graph, n=200

Distributed averaging

The distributed averaging algorithm is a *synchronous* algorithm

- Requires global clock (synchronization)
- Sensitive to changes in network topology

Asynchronous algorithms have the advantage of

- Less sensitive to changes in network topology
- Local clock; no synchronization needed
- Execution time is often substantially reduced compared to synchronous implementations

---

Asynchronous distributed averaging

We want an algorithm where each iteration k is of the form

$$x(k) = W(k)x(k-1),$$

and $x(k)$ converges to the vector of averages $x_{\text{ave}}\mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^n$ is the vector of all ones. That is

$$\lim_{k\to\infty} \phi(k) = \lim_{k\to\infty} W(k)\cdots W(1) = \frac{\mathbf{1}\mathbf{1}^T}{n}$$

Asynchronous distributed averaging

Let the expected value of the matrices $W(k)$ be denoted by $E\{W(k)\} = \overline{W}$. We have

$$E\{\phi(k)\} = \prod_{i=1}^{k} E\{W(k)\} = \overline{W}^k$$

so that $\phi(k)$ *converges in expectation* to $\frac{\mathbf{1}\mathbf{1}^T}{n}$ if $\overline{W}^k \to \frac{\mathbf{1}\mathbf{1}^T}{n}$. This happens if

- $\mathbf{1}^T\overline{W} = \mathbf{1}^T$
- $\overline{W}\mathbf{1} = \mathbf{1}$
- $\rho\left(\overline{W} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) < 1$

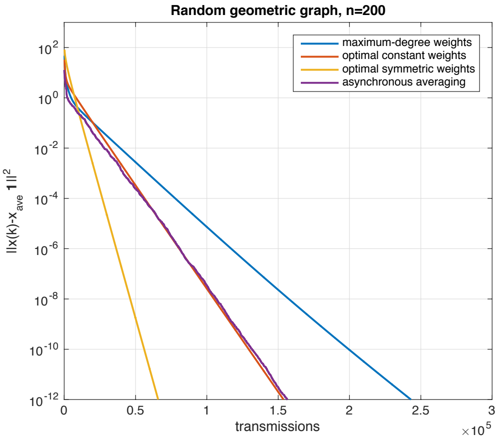
Asynchronous distributed averaging

Consider the following asynchronous distributed algorithm:

Algorithm:

- In the kth time slot, select a node i at random (uniform distribution, probability $P_i = 1/n$) and let it contact its neighboring nodes $j \in \mathcal{N}(i)$.
- At this time, all nodes set their values equal to the average of their current values.

Experimental results



Asynchronous distributed averaging

Conclusions:

- Asynchronous distributed averaging doesn't need *any* global knowledge of the network
- The algorithm converges at a linear rate

Still the algorithm can be quite sensitive to changes in the network topology

Solution:

- In a given time slot, each node can communicate with only *one* of its neighbours *(Gossip algorithms)*

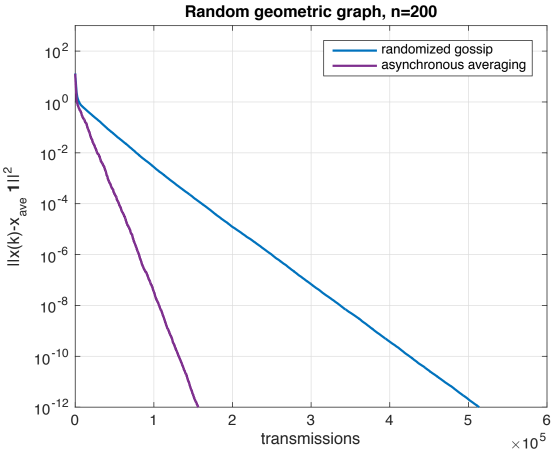
Randomized gossip

Consider the randomized gossip algorithm which is specified by a matrix $P \in \mathbb{R}^{n\times n}$ of nonnegative entries with the condition that $P_{ij} > 0$ if and only if $(i,j) \in E$.

Algorithm:

- In the kth time slot, select a node i at random (uniform distribution, probability $1/n$) and let it contact some neighbouring node j with probability P_{ij}.
- At this time, both nodes set their values equal to the average of their current values.

Experimental Results



Randomized gossip

- Randomized gossip is slow on random geometric graphs
- We can do better by including additional information:
  - knowledge about sensor locations
  - knowledge about states of neighboring nodes
  - use of broadcast protocols

Geographic gossip

Geographic gossip combines gossip with geographic routing

Key assumption: every node i knows its geographic location $l(i)$ within some compact subset of $C \subset \mathbb{R}^d$

- At clock tick k, node i is selected uniformly at random
- Node i chooses a point y uniformly in C (called the target location)
- Node i sends the tuple $(x_i(k), l(i), y)$ to its one-hop neighbor $j \in \mathcal{N}(i)$ closest to y until a node, say v, receives the tuple and has no one-hop neighbor with distance smaller to the random target than its own

Geographic gossip

- Node v decides to accept the tuple or not. If it accepts, it computes a new tuple $(x_v(k), l(v), l(i))$, which is sent back to node i via greedy geographic routing
- If v rejects the tuple it chooses a new point y' selected uniformly in C and repeats the previous steps

Sampling geographic locations uniformly induces a nonuniform sampling distribution. Given the volume ν_v of the Voronoi region of node v:

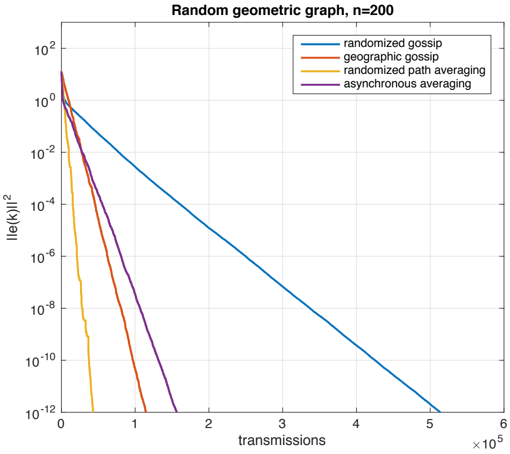
- Sensor v accepts the request with probability $p_a = \min(\tau/\nu_v, 1)$

where τ is a predefined threshold

Geographic gossip

- At a high level, geographic gossip exploits geographic information to create a new complete communication graph $G'(V, E')$ as a overlay on the original graph $G(V, E)$
- In the new graph some edges are more costly than others because of the routing required
- On the other hand, the new communication graph is dense so that gossiping converges more quickly
- The communication costs can be reduced by averaging over the complete path from source i to the destination v. The resulting algorithm is called *randomized path averaging*

Experimental Results



Greedy gossip with eavesdropping

Both geographic gossip and randomized path averaging combine gossip with geographic routing

- Involves overhead due to localization and geographic routing
- The network needs to provide reliable two-way transmission over many hops
- Sensitive to packet loss and changes in network topology

Problem can be solved by using *greedy gossip with eavesdropping*

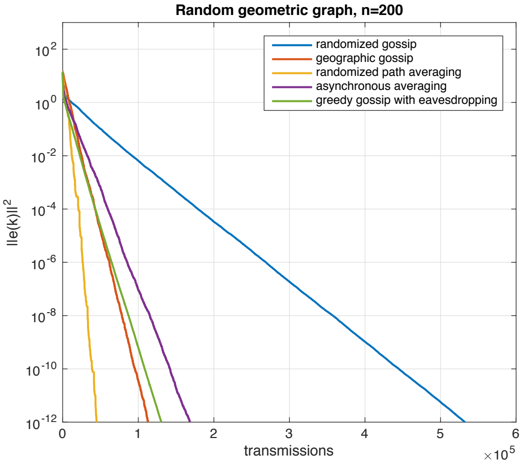
- Exploits the broadcast nature of wireless communications

Greedy gossip with eavesdropping

- Unlike previous randomized gossip algorithms, which perform updates completely at random, greedy gossip with eavesdropping implements a greedy neighbor selection procedure
- Assumes a broadcast transmission model so that all neighbors within range of a transmission node receive the message
- In addition to keeping track of its own value, each node tracks its neighboring values by eavesdropping on their transmissions
- At each iteration, a node is selected uniformly at random and contacts the neighboring node whose value is most different from its own

---

Experimental Results



Sum-weight algorithms

As mentioned before, randomized path averaging combines gossip with geographic routing

- Although efficient in terms of energy consumption, it requires some long distance coordination to make sure that all the values in the route were updated correctly
- Routing information back and forth might as well introduce delay issues, because a node that is engaged in a route needs to wait for the update to come back before it can proceed to another round
- In a mobile network, or in a highly dynamic network, routing the information back on the same route might even not succeed

Sum-weight algorithms

Problem can be solved by using *one-way averaging*:

- Instead of updating one vector $x(k)$ of variables, it updates a vector $s(k)$ of sums, and a vector $w(k)$ of weights
- Initialization $s(0) = x(0)$ and $w(0) = \mathbf{1}$
- At any time, the vector of estimates is $x(k) = s(k)/w(k)$, where the division is performed element-wise

The updates are computed with *column stochastic matrices* $\{D(k)\}_{k\geq 1}$

$$\begin{aligned} s(k) &= D(k)s(k-1) \\ w(k) &= D(k)w(k-1) \end{aligned}$$

Sum-weight algorithms

The algorithm is referred to as *weighted gossip*

The matrix $D(k)$ is column stochastic:

$$D(k) \geq 0, \qquad \mathbf{1}^T D(k) = \mathbf{1}^T$$

which means that sums and weights are conserved. Similar to what we did before, let

$$\phi(k) = D(k) \cdots D(1)$$

Note that $\lim\limits_{k \to \infty} \phi(k) \neq \dfrac{\mathbf{1}\mathbf{1}^T}{n}$ since $D(k)\mathbf{1} \neq \mathbf{1}$; sums and weights do *not* reach a consensus

---

One-way gossip

In general, when we average over 2 neighboring nodes i and j, we have (after re-indexing)

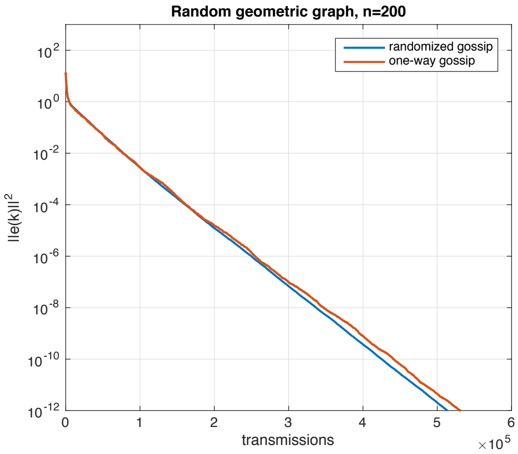
$$D(k) = \left(\begin{array}{cc|c} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \hline 0 & 0 & I_{n-2} \end{array} \right) \in \mathbb{R}^{n \times n},$$

so that the update equations are given by

$$\begin{cases} s_i(k) = \dfrac{1}{2} s_i(k-1) \\ s_j(k) = s_j(k-1) + \dfrac{1}{2} s_i(k-1) \end{cases}$$

and similarly for the weights

Experimental Results



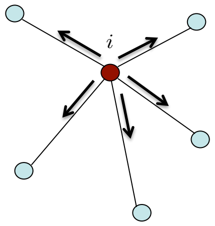
Broadcast weighted gossip

- Broadcast nature of the wireless channel was often not taken into account in the distributed estimation algorithms
- Information propagation is much faster while broadcasting compared to pairwise exchanges

Broadcast algorithm:

At each global clock tick, choose a sensor uniformly at random that broadcasts its pair of values in an appropriate way. Then, the receiving sensors add the received pair of values to their current one

Broadcast weighted gossip



$$D_{i\to\mathcal{N}(i)}(k) = \begin{pmatrix} 1-d_i\alpha^{-1} & 0 & \cdots & 0 \\ \alpha^{-1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \alpha^{-1} & 0 & \cdots & 1 \end{pmatrix}$$

The update equations are given by

$$\begin{cases} s_i(k) = (1-d_i\alpha^{-1})s_i(k-1) \\ s_j(k) = s_j(k-1) + \alpha^{-1}s_i(k-1) \quad \forall j \in \mathcal{N}(i) \end{cases}$$

and similarly for the weights

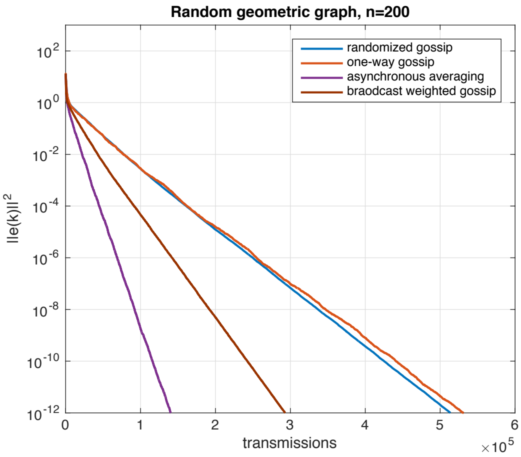
Broadcast weighted gossip

Since $D(k) \geq 0$ we have $\alpha > d_i$. A natural choice is $\alpha = d_i + 1$ so that

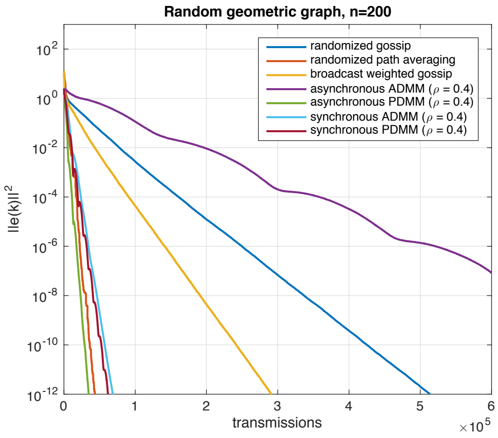
$$D_{i\to\mathcal{N}(i)}(k) = \begin{pmatrix} \frac{1}{d_i+1} & 0 & \cdots & 0 \\ \frac{1}{d_i+1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \frac{1}{d_i+1} & 0 & \cdots & 1 \end{pmatrix}$$

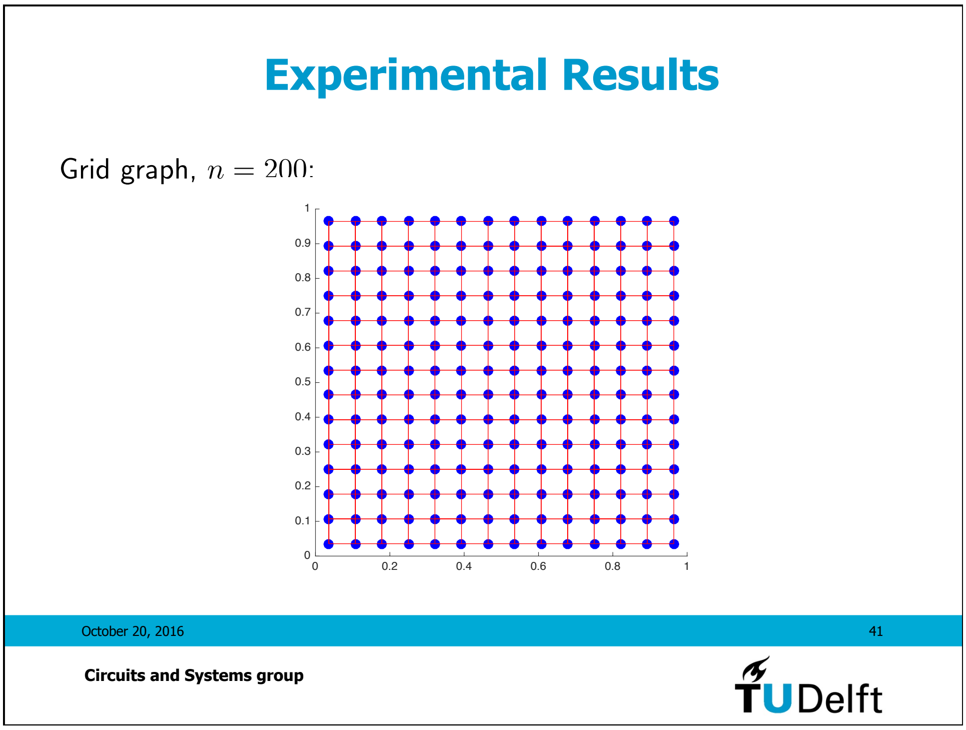
The algorithm is referred to as *broadcast weighted gossip*

Experimental Results



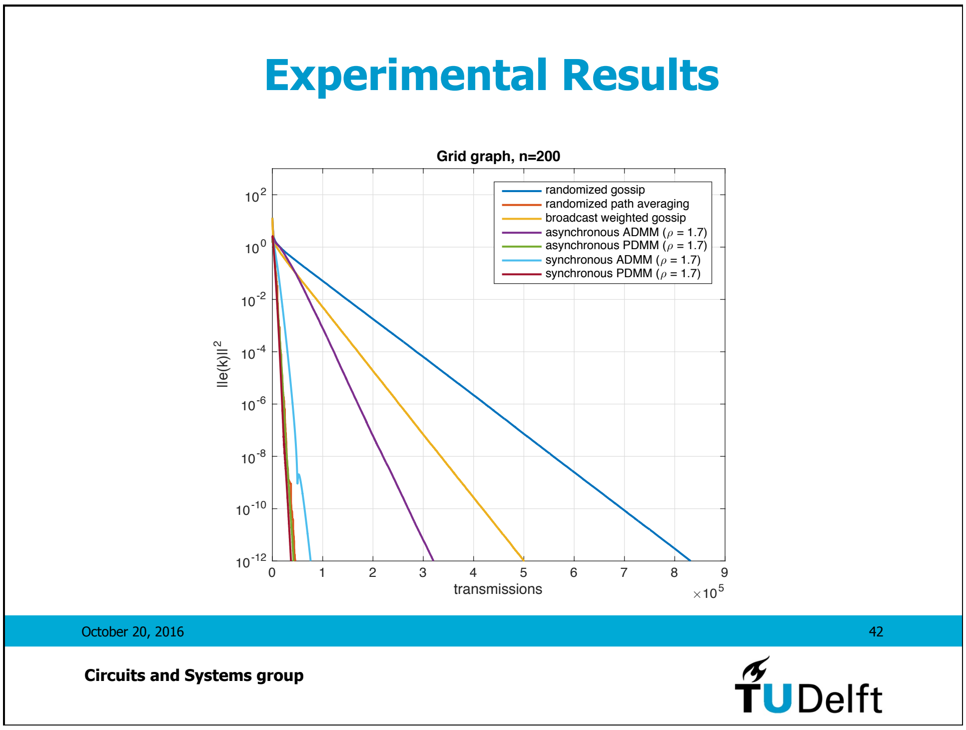
Experimental Results





Experimental Results

Grid graph, $n = 200$:



Experimental Results