CS-541
Wireless Sensor Networks

Lecture 9: Distributed In-network Processing for WSN

Spring Semester 2017-2018

Prof Panagiotis Tsakalides, Dr Athanasia Panousopoulou, Dr Gregory Tsagkatakis

Today's objectives

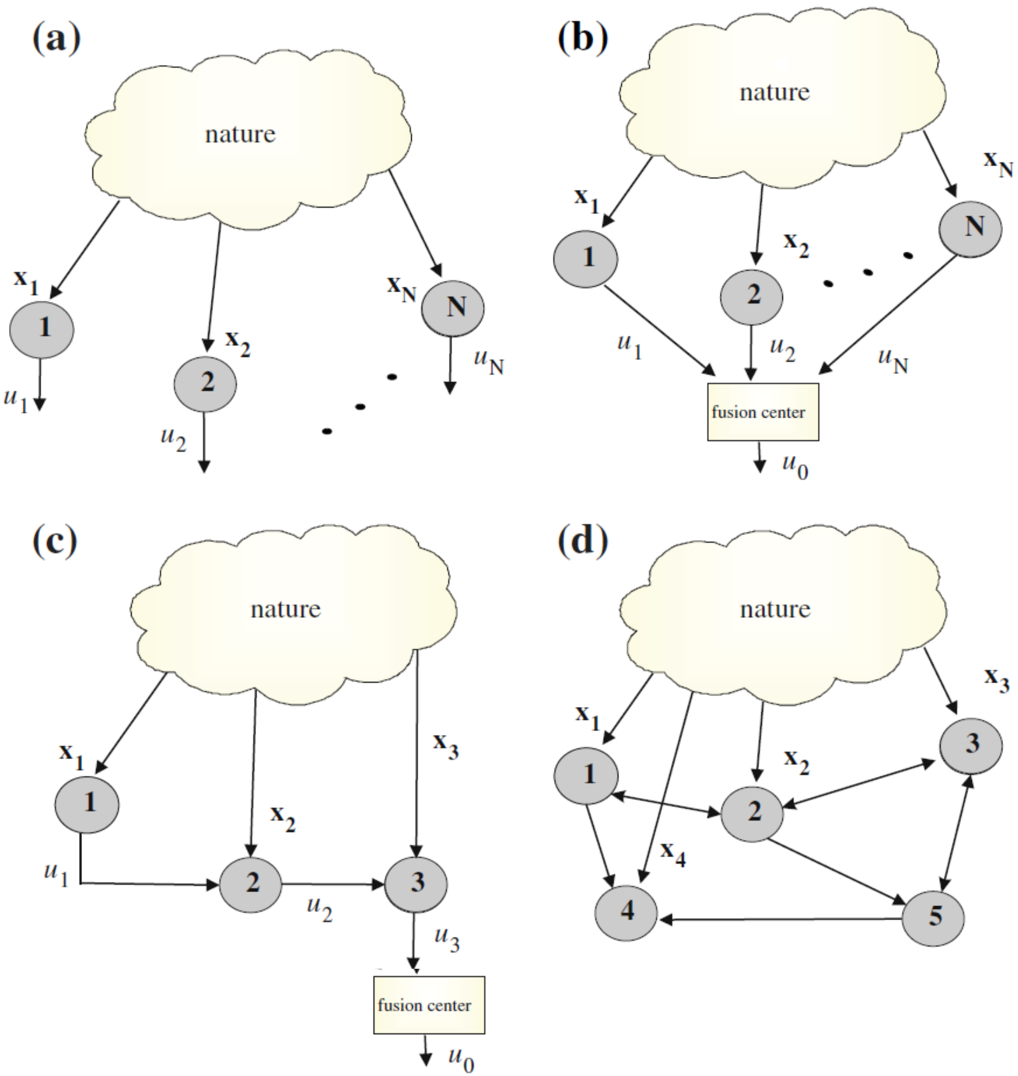
Network signal processing types

Centralized → Graph signal processing

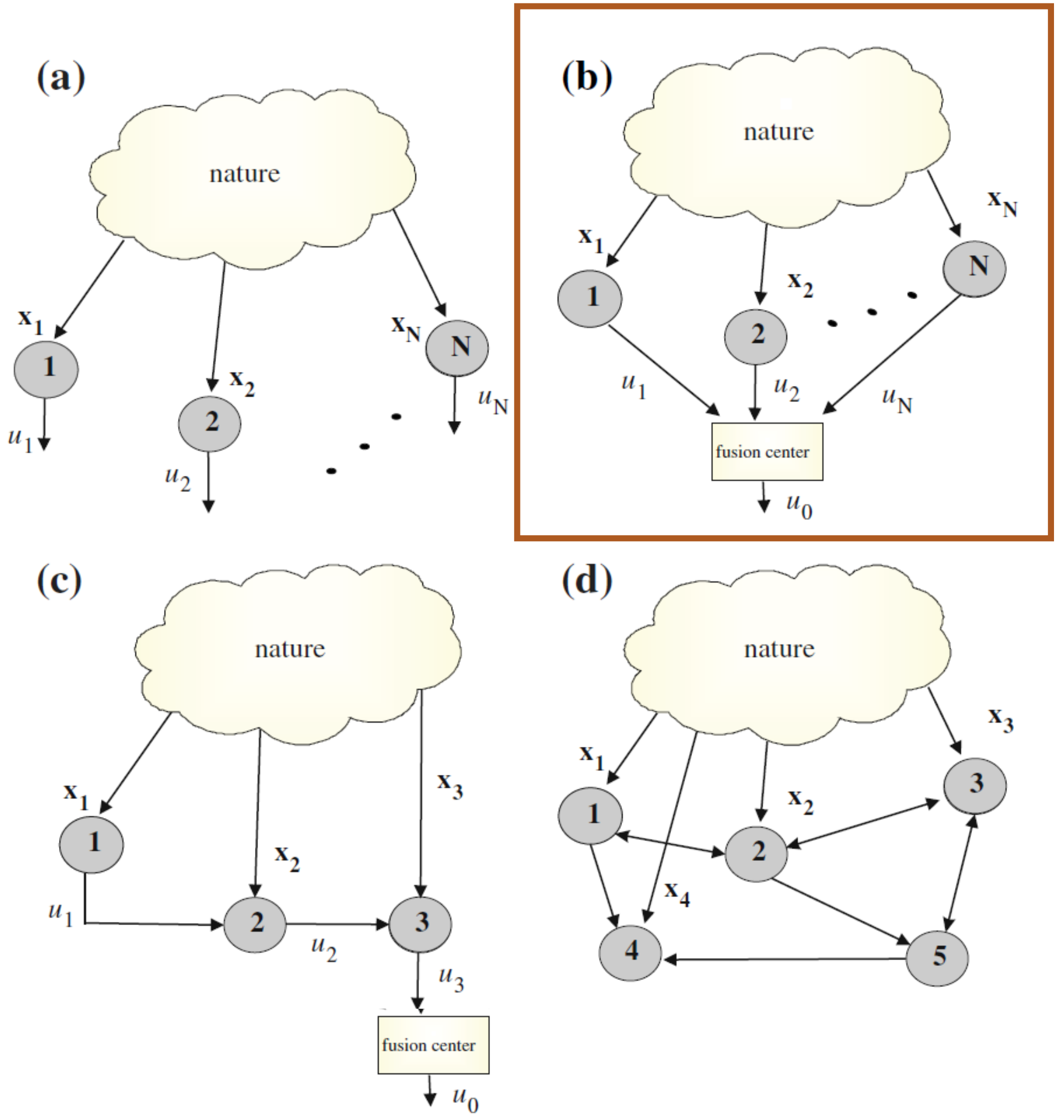
Routing based → Network coding

Distributed → Gossip

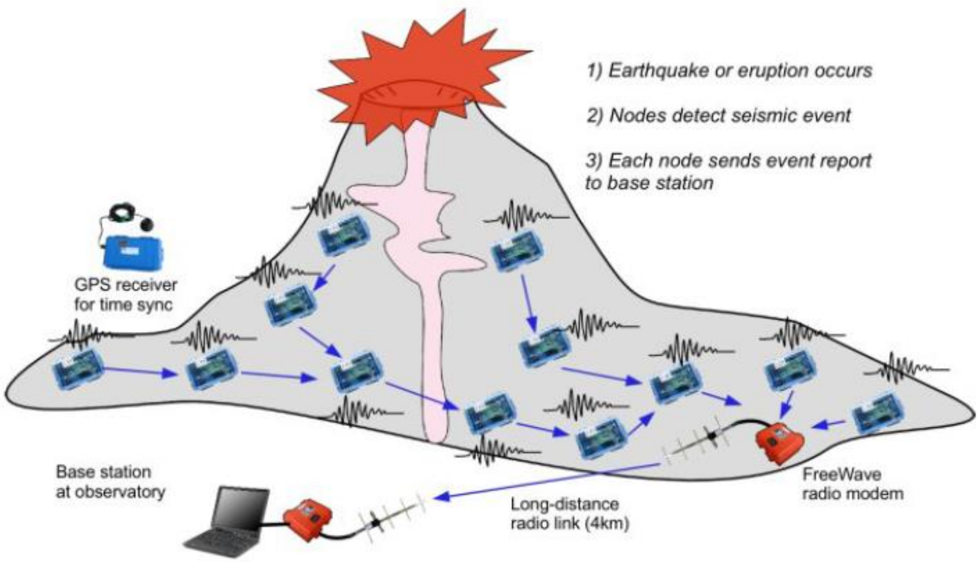
Communication architectures

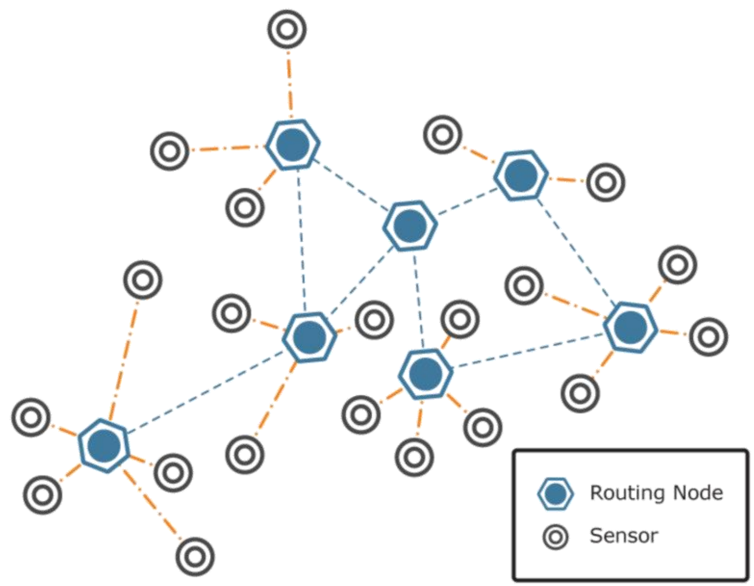


Communication architectures



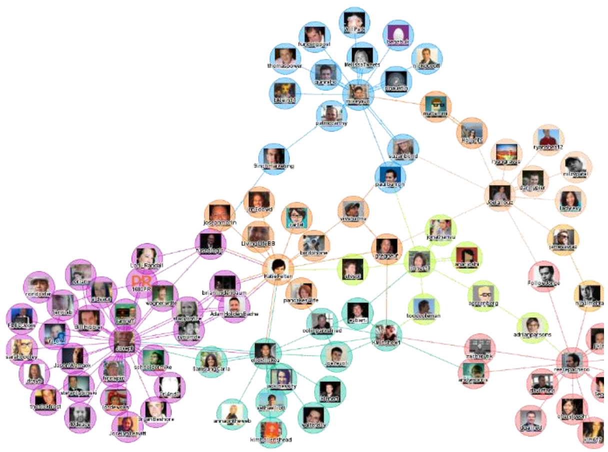
Graph based WSN models



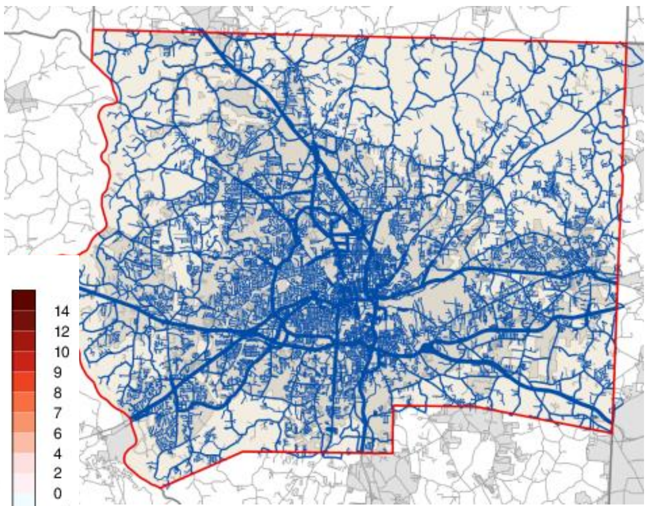


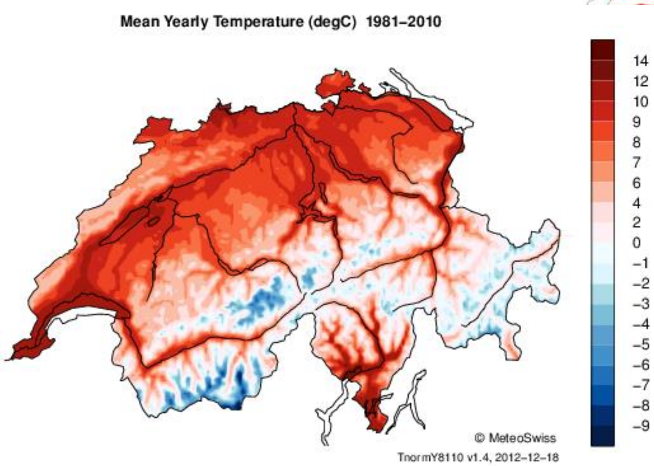
Types of signals on graphs

Social networks



Electrical networks





Environmental monitoring

Modeling signals on graphs

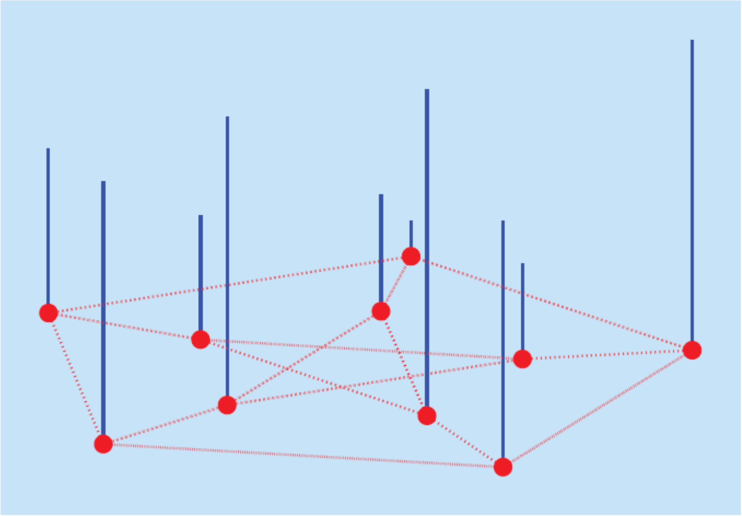
Edge weight <-> similarity between vertices.

Known

- Social media
- Sensor network

Unknown

- Neuroimaging



The height of each blue bar represents the signal value at the vertex.

Common data processing tasks:

- Filtering, denoising, inpainting, compression

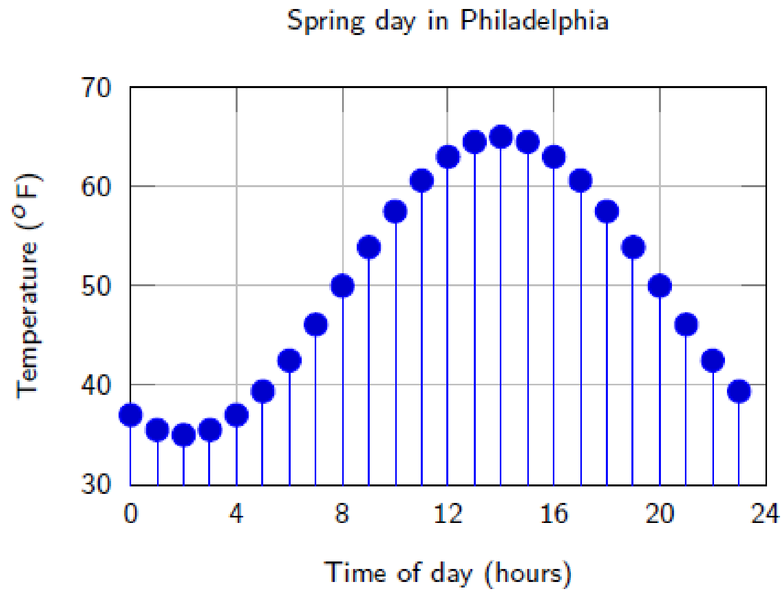
Challenges

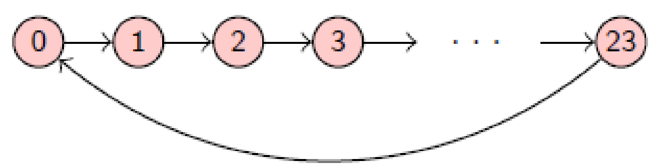
- What is translation, downsampling ?

Regular graph structures

1D Timeseries

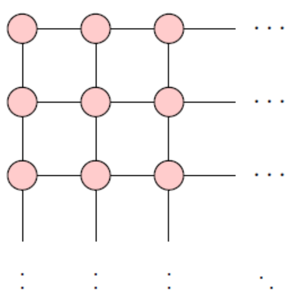
- Nodes <-> time instances
- Edges are unweighted and directed





2D images

- Nodes <-> pixel
- Edges <-> similarity



Signals on Graphs

Graphs: generic data representation forms encoding the geometric structures of data

Applications: social networks, energy distribution networks, transportation network, **wireless sensor network**, and neuronal networks.

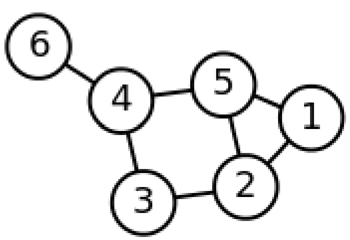
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$$

vertices → $\mathcal{V}$, edges → $\mathcal{E}$, weighted adjacency matrix → W

weights: distance /similarity/relationship

$$W_{i,j} = \begin{cases} \exp\left(-\frac{[\mathrm{dist}(i,j)]^2}{2\theta^2}\right) & \text{if } \mathrm{dist}(i,j) \le \kappa \\ 0 & \text{otherwise,} \end{cases}$$

Undirected graph	Degree matrix: **D**	Adjacency matrix: **A**	Laplacian matrix: **L**



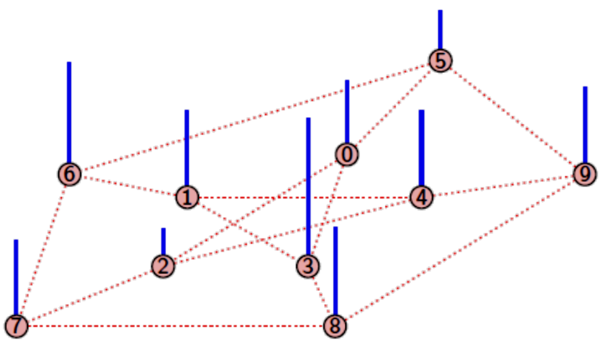
$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

Signals on Graphs

Graph signal f in R^N, where $|V|=N$



$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_9 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.7 \\ 0.3 \\ \vdots \\ 0.7 \end{bmatrix}$$

Graph Laplacian $\mathcal{L} := \mathrm{D} - \mathrm{W}$, D: diagonal with sums of weights

W: weight matrix

Normalized Graph Laplacian $\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$

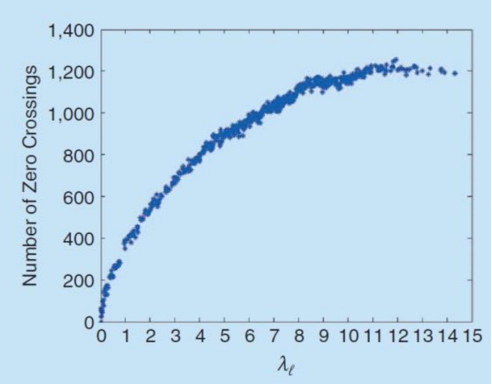
Graph Laplacian

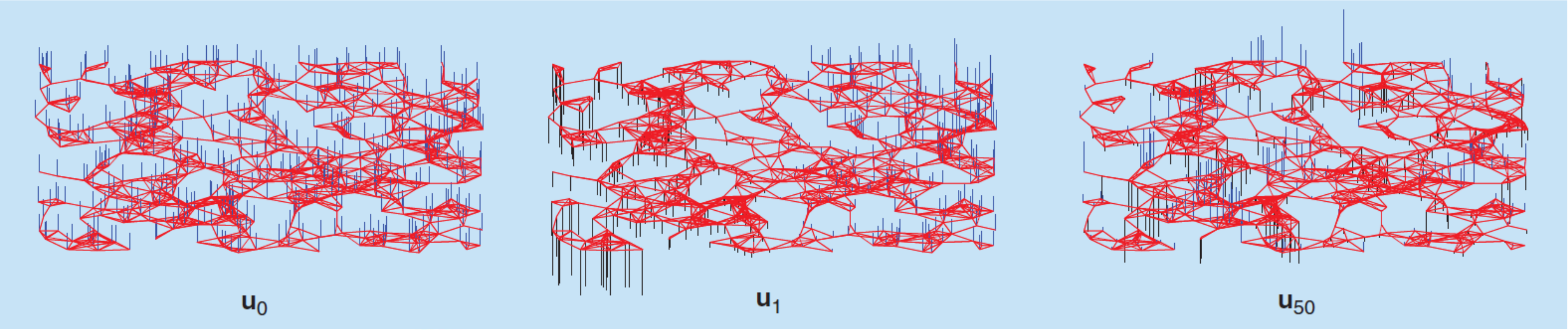
Spectral properties $\mathcal{L}\mathbf{u}_\ell = \lambda_\ell \mathbf{u}_\ell$

- Laplacian is Positive Semi-definite matrix
- Eigenvalues: $0=\lambda_1(L) \leq \lambda_2(L) \leq \ldots \leq \lambda_{N-1}(L)$

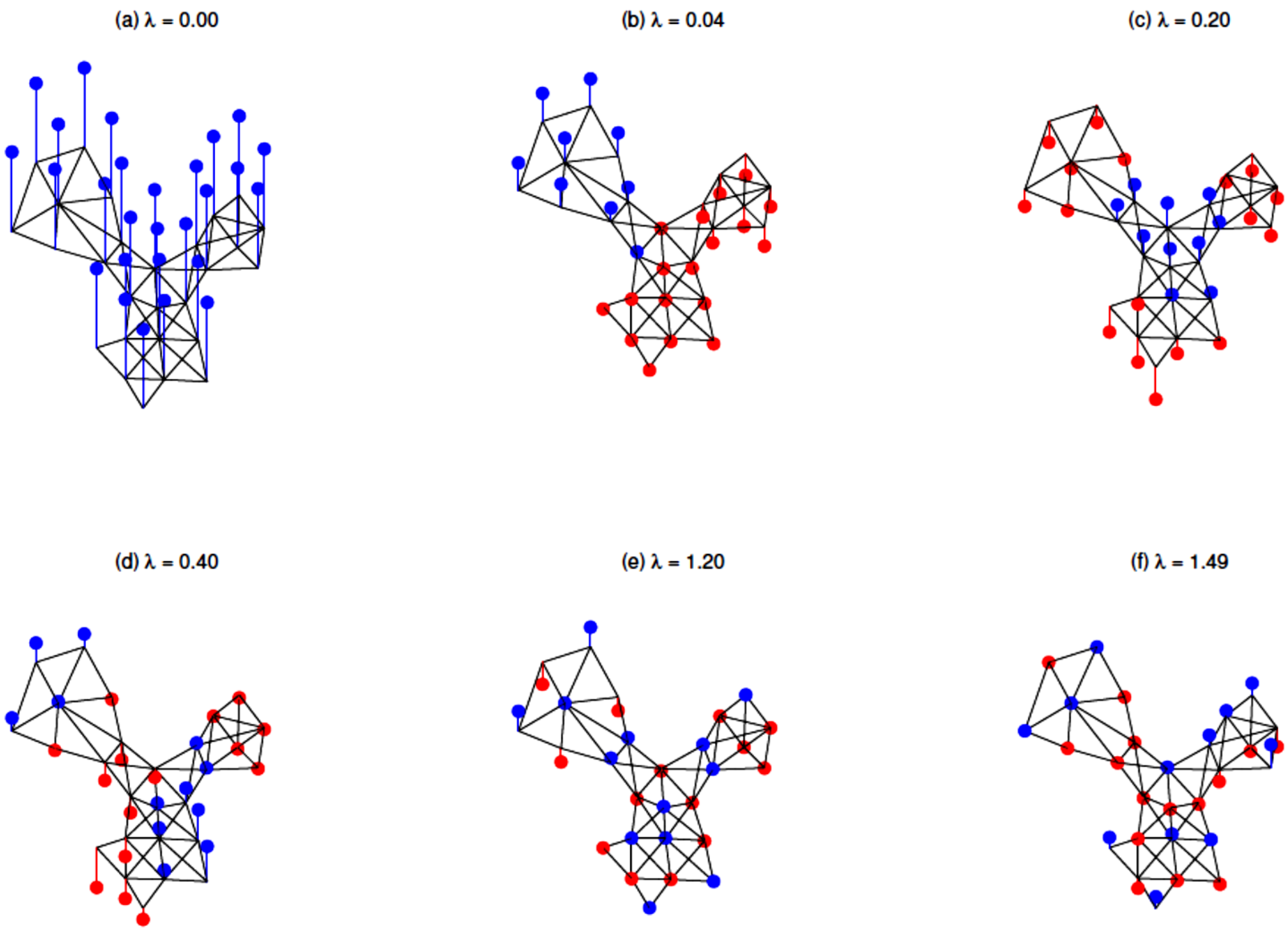
$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} w_{ij}(x_i - x_j)^2 \geq 0, \text{ for all } \mathbf{x}$$

All eigenvalues are nonnegative, i.e. $\lambda_i \geq 0$ for all i

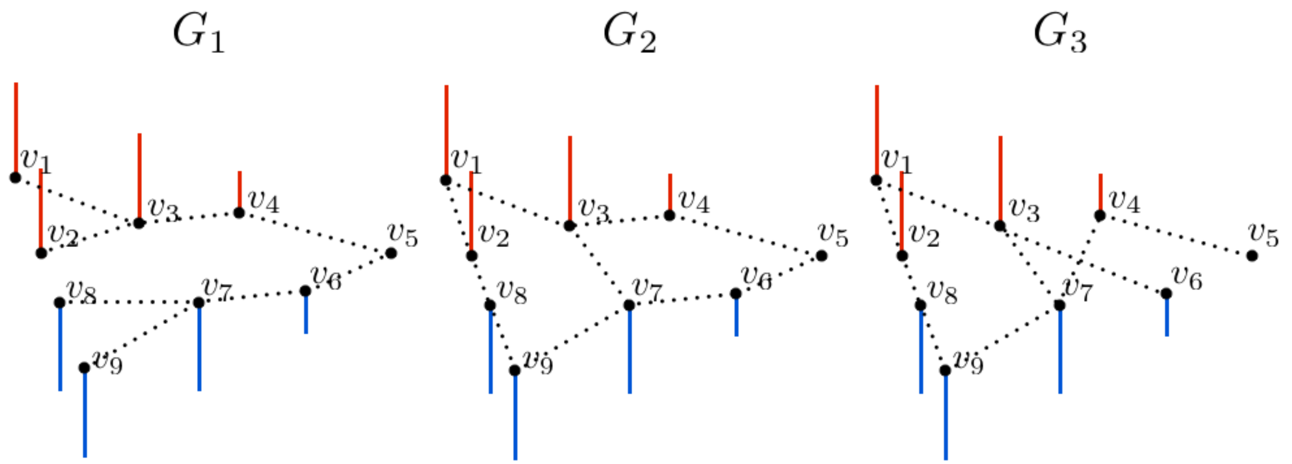




Eigenvectors of Graph Laplacian



Laplacian Regularization



- One signal <-> many different graphs
- Only 1 leads to a smooth graph signal.

➢ Only G1 favors smoothness of the resulting graph signal.

Graph Fourier Transform

Graph based approximation $\hat{f}(\lambda_\ell) := \langle \mathrm{f}, \mathrm{u}_\ell \rangle = \sum_{i=1}^{N} f(i) u_\ell^*(i).$

Smoothness w.r.t. graph $\| \mathrm{f} \|_{\mathcal{L}} := \| \mathcal{L}^{\frac{1}{2}} \mathrm{f} \|_2 = \sqrt{\mathrm{f}^{\mathrm{T}} \mathcal{L} \mathrm{f}} = \sqrt{S_2(\mathrm{f})}.$

Graph spectral filtering (regularization)

$$\min_{\mathrm{f}} \{ \| \mathrm{f} - \mathrm{y} \|_2^2 + \gamma S_p(\mathrm{f}) \},$$

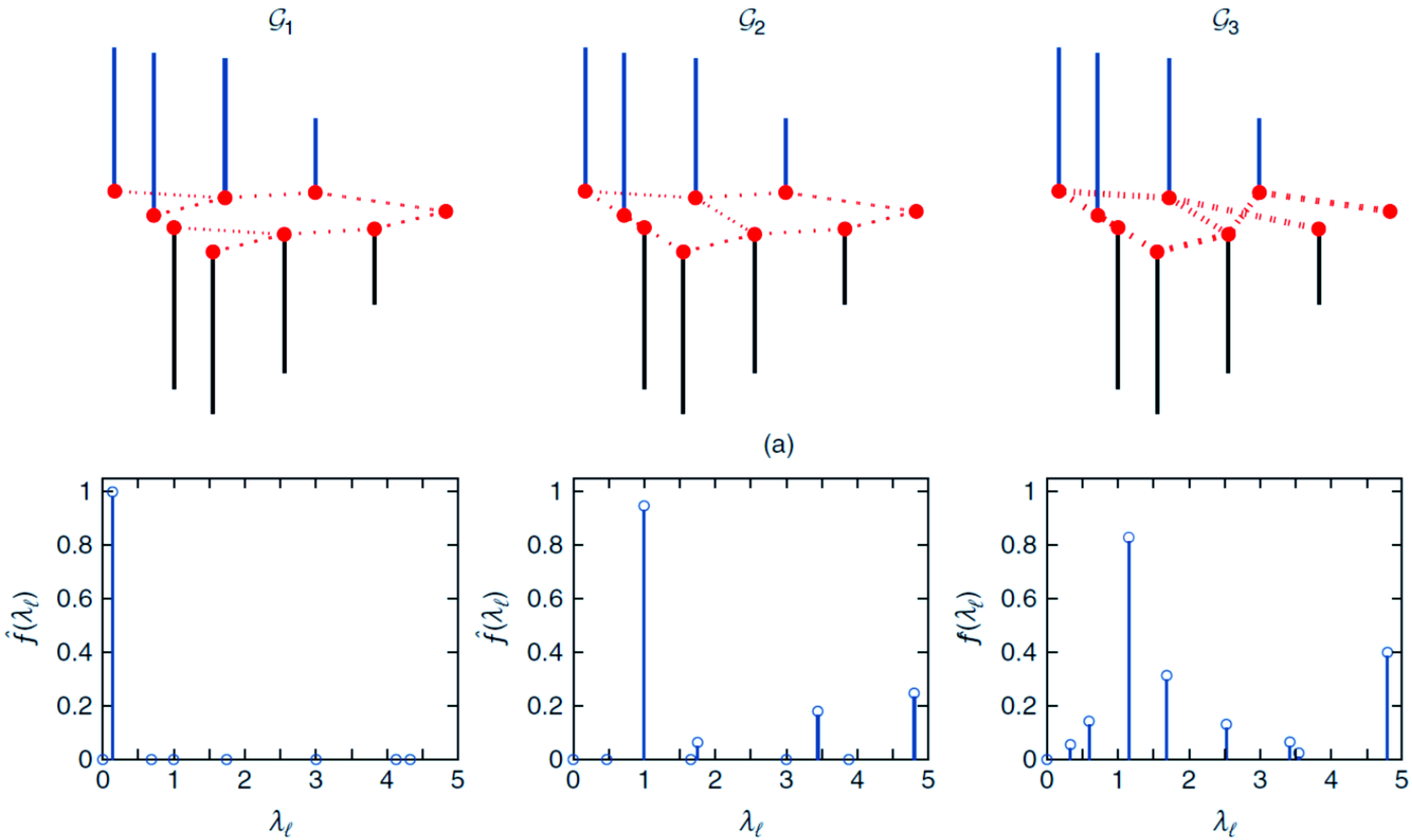
$$\downarrow$$

$$\operatorname{argmin}_{\mathbf{f}} \{ \| \mathbf{f} - \mathbf{y} \|_2^2 + \gamma \mathbf{f}^{\mathrm{T}} \mathcal{L} \mathbf{f} \}.$$

Connectivity of the graph -> encoded in graph Laplacian

Define both a graph Fourier transform (graph Laplacian eigenvectors)

Different notions of smoothness



$$\mathbf{f}^{\mathrm{T}}\mathcal{L}_1\mathbf{f} = 0.14,$$

$$\mathbf{f}^{\mathrm{T}}\mathcal{L}_2\mathbf{f} = 1.31,$$

$$\mathbf{f}^{\mathrm{T}}\mathcal{L}_3\mathbf{f} = 1.81$$

Graph Fourier Transform

A graph filter is a system $\tilde{\mathbf{s}} = \mathbf{H}(\mathbf{s})$

Equivalent $\widetilde{\mathbf{s}} = \mathbf{H}(\mathbf{s}) = h(\mathbf{A})\mathbf{s}.$

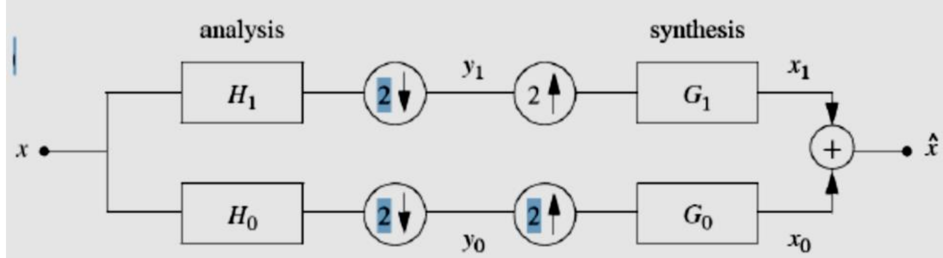
Where $h(\mathbf{A}) = h_0 \mathbf{I} + h_1 \mathbf{A} + \ldots + h_L \mathbf{A}^L.$

Jordan decomposition $\mathbf{A} = \mathbf{V}\,\mathbf{J}\,\mathbf{V}^{-1}$

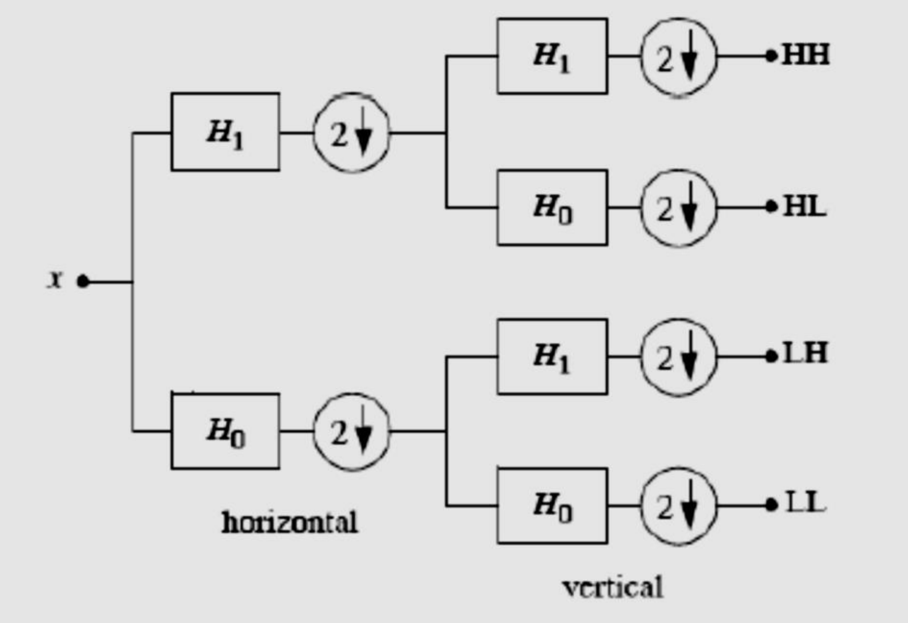
Graph Fourier Transform $\widehat{\mathbf{s}} = \mathbf{F}\,\mathbf{s} = \mathbf{V}^{-1}\mathbf{s}$

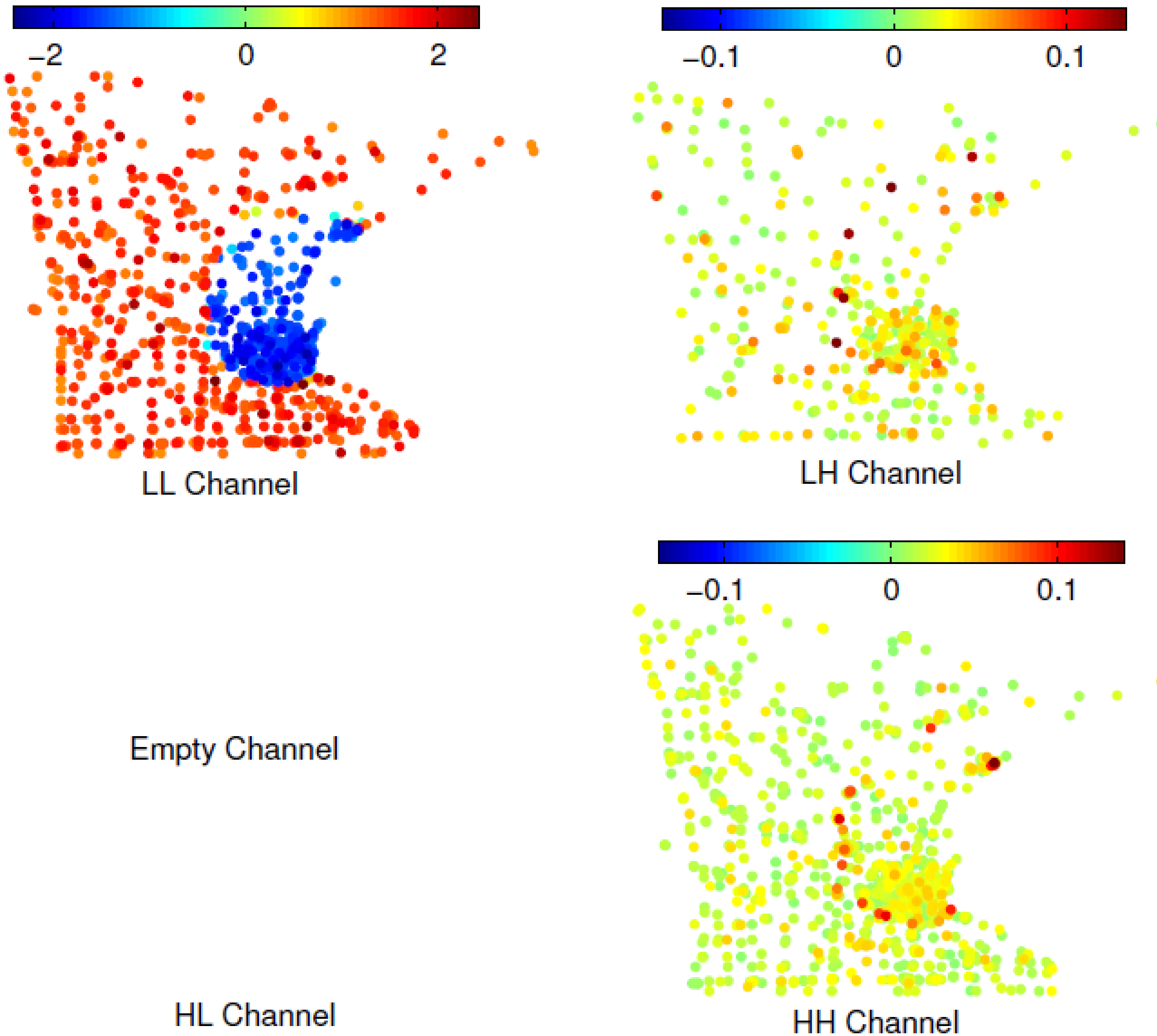
Filters on graphs

Wavelet filterbank



(a) 2 Channel Filterbank





−2
0
2
LL Channel
−0.1
0
0.1
LH Channel
Empty Channel
HL Channel
−0.1
0
0.1
HH Channel

Spatial Signal Graphs

1-hop averaging transform

$$y[n] = \frac{1}{d_n} \sum_{m=1}^{N} A[n, m]x[m]$$

1-hop difference transform

$$y[n] = \frac{1}{d_n} \sum_{m=1}^{N} A[n, m](x[n] - x[m])$$

$$\mathbf{y} = \mathbf{D}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}_{rw}\mathbf{x}$$

$$\mathbf{y} = \mathcal{L}_{\mathbf{rw}}\mathbf{x} = \mathbf{x} - \mathbf{P}_{\mathbf{rw}}\mathbf{x}$$

Spectral anomaly detection in WSN

Decomposition of Laplacian $\mathcal{L}_G = \mathbf{U}_G \mathbf{\Lambda}_G \mathbf{U}_G^t$

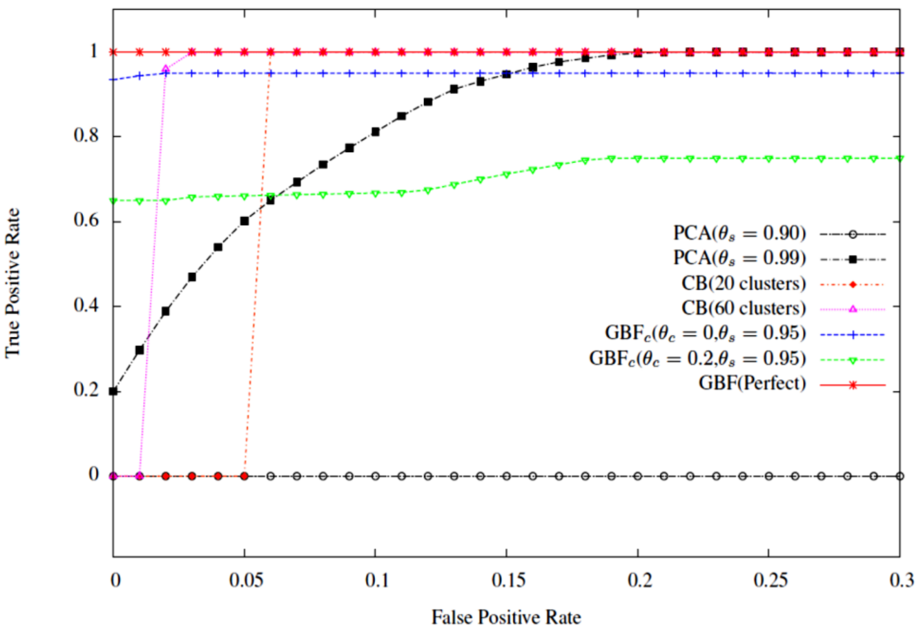
Alternative approach $h(\mathcal{L}_G) = \mathbf{U}_G (h(\mathbf{\Lambda}_G)) \mathbf{U}_G^t$

Graph construction $[w_{i,j}]_b = \exp\left(-\frac{(1 - \|\rho(i,j)\|_1)^2}{\Delta_c^2}\right) \cdot \exp\left(-\frac{\tilde{D}(i,j)^2}{\Delta_d^2}\right)$

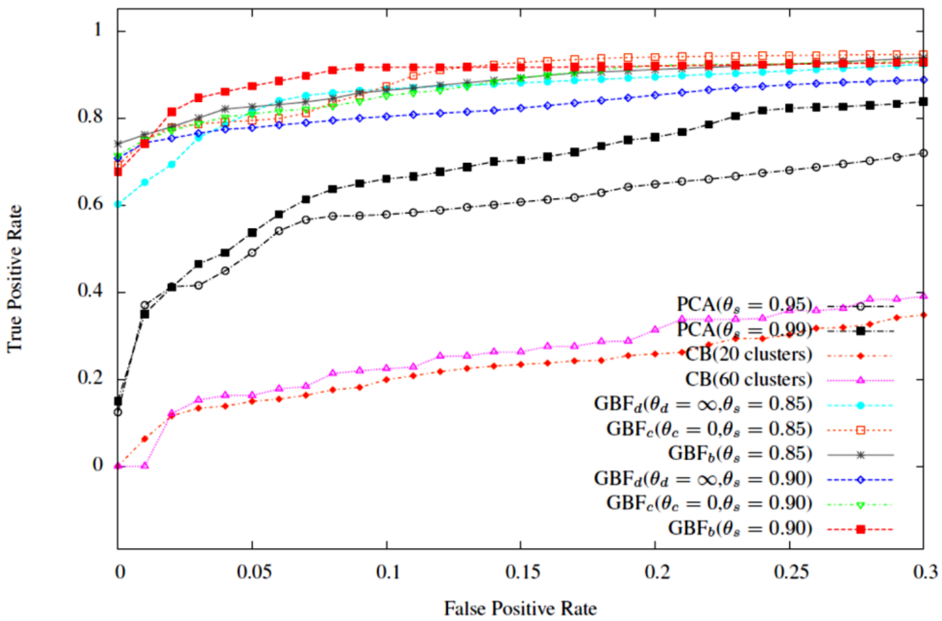
Data fit in graph $\sigma_l^2 = s^2[\mathbf{u}_l^t \mathbf{X}]$ where $s^2[\mathbf{p}^t]$ is the sample variance

Target ratio $\arg\max_c \sum_{l=1}^{c} \frac{\sigma_l^2}{\sigma_T^2}$ subject to $\sum_{l=1}^{c} \frac{\sigma_l^2}{\sigma_T^2} \leq \theta_s$

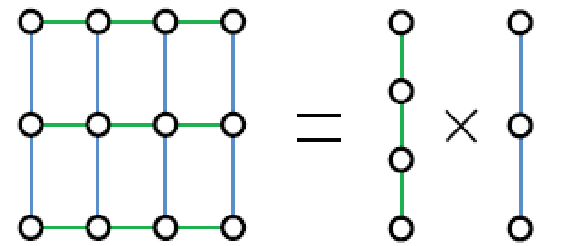
Global



Distributed



Product Graphs

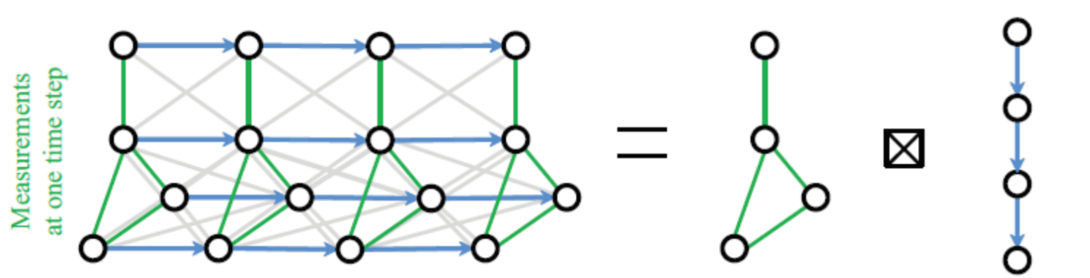


- Assume: $G_1=(V_1,A_1)$ and $G_2(V_2,A_2)$
- Product graph: $G = G_1 \diamond G_2 = (\mathcal{V}, \mathbf{A}_\diamond),$
- Kronecker:

$$\mathbf{A}_\otimes = \mathbf{A}_1 \otimes \mathbf{A}_2.$$

- Cartesian:

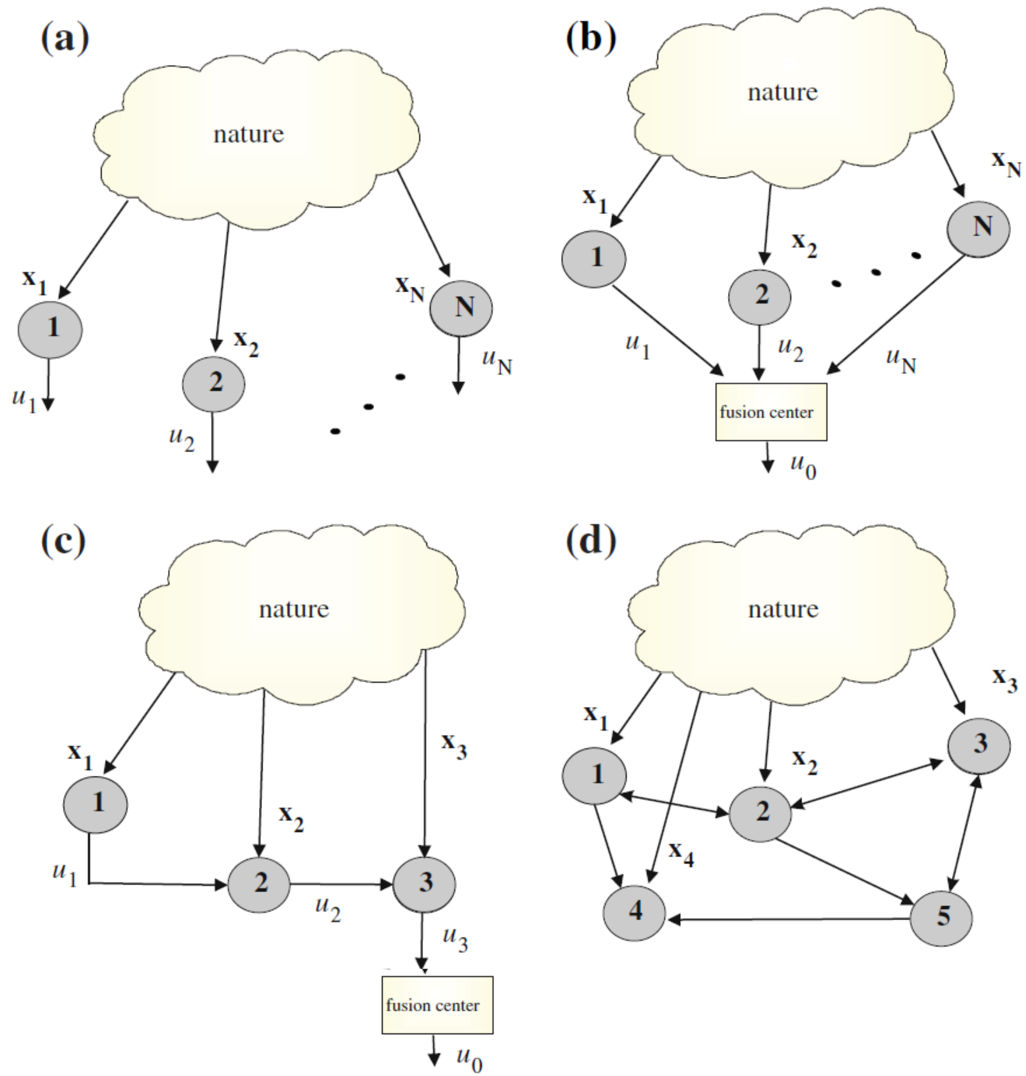
$$\mathbf{A}_\times = \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2.$$



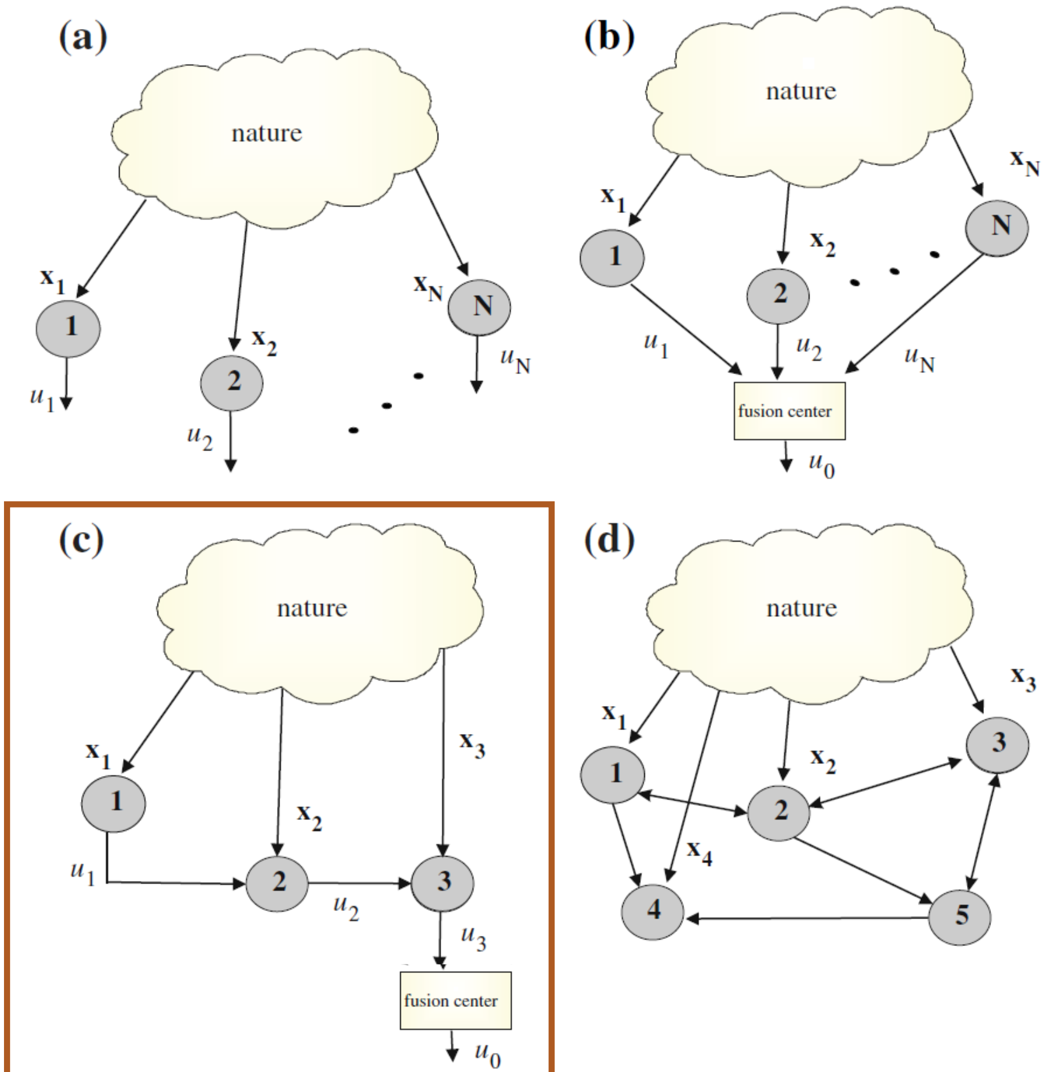
- Strong:

$$\mathbf{A}_\boxtimes = \mathbf{A}_1 \otimes \mathbf{A}_2 + \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2.$$

Communication architectures



Communication architectures

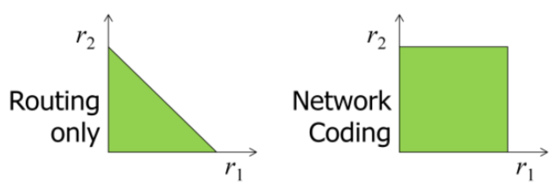


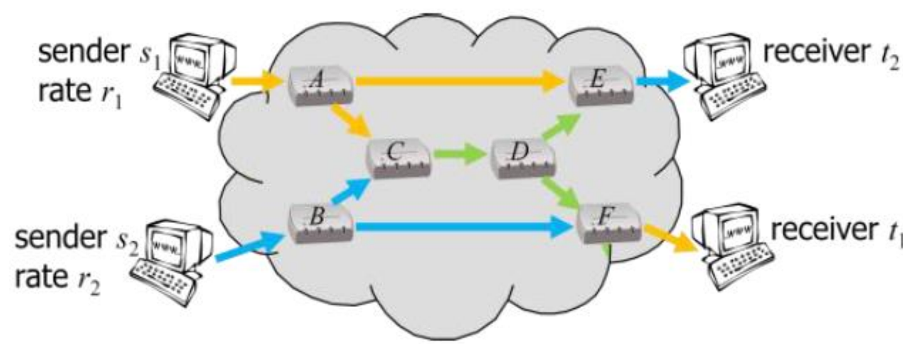
Network Coding (NC)

Typical routing: Each message on an output link must is a copy of a message that arrived earlier on an input link

Network coding: each message sent on a node’s output link can be some function or “mixture” of messages that arrived earlier on the node’s input links

Motivation: **improve throughput**





Minimize

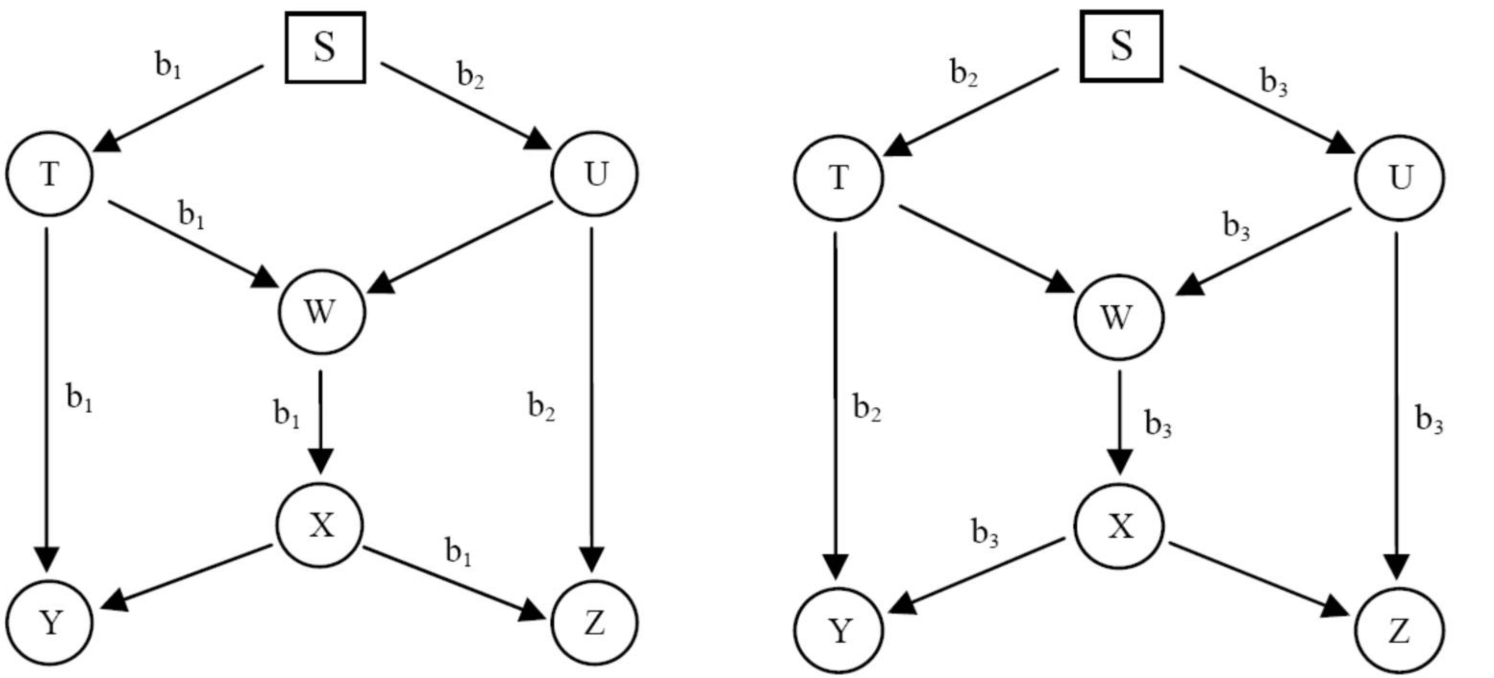
- Energy per bit
- Delay

R. Ahlswede, N. Cai, S.-Y. R. Li, and R.W. Yeung, “Network information flow,” *IEEE Trans. on Information Theory*, vol. 46, no. 4, July 2000

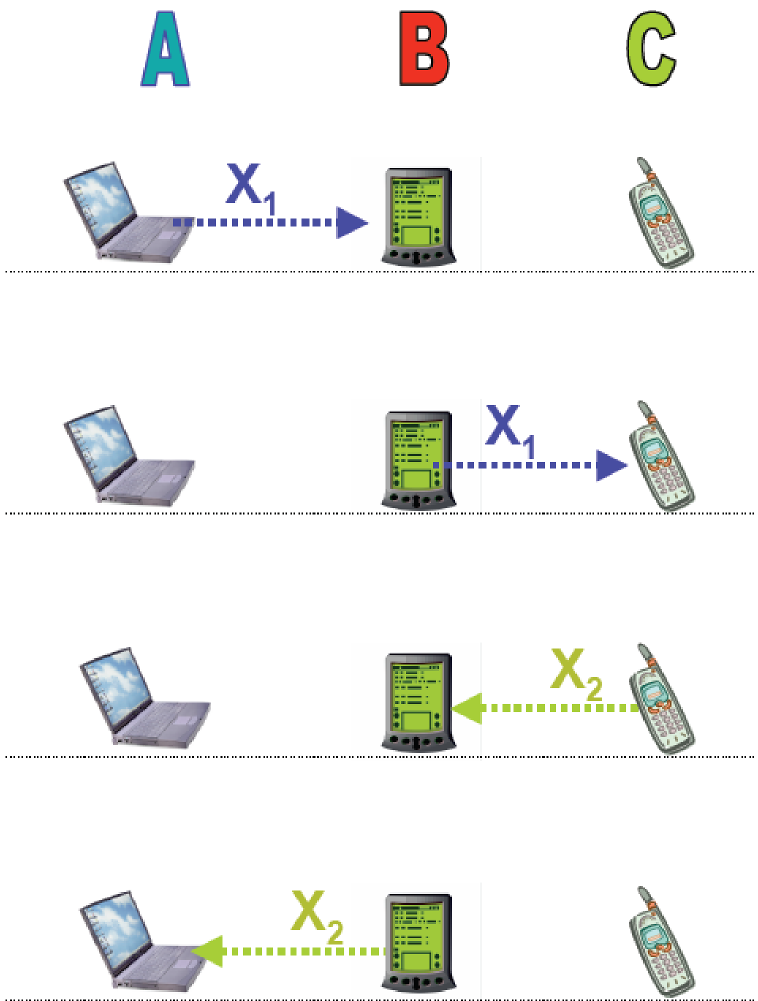
Typical Unicast

Without network coding

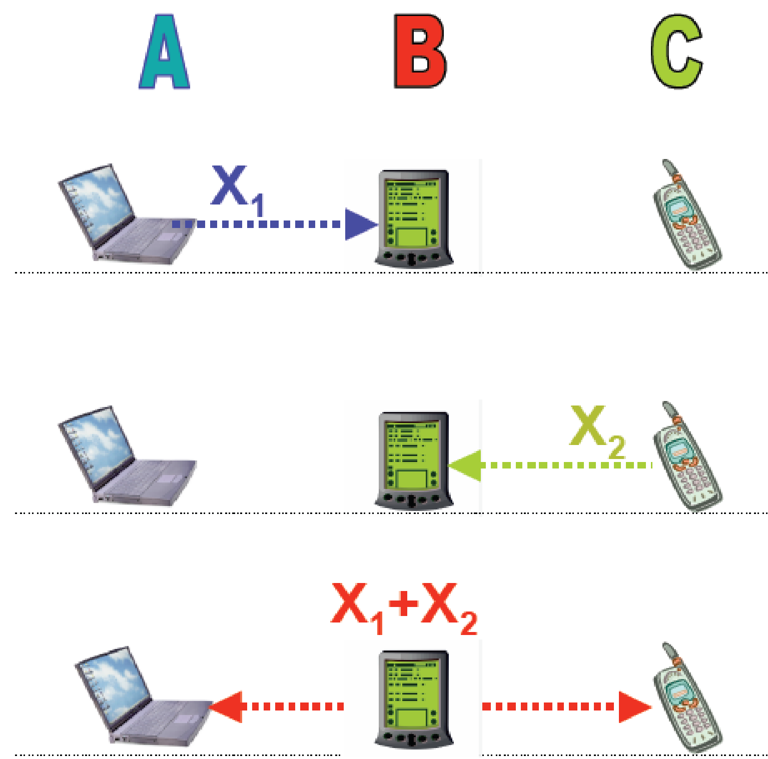
- Simple store and forward
- Multicast rate of 1.5 bits per time unit



Traditional Method



Network Coding



Unicast with NC

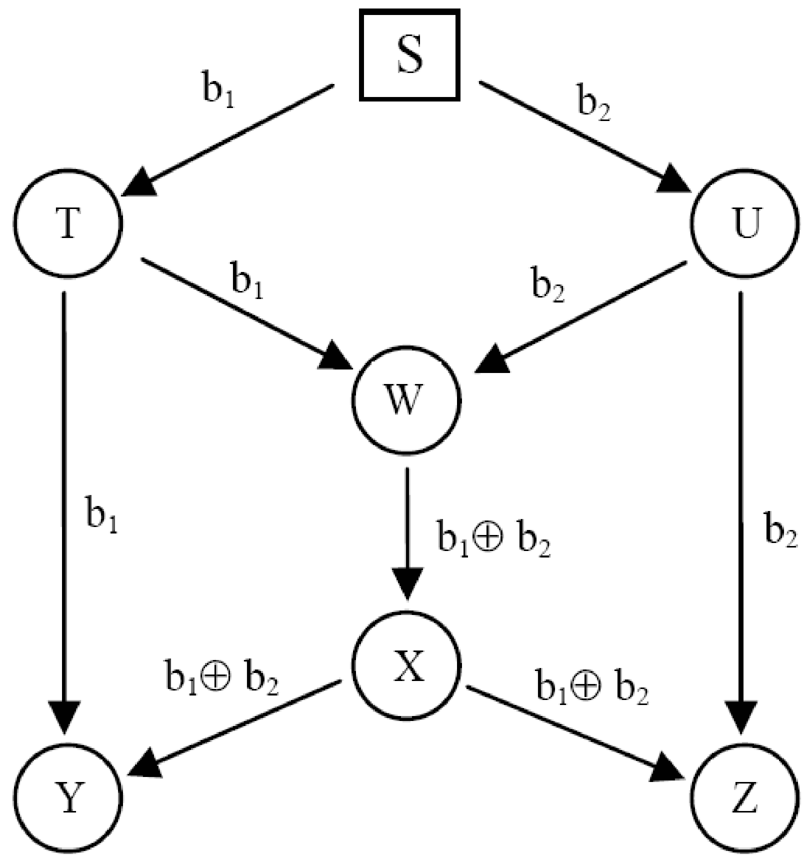
With network coding

X-OR → one of the simplest form of coding

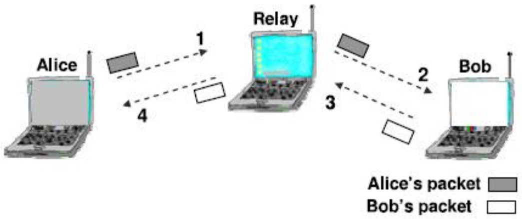
Multicast rate of 2 bits per time unit

Disadvantages:

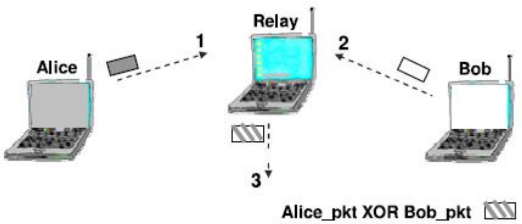
- Coding/decoding scheme has to be agreed upon beforehand



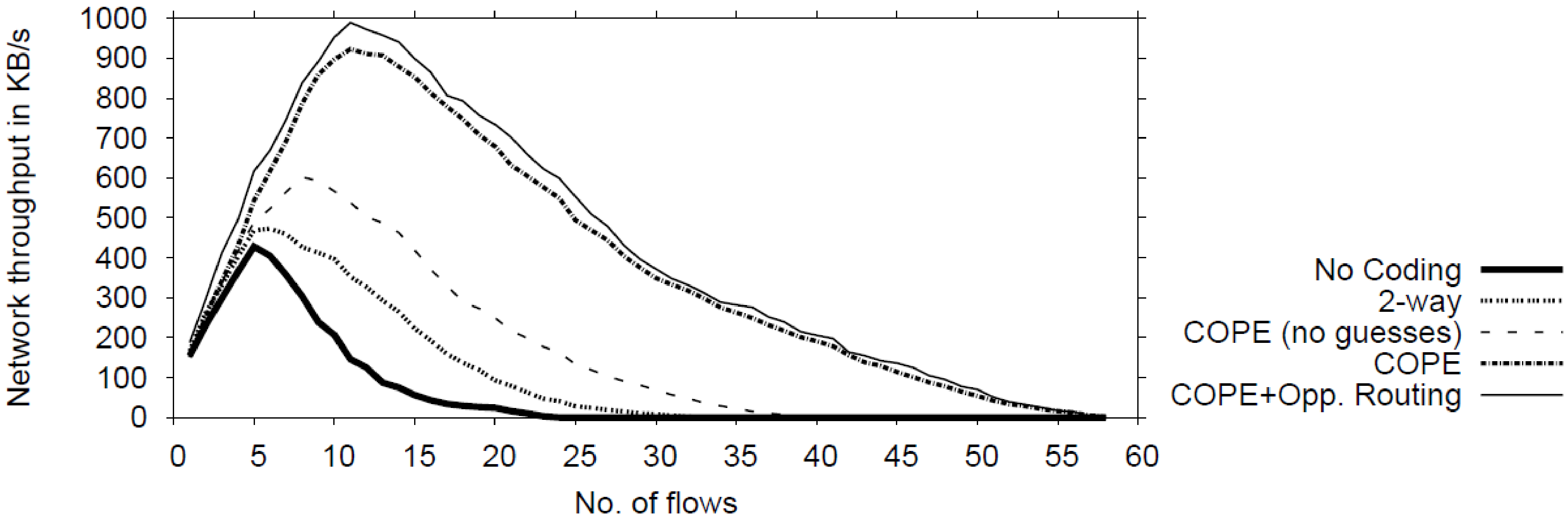
NC performance



(a) No Coding

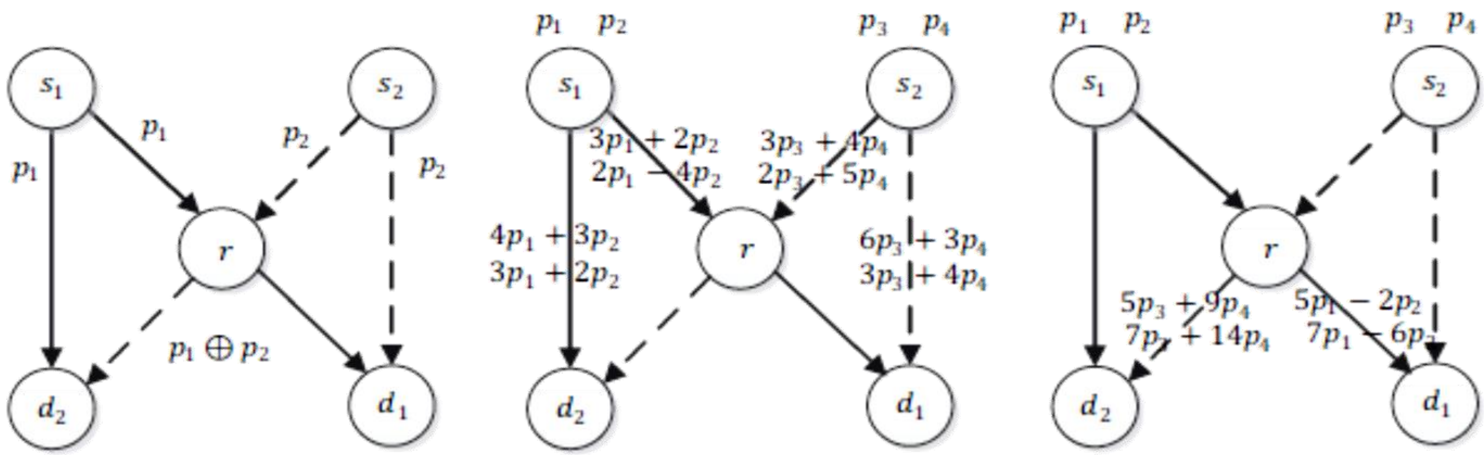


(b) Coding



Generalize to packets

- Operate on packets instead of on bit-streams



(a) (b) (c)

NC encoding & decoding

Message

$$y(e) = \sum_{e':out(e')=v} m_e(e')y(e')$$

$$y(e) = \sum_{i=1}^{h} g_i(e)x_i$$

Encoding vector

$$\overrightarrow{m(e)} = [m_e(e')]_{e':out(e')=v}$$

$$\overrightarrow{g(e)} = [g_1(e), \ldots, g_h(e)]$$

Decoding

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} g_1(e_1) & \cdots & g_h(e_h) \\ \vdots & \ddots & \vdots \\ g_1(e_h) & \cdots & g_h(e_h) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G_t \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix}$$

NC encoding & decoding

Node t can recover the source symbols $x_1, . . . , x_h$ as long as the matrix G_t, formed by the global encoding vectors, has (full) rank.

$$\begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G_t^{-1} \begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix}$$

G_t will be invertible w.h.p. if local encoding vectors are random and the field size is sufficiently large

R. Koetter,M.Medard, "An algebraic approach to network coding", *IEEE/ACM Trans. on Networking*, 2003

Practical NC: Random NC

Issues

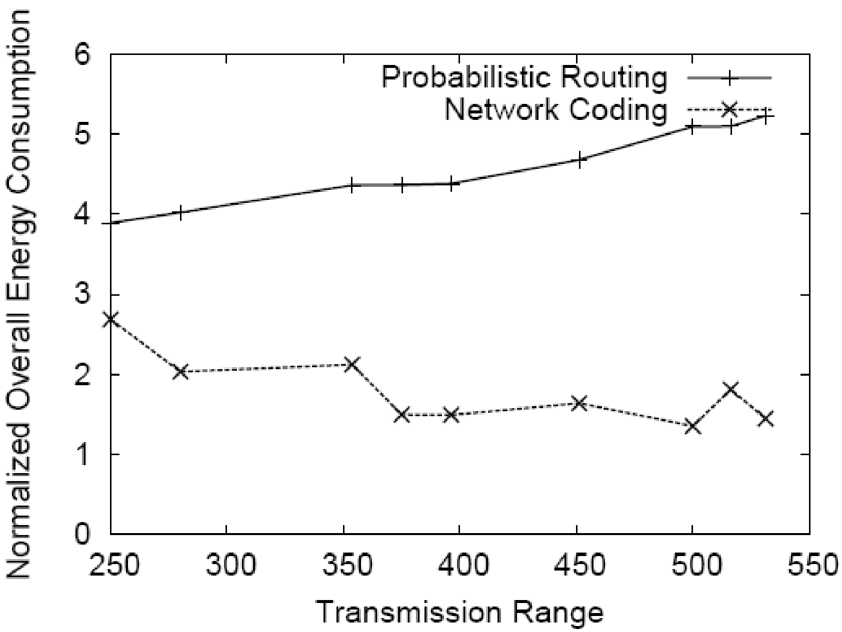
- Synchronous / asynchronous packet delivery
- Varying capacity edges
- Packet delays and drops
- Central coding pattern knowledge

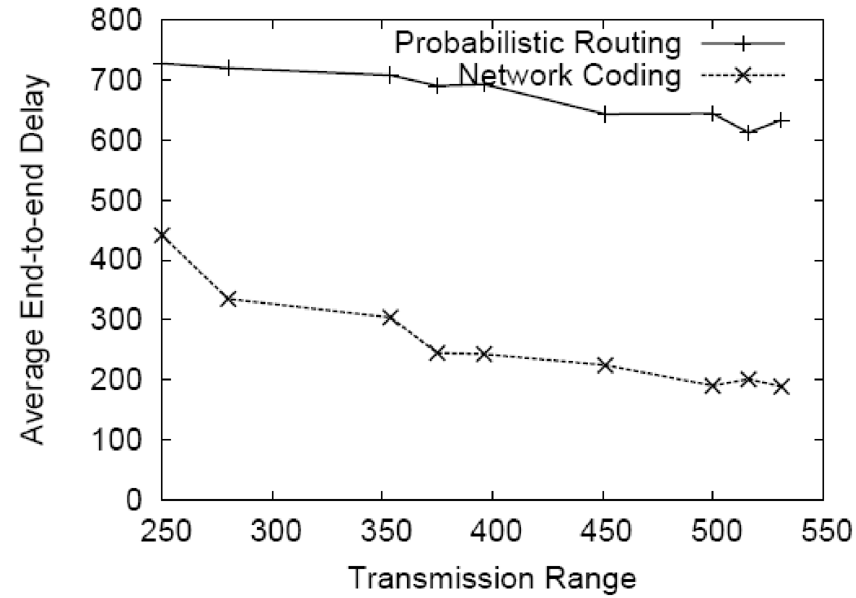
Random NC: random linear coefficients in a finite field and send the encoding vector within the same packet

Packetization: Header removes need for centralized knowledge of graph topology and encoding/decoding functions

Buffering: Nodes stores within their buffers the received packets. Allows asynchronous packets arrivals & departures with arbitrarily varying rates, delay, loss

Random NC simulation results

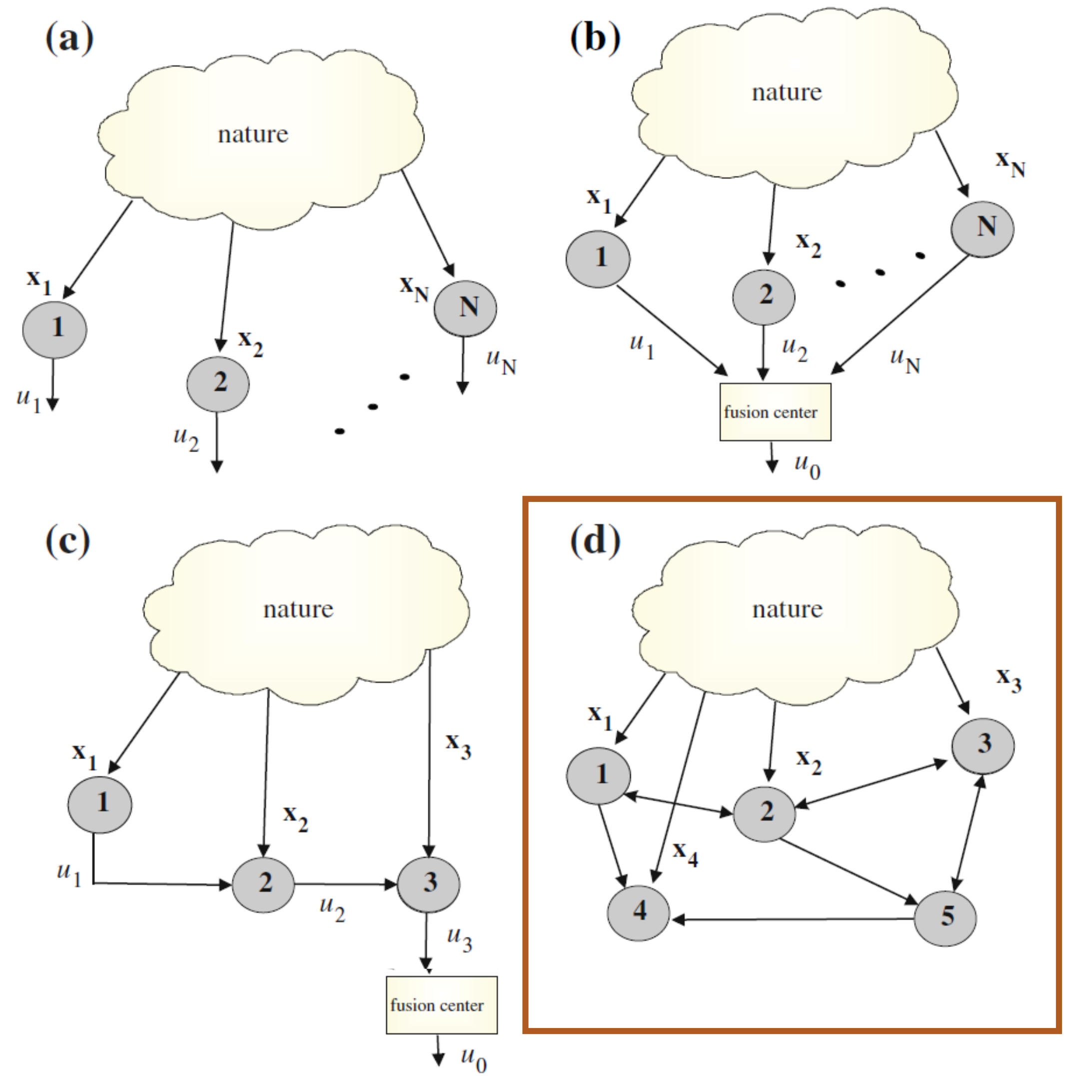




Energy consumption: number of transmissions and receptions needed to gather all the required packets

Delay: number of time units needed to decode all the required packets

Communication architectures



In-network processing

Collaborative signal processing:

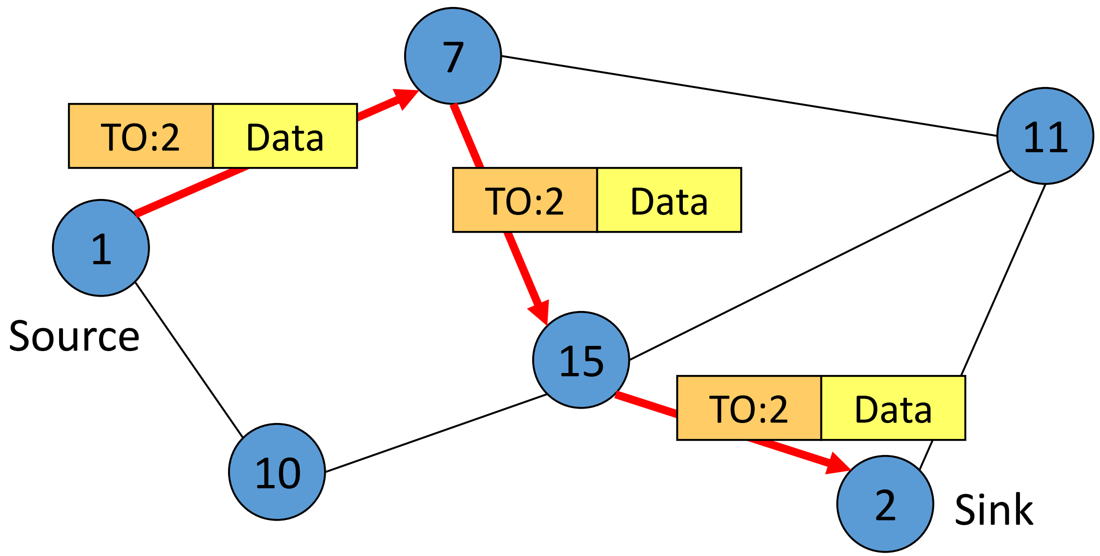
➢ Exploit local computational resources -> reduce data transmissions

Power(Communications) > Power(Processing)

- Applications
  - Detection, Classification, Parameter Estimation, Tracking…
- Assumptions
  - Specialized routing protocols
  - spatio-temporal smoothness

Address-based routing vs. data-centric forwarding

- Address-based routing
  - Directed towards a well-specified ***particular destination*** (sink)
  - Support for unicast, multicast, and broadcast messages



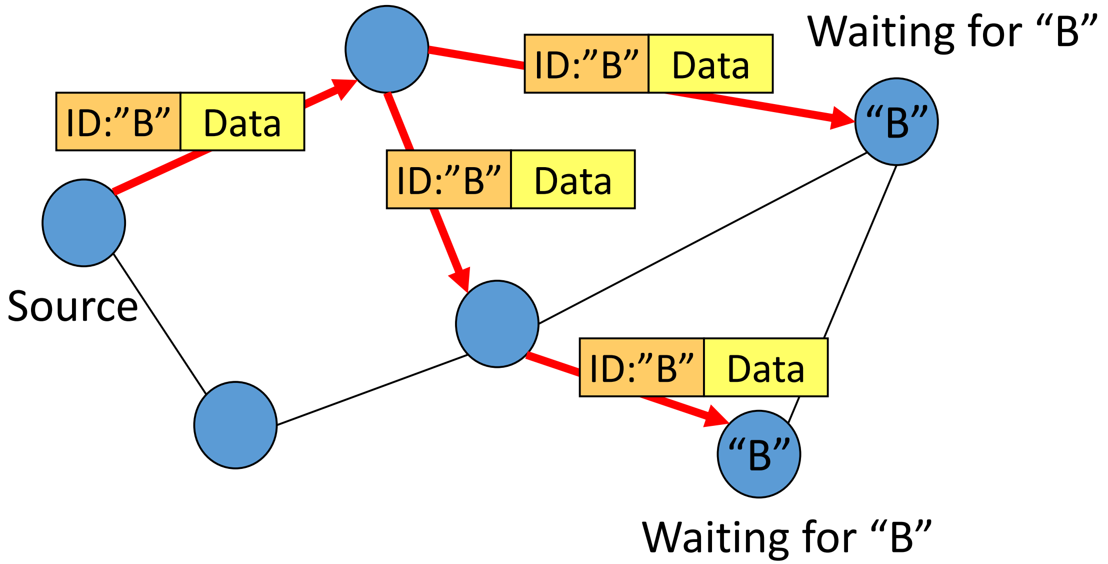
Data-Centric Networking

The traditional communication paradigm focuses on the relationship between communicating peers

In WSNs, the application is not interested in the *identity* of the nodes, but rather in the information about the *physical environment*

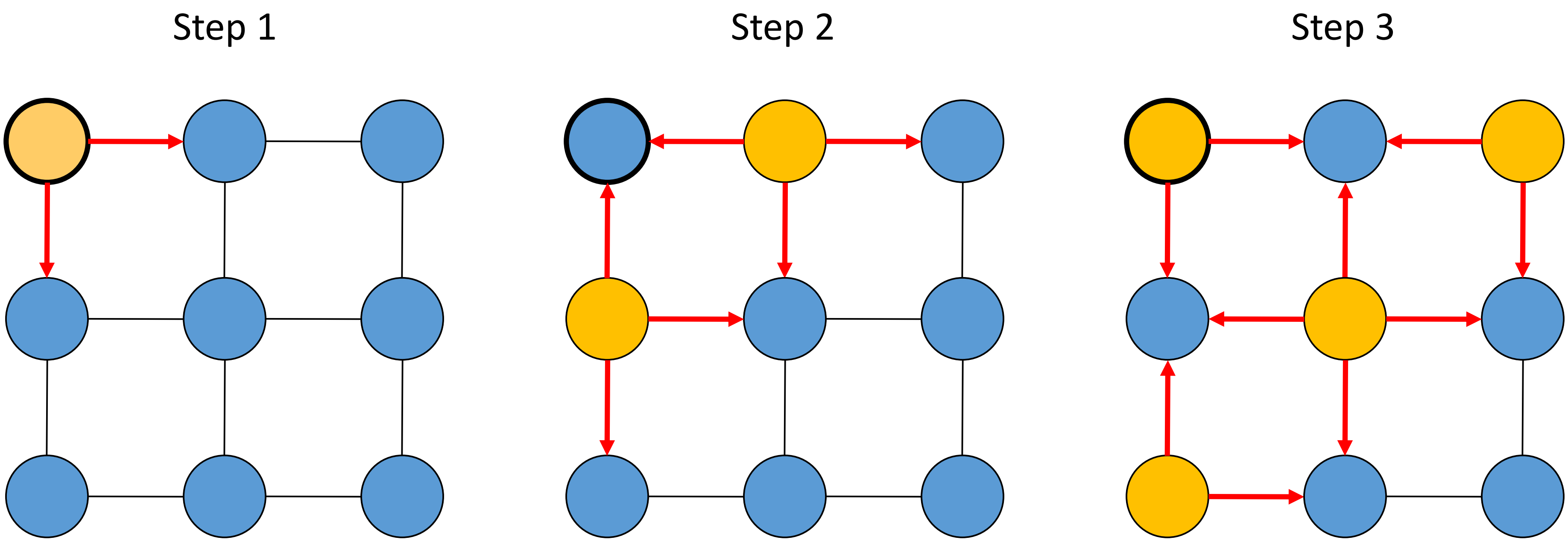
Objectives

- In-network aggregation
- Data-centric addressing
- Decoupling in time
- Fault-tolerance
- Scalability



Flooding

- Basic mechanism:
  - Each node that receives a packet re-broadcasts it to all neighbors
  - The data packet is discarded when the maximum hop count is reached



Flooding

Simplest method for message delivery from observation node to sink

- NO routing table NOR next hop estimation
- On receiving the packet, a sensor just rebroadcasts it

(+) Low computing complexity

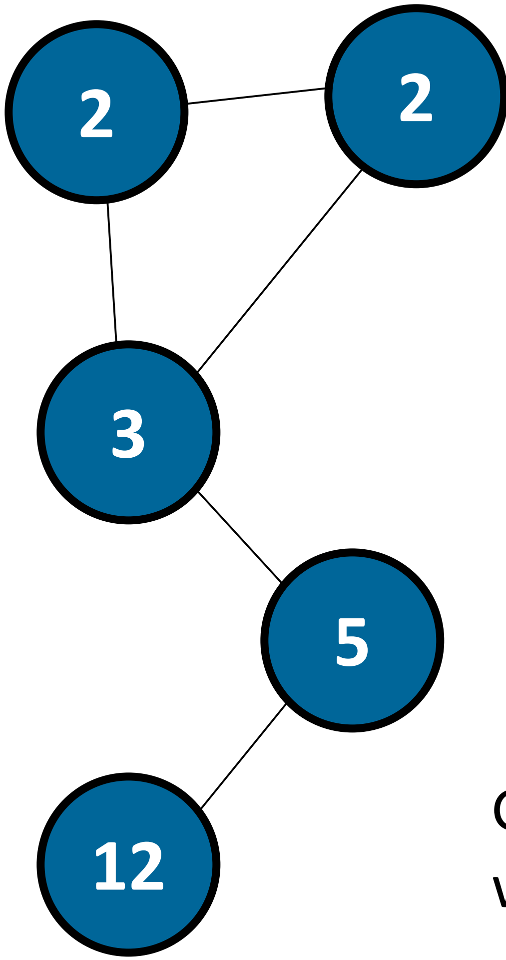
(+) No memory for path caching

(-) Implosion: duplicated messages are received

(-) *Overlay: flooding of* overlapping data

(-) Resource blindness

Distributed Aggregation



- Every node has a measurement (e.g sensing temperature)
- Every node wants to access the global average
- Want a truly distributed, localized and robust algorithm to compute the average.

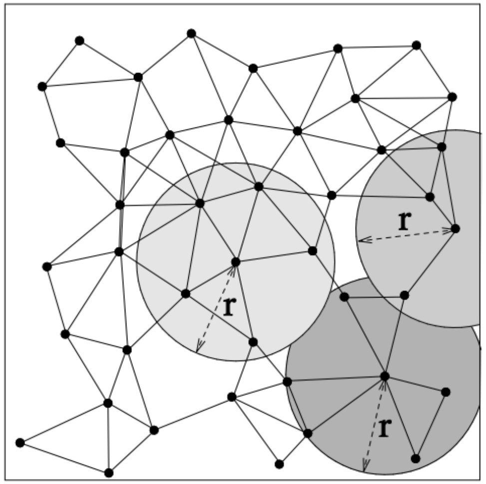
Goal: every node gets (2+2+3+5+12)/5=4.8 with the minimum energy cost

Consensus algorithms

Having a set of agents to agree upon a certain value (usually global function) using only local information exchange (local interaction)

<u>Objectives</u>:

- ➢Distributed computation of general functions
- ➢Computational efficient
- ➢Robust to failures
- ➢Independent of topology



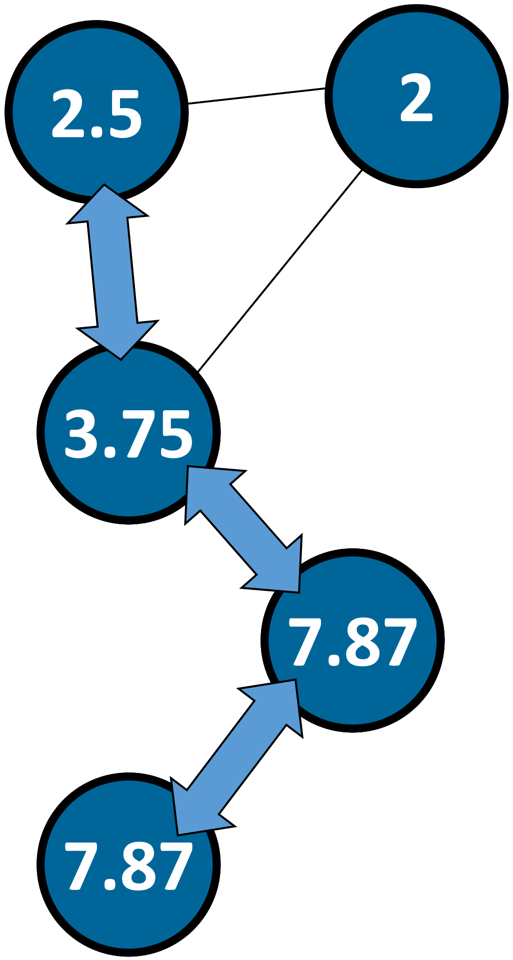
Distributed Average Consensus

Nodes measure → average $x_i \rightarrow x_{ave} = \Sigma x_i/n$

Assumptions

- Nodes their neighbors (location)
- Dynamic network topology

Gossip Algorithms for Aggregations



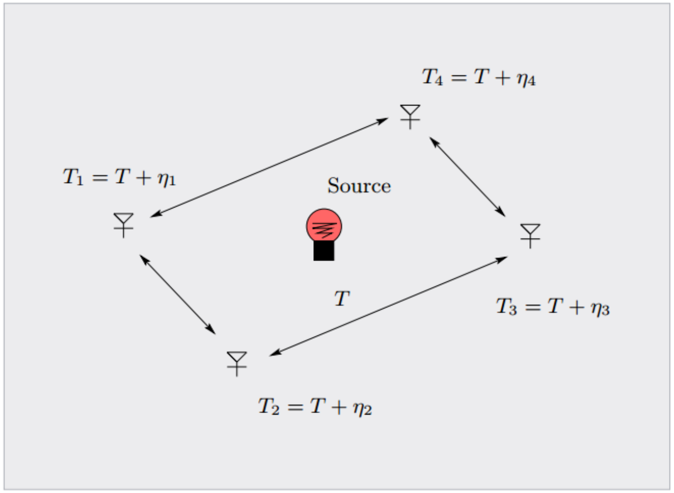
- Start with initial measurement as an estimate for the average and update
- Each node interacts with a random neighbor and both compute **pairwise** average (one update)
- Converges to true average
- Useful building block for more complex problems

Gossip Algorithms

One solution to distributed consensus

- Each node (*n* nodes in total) holds an estimate
- Goal: for every node estimate

  average of all *n* initial values



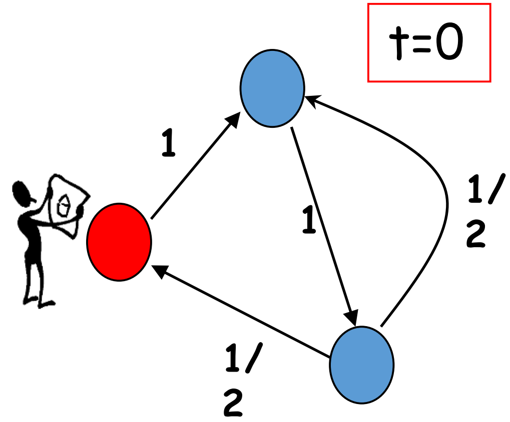
Iterative + random

- At each iteration:
  - ➢ random groups communicate & average
- Local estimation → global consensus
- Q: Time? Packets? Quality? Synchronization?

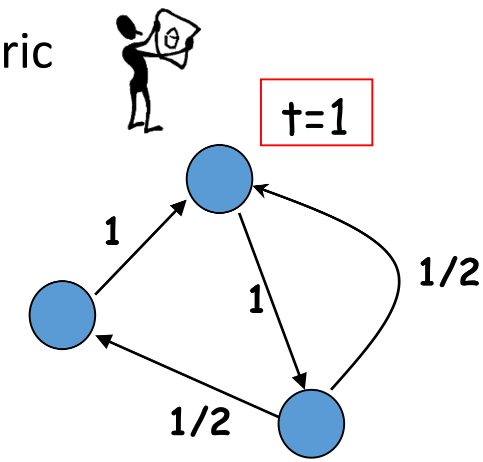
What is a Random Walk

- Given a graph and a starting point (node), we select a neighbor of it at random, and move to this neighbor;
- Then we select a neighbor of this node and move to it, and so on;
- The (random) sequence of nodes selected this way is a random walk on the graph

What is a random walk



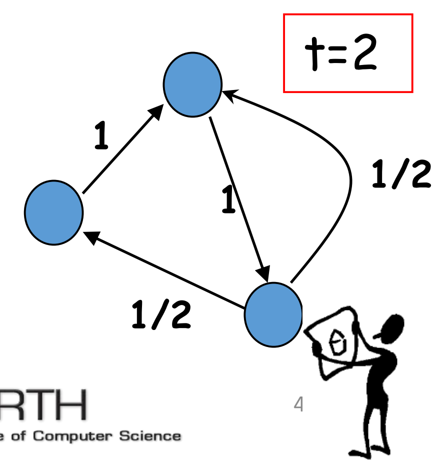
- nxn Adjacency matrix A.
  - A(i,j) = weight on edge from *i* to *j*
  - If the graph is undirected A(i,j)=A(j,i), i.e. A is symmetric



- nxn Transition matrix P.
  - P is row stochastic (doubly for undirected)
  - P(i,j) = probability of stepping on node j from node i

    $= A(i,j)/\sum_i A(i,j)$

- nxn Laplacian Matrix L.
  - $L(i,j)=\sum_i A(i,j)-A(i,j)$
  - Symmetric positive semi-definite for undirected graphs
  - Singular



Probability Distributions

- $x_t(i)$ = prob. that the surfer is at node *i* at time *t*
- $x_{t+1}(i) = \sum_j(\text{Prob. of being at node j}) * Pr(j\text{->}i) = \sum_j x_t(j) * P(j,i)$
- $x_{t+1} = x_t P = x_{t-1} * P * P = x_{t-2} * P * P * P = \ldots = x_0 P^t$

When one keeps walking for a long time?

- For the stationary distribution v_0 we have $v_0 = v_0 * \pi$

For connected, non-bipartite graphs

$\pi(v)$ = node degree/2*#edges = $d(v)/2m$

The more neighbors you have, the more chance you'll be reached

Cover time in Graphs

Given a graph G, let $T_{cover}(u)$ be the expected length of a simple random walk that starts at node *u* and visits every node in G at least once.

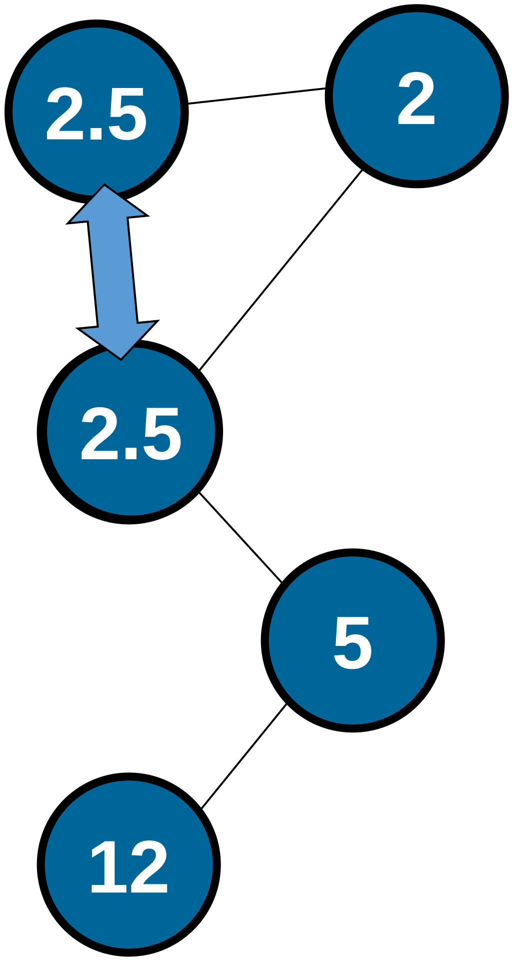
Cover time of G => $T_{cover}(G) = \max_{u \text{ in } G} T_{cover}(u)$.

Given a random geometric graph G with n nodes, if it is a connected graph with high probability, then

$$T_{ave}(\varepsilon, n) = \Theta(n(\log n + T_{mix}(\varepsilon)))$$

A random walk visits each node once by requiring that it makes **C n log n** steps for some C > 0.

Standard gossip



$$
\underbrace{\begin{bmatrix} 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{W(1)} \underbrace{\begin{bmatrix} 2 \\ 2 \\ 3 \\ 5 \\ 12 \end{bmatrix}}_{x(0)} = \underbrace{\begin{bmatrix} 2.5 \\ 2 \\ 2.5 \\ 5 \\ 12 \end{bmatrix}}_{x(1)}
$$

$$x(t) = \mathbf{W}(t)\, x(t-1) = \Pi_t\, \mathbf{W}(t)\, x(0)$$

$\mathbf{W}(t)$ iid random matrices

How many messages

- ε-averaging time: First time where x(t) is ε-close to the normalized true average with probability greater than 1-ε.

$$T_{ave}(n,\varepsilon) = \sup_{x(0)} \inf \left\{ t : P\left(\frac{\| x(t) - x_{ave} \vec{1} \|}{\| x(0) \|} \geq \varepsilon\right) \leq \varepsilon \right\}$$

- x(t)= **W**(t) x(t-1) = $\prod_t$ **W**(t) x(0).
- Define **W**= E **W**(t)

- Theorem: ε-averaging time can be bounded using the spectral gap of **W**:

$$T_{ave}[n,\varepsilon] \leq \frac{3\log(\varepsilon^{-1})}{1-\lambda_2(W)}$$

(Boyd, Gosh, Prabhakar and Shah, IEEE Trans. On Information Theory, June 2006)

Cost of standard Gossip

- Standard Gossip algorithms **require a lot of energy**.

(For realistic sensor network topologies)

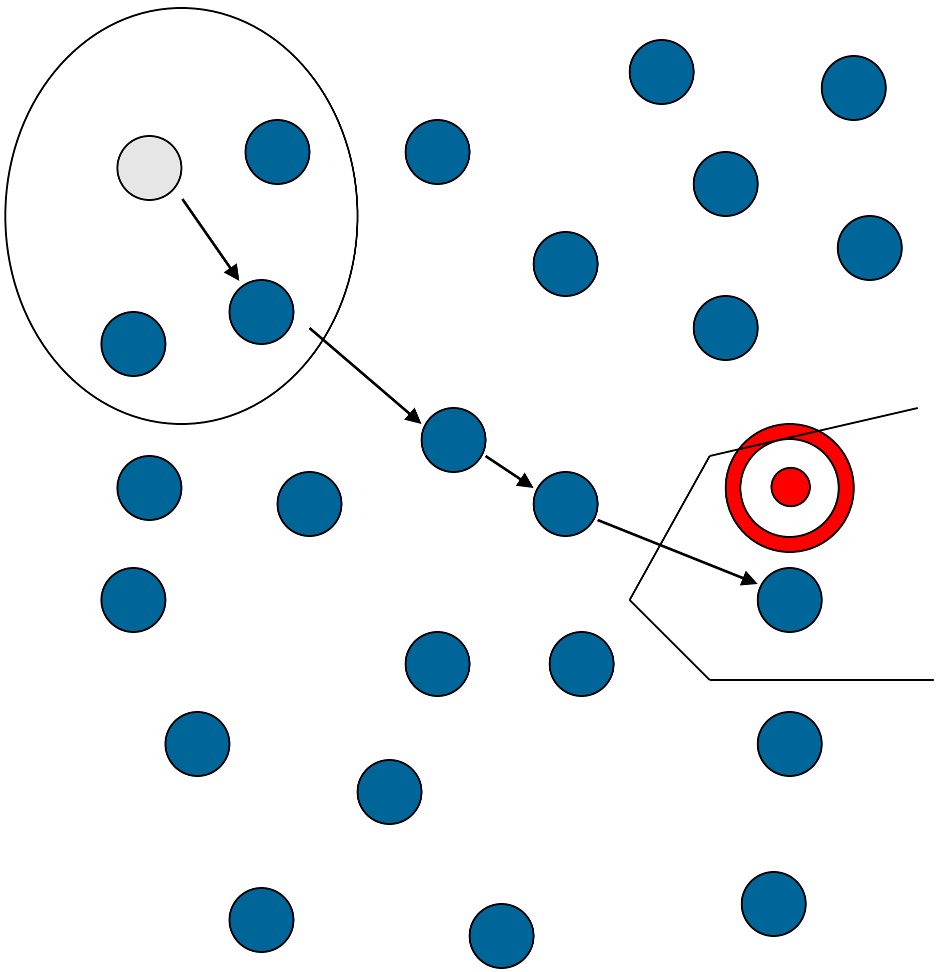
- **Why:** useful information performs random walks, diffuses slowly
- Can we save energy with extra information?

- **Idea:** gossip in random directions, diffuse faster.

- Assume each node knows its location and locations of 1-hop neighbors.

Random Target Routing

- Node picks a random location (="target")
- Greedy routing towards the target
- Probability to receive ~ Voronoi cell area



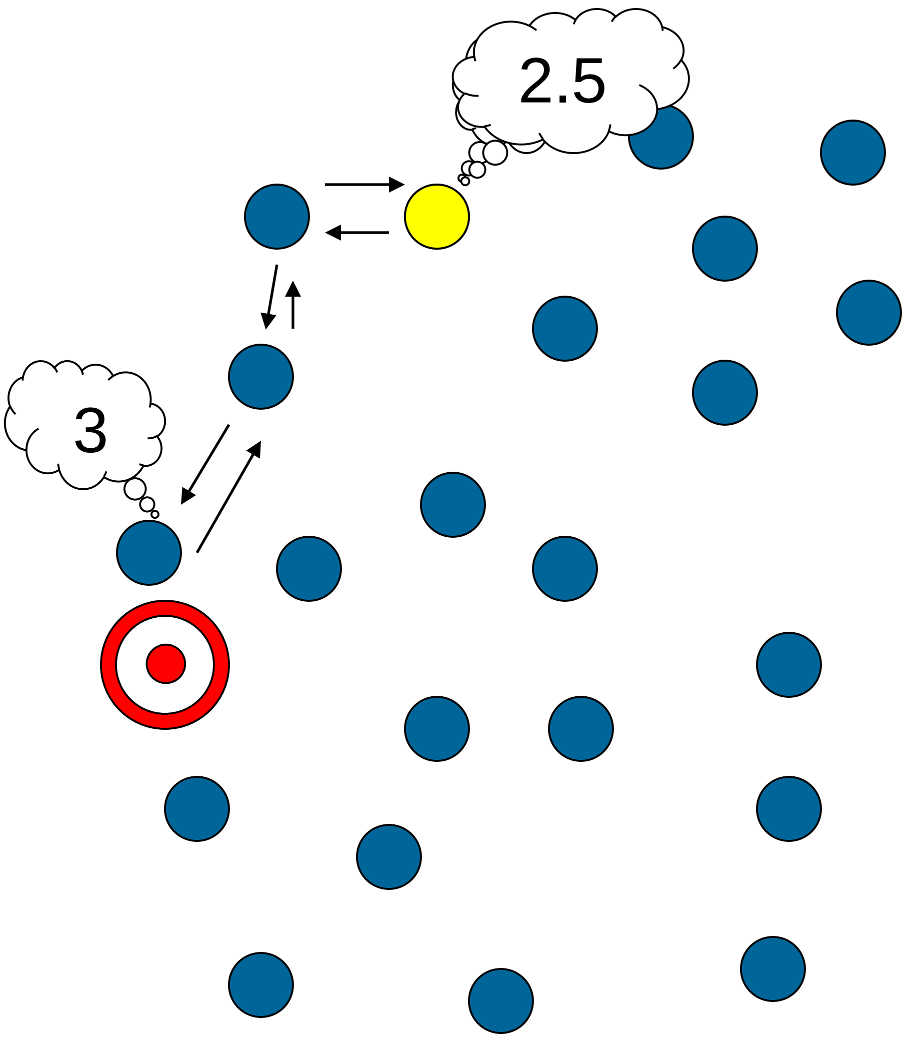
Geographic Gossip

- Nodes use random routing to gossip with nodes far away in the network
- Each interaction costs

$$O\left(\sqrt{\frac{n}{\log n}}\right) = O\left(\frac{1}{r(n)}\right)$$

- But faster mixing
- Number of messages

$$T_{ave}(n) \sim O(n^{1.5})$$



Path averaging

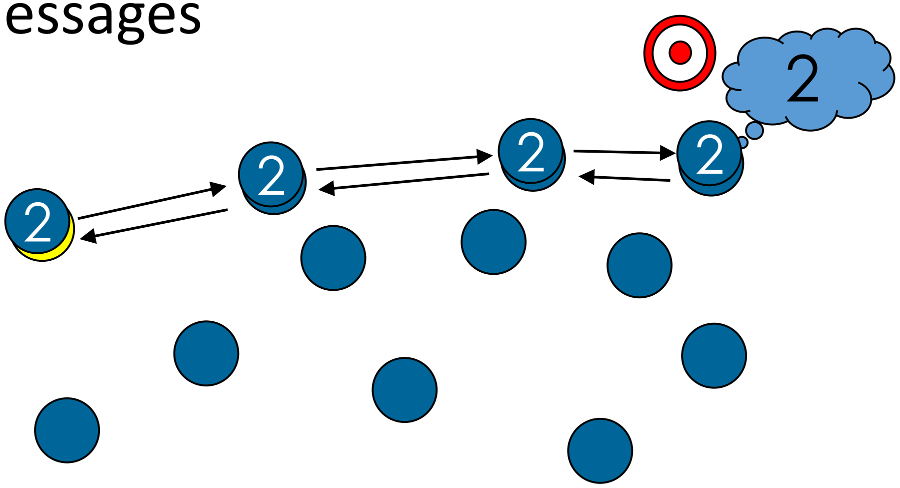
Averaging on the routed path?

The routed packet computes the sum of all the nodes it visits, and a hop-count. The average is propagated backwards to all the nodes on the path.

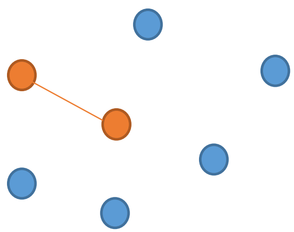
Theorem: Geographic gossip with path averaging on G(n, r) requires expected number of messages

$T_{ave}=\Theta(\mathbf{n} \log 1/\varepsilon)$

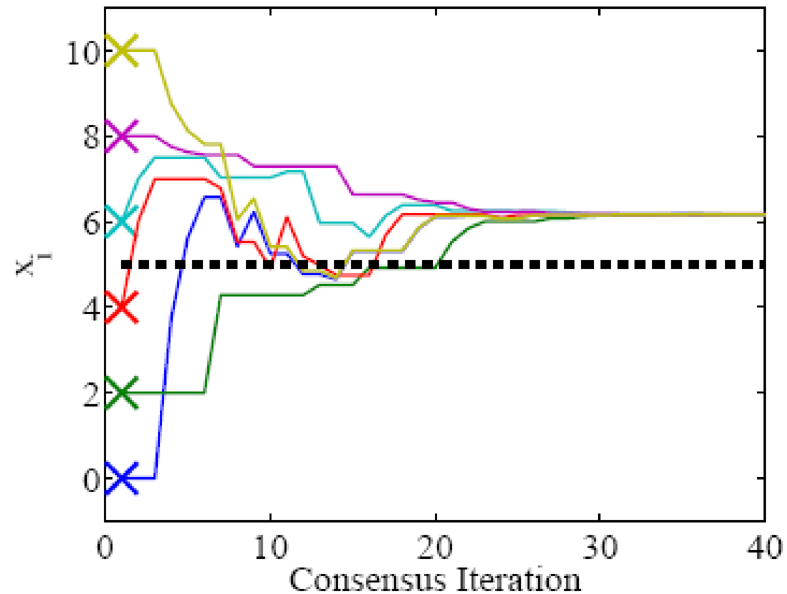
Optimal number of messages

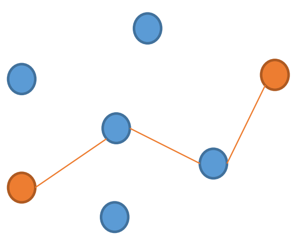


Communication models

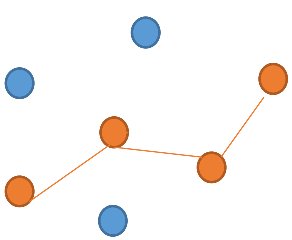


Pair-wise
Θ(n2)

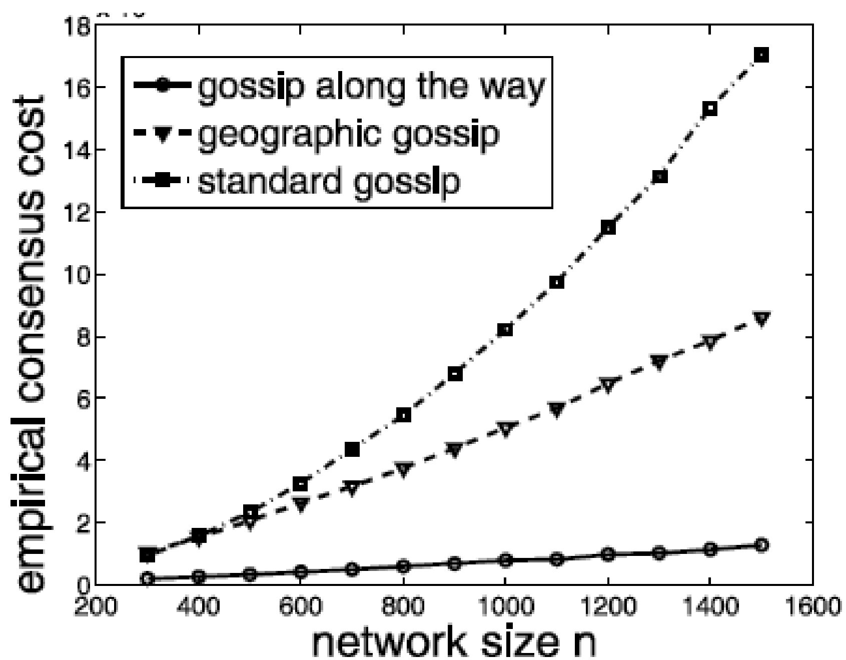




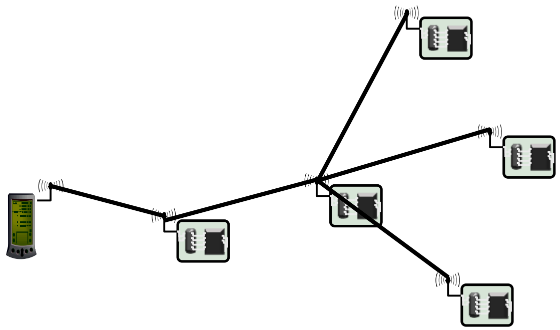
Geographic
routing
Θ(n1.5 √logn)



Path
averaging
nlogn



Data management in WSN



Conventional approach

Sample -> aggregate to sink -> perform analysis

Limitations

- Inefficient for large scale network
- Deployment constraints -> inaccessible sink

Alternative approach

Nodes store data locally -> collector (mobile) gathers

Unreliable & failed nodes

Persistent data storage

Distributed Data Storage

n: nodes in the network

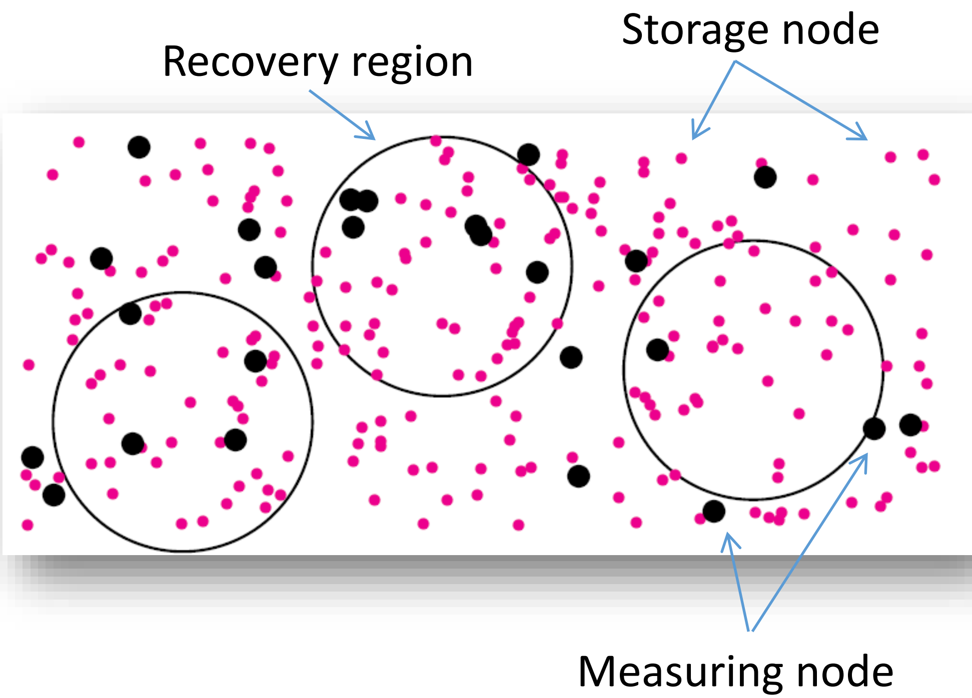
k: sensors take measurements

Objective

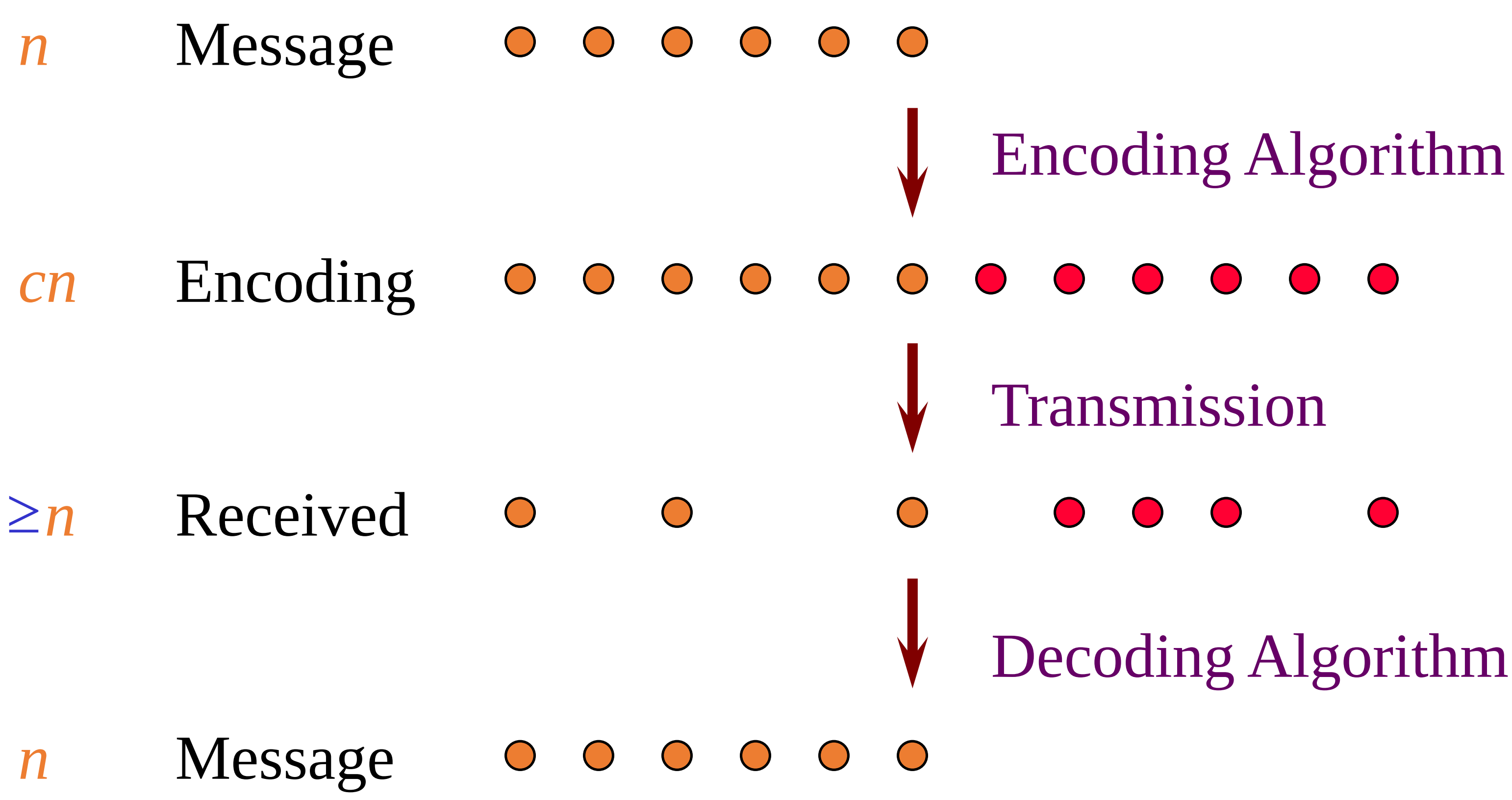
- Each sensors stores *one* packet
- Recovery from any $k(1+\varepsilon)$ nodes
- Decentralized operation

Motivation

- Localized data gathering (energy)
- Recovery from failing networks



Erasure Codes



Data persistence with Fountain Codes

Erasure codes: encode message of k symbols -> n symbols (where k<n) s.t. original message can be recovered from a subset of the n symbols.

Fountain codes (rateless erasure codes)

- limitless sequence of symbols
  from a given set of source symbols
- original source symbols can be
  recovered from *any subset* of the symbols
  of size *equal* to the number of source symbols
- Key representatives: **Luby Transform (LT)** and **Raptor**

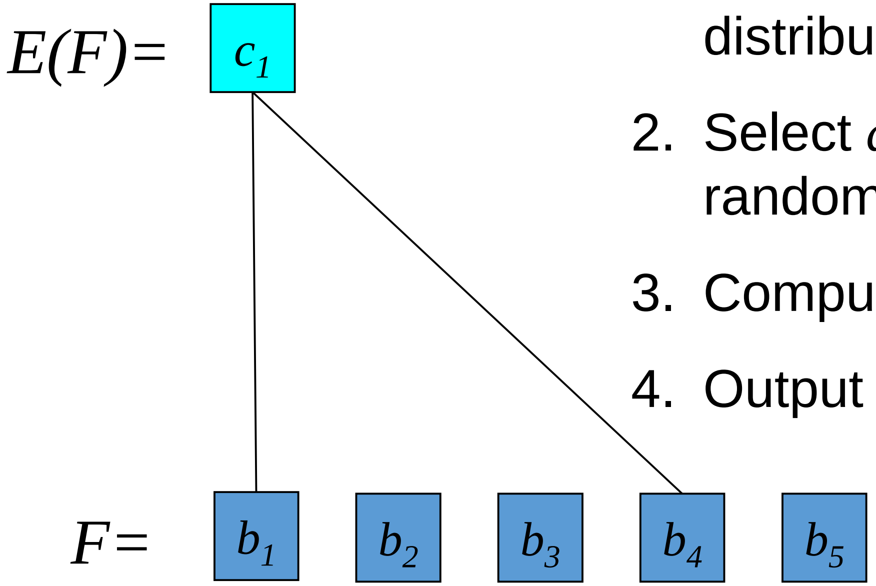
Erasure Codes: LT-Codes

1. Pick *degree* d_1 from a pre-specified distribution. (d_1=2)
2. Select d_1 input blocks uniformly at random. (Pick b_1 and b_4)
3. Compute their sum (XOR).
4. Output sum, block IDs

$F = \; b_1 \; b_2 \; b_3 \; b_4 \; b_5$

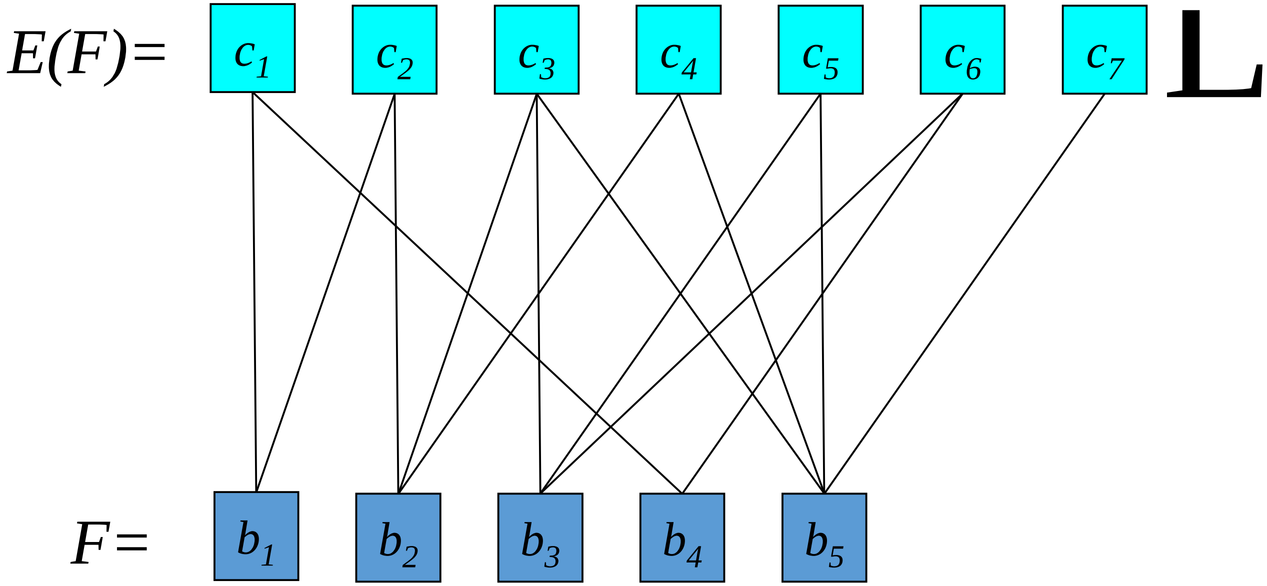
n=5 input blocks

LT-Codes: Encoding

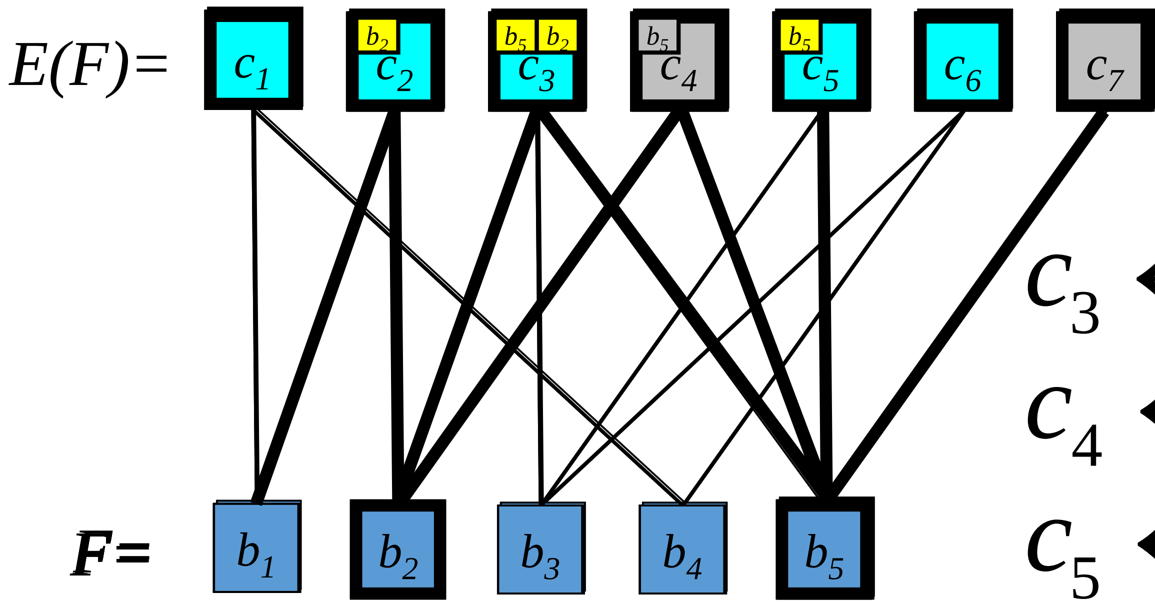


1. Pick *degree* d_1 from a pre-specified distribution. ($d_1=2$)
2. Select d_1 input blocks uniformly at random. (Pick b_1 and b_4)
3. Compute their sum (XOR).
4. Output sum, block IDs

LT-Codes: Encoding



LT-Codes: Decoding



$$c_3 \leftarrow c_3 - b_5$$
$$c_4 \leftarrow c_4 - b_5$$
$$c_5 \leftarrow c_5 - b_5$$

Linear code fundamentals

- Generator matrix **s=mG**
- s = encoded vector, m = input vector, G in $R^{M \times K}$
- K = degree

Ideal Soliton distribution

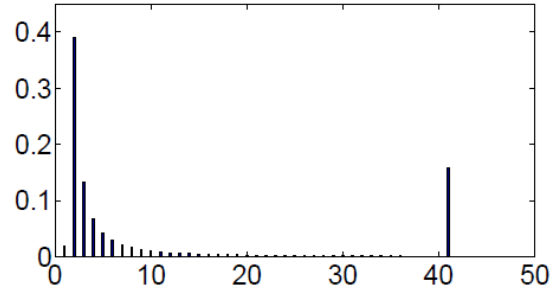
$$\rho(i) = \begin{cases} 1/K & \text{if } i = 1, \\ 1/i(i-1) & \text{for } i = 2, 3, \ldots, K. \end{cases}$$

Recovery

- $k + O(\sqrt{k}\ln^2(k/\delta))$ encoding symbols w.h.p. $(1-\delta)$.

Complexity

- $O(k\ln(k/\delta))$



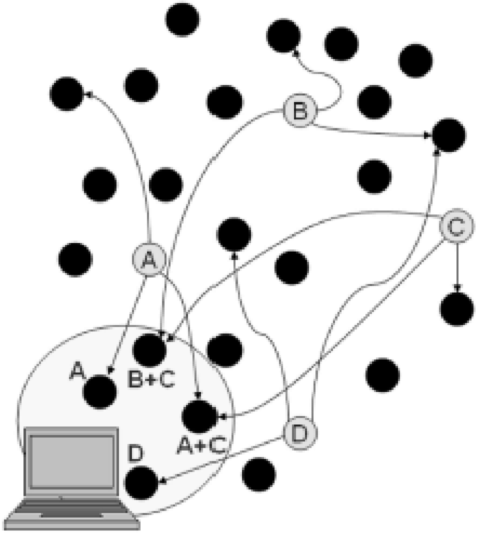
Data dissemination

➢ A random walk with length L will stops at a node.

➢ If the length L of random walk is sufficiently long, then the distribution will achieve steady state.

Algorithmic steps

- Step 1 : Degree generation
- Step 2 : Compute steady-state distribution
- Step 3 : Compute probabilistic forwarding table
- Step 4 : Compute the number of random walks
- Step 5 : Block dissemination
- Step 6: Encoding



Distributed Data Storage with LC

Encoding and Storage Phase (at all nodes u)

1) Node u draws $d_c(u)$ from $\{1, \ldots, \hat{k}(u)\}$ according to Ω.
2) Upon reciving packet x, if $c(x) < C_1 \hat{n} \log \hat{n}$, node u
   - puts x into its forward queue and increments $c(x)$.
   - with probability $d_c(u)/\hat{k}$, accepts x for storage and updates its storage variable y_u^- to y_u^+ as

$$y_u^+ = y_u^- \oplus x_s, \tag{14}$$

   If $c(x) < C_1 \hat{n} \log \hat{n}$, x is removed from circulation.
3) When a node receives a packet before the current round, it forwards its head-of-line (HOL) packet to a randomly chosen neighbor.
4) Encoding phase ends and storage phase begins when each node has seen its $\hat{k}(u)$ source packets.

Experimental results

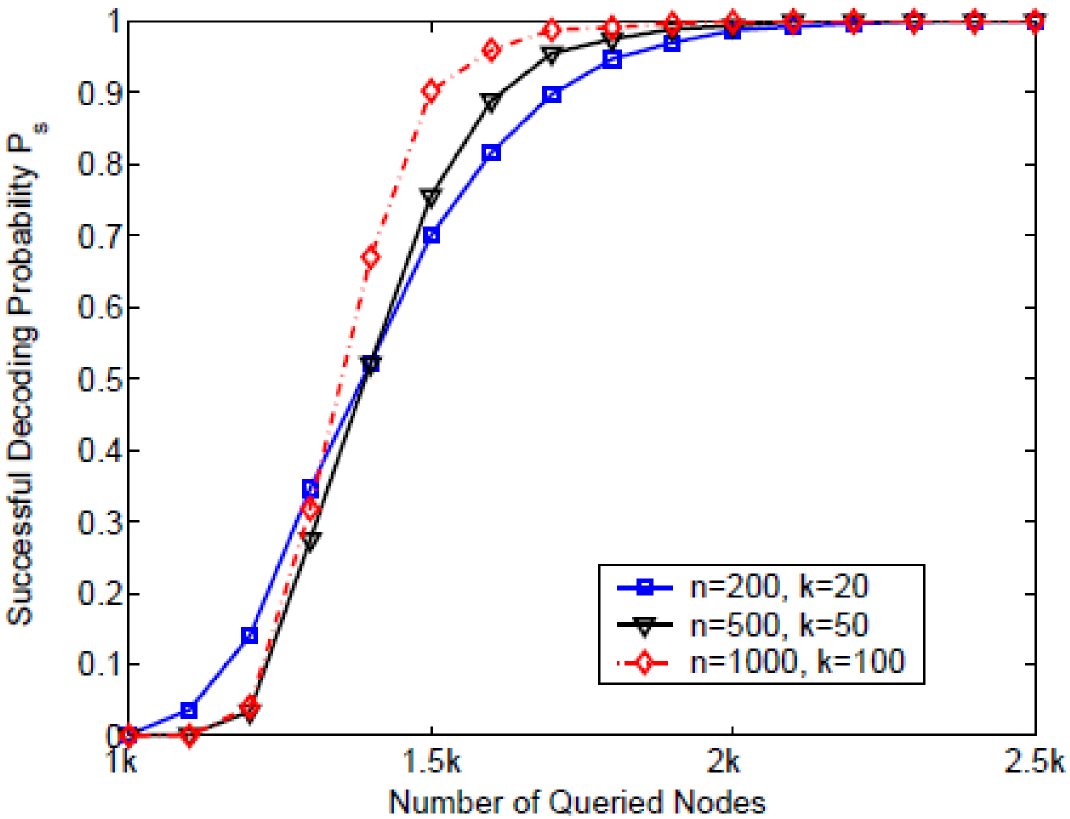
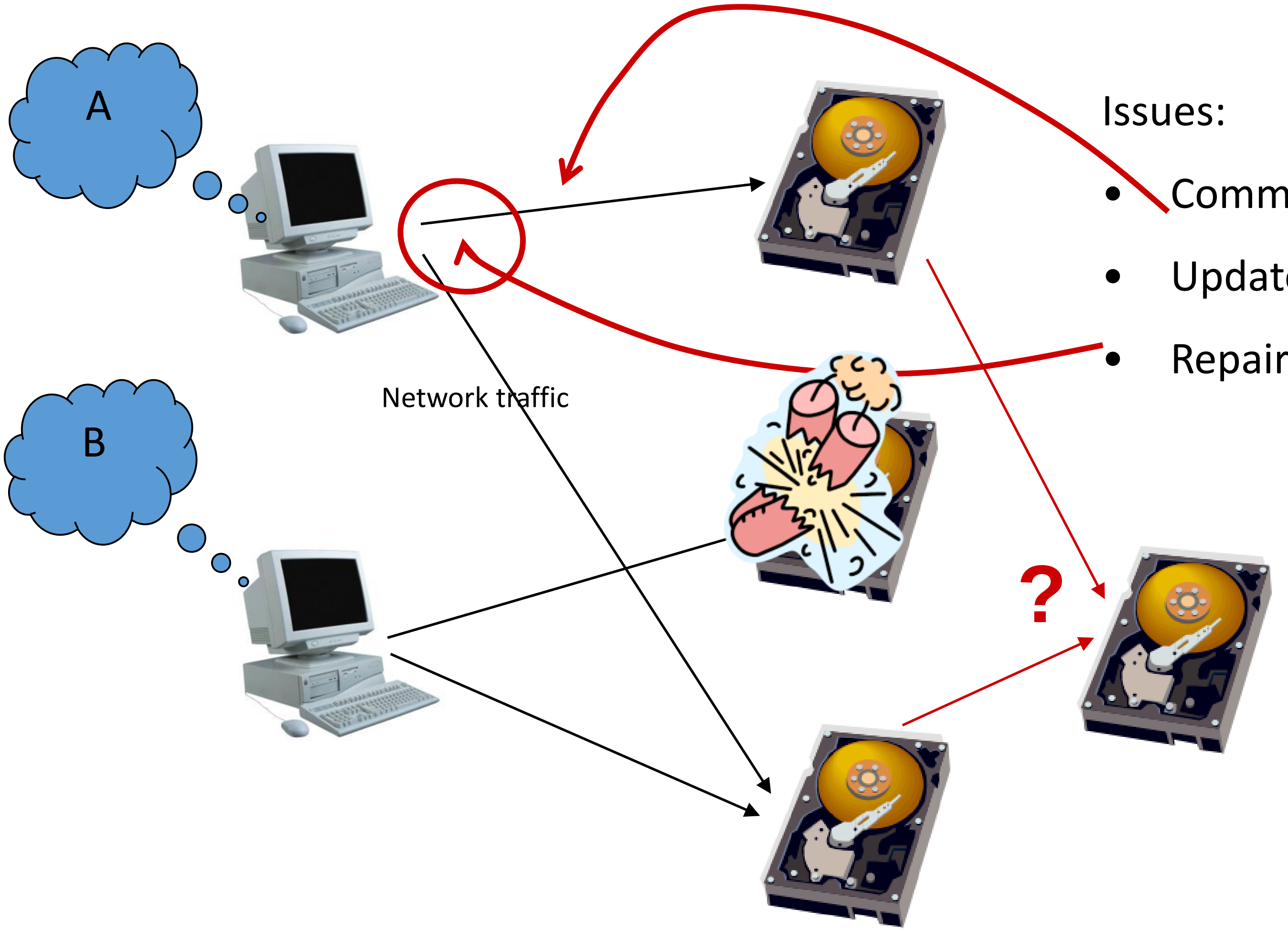


Fig. 2. Performance of LTCDS with known n and k for (a) n=200, k=20; (b) n=500, k=50; and (c) n=1000, k=100.

Coding + Storage Networks = New open problems



Issues:

- Communication
- Update complexity
- Repair communication

Reading List

- Dimakis, Alexandros G., et al. "**Gossip algorithms for distributed signal processing**." *Proceedings of the IEEE* 98.11 (2010): 1847-1864.
- Shuman, David I., et al. "**The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains**." *Signal Processing Magazine, IEEE* 30.3 (2013): 83-98.
- Ostovari, Pouya, Jie Wu, and Abdallah Khreishah. "**Network coding techniques for wireless and sensor networks**." The Art of Wireless Sensor Networks. Springer Berlin Heidelberg, 2014. 129-162.
- Guide to Wireless Sensor Networks eds S. Misra, I. Woungang, S. C. Misra, Chapter 7: Data-Centricity in Wireless Sensor Networks, Abdul-Halim Jallad and Tanya Vladimirova.
- Dimakis, Alexandros G., et al. "A survey on network codes for distributed storage." Proceedings of the IEEE 99.3 (2011): 476-489.