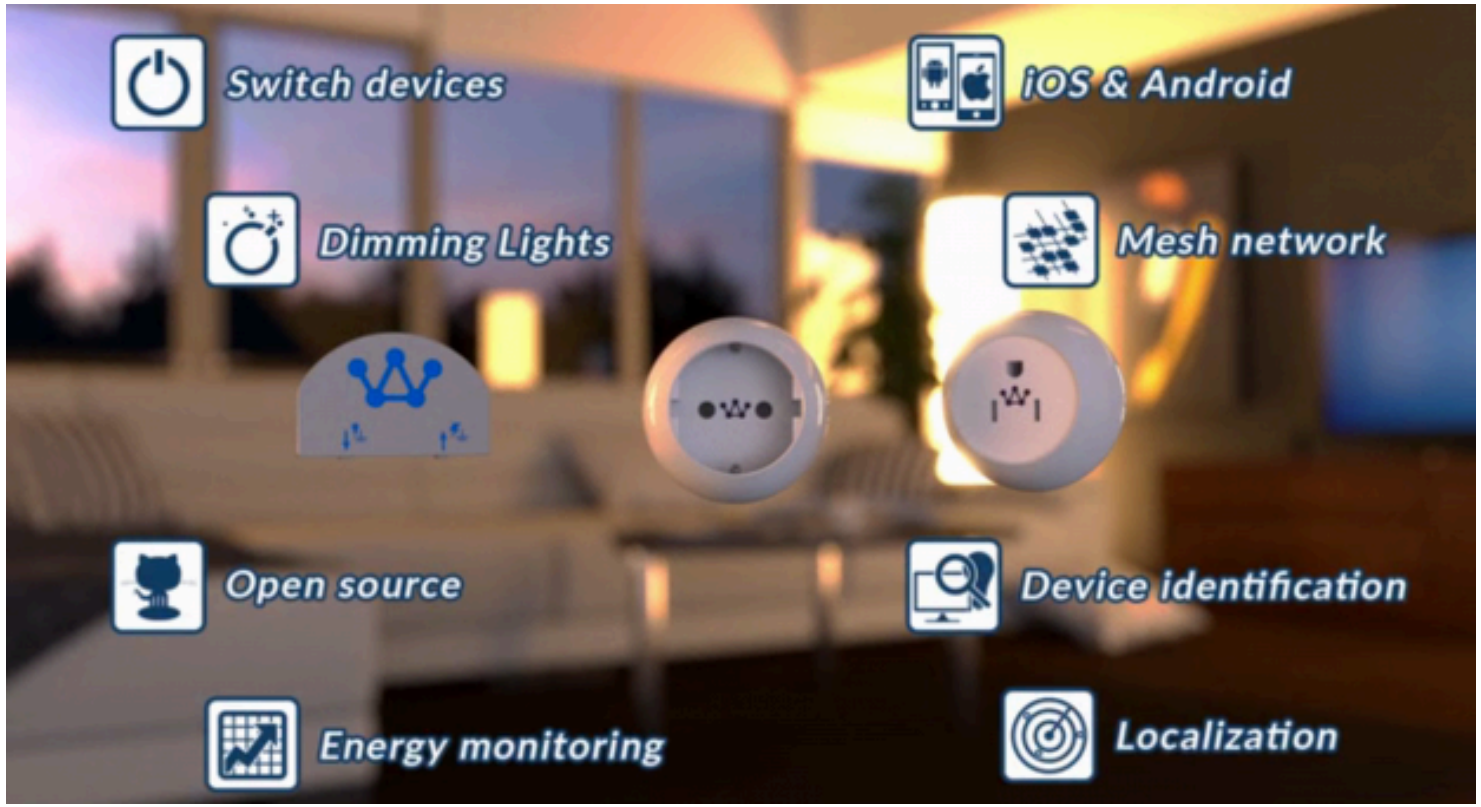
CWI

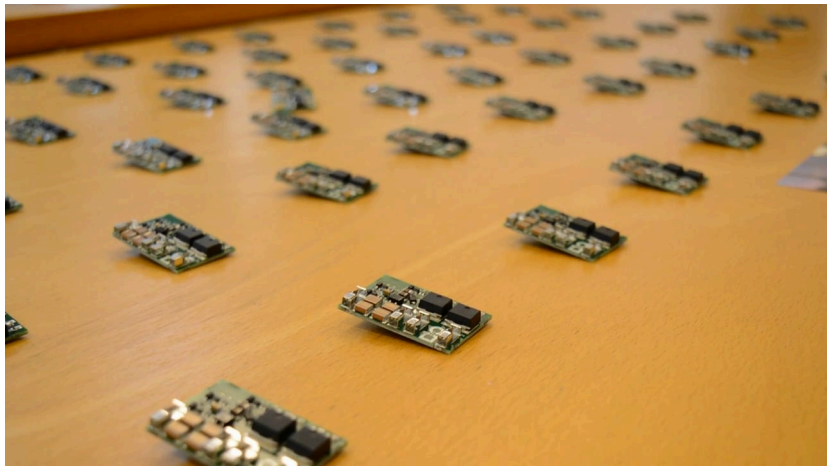
Centrum Wiskunde & Informatica

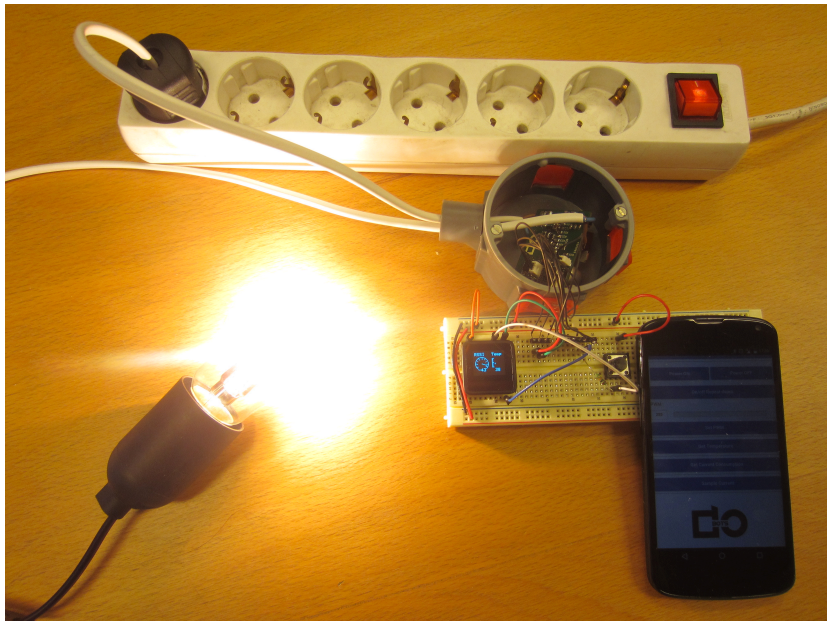
Data science meets embedded systems

Dr. Stefan Dulman

stefan.dulman@cwi.nl



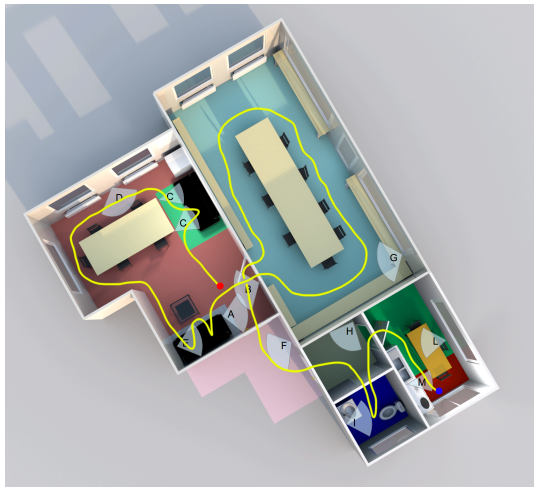




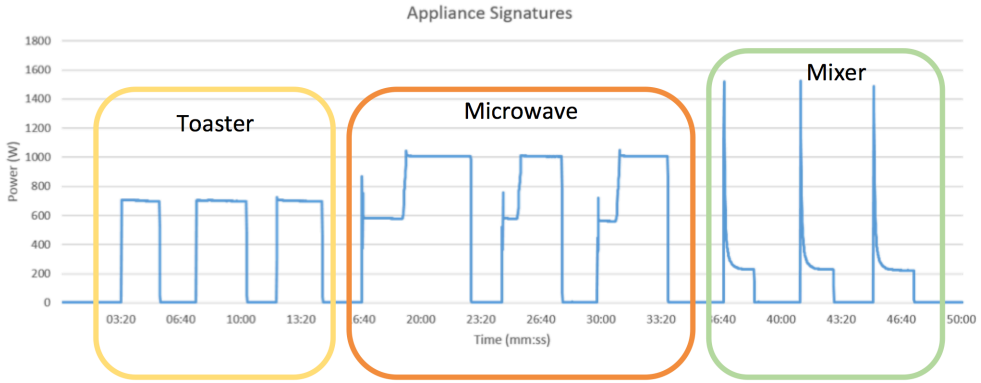
http://www.crownstone.rocks/

Possible applications

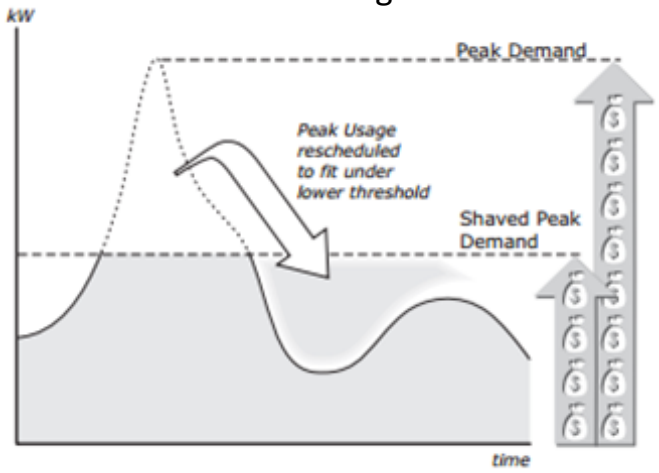
Indoor localization



Energy monitoring



Peak shaving

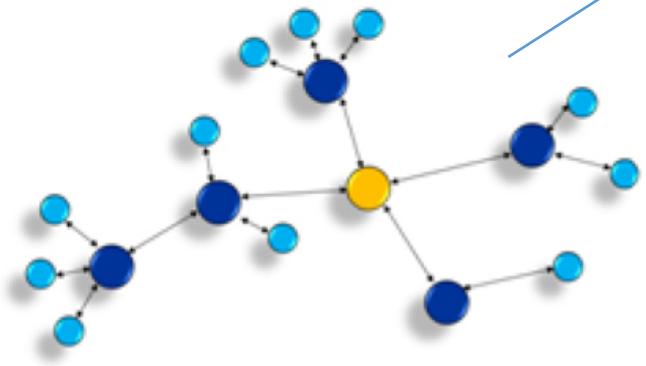


Preventive maintenance

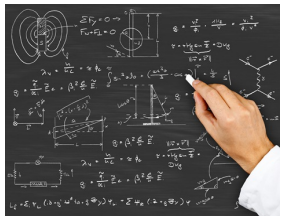
System design

User

The “cloud”



Sensor network



Mathematical model

Data scientist

Lecture outline

- Processing in distributed networks
  - Aggregate computation
  - Fast gossiping algorithms
- *Data science and the internet of things*
  - *Distributed classifiers*
  - *Machine learning in NILM*
- *Dealing with failures in aggregates*
- *Conclusions*

Processing in distributed systems

- Specialists are expensive, simple programming paradigms
- Example: Basic Linear Algebra Subprograms (BLAS)
  - GPU-s, multicore systems, the “cloud”
  - Primitive: $[\ldots]_{m,n} \times [\ldots]_{n,1}$
- BLAS for mesh networks - difficult…
  - Current cloud research - graph processing

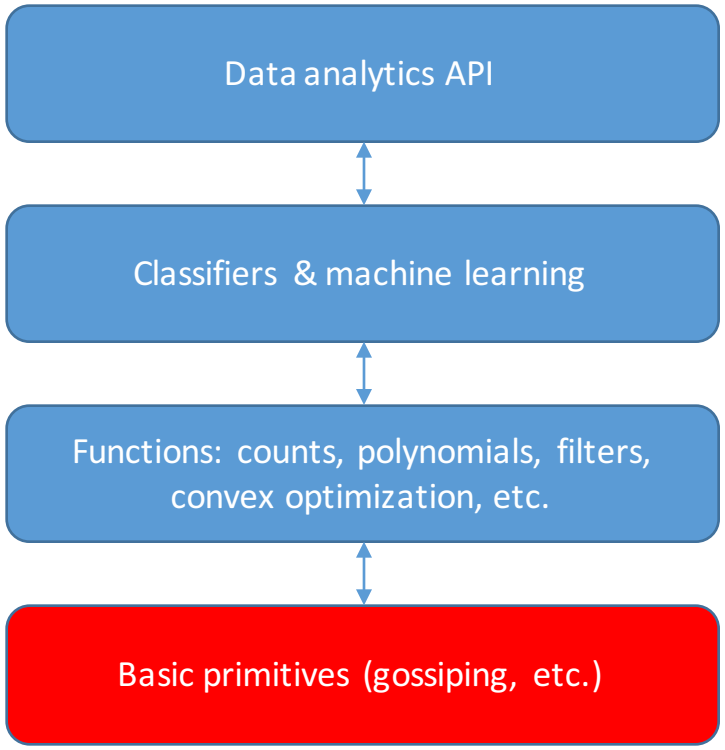
GPUs

Multicore systems

Server farm

Mesh networks

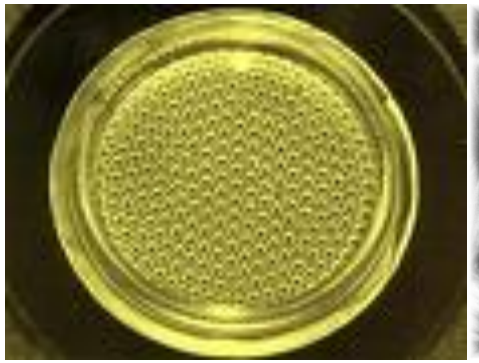
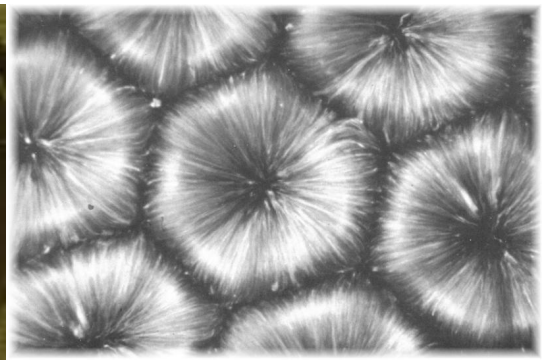
First layer


Data analytics API
Classifiers & machine learning
Functions: counts, polynomials, filters, convex optimization, etc.
Basic primitives (gossiping, etc.)

Mesh networks?

- Which primitives are specific and can be used as basic element of the algebra?



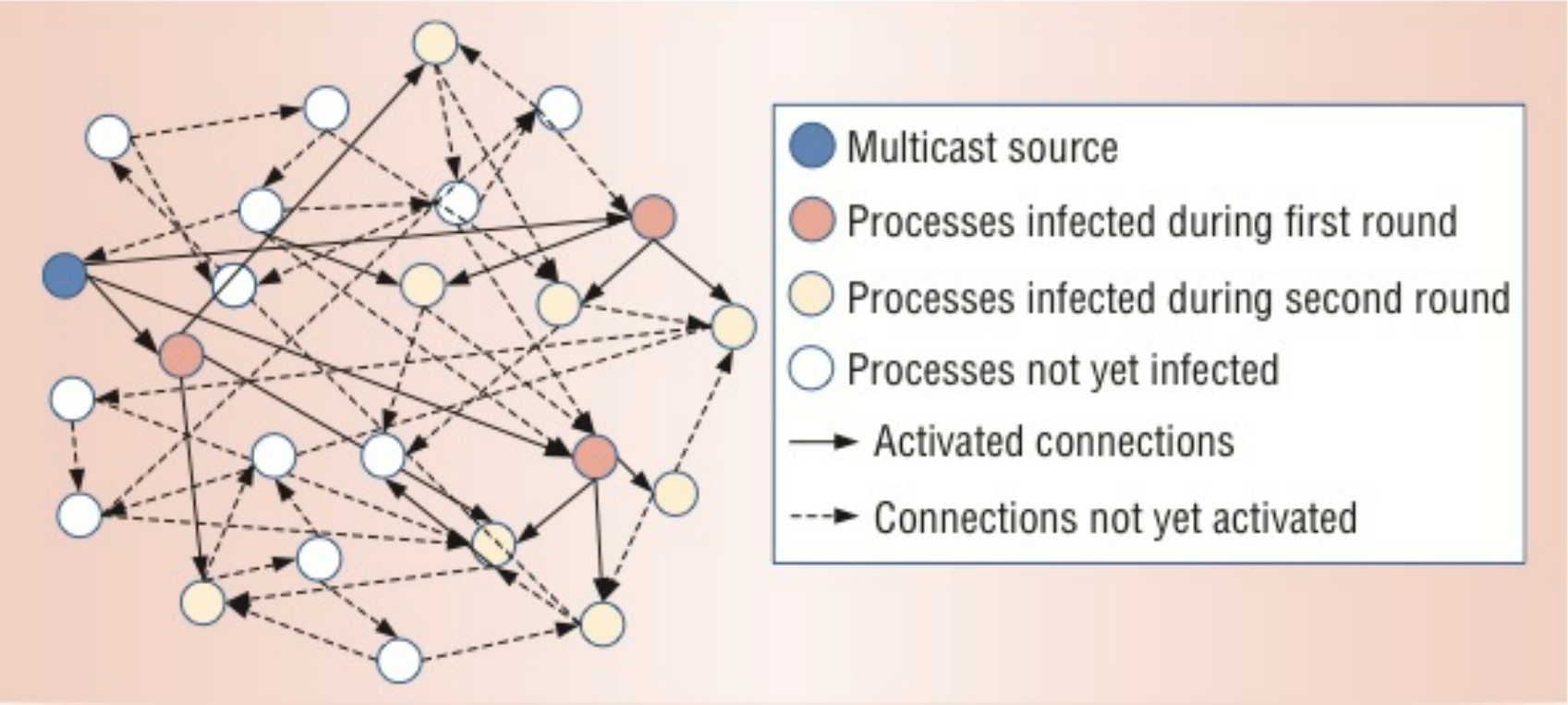




Gossip-based interaction

- Gossip: basic mechanism for spreading and aggregating information
  - Communication takes place locally
  - Nodes exchange information with neighbors
  - Robust, fast, scales well
- Exchanged data can be anything
  - Exchanged data, references to nodes, actual programs, etc.
- No centralized control or management

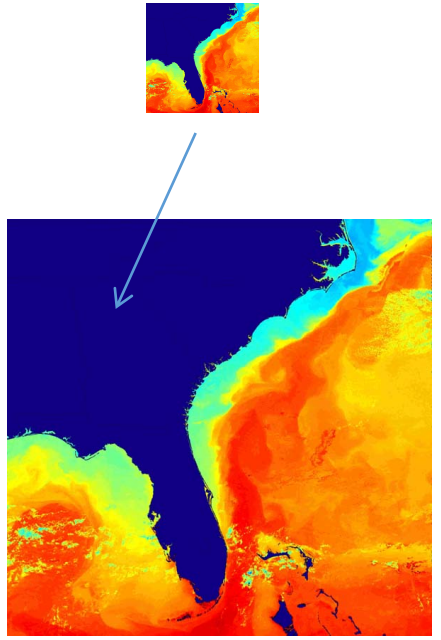
Example – gossiping (multicasting)



P. Eugster: *Epidemic Information Dissemination in Distributed Systems* (IEEE Computer, 2004)

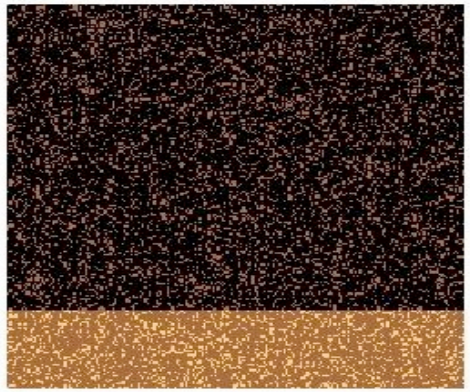
Gossip-based applications

- Examples
  - Raw information dissemination
  - Data aggregation
  - Topology construction for overlay networks
  - Semantic clustering of nodes
  - Realizing storage facilities in ad hoc networks
  - Distributed signal processing
- Not everything can be accomplished via gossiping!

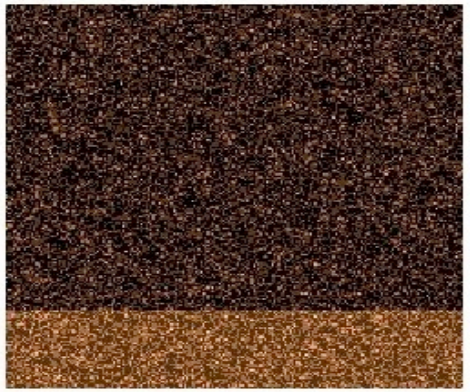


Gossip (push) – fully connected network

Round 1



Round 2



Round 3

Round 4

Round 5

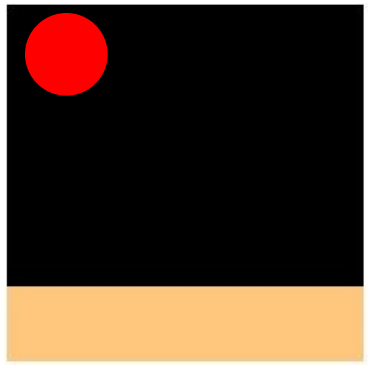
Round 6

Round 7

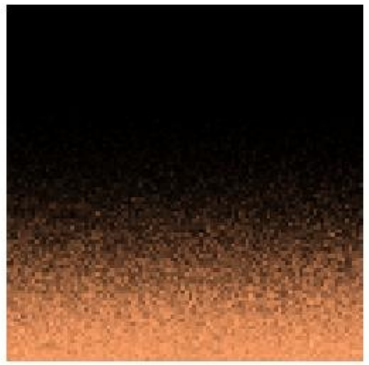
Round 8

200x200 nodes

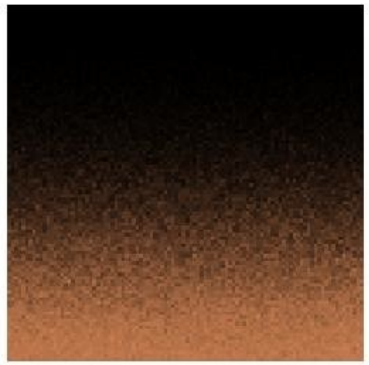
Gossip (push-pull) – mesh network



Round 1



Round 10



Round 20

Round 30

Round 40

Round 50

Round 60

Round 70

100x100 nodes, transmission range = 10 units

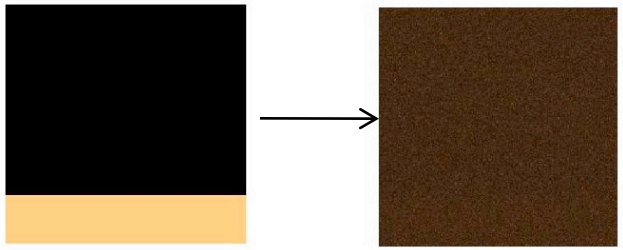
Gossiping mechanism

- Terminology
  - Anti-entropy: each node chooses another node and exchanges information about difference in states
    - Both nodes have access to identical information -> identical states
  - Gossiping: a node informs other nodes about its own state (infects the other nodes) and may stop
- Notations
  - Object O has on node S the value val(O,S) and timestamp T(O,S)

!!! We will use “gossiping” terminology for both mechanisms !!!

Anti-entropy

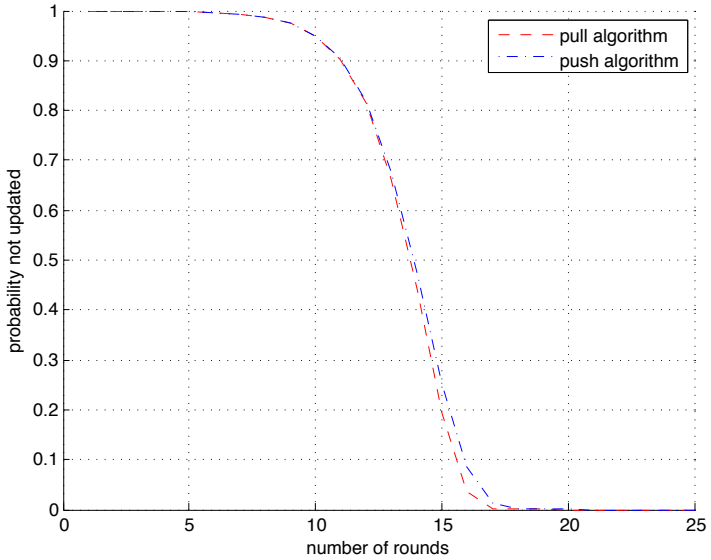
- Node S exchanges information with node S'
  - Push: T(O,S')<T(O,S) -> val(O,S') <- val(O,S)
  - Pull: T(O,S')>T(O,S) -> val(O,S) <- val(O,S')
  - Push-pull: S and S' exchange their updates
- Convergence speed:
  - O(log n) cycles update speed for fully connected network, random unicast model
    - 40000 nodes – 8 cycles to compute the average value
  - O(D^2 log n) in a mesh network – this may be slooooow!!!



Round 1 Round 8

Anti-entropy analysis

- Setup: a source propagates updates in a fully connected network
  - Pi = probability that a node has not received update after i-th cycle
- Two cases:
  - Pull: $p_{i+1} = p_i^2$
    - the node was not updated during the i-th cycle and should contact another node during next cycle
  - Push: $p_{i+1} = p_i \cdot \left(1 - \frac{1}{N}\right)^{N(1-p_i)} \approx p_i \cdot e^{-1}$
    - the node did not update during i-th round and no node chooses it for the next round



Gossiping mechanism

- Basic mechanism
  - Node S (containing the update) contacts node S'
  - If node S' knows the update
    - Node S stops with probability p=1/k (S is effectively removed)
  - Otherwise
    - Node S contacts another random node S''

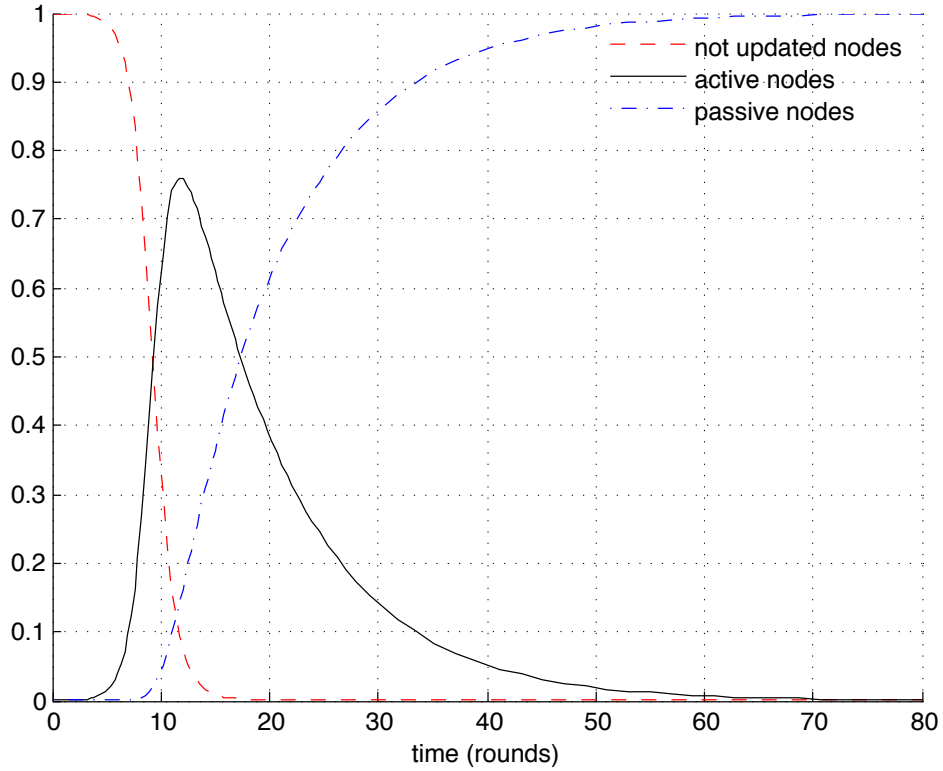
Gossiping – mathematical model

$$i + s + r = 1$$

$$\frac{ds}{dt} = -si$$

$$\frac{di}{dt} = si - \frac{1}{k}(1-s)i$$

$$\frac{dr}{dt} = \frac{1}{k}(1-s)i$$



S – nodes not updated → I – active nodes (updated) → R – passive nodes (updated)

Gossiping – mathematical model

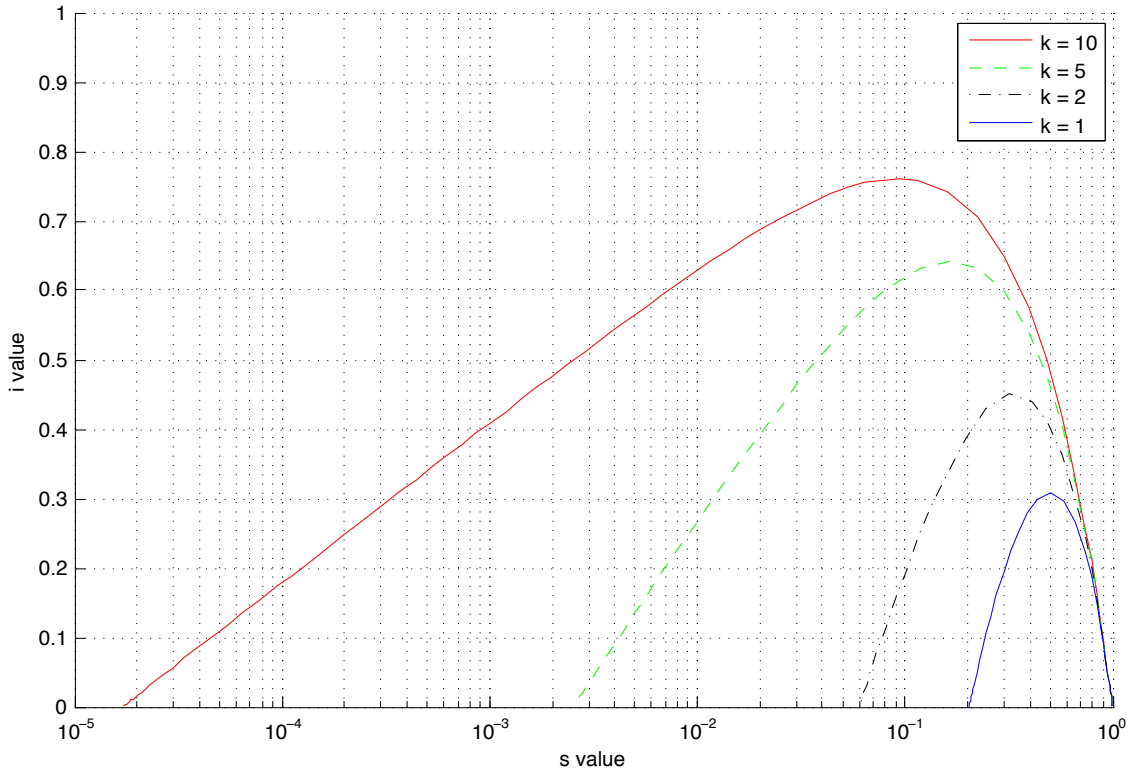
$$\frac{di}{ds} = -\frac{k+1}{k} + \frac{1}{k} \cdot \frac{1}{s}$$

$$i(s) = -\frac{k+1}{k} s + \frac{1}{k} \ln s + C$$

$$i(1) \rightarrow C = \frac{k+1}{k}$$

$$i(s) = \frac{k+1}{k}(1-s) + \frac{1}{k} \ln s$$

$$i(s) = 0 \rightarrow s = e^{-(k+1)(1-s)}$$

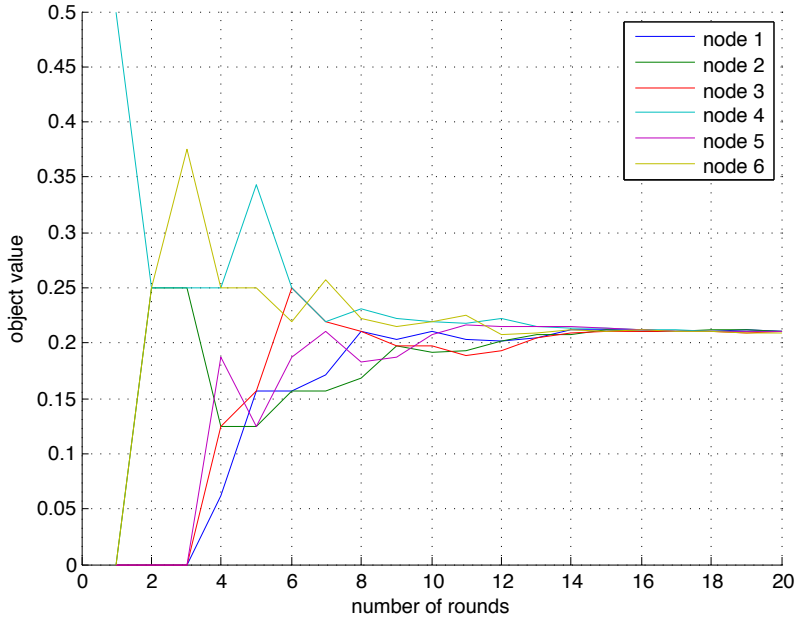


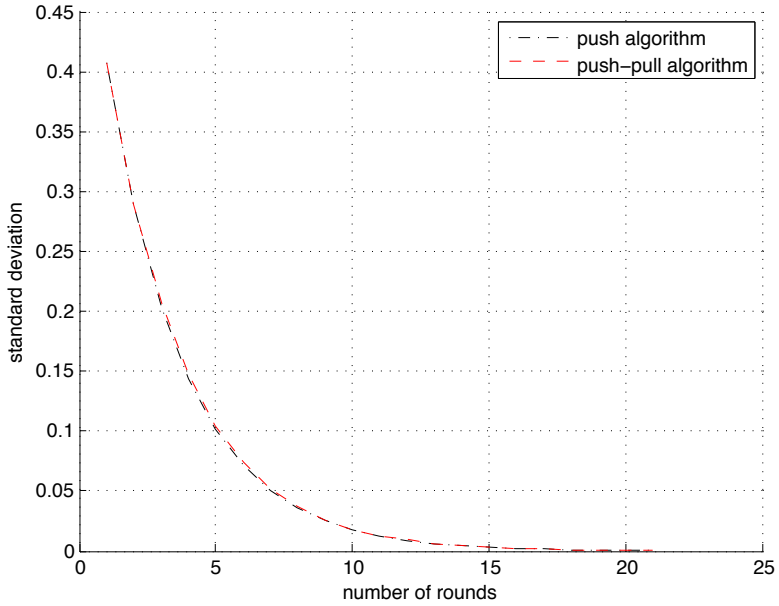
k	s	Ns (n=10000)
1	0.203188	2032
2	0.059520	595
5	0.002516	25

!!! Gossiping is not perfect !!!

Convergence speed

- Demonstration sketch:
  - Consider at each step in time values x_i
  - Compute the standard deviation of x_i at each time round (s_i)
  - Show that s_i decreases exponentially with time





Fully connected networks, anti-entropy

PushSum algorithm

- Simple algorithm of computing average values
  - Each node starts with a (random) value & a weight
  - Nodes exchange pieces of the value with neighbors
  - In time, all values converge to a common mean value
  - PushSum is not strictly an “anti-entropy” algorithm!

Node i:

1: $\{(m_{r,t-1}, \omega_{r,t-1})\}$ – received pairs in round t-1

2: $m_{i,t} = \Sigma_r(m_{r,t-1})$

$\omega_{i,t} = \Sigma_r(\omega_{r,t-1})$

3: choose random neighbor j

4: send to i and j: (½ $m_{i,t}$, ½ $\omega_{i,t}$)

5: estimate in round t is: $m_{i,t}/\omega_{i,t}$

D. Kempe et al.: Gossip-Based Computation of Aggregate Information (FOCS 2003)

PushSum example

m_i	m_j	ω_i	ω_j
7	2	1	1
3.5	5.5	0.5	1.5
6.25	2.75	1.25	0.75
3.125	5.875	0.625	1.375
6.062	2.937	1.312	0.685
3.031	5.969	0.626	1.374
6.016	2.984	1.313	0.627
…	…	…	…
3	6	0.67	1.33
6	3	1.33	0.67
3	6	0.67	1.33
…	…	…	…

← Initial values

3 -> $\frac{1}{3}(m_i + m_j)$
6 -> $\frac{2}{3}(m_i + m_j)$

0.67 → $\frac{1}{3}(\omega_i+\omega_j)$
1.33 → $\frac{2}{3}(\omega_i+\omega_j)$

Final values:

$$\frac{m_i}{\omega_i} = \frac{\frac{1}{3}(m_i + m_j)}{\frac{1}{3}(\omega_i+\omega_j)} = \frac{m_i + m_j}{2}$$

$$\frac{m_j}{\omega_j} = \frac{\frac{2}{3}(m_i + m_j)}{\frac{2}{3}(\omega_i+\omega_j)} = \frac{m_i + m_j}{2}$$

Extension of PushSum

- Computing averages:
  - $m_{i,0}$ = "random", $\omega_{i,0} = 1$
  - $m_{i,\infty} = sum_i(m_{i,0})/n$
- Computing sums:
  - $m_{i,0}$ = "random", $\omega_{i,0} = 0$, $\omega_{0,0} = 1$
  - $m_{i,\infty} = sum_i(m_{i,0})$
- Counting the nodes in the network:
  - $m_{i,0} = 1$, $\omega_{i,0} = 0$, $\omega_{0,0} = 1$
  - $m_{i,\infty} = sum_i(m_{i,0}) = n$
- Main properties hold for all variations:
  - The algorithms are guaranteed to converge
  - The convergence speed is the same in all cases

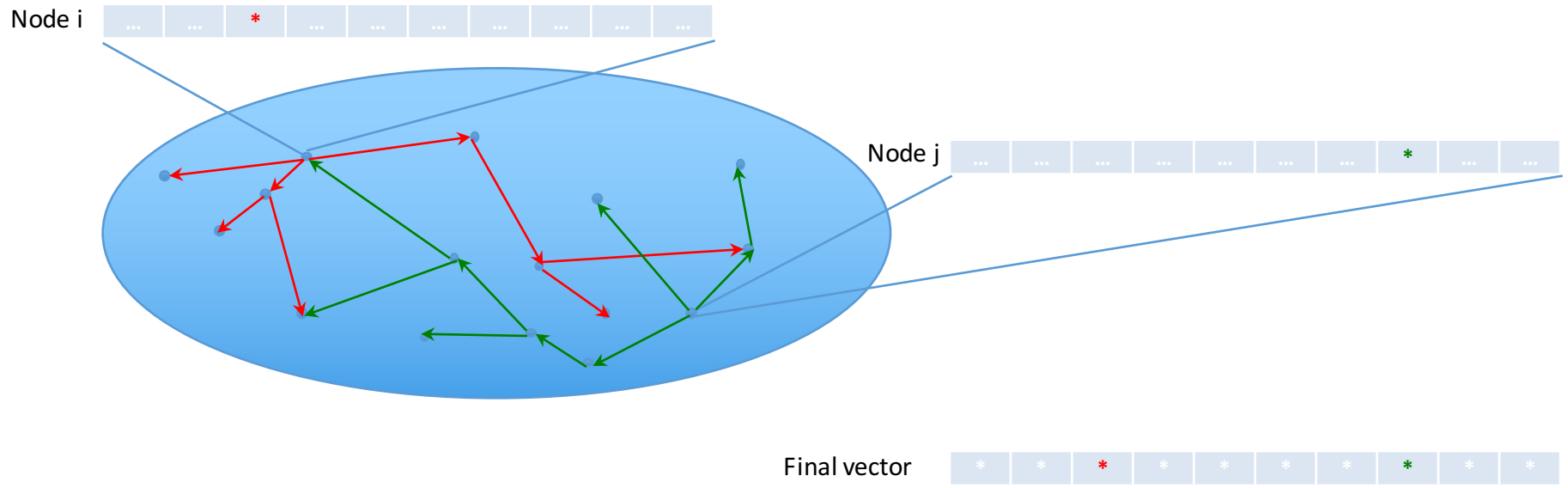
PushSum & network dynamics

- Convergence speed is influenced by:
  - Network diameter – reduced the convergence speed
  - Node mobility – mobility increases convergence speed
  - Number of random neighbors selected
    - Unicast – Multicast – Broadcast

Anand D. Sarwate: The Impact of Mobility on Gossip Algorithms, IEEE Transactions on Information Theory

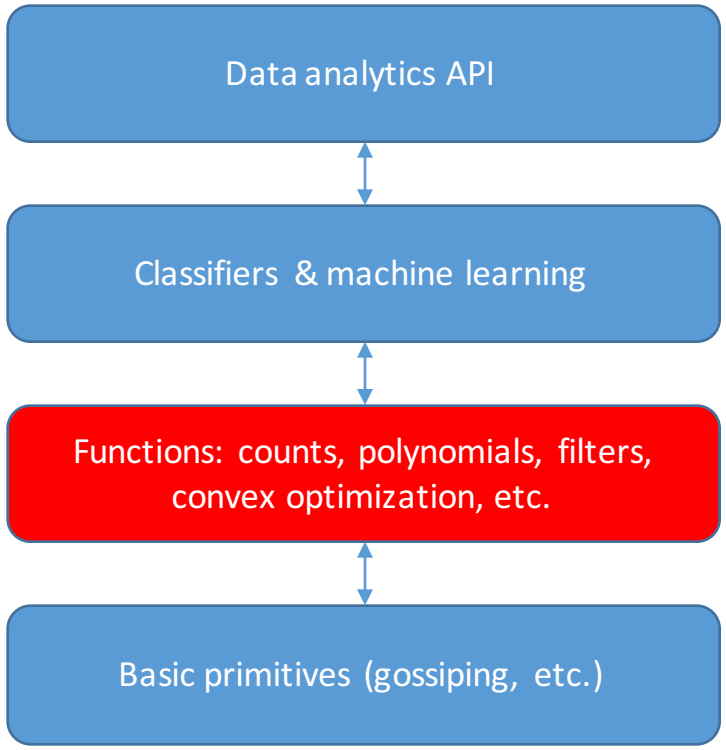
"Fast" gossiping primitives

- Primitive allowing computation of sums in a distributed manner
  - Basic property: minimum value of a series of exponential r.v. with λ_i is a r.v. with parameter $\lambda = \Sigma \lambda_i$
  - Positive aspects: very fast propagation – O(D) time steps
  - Negative aspects: message size affects precision $O(\delta^{-2})$



Shah, Devavrat. Gossip algorithms. Now Publishers Inc, 2009

Second layer



Computability with locally checkable functions

- Simple aggregates can be layered to compute any polynomial function
- Example: compute $\Sigma\, x_i\, x_j$
  - Idea: use the identity $(x_1+x_2)^2 = x_1^2+x_2^2+2x_1x_2$
  - $v_1 = \Sigma\, x_i$
  - $v_2 = \Sigma\, x_i^2$
  - Result = $(v_1^2 - v_2)/2$

What is computable?

- Straight-forward single-step computations
  - minimum and maximum, sums and products [9]
  - averages, quantiles, random sampling [6]
  - generalized means, variance and other moments, counting, rank statistics [5]
  - Presburger algebra [1]
- Multi-step computations
  - find the nth element, quantiles [6]
  - median element [10]
  - system identification (transfer function) [3]
  - first k eigen vectors [7]
  - principal component analysis (via sparsification), MDS-MAP localization [8]
  - Fiedler vector [2]
  - frequency moments of data [12]
  - eigenvectors via wave propagation; network clustering [4], [11]

What is computable?

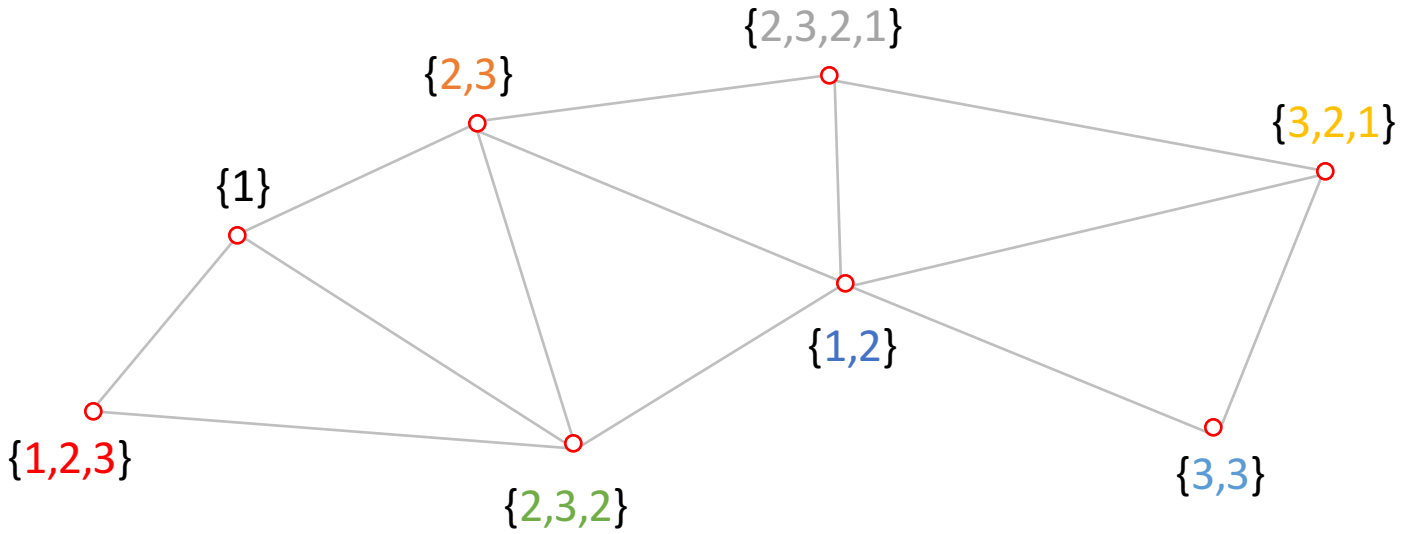
- [1] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert, "The computational power of population protocols," Distributed Computing, vol. 20, no. 4, pp. 279–304, 2007.
- [2] A. Bertrand and M. Moonen, "Distributed computation of the fiedler vector with application to topology inference in ad hoc networks," Signal Processing, vol. 93, no. 5, pp. 1106–1117, 2013.
- [3] A. Carzaniga, C. Hall, and M. Papalini, "Fully decentralized estimation of some global properties of a network," in INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 630–638.
- [4] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of laplacian eigenvalues in multi-agent systems," Automatica, vol. 49, no. 4, pp. 1031–1036, 2013.
- [5] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," ACM Transactions on Computer Systems (TOCS), vol. 23, no. 3, pp. 219–252, 2005.
- [6] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on. IEEE, 2003, pp. 482–491.
- [7] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," in Proceedings of the thirty-sixth annual ACM symposium on Theory of computing. ACM, 2004, pp. 561–568.
- [8] S. B. Korada, A. Montanari, and S. Oh, "Gossip pca," in Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems. ACM, 2011, pp. 209–220.
- [9] D. Mosk-Aoyama and D. Shah, "Fast distributed algorithms for computing separable functions," Information Theory, IEEE Transactions on, vol. 54, no. 7, pp. 2997–3007, 2008.

Random sampling

- Problem: extract random samples from a set
- Example:
  - The set: {1,2,3,1,2,3,2,1,2,3,2,1,2,3,2,1,2,3,3,3}
  - Solutions:
    - Pick randomly *i* in the range 1-20 and pick the i^{th} element
    - Etc…
  - Solution needs to generate random samples according to the distribution of the set:
    - Element frequencies {1,2,3}→ {0.25,0.4,0.35}

Distributed random sampling

- Set is distributed over n nodes
- Problem: pick random samples from the set without gathering all the data at a central point



Push-Random algorithm

- Protocol uses short messages
  - Each node holds only one element
  - Messages contain one element + one weight

Node i:

1: $\{(q_{r,t-1}, \omega_{r,t-1})\}$ – received pairs in round t-1

2: $\omega_{i,t} = \Sigma_r (\omega_{r,t-1})$

$q_{i,t}$ = picked at random from $\{q_{r,t-1}\}$ with prob. $w_{r,t-1}/\omega_{i,t}$

3: choose shares $\alpha_{i,j,t}$ for each neighbor j

4: send to each neighbor j: $(q_{i,t}, \alpha_{i,j,t} \cdot \omega_{i,t})$

5: random sample in round t is: $q_{i,t}$

Lecture outline

- *Processing in distributed networks*
  - *Aggregate computation*
  - *Fast gossiping algorithms*
- Data science and the internet of things
  - Distributed classifiers
  - Machine learning in NILM
- *Dealing with failures in aggregates*
- *Conclusions*

Processing in distributed systems

- Specialists are expensive, simple programming paradigms
- Example: Basic Linear Algebra Subprograms (BLAS)
  - GPU-s, multicore systems, the “cloud”
  - Primitive: $[\ldots]_{m,n} \times [\ldots]_{n,1}$
- BLAS for mesh networks - difficult…
  - Current cloud research - graph processing

GPUs

Multicore systems

Server farm

Mesh networks

Moving towards data science applications

- What is data science?
  - Big data, cloud computing, …
  - Statistics, analytics...
  - Machine learning
- Data, data, data… Who produces large amounts of data?
  - Embedded sensors
    - Wireless sensor networks
    - Internet of things
    - Machine-to-machine
  - “Others”: human social activities (networks, videos, images, songs, etc)

CWI

Centrum Wiskunde & Informatica

Kaggle and data science competitions

Active Competitions

All Competitions

Active Competitions

Ultrasound Nerve Segmentation Identify nerve structures in ultrasound images of the neck	**44 days** **436** teams **428** scripts **$100,000**
State Farm Distracted Driver Detection Can computer vision spot distracted drivers?	**27 days** **1175** teams **724** scripts **$65,000**
Grupo Bimbo Inventory Demand Maximize sales and minimize returns of bakery goods	**56 days** **974** teams **1107** scripts **$25,000**
Avito Duplicate Ads Detection Can you detect duplicitous duplicate ads?	**6.5 days** **548** teams **322** scripts **$20,000**
Facebook V: Predicting Check Ins Machine Learning Software Engineer at **Facebook** Menlo Park, CA or Seattle, WA	**36 hours** **1235** teams **2644** scripts **Jobs**
Integer Sequence Learning 1, 2, 3, 4, 5, 7?!	**2 months** **110** teams **138** scripts **Knowledge**
Painter by Numbers Does every painter leave a fingerprint?	**3 months** **17** teams **61** scripts **Knowledge**
Shelter Animal Outcomes Help improve outcomes for shelter animals	**26 days** **1211** teams **795** scripts **Knowledge**

Telstra

Completed • Jobs • 974 teams

Telstra Network Disruptions

Wed 25 Nov 2015 – Mon 29 Feb 2016 (4 months ago)

Dashboard

Home
- Data
- Make a submission

Information
- Description
- Evaluation
- Rules
- Timeline

Forum

Leaderboard
- Public
- Private

Private Leaderboard

1. Mario Filho

Competition Details » Get the Data » Make a submission

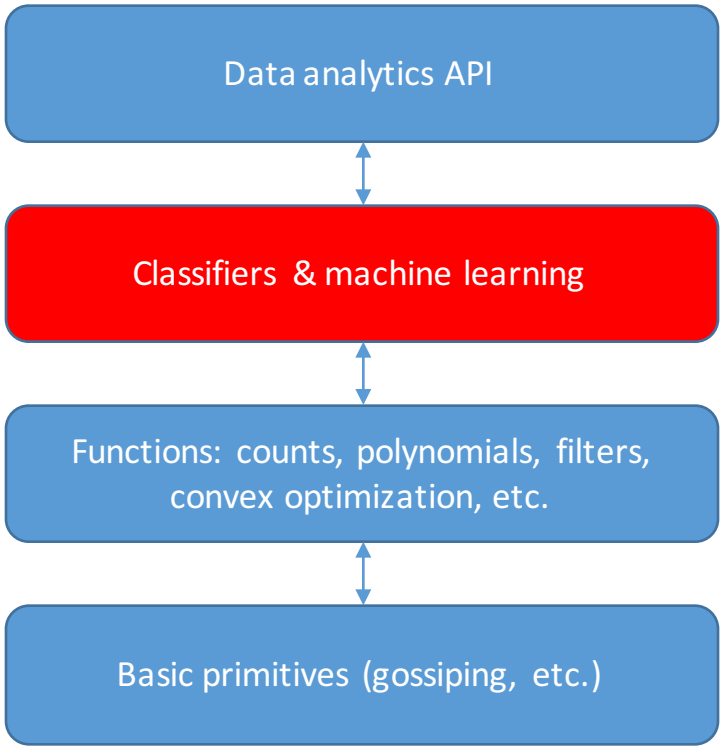
Predict service faults on Australia's largest telecommunications network

In their first recruiting competition, Telstra is challenging Kagglers to predict the severity of service disruptions on their network. Using a dataset of features from their service logs, you're tasked with predicting if a disruption is a momentary glitch or a total interruption of connectivity.

Telstra is on a journey to enhance the customer experience - ensuring everyone in the company is putting customers first. In terms of its expansive network, this means

kaggle™

Third layer


Data analytics API
Classifiers & machine learning
Functions: counts, polynomials, filters, convex optimization, etc.
Basic primitives (gossiping, etc.)

Moving towards data science applications

- Motivation:
  - Traditionally
    - Get the data out of the network fast and clean
    - Data scientists will process it in the cloud afterwards
  - Opportunities
    - The network is the tool - traffic, resolution, lifetime, don't hold two infrastructures
    - Unexpected additional benefit: privacy (!)

Use case: operational smart grid

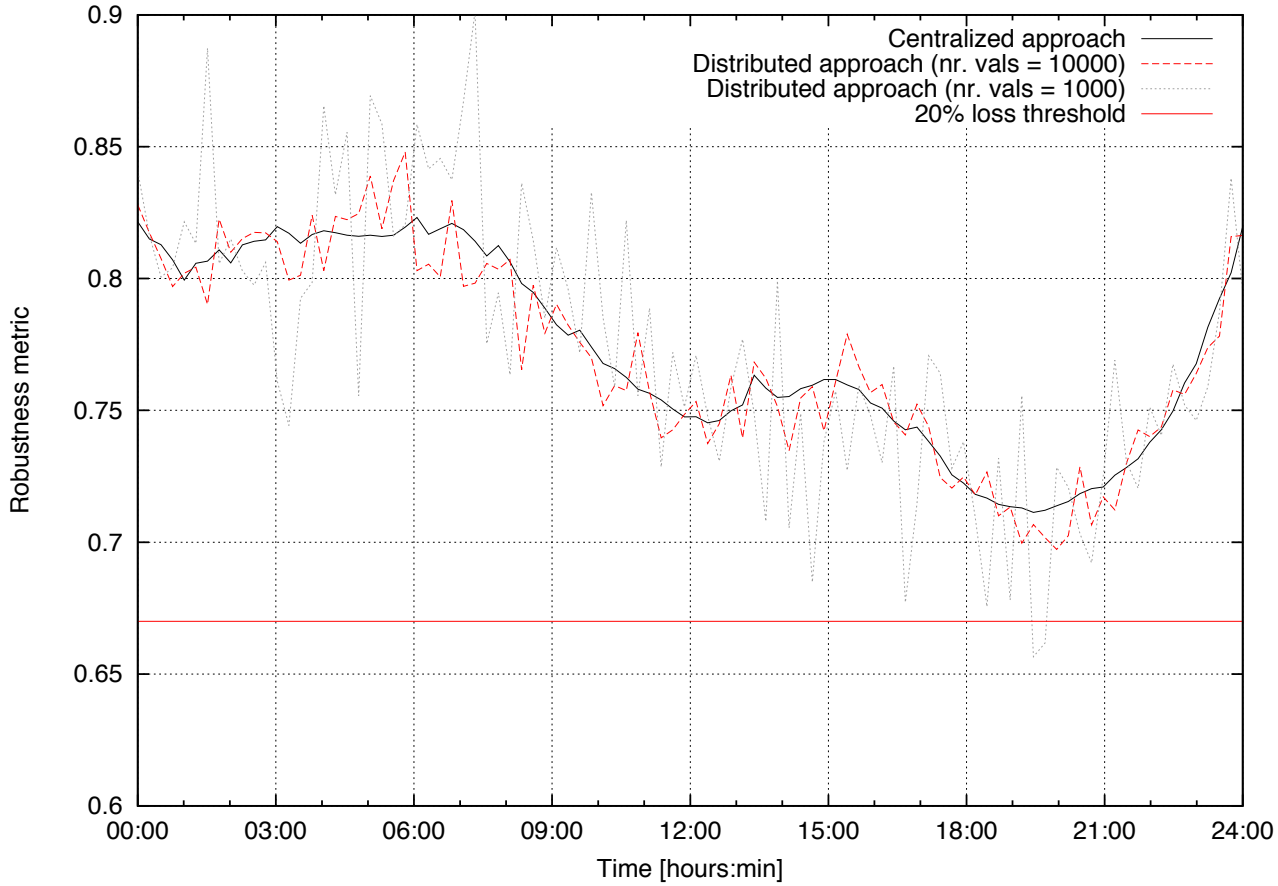
- Robustness metrics
  - Real time results needed
  - Highly dynamic network (continuously changing load)
  - Dynamic network topology
  - Tremendous size – country wide deployment
- Self-stabilizing epidemic algorithms
  - Achieve real time monitoring
  - Use insignificant resources
  - (Almost) constant behavior function of network scale

Koç, Yakup, et al. "A robustness metric for cascading failures by targeted attacks in power networks.", 2013

Example: computing robustness in the smart grid

$$R_{n,i} = -\sum_{j=1}^{d} \alpha_{ij} p_{ij} \log p_{ij}$$

$$R_{CF} = \frac{\sum_{i=1}^{N} R_{n,i} P_i}{\sum_{j=1}^{N} P_j}$$

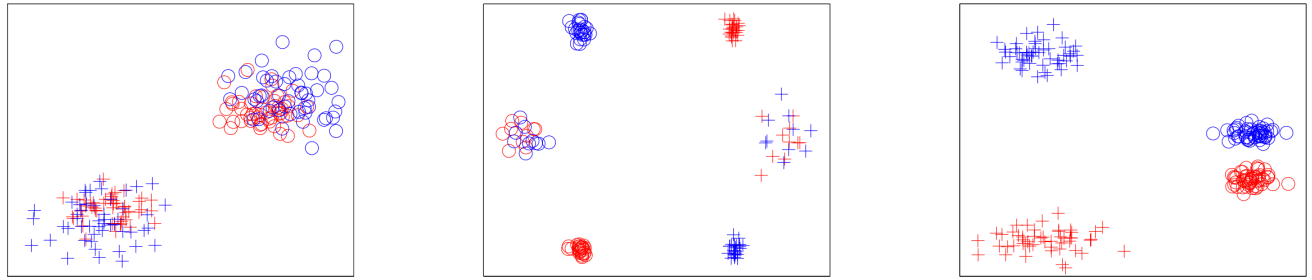


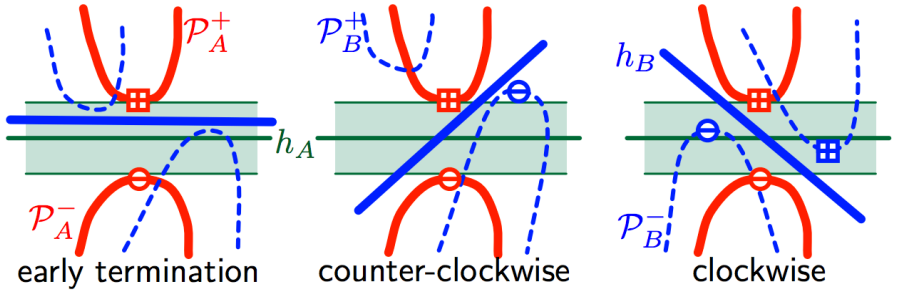
M. Warnier et al. - Distributed Monitoring for Prevention of Cascading Failures in Operational Power Grids

Ideas from different fields - data mining

- Problem:
  - Banks need to agree on the profile of the customer (defaulter)
  - Banks cannot exchange client data
- Idea:
  - What if data never leaves the nodes?
  - Gossiping may be used for computation of classifier parameters

Distributed classifiers

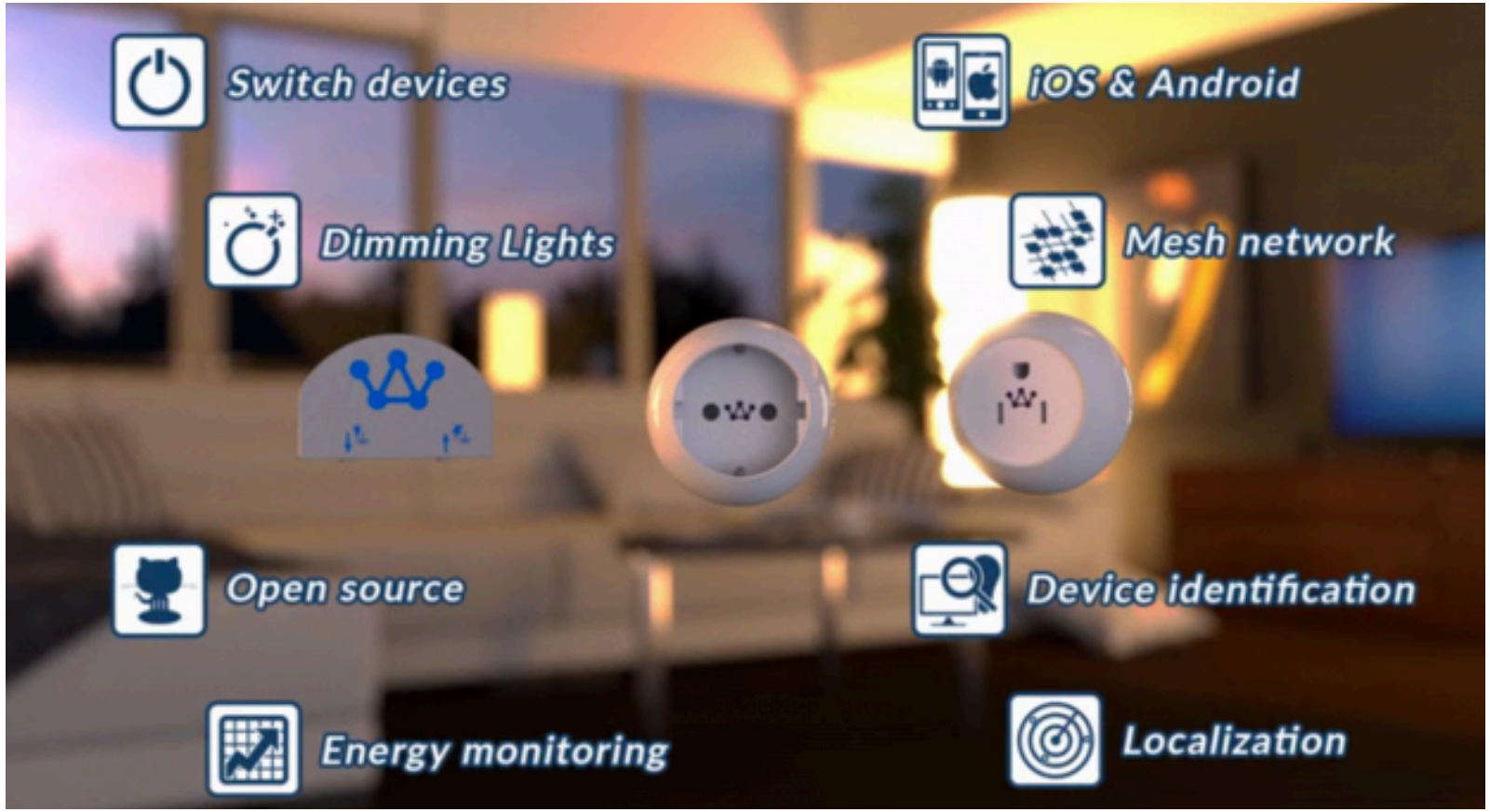




Method	DATA1		DATA2		DATA3	
	Acc	Cost	Acc	Cost	Acc	Cost
NAIVE	100%	1500	100%	1500	100%	1500
VOTING	98.75%	1500	100%	1500	50%	1500
RANDOM	100%	195	100%	195	99.76%	195
MAXMARG	**97.61%**	**14**	**100%**	**2**	**97.38%**	**38**
MEDIAN	**99.0%**	**36**	**100%**	**6**	**98.75%**	**29**

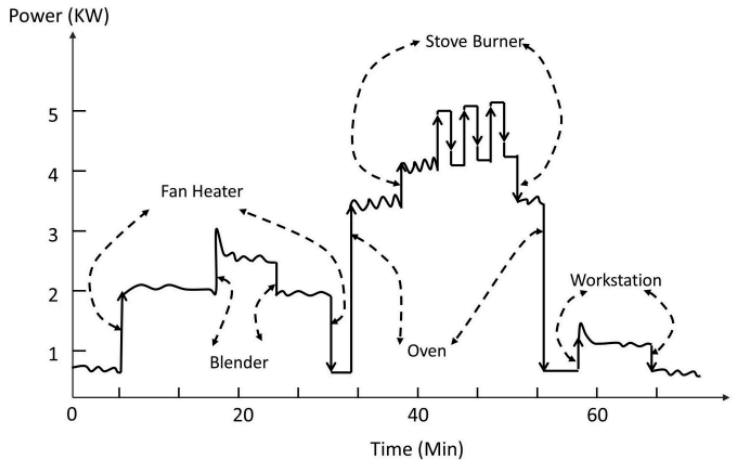
H. Daume - Protocols for Learning Classifiers on Distributed Data, 2012

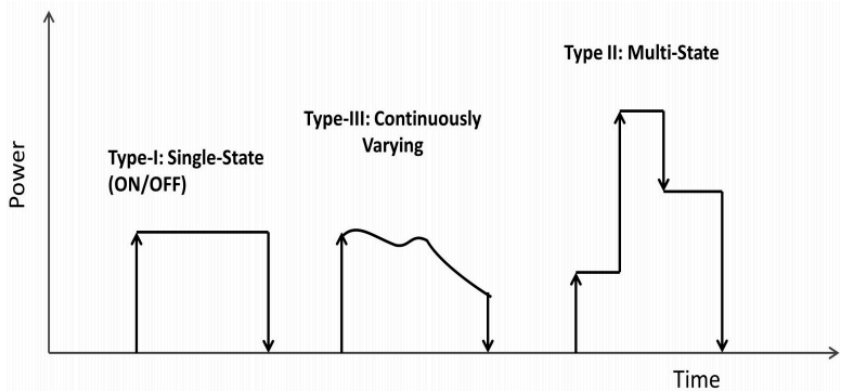
Preventive maintenance



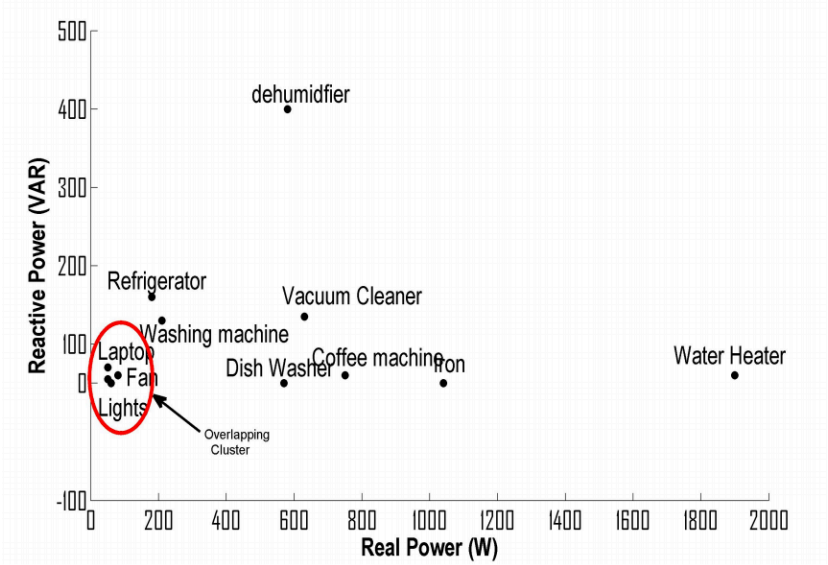
Machine learning & electricity networks

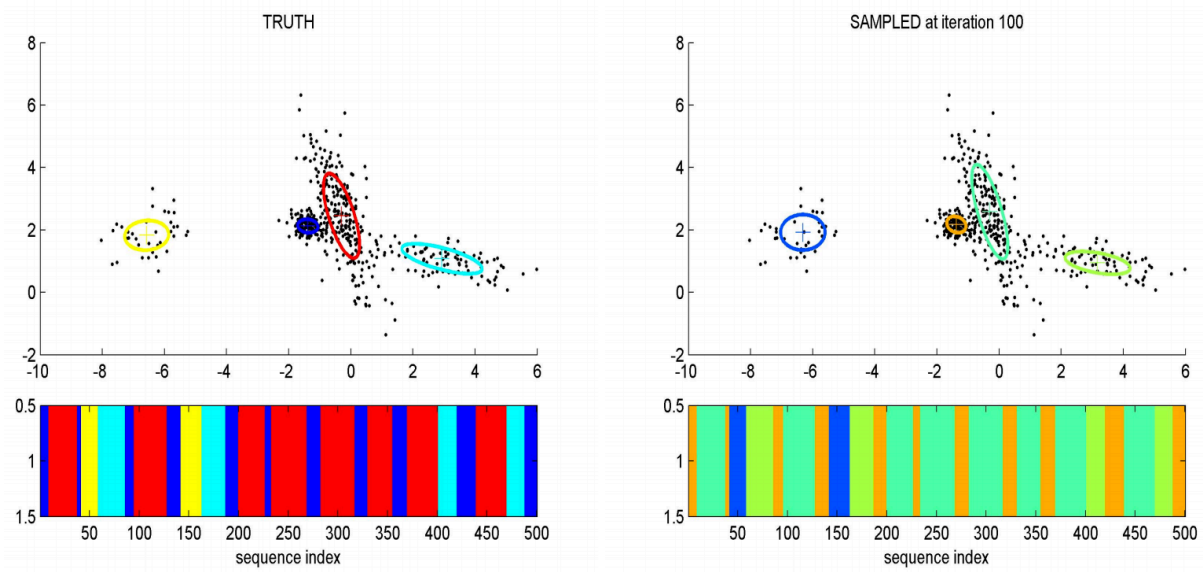
- Non intrusive load monitoring (NILM)
- Problem:
  - Detect which devices are active at given moments of time
- Solution:
  - Employ machine learning to estimate the number of devices and/or identify specific devices





NILM and machine learning





A. Zoha - Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey, 2012

Lecture outline

- *Processing in distributed networks*
  - *Aggregate computation*
  - *Fast gossiping algorithms*
- *Data science and the internet of things*
  - *Distributed classifiers*
  - *Machine learning in NILM*
- Dealing with failures in aggregates
- *Conclusions*

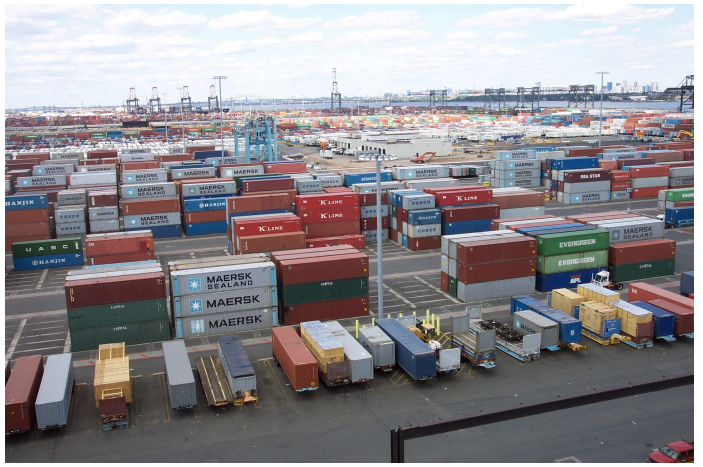
Use case: dynamic crowd monitoring

- Cardinality as value of interest
  - Number of people, density, flux, etc.
  - Infrastructure – congested or unavailable
  - Worst dynamics from network perspectives
- Use available technology
  - Smartphones in ad-hoc communication mode
  - Low resource consumption
  - High availability, robust estimates



More application examples

- More possible applications:
  - Detecting average speed, traffic jam size, number of cars in traffic
  - Monitoring number of persons in a public place, detecting stampedes
  - Optimizing resource allocation in transport and logistics scenarios



Even more applications

- Target networks where large data dissemination is not the main goal
- Example:
  - Large-scale networks
  - Running non-standard routing algorithms
  - User needs to access all data for:
    - Signaling exceptions
    - Estimating network parameters
    - Computing parameters for optimization algorithms

Robbert Van Renesse et al. - Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining, ACM transactions on computer systems (TOCS) 2003

Influence of failures

- Under ideal conditions (no failures):

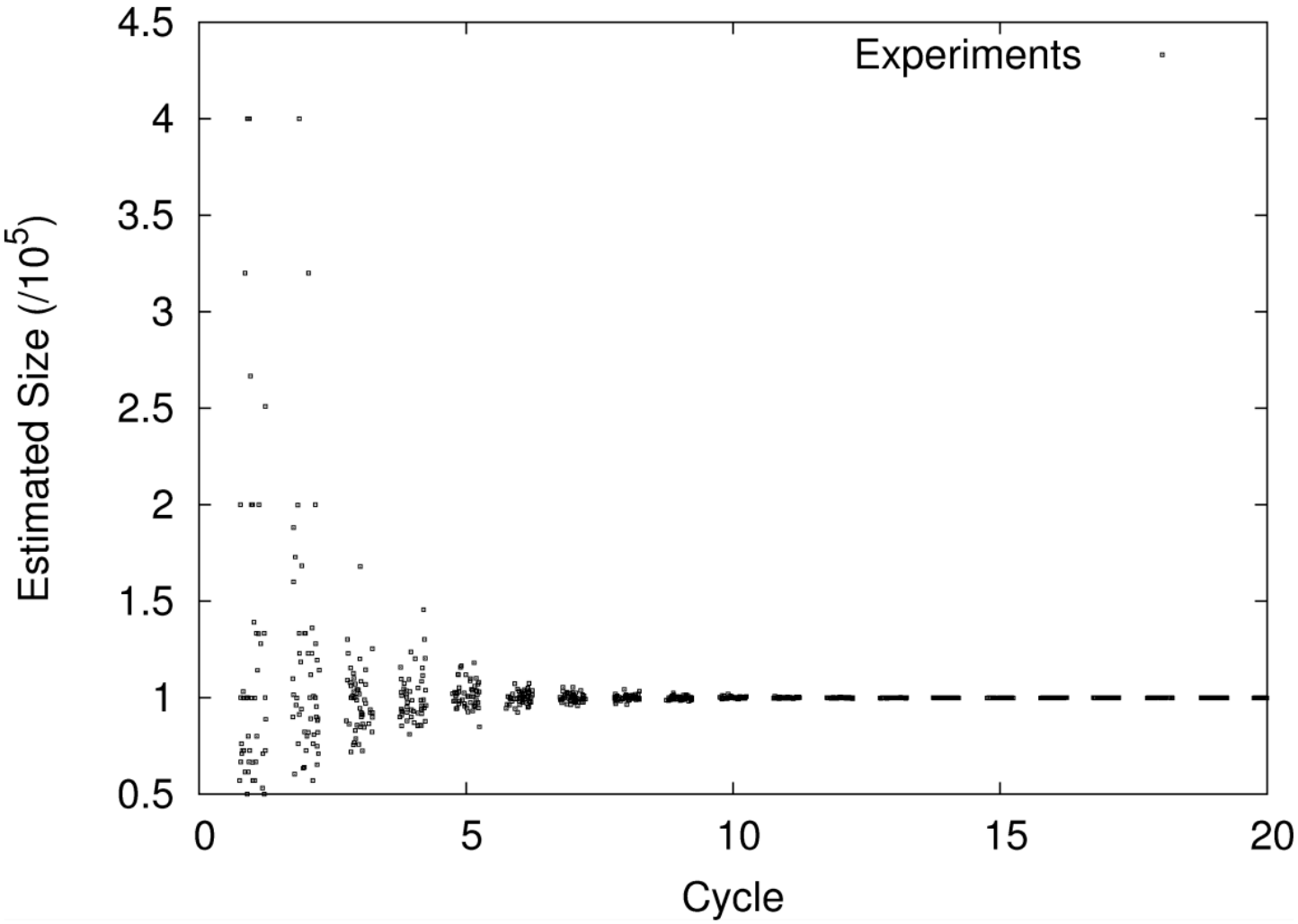
  $\Sigma_i m_{i,t-1} = \Sigma_i m_{i,t}$ – mass conservation

  $\Sigma_i \omega_{i,t-1} = \Sigma_i \omega_{i,t}$ – weight conservation

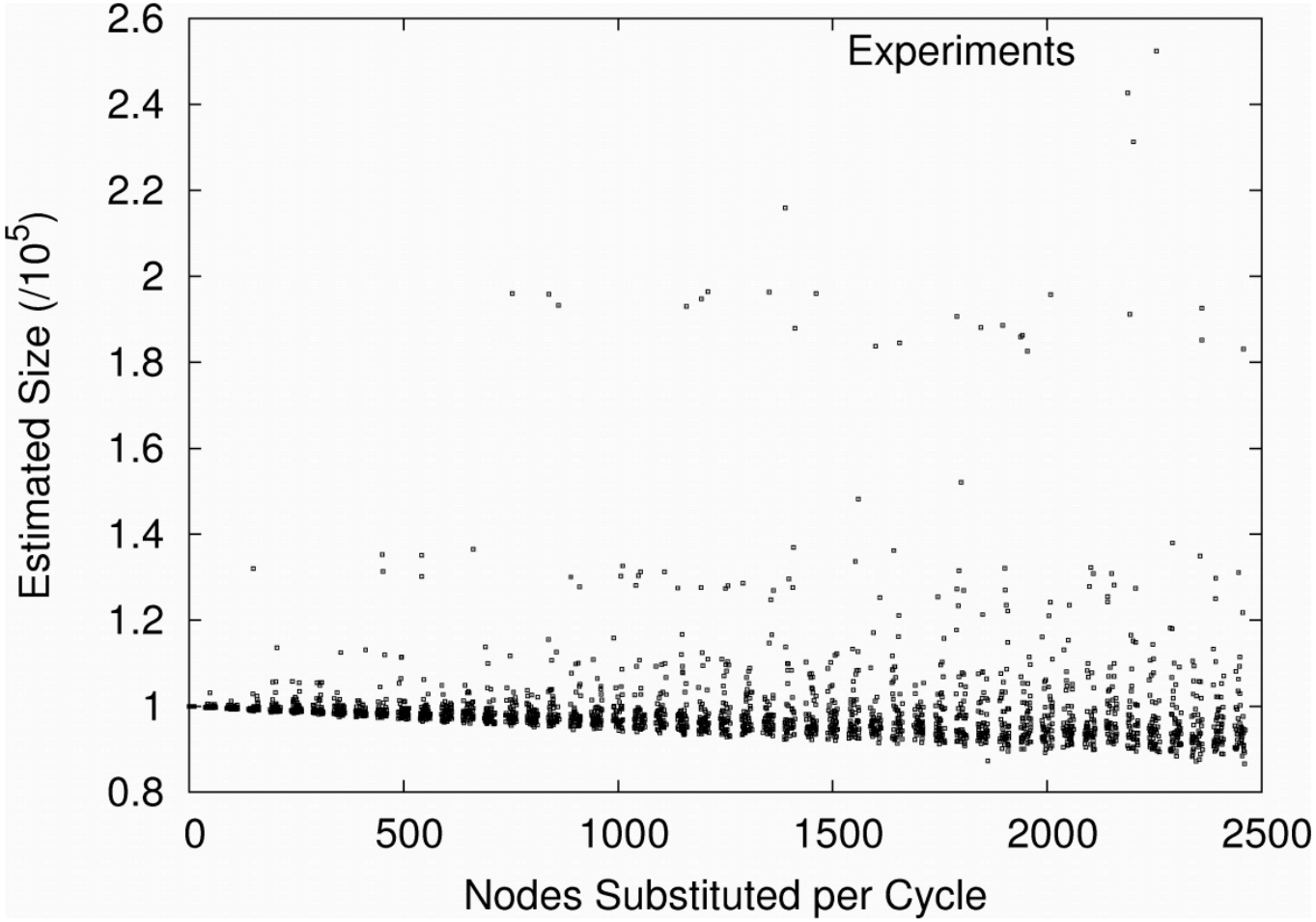
- In real environment:
  - Node failures: mass loss
  - Communication failures: mass loss
  - Churn: severe mass loss

Márk Jelasity: *Gossip-based aggregation in large dynamic networks*. ACM Trans. Computer Systems, 2005

Network size estimation with 50% nodes crashes



Network size estimation under node churn



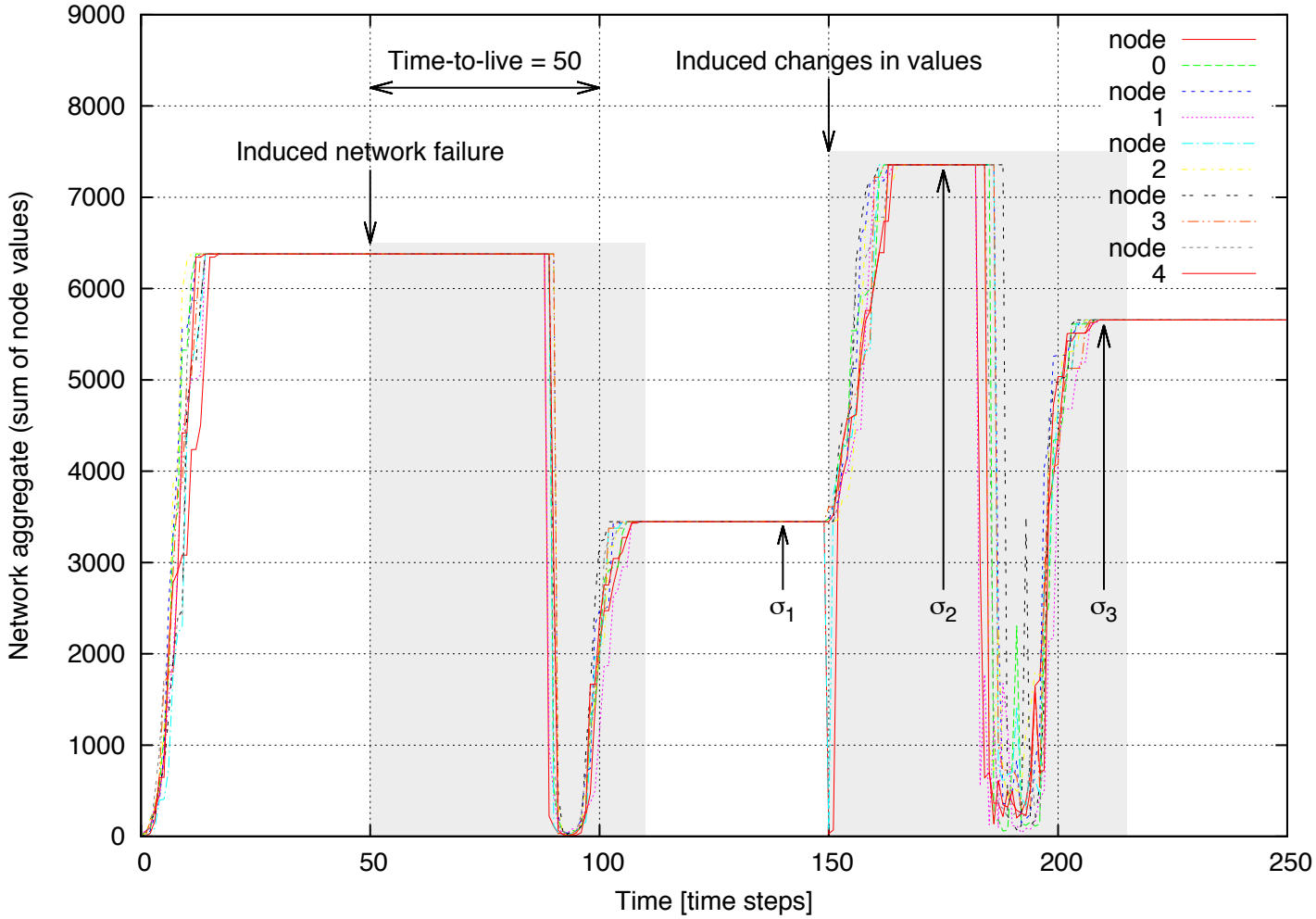
What to do about failures?

- Introduce the concept of synchronized epochs
  - Márk Jelasity: *Gossip-based aggregation in large dynamic networks*, 2005
- Run several algorithms in parallel
  - N. Bicocchi et al - Handling dynamics in diffusive aggregation schemes: An evaporative approach, 2010
- Introduce on purpose periodic failures (resets)
  - A. Pruteanu et al - ChurnDetect: A Gossip-Based Churn Estimator for Large-Scale Dynamic Networks, 2011

Counter-based self-stabilization

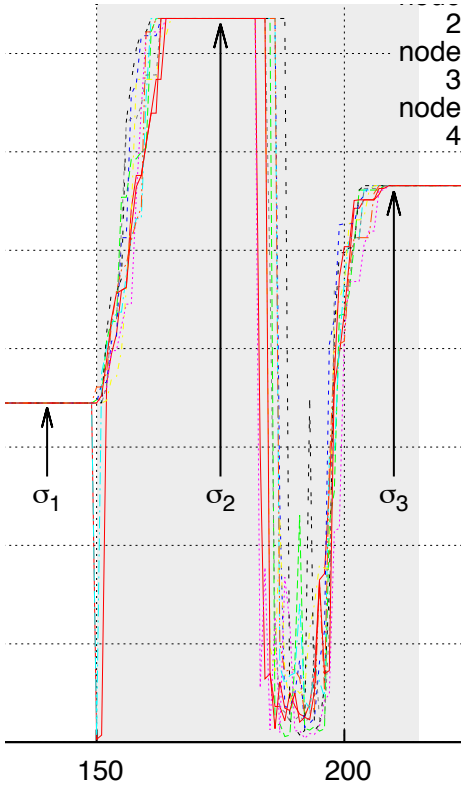
- Each value in the network has a time-to-live associated
  - Time-to-live is initialized with a maximum T
  - Time-to-live decreases with time (and hopcount)
  - When time-to-live reaches 0, values are deleted and replaced
- Effects
  - Values belonging to failed nodes will disappear – no tracking needed
  - Old values which got changed will also disappear with time-to-live
- Choosing the T constant
  - T should be as large as possible
    - Larger than the diameter of the graph
    - Messages should be allowed to span the whole network before expiring
  - T should be as small as possible
    - Size of T is linked directly to the “refresh” speed of the network

Final vector	*	*	*	*	*	*	*	*	*	*
TTL field	17	21	99	64	57	21	75	19	83	95

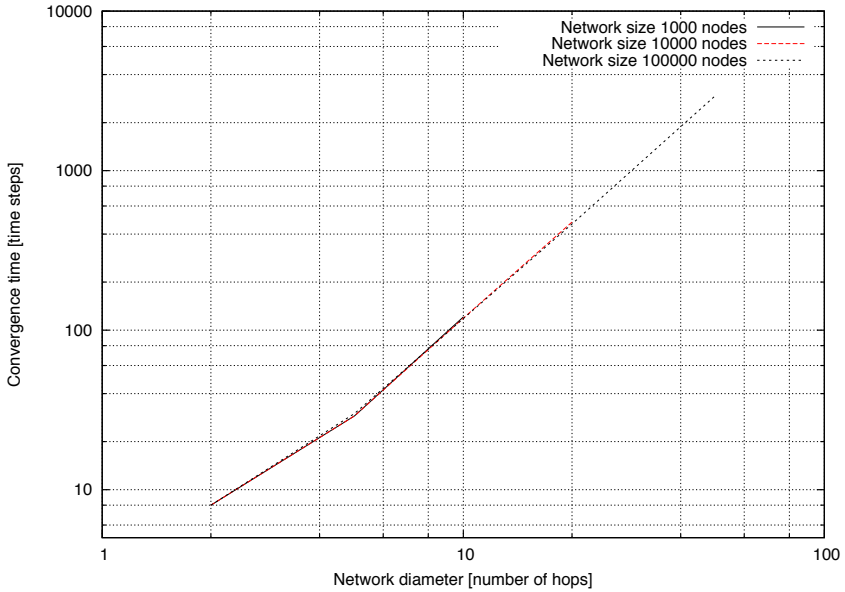

Time-to-live = 50
Induced changes in values
Induced network failure
node 0
node 1
node 2
node 3
node 4
σ1
σ2
σ3
Network aggregate (sum of node values)
Time [time steps]
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
50
100
150
200
250

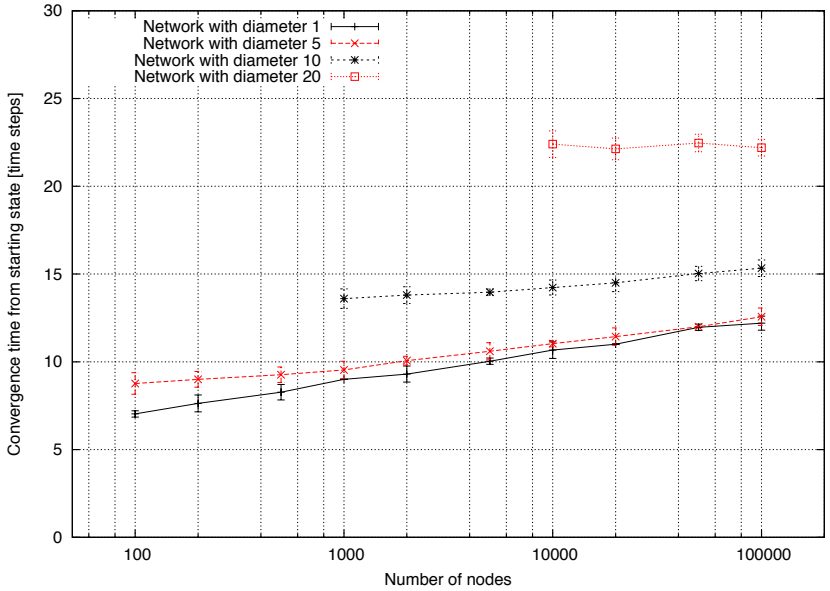
Propagation of fast/slow values

- σ2 – σ1 reflects the sum of the changed values in the network
- σ3 – σ2 reflects the sum of the replaced values in the network

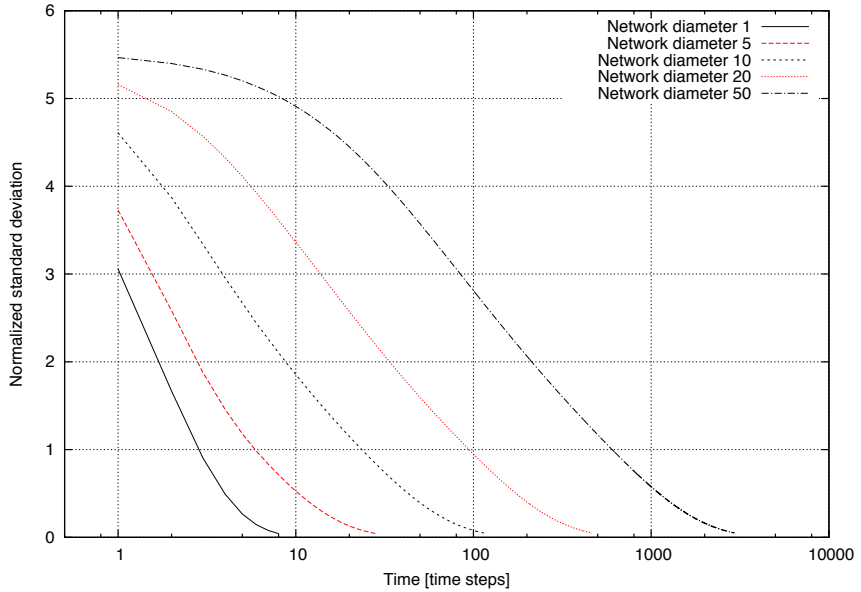


Simulation results





SFC convergence



AVG convergence

Conclusions

User

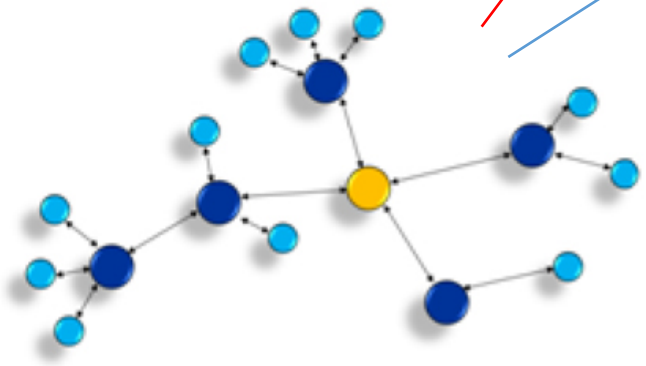
Data analytics API

Classifiers & machine learning

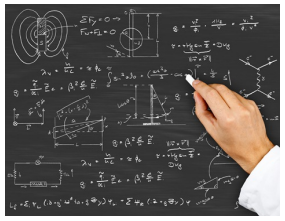
Functions: counts, polynomials, filters, convex optimization, etc.

Basic primitives (gossiping, etc.)

The “cloud”



Sensor network



Mathematical model

Data scientist

Conclusions

- System design to match the new data processing requirements
  - Reviewed a possible basic primitive: gossiping
  - Basic primitive allows complex functions and classifiers to be built directly in the network

- Additional information for this lecture:
  - Maarten van Steen: *Gossiping in Large-Scale Distributed Systems*
  - Ozalp Babaoglu: *Complex Systems*
  - Giovanna di Marzo Serugendo: *Adaptive Systems*