Gossip beyond broadcasting: gossip based aggregation

Márk Jelasity
Department of Computer Science
University of Bologna
Italy

Project funded by the Future and Emerging Technologies arm of the IST Programme

Outline

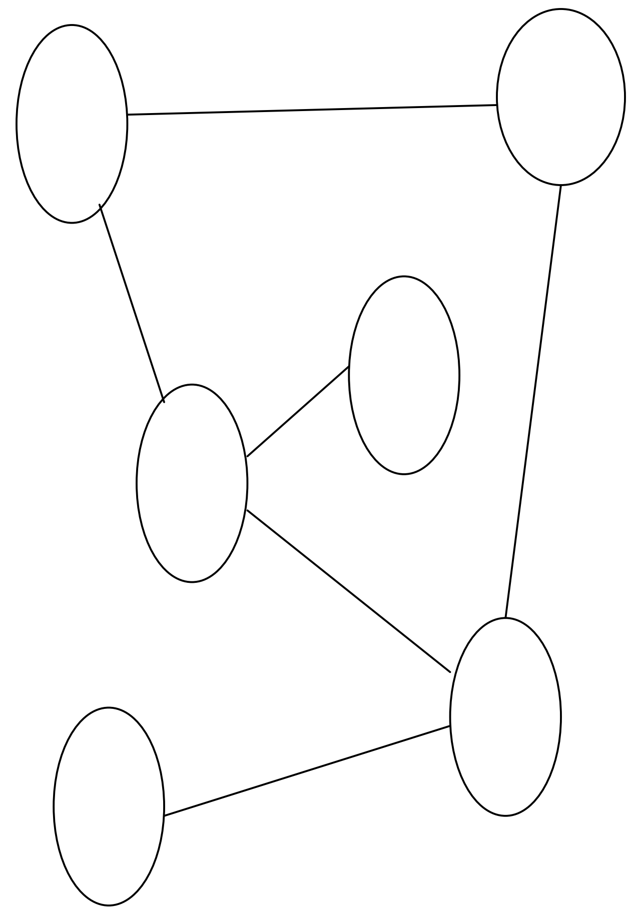
- The gossip based communication model
- An example protocol: average calculation
- Components: characterization and combination
- Future work

Outline

- The gossip based communication model
- An example protocol: average calculation
- Components: characterization and combination
- Future work

Basic Concepts

- Nodes
  - have state
  - perform computations
- Communication Topology
  - Neighbors (“knows about” relation)
  - Maintained by specific protocols



Push-Pull Gossip based Communication Model

```
// active thread
do forever
   wait(T time units)
   peer = selectRandomNeighbor()
   send state to peer
   receive peer.state from peer
   state = updateState(state,peer.state)

// passive thread
do forever
   (peer,peer.state) = waitMessage()
   send state to peer
   state = updateState(state,peer.state)
```

Gossip as Communication Model

- Gossip is a communication model, like eg the client-server model (as opposed to protocol)
- Its main properties are
  - proactive
  - democratic
  - potentially (depends on application)
    - scalable
    - robust
    - reliable

Gossip as Cellular Automaton

- Similarities:
  - Cycles (each T time units interval)
  - State updates based on neighborhood state
- Differences:
  - Only one neighbor is used in a cycle
  - Not `generational’ but `steady state’
  - Topology can be arbitrary
  - Topology can dynamically change over time

Expressivity

- It supports lots of very different protocols and design philosophies, not only information dissemination
  - epidemics
    - infomation dissemination, aggregation (max)
  - diffusion
    - aggregation (avg), load balancing
  - topology management
  - synchronization
  - etc.

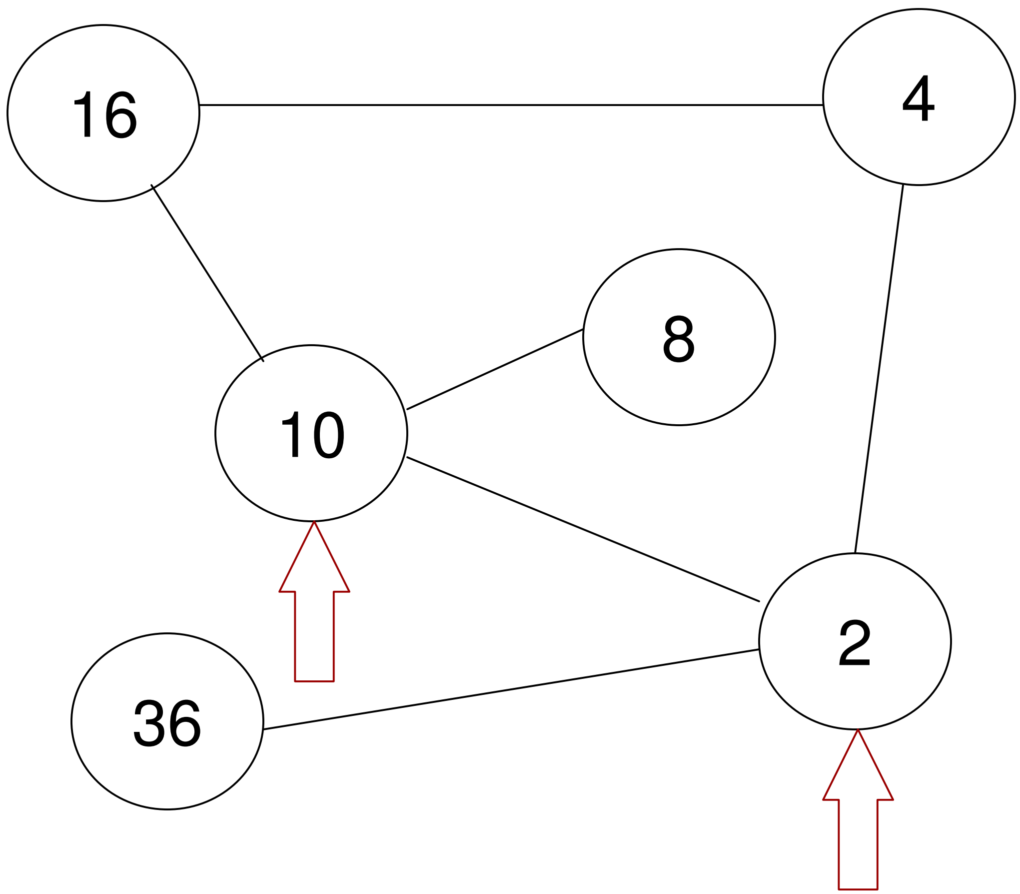
Outline

- The gossip based communication model
- An example protocol: average calculation
- Components: characterization and combination
- Future work

The Protocol

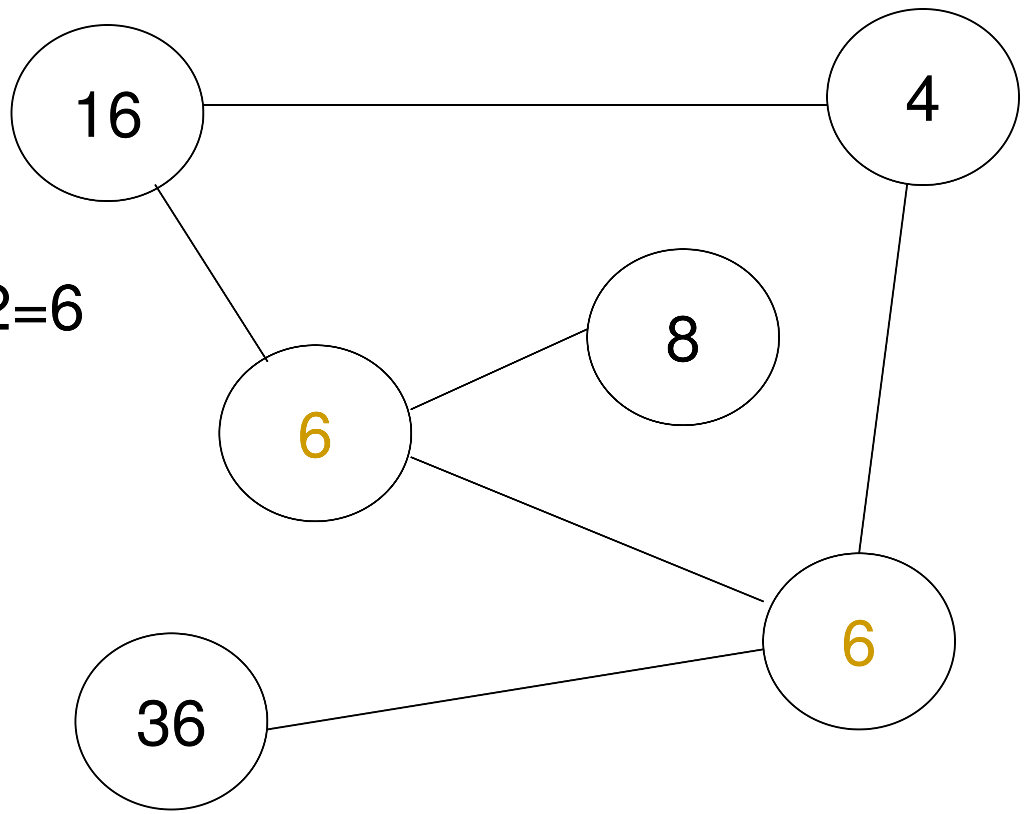
- We apply diffusion for calculating the average
- state: current approximation of average in the whole system
- `updateState(s1, s2):= (s1+s2)/2`
- Diffusion has lots of other applications (we will discuss them later) including
  - network size estimation
  - calculating variance (or any moments)
  - load balancing

Basic operation

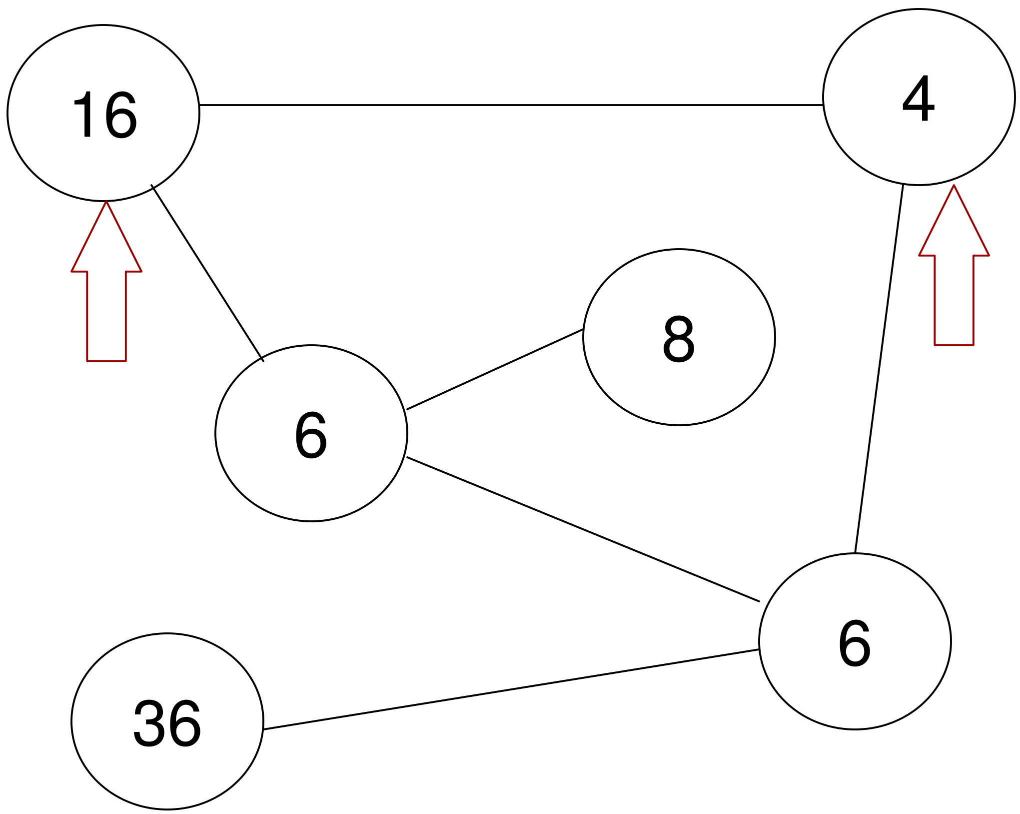


Basic operation

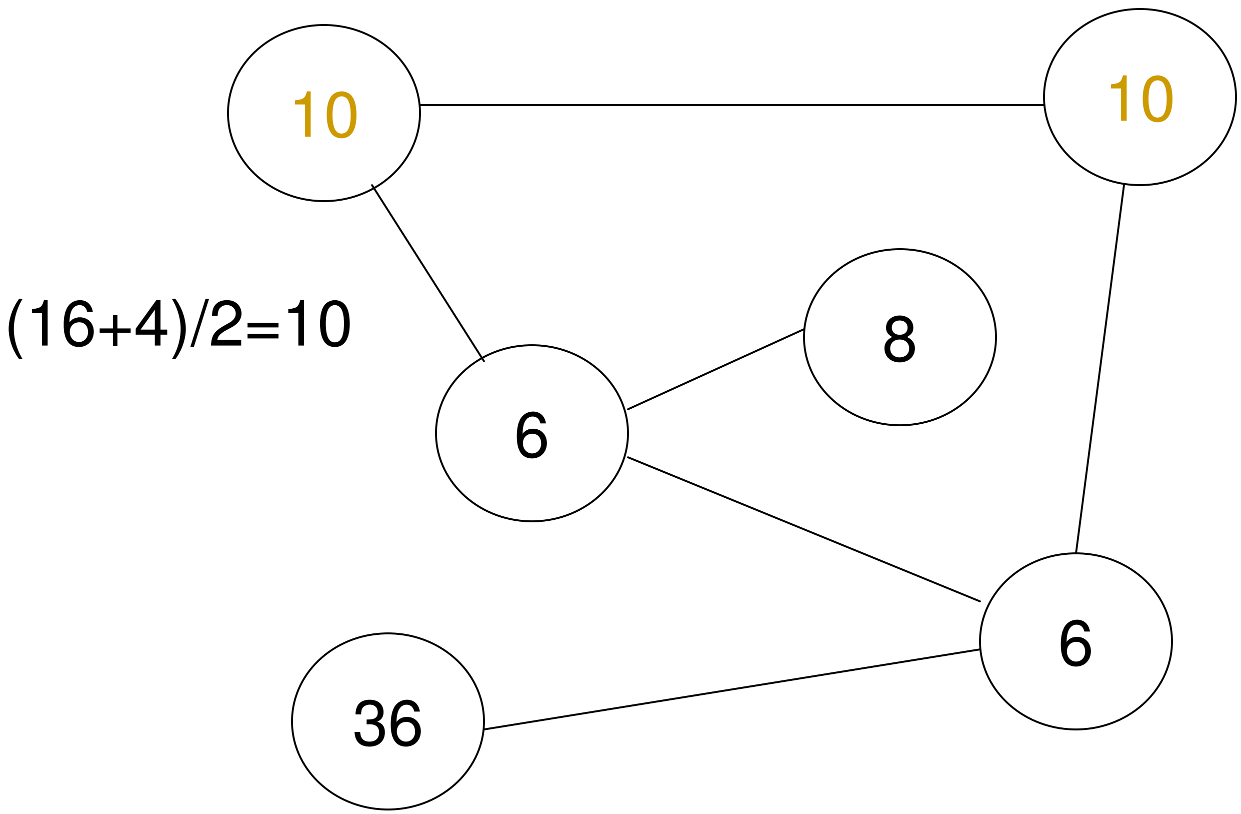
(10+2)/2=6



Basic operation



Basic operation



Applications

The averaging protocol can compute any means in the form

$$m = f^{-1}\left(\frac{f(a_1) + \ldots + f(a_n)}{n}\right)$$

These include the following means

average	$f(x) = x$
quadratic	$f(x) = x^2$
harmonic	$f(x) = \frac{1}{x}$
geometric	$f(x) = \ln x$

Applications

- The averaging protocol can compute any aggregate that can be expressed by a function of some means. For instance
  - variance: using avg. and avg. of squares
  - network size: 1/average if only one node holds 1 the rest 0
  - sum: network size times average
  - any n-th moment: using n-th power averages
  - coefficients of mathematical models like linear regression
  - statistical tests
  - etc...

Some Observations and Questions

- The procedure is convergent if the graph is connected
- Each node converges to the average of the original values
- How fast is convergence on different topologies?
- Which topology is optimal?
- What are the key features of a topology that determine the speed of convergence?
- What are the effects of node/link failure?

Some Answers

- On the fully connected topology convergence speed is exponential.
- On a random topology it is practically exponential.
- Node failure is not critical.
- Link failure is not critical.

Framework

A local protocol

```
do forever
   wait(getWaitingTime())
   nj = selectRandomNeighbor()
   // perform elementary aggregation step
   send a[i] to nj
   receive a[j] from nj
   a[i] = (a[i] + a[j])/2
```

A global translation

```
do N times
   (i, j) = getPair()
   // perform elementary aggregation step
   a[i] = a[j] = (a[i] + a[j])/2
```

The base theorem

IF:

- Each pair of values selected by the index pairs returned by each call to `getPair` are uncorrelated,
- the random variables $\varphi_1, \ldots, \varphi_N$ are identically distributed. Let φ denote a random variable with this common distribution,
- After (i, j) is returned by `getPair` the number of times i and j will be selected by the remaining calls to `getPair` has identical distribution.

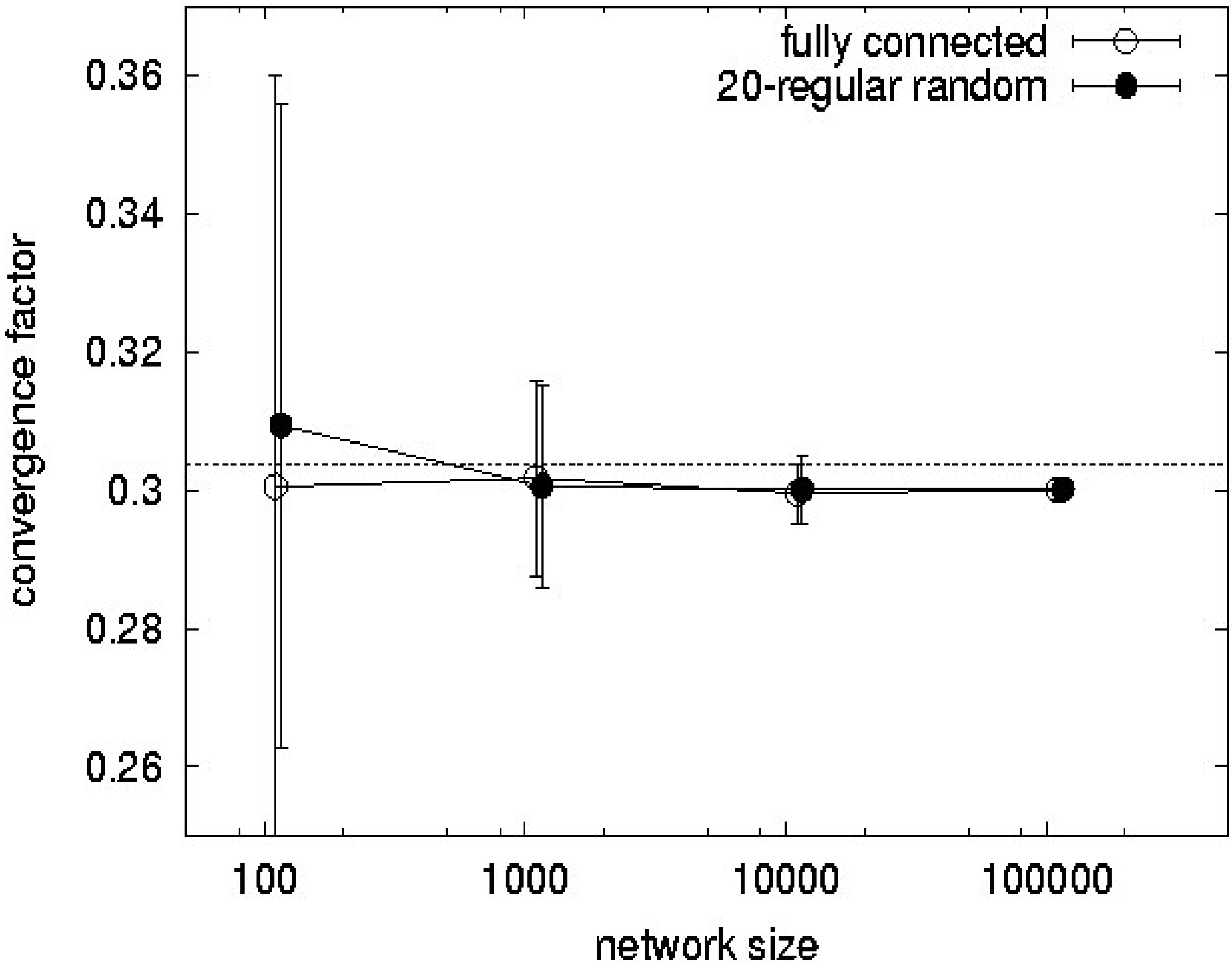
THEN:

$$E(\sigma_{i+1}^2) \approx E(2^{-\varphi})E(\sigma_i^2)$$

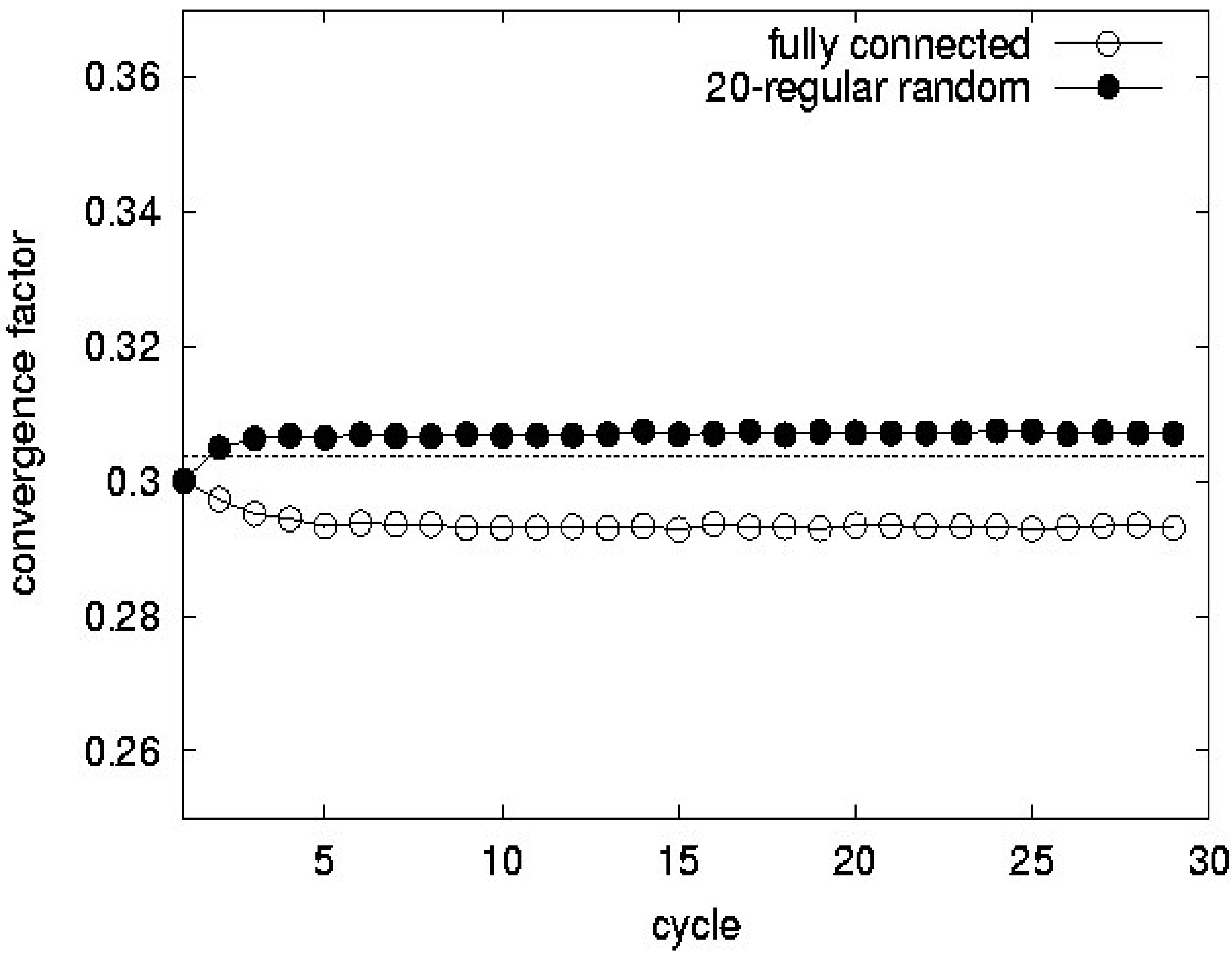
Convergence factor

- `getPair` defined by the local protocol when each node contacts a peer regularly
- A local corresponding protocol exists

$$P(\varphi = j) = \frac{1}{(j-1)!}e^{-1} \rightarrow E(2^{-\varphi}) = \frac{1}{2\sqrt{e}}$$



fully connected
20-regular random
convergence factor
0.36
0.34
0.32
0.3
0.28
0.26
100
1000
10000
100000
network size



fully connected
20-regular random
0.36
0.34
0.32
0.3
0.28
0.26
5
10
15
20
25
30
convergence factor
cycle

Link Failure

- Any given link fails with probability P_d
- The effect of this is only slowdown. In particular the rate can be bounded as follows

if $E(\sigma_{i+1}^2) = \rho E(\sigma_i^2)$ then

$$\rho_d \leq \left(\frac{1}{e}\right)^{1-P_d} = e^{P_d - 1}$$

Outline

- The gossip based communication model
- An example protocol: average calculation
- Components: characterization and combination
- Future work

Topology (lpbcast, newscast)

- State: neighbor list: constant sized list of peer adddresses
- `updateState(s1,s2)`: select new list randomly or based on some additional information
- `selectRandomNeighbor()` can be biased based on (partial) information on the state of the peers
- based on particular implementation details, generates different topologies

Broadcasting

- State: `infected' or not, ie received information or not
- `updateState(s1,s2)`: if received state is infected, set state to infected
- pull brodcast is also possible: efficient, not adaptive
- `selectRandomNeighbor()` can be biased based on (partial) information on the state of the peers
- not flooding, even with random neighbor selection

Aggregation

- State: current approximation of aggregate
- `updateState(s1,s2)`: elementary aggregation step, examples include
  - (s1+s2)/2 for average
  - $(s1s2)^{1/2}$ for geometric mean
  - max(s1,s2) for maximum
  - min(s1,s2) for minimum
- combining elementary aggregations more complex functions can be computed such as sum, set size, variance, etc.

Synchronization

- State: current epoch. The synchronization point is the beginning of each epoch.
- An epoch has a fixed time length, and incremented based on a local clock
- `updateState(s1,s2)`: the maximal epoch identifier: max(s1,s2)
- solves problems of clock drift, joinings, failures, message delays
- General building block to be used by all applications needing synchronization

Characterization of Components

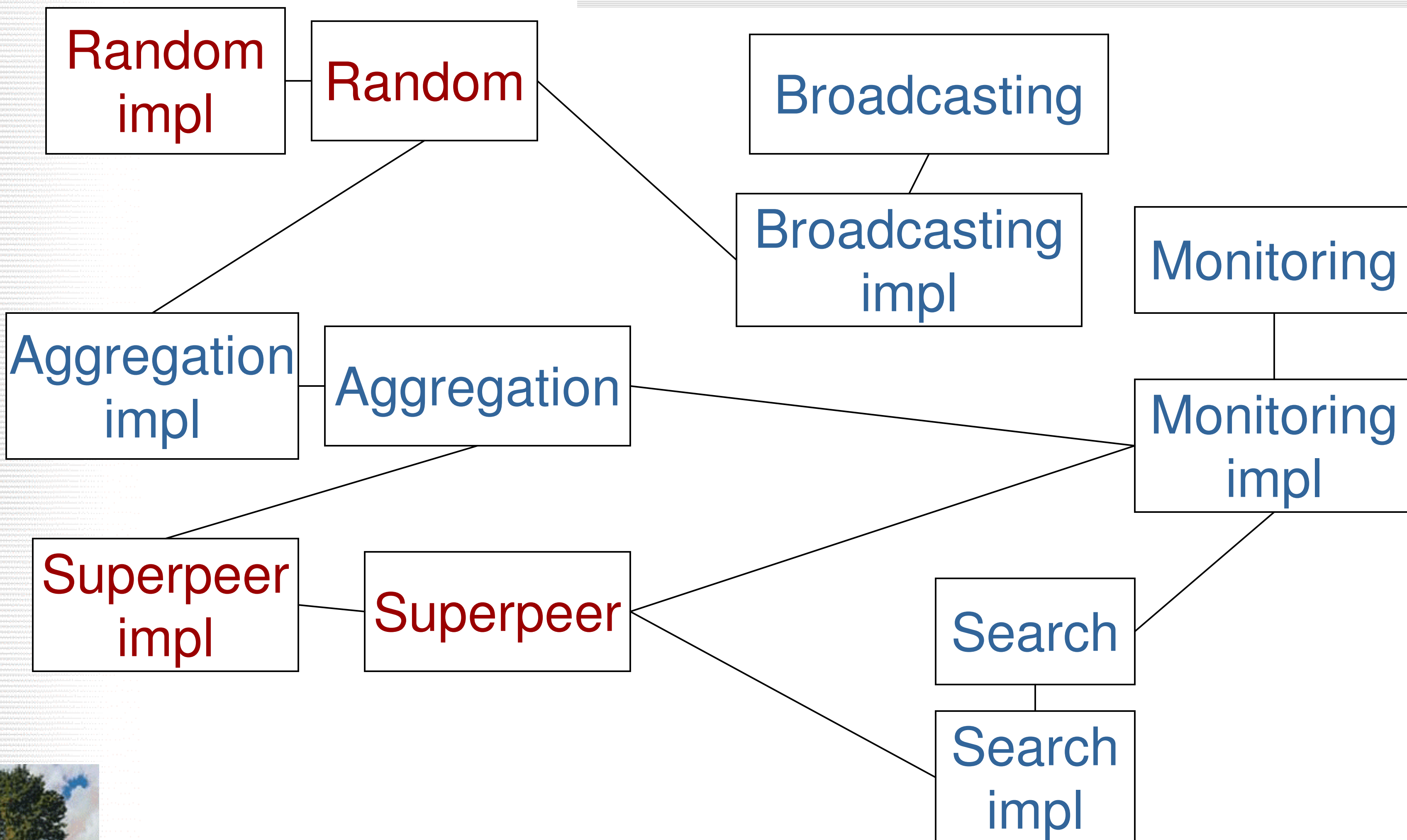
A component or building block is a protocol defining a self organizing system that provides a function through a standard interface. (Eg average calculation.)

- topology or function
- fast or slow
  - self organizing systems need time to converge or adapt; this process can be fast ot slow
  - slow protocols may rely on fast ones
- adaptive or convergent (static)
  - a self organizing system can converge to a stable state or it can react to the environment

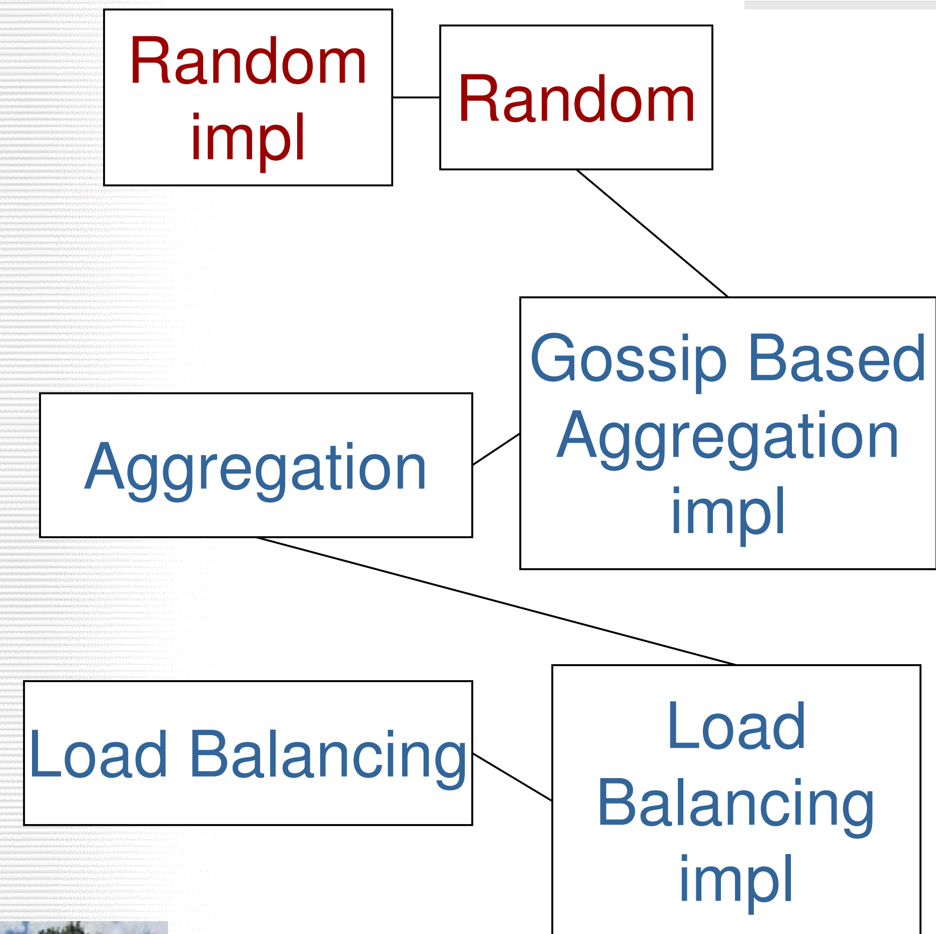
Combination of Components

- The goal is reusability to facilitate research (simpler problems) and development (off-the-shelf components)
- Some rules of thumb for combination
  - slow functions can utilize fast functions on the fly (topology, aggregation)
  - expensive functions can utilize cheap functions for optimization
- At the root there is always a topology (membership) protocol

Combination of components



Optimal Load Balancing



- Fast average calculation provides optimal load (fast, not optimal, cheap)
- Slow load balancing optimized based on the knowledge of optimal load (slow, optimal, expensive)

Outline

- The gossip based communication model
- An example protocol: average calculation
- Components: characterization and combination
- Future work

Future Work

- New functions
- More formal framework for composition
- Security
  - security of components: mutual auditing
  - security component?
- Simulation
  - realism vs. scalability: the study of the simplifying assumptions of peersim
- Visulization
- AHN, sensor networks?

Conclusions

- Gossip comunication model is a general paradigm
- Gossip based aggregation is shown to be
  - scalable (results independent of N)
  - fast
  - robust
- Possibility to combine functions
  - topology management
  - information processing/control