

Mean-field Method for Large Dynamic Gossip Networks

Rena Bakhshi

Theoretical Computer Science Section
Department of Computer Science,
Vrije Universiteit Amsterdam, The Netherlands

Dagstuhl seminar

July 4–9, 2010

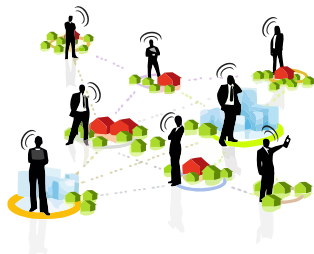
Gossip-based Protocols

- Inspired by gossip in social networks
- Nodes maintain local knowledge
- **S**imple, **S**calable, **S**ymmetric
- Periodic communication
- Randomly picked gossip partner
- Pairwise interaction
- Exchange of randomly selected data



Motivation

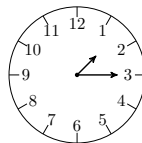
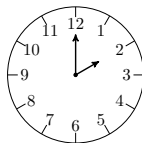
- Large-scale networks
- Mobile users
 - handhelds
 - laptops
 - GPS devices, etc.
- Interactions between users
 - sharing data
 - web services, etc.
- Problem with analysis of these networks:
 - state space explosion
 - model checking with 3–5 nodes
 - with abstraction up to 15 nodes



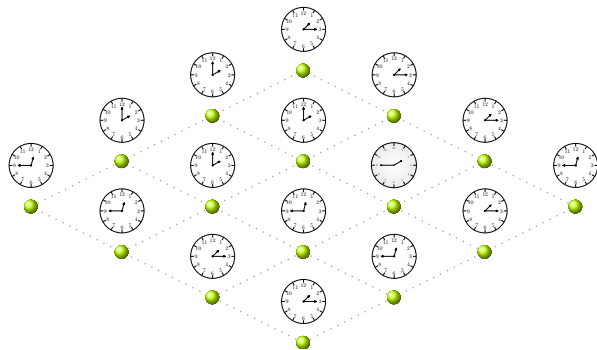
Scalable techniques for scalable protocols are needed!

Gossip Time Protocol

- Self-managing time synchronization protocol
- Synchronization of clocks in both time and frequency
- Nodes use peer-sampling service
- Presence of “time source”



Gossip Time Protocol



- Network of nodes, each equipped with a local clock
- Nodes periodically exchange time info in random fashion
- Node with the worse-quality time adopts the higher-quality time of its peer

States

Represented by a pair (g, h)

- gossip delay g
- hop count h (serves as metric for time precision)

Nodes

- Time source: $h = 0$
- Unsynchronized node: $h = \infty$
- Active node: $g = 0$

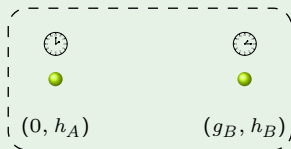
Pull

Active node $(0, h)$ picks random partner (g', h') :

- if $h' < h$ then $h := h' + 1$ and $g := G_{\max}$
- if $h' \geq h$ then $g := G_{\max}$ (sample rejected)

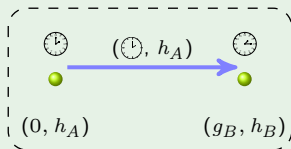
GTP (example)

Example



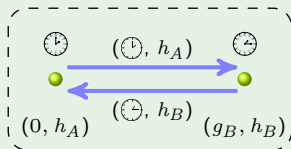
GTP (example)

Example



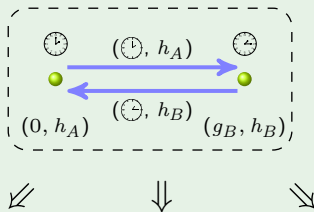
GTP (example)

Example



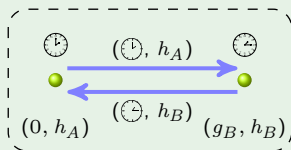
GTP (example)

Example

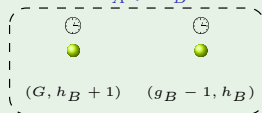


GTP (example)

Example

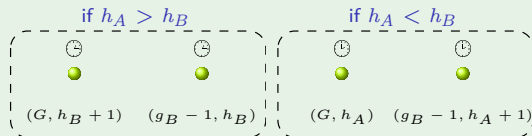
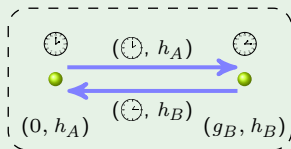


if $h_A > h_B$



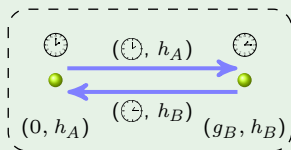
GTP (example)

Example

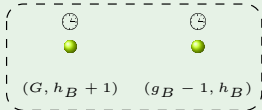


GTP (example)

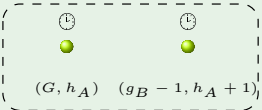
Example



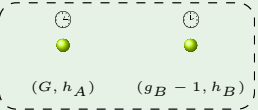
if $h_A > h_B$



if $h_A < h_B$

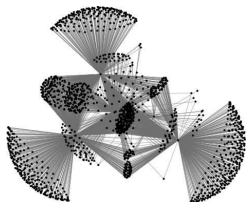


otherwise



Mean-field analysis

- Very large **probabilistic** systems ($N \rightarrow \infty$)
- **Deterministic** behaviour in the limit
- Fraction of nodes in possible states
 - fraction of nodes in states (g, h)



Example

	initial	step 1	step 2	...
time source	$\begin{pmatrix} 0.01 \\ 0 \\ 0 \\ \vdots \\ 0.99 \end{pmatrix}$	$\begin{pmatrix} 0.01 \\ 0.005 \\ 0 \\ \vdots \\ 0.985 \end{pmatrix}$	$\begin{pmatrix} 0.01 \\ 0.0099 \\ 0.0025 \\ \vdots \\ 0.9776 \end{pmatrix}$...
hop 1				
hop 2				
\vdots				
unsynchronized				

Transition matrix

- Probability to go from one state (column) to another (row)

$$\begin{pmatrix} \ddots & & & & \\ & P_{(g,h),(g',h')} & & & \\ & & \ddots & & \\ & & & \ddots & \\ \ddots & & & & \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ \mathbf{m}_{(g,h)} \\ \vdots \end{pmatrix}$$

Example (simplified calculation)

$$P_{(0,5),(G_{\max},2)} = \sum_{g=1}^{G_{\max}} \mathbf{m}_{(g,1)}$$

Measures of interest

What can we analyse?

Example

- how fast nodes get synchronised (GTP)
- number of replicas of datum (Shuffle)

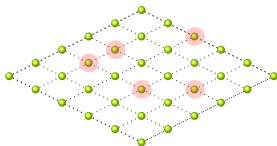


Figure: Replication

GTP Mean-field vs. Emulation¹

- $N = 1500$, 1 time source
- $G_{\max} = 25$

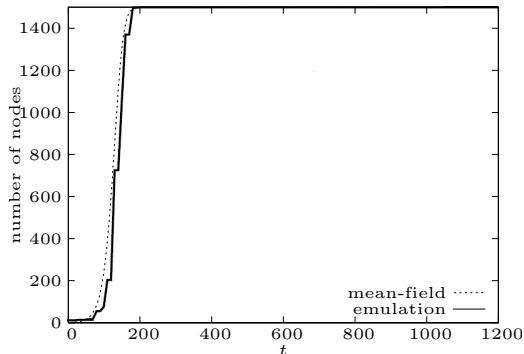


Figure: # nodes discovering the time source over time

¹Emulation results taken from [Iwanicki]

GTP Mean-field vs. Emulation¹

- $N = 1500$
- $G_{\max} = 25$

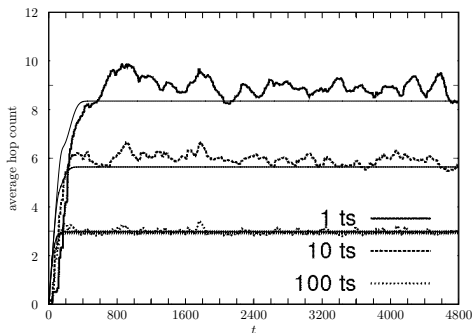


Figure: Average hop count for different # time sources

¹Emulation results taken from [Iwanicki]

GTP Further Measurements

- $G_{\max} = 25$ sec, $L = 25$ sec, $H = 15$

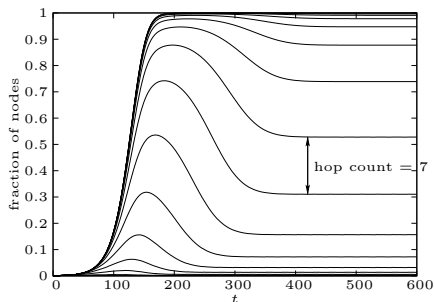


Figure: Hop count distribution over the first 10 min

Very large networks (Shuffle protocol)

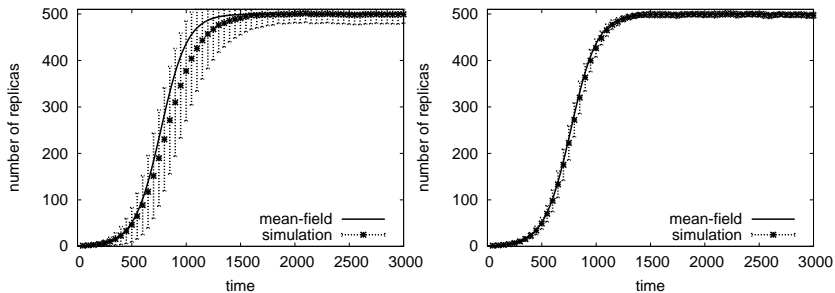


Figure: Replication for a network of 2500 and 25000 nodes.

Mean-field analysis: Summary

Assuming large network size: " $N \rightarrow \infty$ "

- Limit behaviour of complete system

Gossip protocols naturally fit for mean-field analysis

- Operate in large-scale, decentralized network
- Symmetrical behaviour of nodes
- Data exchange in a random fashion

Mean-field is hardly applied to communication networks:

- transition probabilities have to be derived by hand

Mean-field analysis manually

(Some) Transition probabilities

$$P_{s(l=0),s'}^N(\mathbf{m}) = \frac{\mathbf{m}_{(g',l',h'|g'>0)} \cdot N}{N-1} \cdot (1 - \text{noc}^N(\mathbf{m})) \\ + \frac{\mathbf{m}_{(0,l',h')}}{N-1} + \frac{\mathbf{m}_{(g',l',\infty|g'>0)} \cdot N}{N-1} \cdot \text{noc}^N(\mathbf{m}),$$

$$P_{s(l>0),s'}^N(\mathbf{m}) = \frac{\mathbf{m}_{(g',l',h'|g'>0)} \cdot N}{N-1} \cdot (1 - \text{noc}^N(\mathbf{m})) \\ + \frac{\mathbf{m}_{(0,l',h')}}{N-1} + \frac{\mathbf{m}_{(g',l',h'|g'>0,h' \geq h)}}{N-1} \cdot \text{noc}^N(\mathbf{m}).$$

$$s(\cdot) = (0, \cdot, h \mid h > 0), s' = (G(h), 0, h)$$

$$\text{noc}(\mathbf{m}) = \lim_{N \rightarrow \infty} \text{noc}^N(\mathbf{m}) = \lim_{N \rightarrow \infty} \left(\frac{N-3}{N-1} \right)^{\mathbf{m}_{(0,l,h)} \cdot N-1} = e^{-2 \cdot \mathbf{m}_{(0,l,h)}}.$$

$$P_{s(l=0),s'}^N(\mathbf{m}) = \mathbf{m}_{(g',l',h'|g'>0)} \cdot \frac{N}{N-1} \cdot \text{noc}^N(\mathbf{m}), \quad \forall h' < \infty \dots$$

$$s(\cdot) = (0, \cdot, h \mid h > 0), s' = (G(h'+1), L, h'+1)$$

Mean-field analysis manually

(Some) Transition probabilities

$$P_{s(l=0),s'}^N(\mathbf{m}) = \frac{\mathbf{m}_{(g',l',h'|g'>0)} \cdot N}{N-1} \cdot (1 - \text{noc}^N(\mathbf{m})) \\ + \frac{\mathbf{m}_{(0,l',h')}}{N-1} + \frac{\mathbf{m}_{(g',l',\infty|g'>0)} \cdot N}{N-1} \cdot \text{noc}^N(\mathbf{m}),$$

$$P_{s(l>0),s'}^N(\mathbf{m}) = \frac{\mathbf{m}_{(g',l',h'|g'>0)} \cdot N}{N-1} \cdot (1 - \text{noc}^N(\mathbf{m})) \\ + \frac{\mathbf{m}_{(0,l',h')}}{N-1} + \frac{\mathbf{m}_{(g',l',h'|g'>0,h' \geq h)} \cdot N}{N-1} \cdot \text{noc}^N(\mathbf{m}).$$

$$s(\cdot) = (0, \cdot, h \mid h > 0), s' = (G(h), 0, h)$$

$$\text{noc}(\mathbf{m}) = \lim_{N \rightarrow \infty} \text{noc}^N(\mathbf{m}) = \lim_{N \rightarrow \infty} \left(\frac{N-3}{N-1} \right)^{\mathbf{m}_{(0,l,h)} \cdot N-1}$$

$$P_{s(l=0),s'}^N(\mathbf{m}) = \mathbf{m}_{(g',l',h'|g'>0)} \cdot \frac{N}{N-1} \cdot \text{noc}^N(\mathbf{m})$$

$$s(\cdot) = (0, \cdot, h \mid h > 0), s' = (G(h), 0, h)$$

to automate

Mean-field automation

Automated computations

- transition matrix in the limit
- matrix-vector multiplication

Extension

- classes of states
 - group membership
 - physical node position
- ...

Measures of interest

- speed of information spread
- infection level in different locations

Example

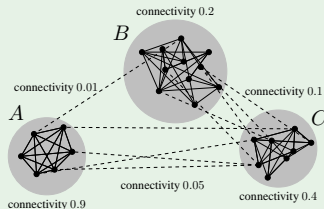
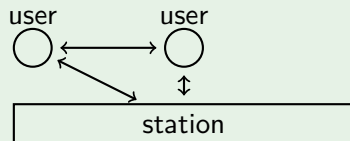


Figure: Clustered network

Concept of classes

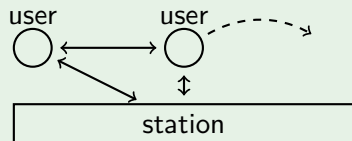
Example



- state space is divided into classes
- states can be in different classes
- nodes can move between different classes

Concept of classes

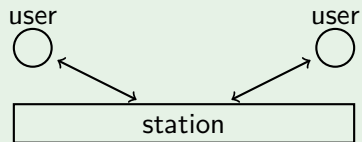
Example



- state space is divided into classes
- states can be in different classes
- nodes can move between different classes

Concept of classes

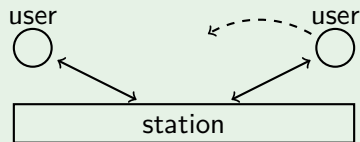
Example



- state space is divided into classes
- states can be in different classes
- nodes can move between different classes

Concept of classes

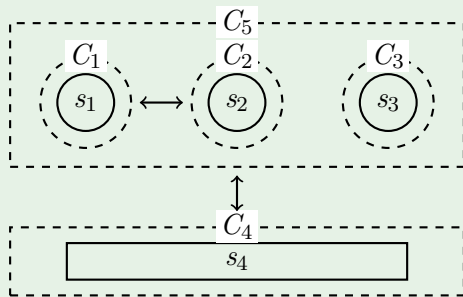
Example



- state space is divided into classes
- states can be in different classes
- nodes can move between different classes

Concept of classes

Example



- states s_2 and s_3 are positions of **mobile** user
- **station** s_4 communicates with all users
- **mobile** user communicates with **stationary** user s_1 only in s_2

Mean-field framework

What communication systems fit into the framework?

Class of systems

- uniform contact probability inside and between groups

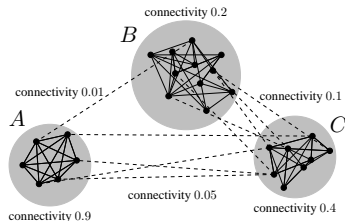
Case studies

- Uniform networks
- Mobile networks
- Dynamic networks

How to specify system in the framework?

Specification

- Relative cluster size
- Contact classes as user input



User input

```
object PredatorNetDef extends NetDef {  
  case class Water() extends Node { }  
  
  case class Prey() extends Node {  
    talk = { case b: Water => Array(1.04 -> Prey(), 1 -> Water())  
            case _ => Array() }  
    contacts = () => Array(0.5 -> ContactClass(Water())) }  
  
  case class Predator() extends Node {  
    talk = { case b: Prey => Array(1.5 -> Predator(), 0 -> Prey())  
            case b: Water => Array(0.95 -> Predator(), 1 -> Water())  
            case _ => Array() }  
  
    idle = () => Array(0.95 -> Predator())  
    collision = idle  
  
    contacts = () => Array(1 -> ContactClass(Prey(), Water())) }  
  
  initial ( 0.8 -> Water(), 0.14 -> Prey(), 0.06 -> Predator() ) }
```

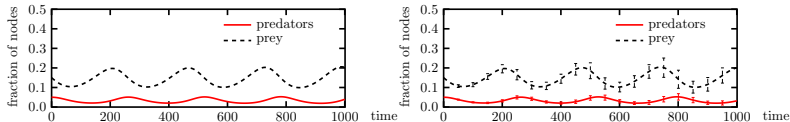


Figure: Mean-field vs. Monte-Carlo (500 predator and 1500 prey)

User input

```
object PredatorNetDef extends NetDef {  
  case class Water() extends Node { }  
  
  case class Prey() extends Node {  
    talk = { case b: Water => Array(1.04 -> Prey(), 1 -> Water())  
            case _ => Array() }  
    contacts = () => Array(0.5 -> ContactClass(Water())) }  
  
  case class Predator() extends Node {  
    talk = { case b: Prey => Array(1.5 -> Predator(), 0 -> Prey())  
            case b: Water => Array(0.95 -> Predator(), 1 -> Water())  
            case _ => Array() }  
  
    idle = () => Array(0.95 -> Predator())  
    collision = idle  
  
    contacts = () => Array(1 -> ContactClass(Prey(), Water())) }  
  
  initial ( 0.8 -> Water(), 0.14 -> Prey(), 0.06 -> Predator() ) }
```

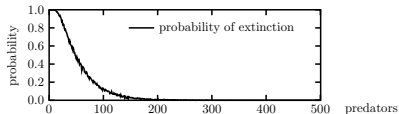


Figure: Probability of extinction within a period of 1000 time steps

Back to GTP

The power of programming language:

```
object ClockNetDef extends NetDef { ...  
  val allContacts = Array(  
    1.0 -> ContactClass({ for (g <- (0 to MAX_DELAY); t <- (0 to MAX_TIMEOUT);  
      h <- (0 to MAX_HOP)) yield Clock(g, t, h)}))  
  case class Clock(delay: Int, timeout: Int, hop: Int) extends Node {  
    talk = { case b: Clock => Array(1 -> update(this,b), 1 -> update(b,this))}  
    idle = () => Array(1 -> Clock(ndelay(delay), 0 max (timeout - 1), hop))  
    ...  
    contacts = () => if (delay == 0) allContacts else Array() }  
  def update(a: Clock, b: Clock): Clock = {  
    if(a.hop == 0) return Clock(ndelay(a.delay), a.timeout, a.hop);  
    if(a.timeout == 0) { ... } else { ... }  
    ...  
  }  
  initial (  
    (List(SOURCE -> Clock(12, MAX_TIMEOUT, 0)) :::  
      ((0 to 11) ++ (13 to MAX_DELAY)).map{  
        delay => (NOSOURCE -> Clock(delay, MAX_TIMEOUT, MAX_HOP)) }.toList ))}
```

Figure: GTP specification

Back to GTP

The power of programming language:

```
object ClockNetDef extends NetDef { ...  
  val allContacts = Array(  
    1.0 -> ContactClass({ for (g <- (0 to MAX_DELAY); t <- (0 to MAX_TIMEOUT);  
      h <- (0 to MAX_HOP)) yield Clock(g, t, h)})  
  case class Clock(delay: Int, timeout: Int, hop: Int) extends Node {  
    talk = { case b: Clock => Array(1 -> update(this,b), 1 -> update(b,this))}  
    idle = () => Array(1 -> Clock(ndelay(delay), 0 max (timeout - 1), hop))  
    ...  
    contacts = () => if (delay == 0) allContacts else Array() }  
  def update(a: Clock, b: Clock): Clock = {  
    if(a.hop == 0) return Clock(ndelay(a.delay), a.timeout, a.hop);  
    if(a.timeout == 0) { ... } else { ... }  
    ...  
  initial (  
    (List(SOURCE -> Clock(12, MAX_TIMEOUT, 0)) :::  
      ((0 to 11) ++ (13 to MAX_DELAY)).map{  
        delay => (NOSOURCE -> Clock(delay, MAX_TIMEOUT, MAX_HOP)) }.toList ))}
```

Figure: GTP specification

Back to GTP

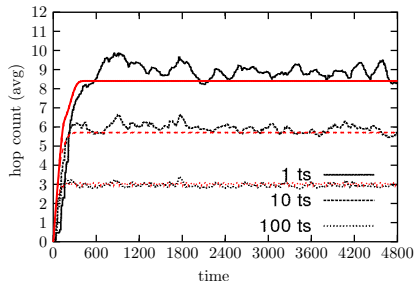
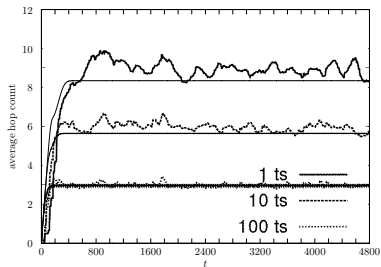


Figure: Mean-field manual and automated (GTP)

Conclusions

- tradeoff between accuracy and scalability
- our holy grail or too abstract
- framework automation
 - first step
 - more case studies
 - possible extensions

Thank you for your attention!

