Gossip-based Self-Organizing Overlay Networks

Márk Jelasity
PART I

A bird’s eye view of grassroots autonomic computing
Motivation

- IT infrastructure and applications became
  - large
  - heterogeneous
  - dynamic
  - complex
- as a result they are expensive to manage
- automation is necessary: self-management
  - self-organization
  - self-healing
  - self-stabilizing
  - self-*...
The industry is interested

- IBM: autonomic computing
- HP: adaptive enterprise
- Intel: proactive computing
- Microsoft: dynamic systems initiative
- Sun: N1
``autonomic'' comes from autonomic nervous system (and thus biology inspired)

Said nervous system

- operates subconsciously, without intervention (its autonomous)
- takes care of basic functions like heart rate, blood pressure, digestion, etc.

(This analogy does not cover some self-* functions like self-repair or self-protection)
``The autonomic computing architecture starts from the premise that implementing self-managing attributes involves an intelligent control loop’’

An architectural blueprint for autonomic computing, IBM

---

2004 sept 28, Szeged
the grassroots alternative

Motivation and tools

- biology
  - swarm intelligence
  - evolution
  - regeneration
  - pattern formation
  - embriogenesis
- social systems
- complex systems and networks
- P2P computing

2004 sept 28, Szeged
comparison of grassroots and top down self-*

- emergence
- extreme simplicity
- no explicit knowledge (representation)
- no explicit decisions
- no separation of manager/managed at any level
- low predictability (?)

ML, AI (even GOFAI):
- symbolic approach
- high (extreme?) complexity
- knowledge based, often explicit conceptual, rule based knowledge representation (policy)
- separation of managed entity and manager (homunculus)
- predictability (?)

2004 sept 28, Szeged
Some comments on grassroots self-management

Not a universal solution, probably appropriate in very large scale, highly dynamic, highly distributed systems

- Potentially robust and scalable
- Much easier to implement
- Therefore computer architectures do not become orders of magnitude more complex (like in autonomic computing)
- Potentially more efficient and effective
- Has its downsides too:
  - predictability and controllability
  - new design philosophy: not a gradual transition
European Activity

- Workshops
  - Engineering Self-Organising Systems (ESOA)
  - Self-*
- Recent related European 6th framework IST FET calls for IP proposals (EUR 8 million each)
  - complex systems
  - global computer
  - autonomic communication (starting Nov. 2004)
PART II

An example: overlay topology management
The Four Main Theses

1. **Topology** (network structure) is a key abstraction in distributed systems

2. **Gossip protocols** can maintain topologies in a scalable and robust manner

3. In particular, **random topology** and

4. **structured topologies** (ring, torus, binary tree).

2004 sept 28, Szeged
Topology as key abstraction (1)

- New field of complex networks: topology is key to explain robustness and function not only in computer science
  - spreading of epidemics and information (gossip): (human societies (sexual and other contact networks))
  - tolerance to random damage or attacks (food chains, chemical reactions (DNA), power grid)
  - search (strong and weak ties, 6 degrees of separation, small worlds)
Examples: WWW, Internet, P2P overlay networks (FastTrack, Gnutella, DHTs)

Topology is responsible for
- performance: function dependent: can be small diameter, randomness, hierarchy
- robustness: proximity, redundancy, no hotspots
Topology as key abstraction (3)
Other uses of Topology in CS

Sorting

Clustering

2004 sept 28, Szeged
Nodes: components of a large distributed system

Neighbor: if x ``knows about'' y, y is a neighbor of x

view: list of neighbors
Goal

- In some cases topology is given or is evolving out of control of any entity (WWW, Internet)
- In other cases it can be designed explicitly and built in a static manner (computer architectures, LAN)
- **Dynamic and large** distributed systems (overlay networks): efficient and robust protocols for topology maintenance are needed to support several different topologies (communication, sorting, clustering).
push pull gossip scheme: information exchange is symmetric

many applications
  broadcast: state is info update
  data aggregation: state is numeric value
  topology: state is the view

active thread

do once in each $T$ time units at a random time
  $p = \text{selectPeer}()$
  send state to $p$
  receive state$_p$ from $p$
  state = update(state$_p$)

passive thread

  $(p,\text{state}_p) = \text{waitMessage}()$
  send state to $p$
  state = update(state$_p$)
Gossip protocols for topology management (2)

active thread

do once in each $T$ time units at a random time

\[
\begin{align*}
  p &= \text{selectPeer()} \\
  \text{buffer} &= \text{merge}(<\text{view}, \{\text{myDescriptor}\}>)) \\
  &\text{send} \text{buffer} \text{to} \ p \\
  &\text{receive} \text{view}_p \text{from} \ p \\
  \text{buffer} &= \text{merge}(<\text{view}, \text{view}_p>) \\
  \text{view} &= \text{selectView}(<\text{buffer}>)
\end{align*}
\]

passive thread

\[
\begin{align*}
  (p,\text{view}_p) &= \text{waitMessage()} \\
  \text{buffer} &= \text{merge}(<\text{view}, \{\text{myDescriptor}\}>)) \\
  &\text{send} \text{view} \text{to} \ p \\
  &\text{buffer} = \text{merge}(<\text{view}, \text{view}_p>) \\
  \text{view} &= \text{selectView}(<\text{buffer}>)
\end{align*}
\]
merge: implements set operation merge

myDescriptor: contains address and application dependent information

selectPeer: uses the actual view to select a peer to contact

selectView: based on the information contained in the descriptors constructs the next view
Gossip protocols for topology management (4)
Gossip protocols for topology management (4)

SelectPeer

2004 sept 28, Szeged
Gossip protocols for topology management (4)

Exchange of views

2004 sept 28, Szeged
Gossip protocols for topology management (4)

Both sides apply `selectView` thereby redefining topology
Gossip protocols for topology management (5)

- Notion of **cycle**: any time interval when on average a node performs one view exchange: \( T/2 \) time units

- `selectPeer` and `selectView` are crucial: different implementations result in different topologies
  - `newscast`: random networks
  - `T-Man`: structured networks
Newscast: a gossip protocol for random topologies (1)

- descriptor: contains timestamp of creating the descriptor
- selectPeer: randomly selects a neighbor from the view
- selectView: fills the view with the freshest descriptors. New information gradually replaces old information: high robustness and adaptivity.
Newscast: a gossip protocol for random topologies (2)

- simulation results: N=100 000, view size c=30

- scenarios:
  - growing: start with no nodes, add 5000 nodes each cycle (connecting them to first node only)
  - lattice: start with regular linear lattice (each node connected to k nearest nodes)
  - random: start with random topology
  - dynamic: random, but between cycle 20 and 40 replace 10% of nodes each cycle

2004 sept 28, Szeged
Newscast: a gossip protocol for random topologies (3)

- robustness to failure simulation results: N=100,000, view size c=20, 40 and 80
- allowed the newscast topology to converge
- size of largest connected cluster after removing a given proportion of random nodes
Newscast: a gossip protocol for random topologies (4)

- Newscast generates a closely random topology irrespective of starting conditions.
- The convergence is fast, a few cycles.
- Scalable (though not demonstrated here).
- Robust to node dynamism (churn).
- Robust to node failure.
- A reliable source of random nodes from the whole system, a service many applications need, including T-Man.
T-Man: a gossip protocol for structured topologies

- **Ranking function**: \( R(y, \{x_1, \ldots, x_m\}) \) ranks a set of nodes \( \{x_1, \ldots, x_m\} \) with respect to a base node \( y \)

- For a fixed base node \( y \) and node set \( S = \{x_1, \ldots, x_m\} \) the ranking function defines an ordering relation \( (\leq) \) over \( S \)

- if \( d \) is a distance function over the set of all nodes \( \{x_1, \ldots, x_N\} \) then it can be used to define a ranking function (backwards not true)
T-Man: a gossip protocol for structured topologies

- **Problem 1 (construction)** Construct the view of node $x$ such that they contain the highest ranked elements when ranking is applied over the whole node set with $x$ as base node.

- **Problem 2 (embedding)** Construct the view of node $x$ such that it contains at least $k$ elements which have a highest rank when ranking over the whole nodes set with $x$ as base node.

2004 sept 28, Szeged
T-Man: a gossip protocol for structured topologies

- descriptor: contains profile of the node (real number, 2d vector, etc)
- selectPeer: Ranks view and selects a neighbor from the first half according to ranking
- selectView: fills the view with the lowest rank descriptors
- View initialization: random initial nodes are desirable: newscast is used
Example 1: ring and line Let the nodes be real numbers. Let the ranking function be defined by the distance \( d(a,b) = |a-b| \). For the ring, apply periodic boundary conditions, assuming nodes are from \([0,N]\).

Example 2: mesh and torus Let the nodes be two dimensional real vectors. Similarly to the ring, let the Manhattan distance define the topology.
Example 3: binary tree
Let the nodes be binary strings of length m. Let the ranking function be defined by the distance given by the hop count in the binary undirected rooted tree as illustrated.

```
001
 /    /
010   011
 /  /  /  /
100 101 110 111
```
Example 4: sorting Let $\leq$ be a total ordering over the nodes. Let the ranking function apply a distance function consistent with $\leq$ separately to those $<$ and $>$ than the base node, and merge the ranked two subsets.

For example $R(10,\{1,2,4,100,300\})$ could return $(4,100,2,300,1)$. No distance function over the set of nodes generates this ranking function!
T-Man: a gossip protocol for structured topologies

- Irregular initialization (nodes are random, with possible gaps in the distribution): clustering and sorting can be achieved.
  - Using the line ranking function: clustering
  - Using the ordering ranking function: sorting
- Regular initialization (nodes are 1,...,N for ring, similarly for other topologies): rest of the results about this case.
  - N=2^{14}, 2^{17}, 2^{20}; c=20, 40, 80; ranking function is ring, 2-d torus and binary tree.
Observed exponential behavior can be explained using an intuitive approximate model:

1) the view after the first contact is the first (lowest ranking) half of a random sample twice as large as the view

2) the extension of this idea is that after the ith contact the view is the first c elements of $c2^i$ random samples

this simple model predicts exponentially increasing probability of embedding good links, in particular, it doubles in each cycle (after each contact).
T-Man: a gossip protocol for structured topologies

We predict the time of embedding by calculating the time for reaching probability one for the inclusion of the target links, according to the simple model which gives the formula $i < \log_2(N-1) - \log_2 c$. This yealds the following predictions:

<table>
<thead>
<tr>
<th></th>
<th>C=20</th>
<th>C=40</th>
<th>C=80</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N=2^{14}$</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>$N=2^{17}$</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>$N=2^{20}$</td>
<td>16</td>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>
(c) binary tree

- convergence factor
- cycles
T-Man: a gossip protocol for structured topologies

- End phase: due to deviations from the approximate model, a few nodes have to find their place, using the already converged structure, to reach perfection.
- To improve end phase we apply:
  - **contact balancing**: in cycle i find peer that communicated less than i times and reject connections if communicated i times already.
  - **endgame**: more aggressive peer selection: select the closest peer (instead of random from first half).
(d) $N = 2^{14}$

- Binary tree, $c=20$ (red)
- Binary tree, $c=40$ (red)
- Binary tree, $c=80$ (red)
- Ring, $c=20$ (green)
- Ring, $c=40$ (green)
- Ring, $c=80$ (green)
- Torus, $c=20$ (cyan)
- Torus, $c=40$ (cyan)
- Torus, $c=80$ (cyan)

Number of missing target links vs. cycles.
(e) $N=2^{17}$

The graph shows the number of missing target links over cycles. The y-axis represents the number of missing target links on a logarithmic scale, ranging from $10^1$ to $10^5$. The x-axis represents the cycles, ranging from 15 to 100.

Different curves represent different scenarios or conditions, with colors indicating distinct data sets or parameters. The curves demonstrate how the number of missing target links decreases over time.
T-Man: a gossip protocol for structured topologies

- T-Man generates a wide range of topologies
- The convergence is fast
  - Logarithmic in the number of nodes,
  - Independently of the topology
- Not only approximate, but **perfect** embedding can be achieved
- Applications include communication topology, sorting and clustering
Future Work

- fault tolerance: how T-Man reacts to node and link failures
- adaptivity: how T-Man adapts to a continuously changing node set
- our results with other gossip protocols suggest solutions
  - restarting
  - concurrently running protocol instances and usage of past results when restarting
- exploring applications
For more info check out my homepage:

http://www.cs.unibo.it/~jelasity/