Grassroots Approach to Self-Management

Ozalp Babaoglu
Márk Jelasity
Alberto Montresor

Department of Computer Science
University of Bologna
Italy

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

Basic premise

- Our (in)ability to deploy, configure, tune, maintain and manage effectively large-scale networked information systems is the principal obstacle to exploiting their potential
- We have reached the limits of traditional techniques
- Systems have to be self-organizing, self-configuring, self-tuning, self-healing, self-managing — *self-**

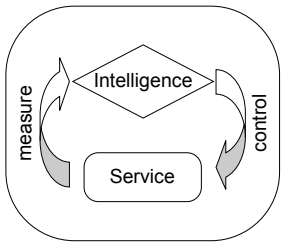
Autonomic computing

- *Autonomic computing* proposes to achieve self-management by replacing the human element with software/hardware components
- Analogy to the *autonomic nervous system* which
  - operates subconsciously, without intervention — it is *autonomous*
  - takes care of routine functions like heart rate, blood pressure, hormone production, digestion, etc.
- This analogy fails for certain other self-* functions like self-repair or self-protection

Autonomic computing: implementation

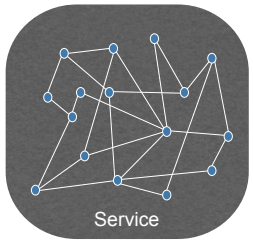
"The autonomic computing architecture starts from the premise that implementing self-managing attributes involves an intelligent control loop"

An architectural blueprint for autonomic computing, IBM

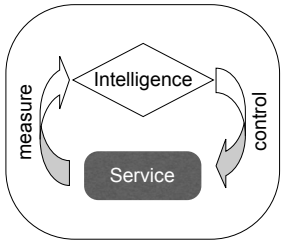


Autonomic service

The “grassroots” alternative



Grassroots



Autonomic

Grassroots approach

- “Service” implemented as a large number of simple entities that interact in simple ways (through an underlying communication structure)
- No distinction between “managed” and “manager” entities — only “peers”
- No control loop

Grassroots approach

- Inspired by *complex adaptive systems* that are found in
  - nature and biological processes
  - social structures
  - economies, financial markets
- These systems are *decentralized*, *self-organizing*, *adaptive* and *robust* through *emergence*, rather than explicit programming (control loop)

Autonomic vs grassroots

- Intrinsically centralized
  - Intrinsically decentralized
- Separation of managed entity and manager (homunculus)
  - “Peer” status for all components
- Knowledge based with Machine Learning, AI techniques
  - Emergence without explicit knowledge representation
- High complexity
  - Extreme simplicity
- Higher predictability
  - Lower predictability

Grassroots approach

- Not a universal solution
- Appropriate for very large scale, highly dynamic, highly distributed systems
- Potentially very robust and scalable
- Much simpler and easier to implement
- Potentially more efficient and effective
- Has its downsides:
  - Lower predictability and lower controllability
  - Not a gradual transition but a paradigm shift

The “psychological” barrier

- Users and administrators often mistrust emergent systems because
  - There are no hard guarantees that they will do what they are supposed to
  - Even when they appear to do what they are supposed to do, there is usually no explanation of why — gap between microscopic and macroscopic behavior
  - Difficult to exert control over them: what actions are necessary to achieve a desired behavior?

Project BISON

- Funded by IST-FET under FP5
- Partners
  - University of Bologna, Italy (Coordinator)
  - Telenor Communication AS, Norway
  - Technical University of Dresden, Germany
  - IDSIA, Lugano, Switzerland
- 1 January 2003 start date, duration 36 months
- Total cost €2,251,594
- EU funding €1,128,000
- URL: `http://www.cs.unibo.it/bison`

BISON objectives

- Develop tools and techniques suitable for building network information systems that exhibit “organic” or “life-like” properties
- Do this by drawing inspiration from complex systems that arise in nature (biology)

Technological niche

- Modern dynamic network structures
  - Mobile ad-hoc networks (MANET)
  - Overlay Networks
    - Peer-to-Peer systems
    - Grid computing

BISON expected results

- Decentralized, self-organizing, adaptive and robust solutions to important technological problems that arise in dynamic networks
- Systematic framework and a coherent set of heuristics to guide the synthesis of complex systems that solve interesting technological problems

Timely and relevant

essay concepts

Engineering complex systems

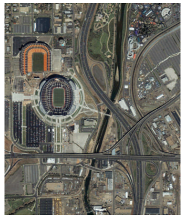
The emergent properties of complex systems are far removed from the traditional preoccupation of engineers with design and purpose.

J. M. Ottino

Complex systems can be identified by what they do (display organization without a central organizing authority — emergence), and also by how they may or may not be analysed (as decomposing the system and analysing sub-parts do not necessarily give a clue as to the behaviour of the whole). Systems that fall within the scope of complex systems include metabolic pathways, ecosystems, the web, the US power grid and the propagation of HIV infections.

Complex systems have captured the attention of physicists, biologists, ecologists, economists and social scientists. Ideas about complex systems are making inroads in anthropology, political science and finance. Many examples of complex networks that have greatly impacted our lives — such as highways, electrification and the Internet — derive from engineering. But although engineers may have developed the components, they did not plan their connection.

The hallmarks of complex systems are adaptation, self-organization and emergence — no one designed the web or the metabolic processes within a cell. And this is where the conceptual conflict with engineering arises. Engineering is not about letting systems be. Engineering is about making things happen, about convergence, opti-



More than the sum of its parts: complex systems, such as highways, are constantly evolving.

SPACE IMAGING

self-organize to operate in a state of optimum performance, in the face of effects that may potentially destroy it. However, the optimal state is a high-risk state — good returns at the price of possible ruin. Most engineers are risk averse, and would prefer to eliminate the probability of catastrophic events. Recent work borrows concepts from economic theories (risk aversion, subjective benefit of outcomes) and argues that one can com-

work in directed self-assembly and complex dissipative systems, which organize when there is energy input. However, practical processing by self-assembly is still not a reality, and there is work here for engineers.

But the choice need not be just between designing everything at the outset and letting systems design themselves. Most design processes are far from linear, with multiple decision points and ideas 'evolving' before the final design 'emerges'. However, once finished, the design itself does not adapt. Here, engineers are beginning to get insight from biology. The emergence of function — the ability of a system to perform a task — can be guided by its environment, without imposing a rigid blueprint. For example, just like the beaks of Darwin's finches, a finite-element-analysis of a component shape such as an airfoil can evolve plastically through a continuum of possibilities under a set of constraints, so as to optimize the shape for a given function.

Engineers calculate, and calculation requires a theory, or at least an organized framework. Could there be laws governing complex systems? If by 'laws' one means something from which consequences can be derived — as in physics — then the answer may be no. But how about a notch below, such as discovering relationships with caveats, as in the ideal gas 'law', or uncovering power-law relationships? Then the answer is clearly yes.

Nature, 29 January 2004

BISON biological inspirations

- Social insects, ants
- Immune system
- Amoebae
- Neurons
- Diffusion
- Chemotaxis
- Adhesion
- Regeneration
- Epidemics (gossip)

BISON mechanisms, services

- Routing (MANET)
- Power management (MANET)
- Aggregation
- Topology management
- Load balancing
- Searching
- Monitoring

Gossip-style communication

- Each node periodically selects another (random) peer and exchanges local state information
- Each node updates its local state based on the information exchanged
- System fully symmetric — all nodes act identically
- Communication is symmetric — “push-pull” gossip
- Proactive
- Many uses in distributed systems

Gossip-style communication

```
// active thread
do forever
     wait(T time units)
     q = selectPeer()
     send S to q
     receive Sq from q
     S = update(S,Sq)


// passive thread
do forever
     (p,Sp) = waitMessage()
     send S to p
     S = update(S,Sp)
```


Gossip-based components

- Protocols based on the same probabilistic gossip communication model:
  - data aggregation (e.g. average, maximum, etc)
  - topology management
    - unstructured: *newscast*
    - structured: *T-Man*
  - load balancing
  - etc.

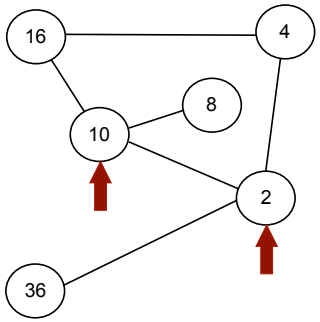
Aggregation

- Each node *p* has a (numeric) local state S_p
- Compute (global) aggregate function over the initial values at *all* nodes
- The aggregate value to be known (locally) at each node
- Examples of aggregate functions:
  - Average
  - Min-max
  - Geometric mean
  - Variance
  - Network size

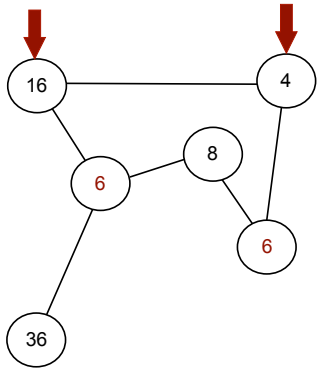
Aggregation through gossipping

- Local variable S_p contains current estimate of the aggregate
- Need to give implementations for
  - selectPeer()
  - update(S_p, S_q)
- selectPeer() picks a random neighbor
  - average: update(S_p, S_q) = $\frac{(S_p+S_q)}{2}$
  - geometric mean: update(S_p, S_q) = $\sqrt{(S_p S_q)}$
  - maximum: update(S_p, S_q) = *max*(S_p, S_q)
- Other, more complex functions built by combining elementary functions

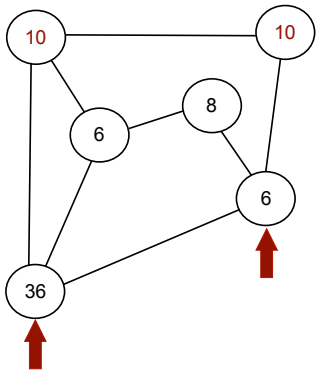
Aggregation example: averaging



Aggregation example: averaging



Aggregation example: averaging



Properties of aggregation

- In gossip-based averaging, if the selected peer is a globally random sample, then the variance of the set of estimates decreases exponentially
- Extreme robustness to node and link failure and node dynamism (churn)

Márk Jelasity and Alberto Montresor. Epidemic-style proactive aggregation in large overlay networks. In Proceedings of The 24th International Conference on Distributed Computing Systems (ICDCS 2004), Tokyo, Japan, 2004.

Alberto Montresor, Márk Jelasity, and Ozalp Babaoglu. Robust aggregation protocols for large-scale overlay networks. In Proceedings of DSN 2004, Florence, Italy.

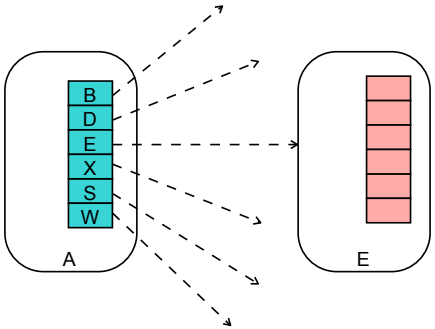
Overlay networks, views

- The set of nodes that a peer knows about is called its view
- Typically, views are a (very) small subset of all nodes
- Views are typically dynamic since the set of nodes and the "knows" relation are highly dynamic (churn)
- Views define an overlay network with dynamic topology on top of the basic communication substrate

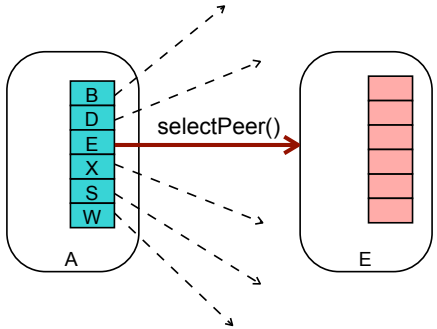
Topology management

- How to ensure that the overlay network topology satisfies certain properties:
  - has a desired structure (connected, random graph, ring, torus, binary tree, etc.)
  - *maintains* the desired structure in a dynamic setting (churn)
- Problem to be solved is *topology management*
- Solution based on gossipping
  - Local state: *view*
  - selectPeer: uses the actual view to select a peer
  - updateState($view_p$, $view_q$): construct new view based on local view and that of the selected peer

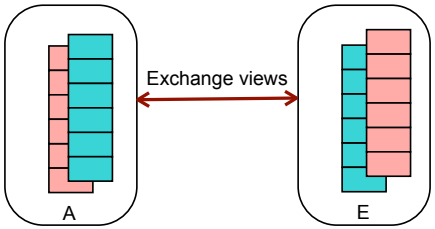
Topology management: example



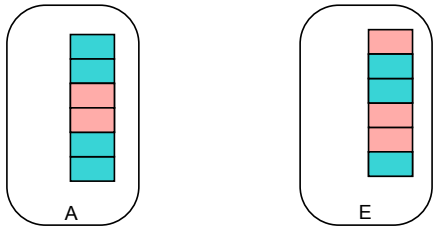
Topology management: example



Topology management: example



Topology management: example



Both peers apply updateState thereby redefining topology

Newscast: a gossip protocol for random topologies

- Node descriptors stored in the view contain timestamps
- selectPeer: randomly selects a peer from the curent view
- updateState: fills the view with the freshest descriptors (based on timestamps) from the union of the two views
- New information gradually replaces old information

Newscast: a gossip protocol for random topologies

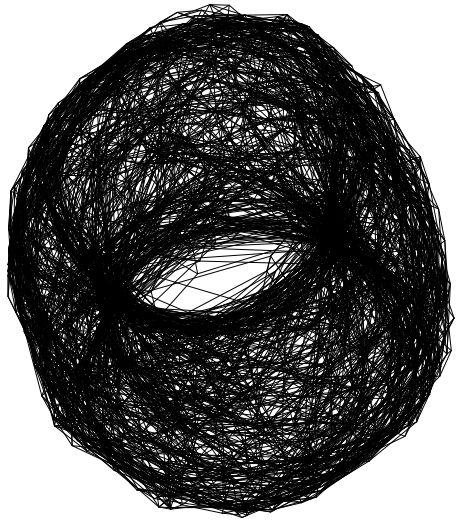
- Extremely robust to node and link failure and node dynamism (churn)
- Maintains a connected, approximately random topology
- Scalable

Márk Jelasity, Wojtek Kowalczyk, and Maarten van Steen. Newscast computing. Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, November 2003.

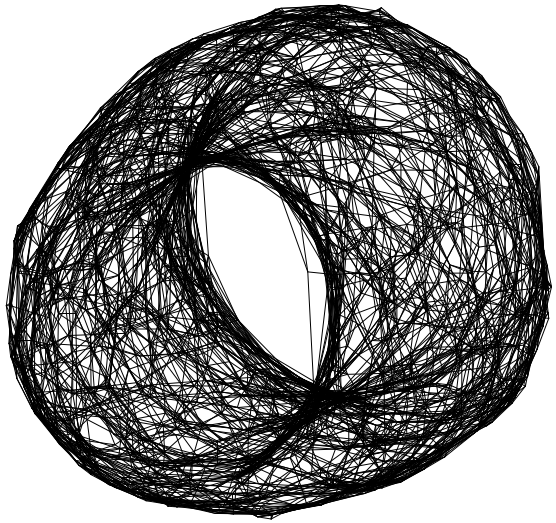
T-Man: a gossip protocol for structured topologies

- Node descriptors stored in the view contain the profile of the node (real number, 2-d vector, etc)
- selectPeer: Orders the view using a ranking function that defines the target topology and selects a neighbor from the first half according to ranking
- selectView: Fills the view with the lowest rank descriptors
- View initialization: a random set of initial nodes are desirable (use newscast)

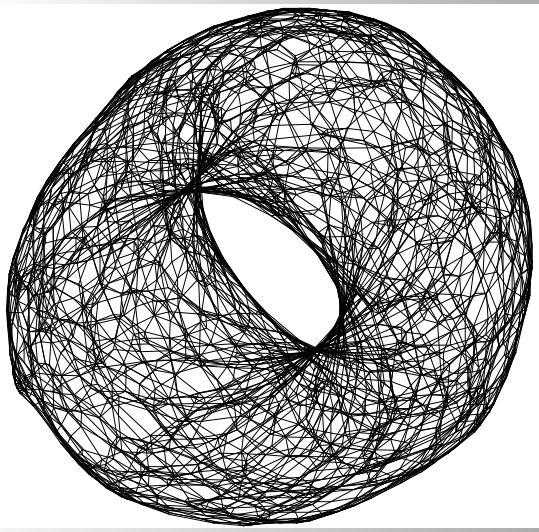
T-Man example: after 3 cycles



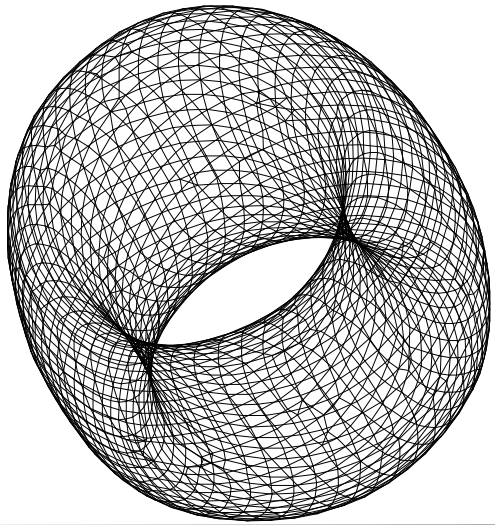
T-Man example: after 5 cycles



T-Man example: after 8 cycles



T-Man example: after 15 cycles



T-Man: a gossip protocol for structured topologies

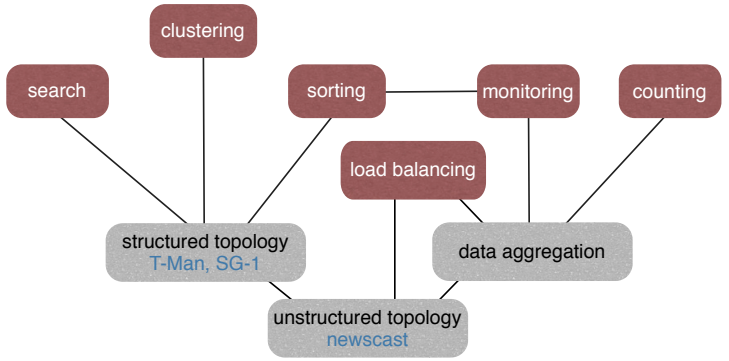
- Capable of generating a wide range of topologies (small and large diameter, clustered, sorted, etc)
- Preliminary results show that T-Man is scalable: converges with high accuracy in approximately logarithmic time

Márk Jelasity and Ozalp Babaoglu. T-Man: Fast gossip-based construction of large-scale overlay topologies. Technical Report UBLCS-2004-7, University of Bologna, Department of Computer Science, Bologna, Italy, May 2004.

Compositional grassroots self-management

- Make grassroots protocols managable through modularity
- Simple components (building blocks, services) for a specific simple function
- Due to their simplicity, they can be thoroughly understood, described and explained even to non-specialists
- They can be combined (now in a non-emergent manner) to form new, more complex functions keeping the benefits of simplicity, robustness and scalability

Dependencies among gossip-based components



Load balancing

- Use the aggregation component to calculate optimal load (global information)
- Use newscast to maintain dynamically changing random neighborhood
- Transfer load only from overloaded nodes to underloaded nodes
- Minimizes actually transferred amount of load

Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. A modular paradigm for building self-organizing peer-to-peer applications. In G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli (Eds), Engineering Self-Organising Systems, Vol. 2977 in Lecture Notes in Artificial Intelligence, 265-282, Springer-Verlag, 2004.

Summary

- Grassroots self-management has potential in very large scale, highly dynamic distributed systems
- Problems of trust and controllability can be tackled by breaking up functions into basic building blocks
- Gossiping is amazingly effective for building decentralized, scalable, robust, adaptive solutions to important problems in highly dynamic distributed systems
- We have shown building blocks for topology management (structured and unstructured), data aggregation and load balancing