Nature-Inspired Techniques for Self-Organization in Dynamic Networks

Ozalp Babaoglu

Department of Computer Science
University of Bologna
Italy

ALMA MATER STUDIORUM — UNIVERSITA' DI BOLOGNA

Technological progress

- Typically, new technologies are "geeky" and require expertise to understand, install and maintain through significant human involvement
  - Power
  - Transportation
  - Music recording and reproduction
  - Communication

Technological progress

- Eventually, humans are partially or completely removed from deployment and maintenance as the technology matures and becomes simpler (for users)
  - To increase adoption and sales
  - To decrease cost (industrial revolution, agriculture)
  - To allow super-human performance (transportation, aviation)
- Simplicity of usage often means increased overall system complexity

Case for autonomic computing

- Information systems have been less successful in following this trend
- They tend to be very complex and costly to install, configure and maintain by humans
- This is a major obstacle to progress
  - For industry
    - IT costs are becoming prohibitive, no new systems, only maintenance
    - Integrating multiple systems is extremely difficult
  - For consumers
    - Electronic gadgets, computers, etc. cause frustration and inconvenience, which hinders adoption

Case for autonomic computing

- Need information systems that are
  - Self configuring
  - Self optimizing
  - Self healing
  - Self protecting
  - Self managing
- "Self-∗" proposed as a catch-all term for most desirable properties
- Outlined in IBM's "Autonomic Computing" vision

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.

Emergence is everywhere

- Unforeseen and uncontrolled interactions among the system components may lead to *emergent behavior*
- In large and complex systems, even if they are centrally controlled, emergence is often inevitable
  - Power grids
  - Telephone switching networks
  - Retail, supply chains
- Manifests itself in phenomenon like "cascading failures" or *parasitic emergence*

Building on interactions

- Make interactions among components work *for* you rather than *against* you
- By planning and building interactions into our components, we can achieve desirable global properties that emerge without explicit action or control

Grassroots approach

- "Service" implemented as a large number of simple entities that interact in simple ways
- Totally decentralized with no distinction between "managed" and "manager" entities — only "peers"
- Can be *self-organizing*, *adaptive* and *robust* through *emergence*, rather than explicit programming — no control loop
- May be the only viable option in large scale, dynamic network environments with multiple administrative domains (grid, peer-to-peer, cloud computing)
- Can draw inspiration from natural, biological, social, economical structures or processes

Grassroots approach

- Key ideas
  - (Simple) actions based only on local information
  - (Simple) interactions with small number of other components
  - Power of randomization
  - Power of large numbers

Gossip-style interaction

- Model for structuring decentralized solutions to problems in large systems
- Interactions limited to small number of *peers* that know of each other
- System fully symmetric — all peers act identically
- Gossiping can be
  - Reactive, proactive
  - Push, pull, push-pull
- The set of peers that a node "knows" is called a *view* and defines an overlay network

Proactive gossip framework

```
// active thread
do forever
    wait(T time units)
    q = SelectPeer()
    push S to q
    pull Sq from q
    S = Update(S,Sq)


// passive thread
do forever
    (p,Sp) = pull * from *
    push S to p
    S = Update(S,Sp)
```

Proactive gossip framework

- To instantiate the framework, need to define
  - Local state ***S***
  - *Peer sampling service* implementing method **SelectPeer()**
  - Style of interaction
    - **push**
    - **pull**
    - **push-pull**
  - Method **Update()**

PSS: Peer sampling service

- Return a small number (usually one) of nodes selected at *random* among the *entire* population of peers
- Conceptually simple but difficult in practice since it is not feasible for peers to maintain the entire population locally
  - Extremely large
  - Extremely dynamic
- PSS approximates a random sample from a global population using only local views which have a small, constant size
- PSS itself can be based on gossiping

What we have done

- We have used gossiping to obtain fast, robust, decentralized solutions for
  - Peer sampling service
  - Aggregation
  - Overlay topology management
  - Heartbeat synchronization
  - Formation creation in ubiquitous computing
  - Cooperation in selfish environments

#1
Overlay Topology Management

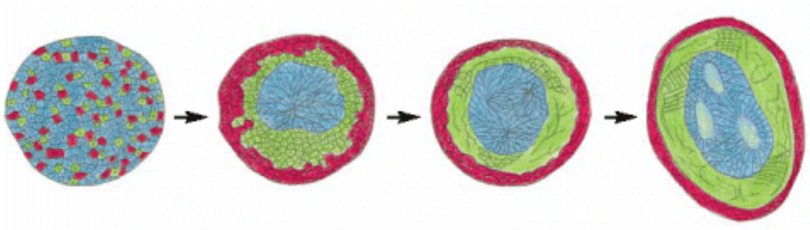
Topology management

- *Overlay networks* are key abstractions for building large, decentralized systems (grid computing, peer-to-peer, cloud computing)
- How to construct and maintain an overlay network that satisfies desired topological properties in a manner that is
  - Decentralized
  - Self-organizing (insensitive to initial state)
  - Scalable (insensitive to network size)
  - Robust (insensitive to churn)
- If this topology management problem can be solved efficiently and rapidly, it can be used to satisfy application topological needs on-demand

Developmental biology

- *Morphogenesis* attempts to understand the processes that control the organized spatial distribution of cells during embryonic development and that give rise to the characteristic forms of tissues, organs, and overall body anatomy
- An interesting theory based on “differential cell adhesion”
  - different cell types “sort out” based on “likes” and “dislikes” for each other
  - any cell configuration has an energy level
  - cells try to minimize the free energy in the system by a stochastic movement process

Developmental biology



Cells from different parts of an early amphibian embryo sort out according to their origins (Townes & Holtfreter 1955)

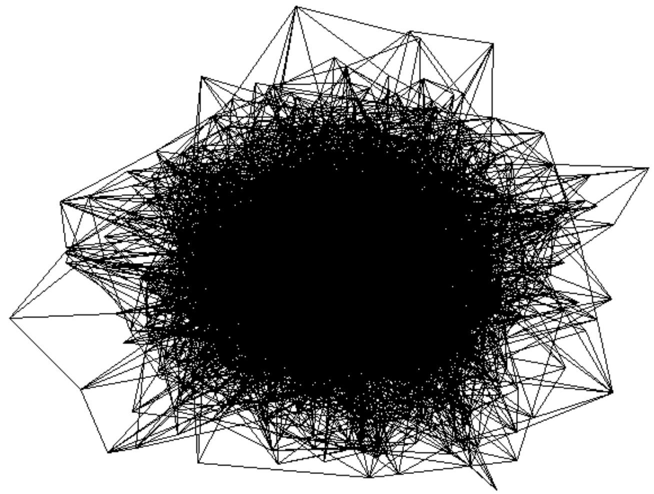
Back to topology management

- In biological systems, adhesion constrained by physical constraints
- In overlay networks, we can define peer relationships as we wish, resulting in a vast range of potential target topologies
- Notion of “like” and “dislike” captured by a *ranking function*
- Each ranking function encodes a particular target topology
- Target topology can be changed on-the-fly by informing all nodes to start using the appropriate ranking function (perhaps after having distributed it)

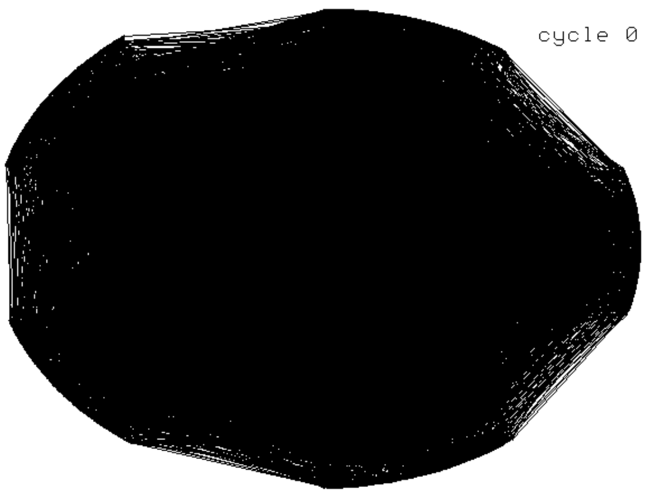
Gossip framework instantiation

- Style of interaction: push-pull
- Local state ***S***: Current neighbor set
- Method **SelectPeer()**: Single random neighbor
- Method **Update()**: Ranking function defined according to desired topology (ring, mesh, torus, DHT, etc.)

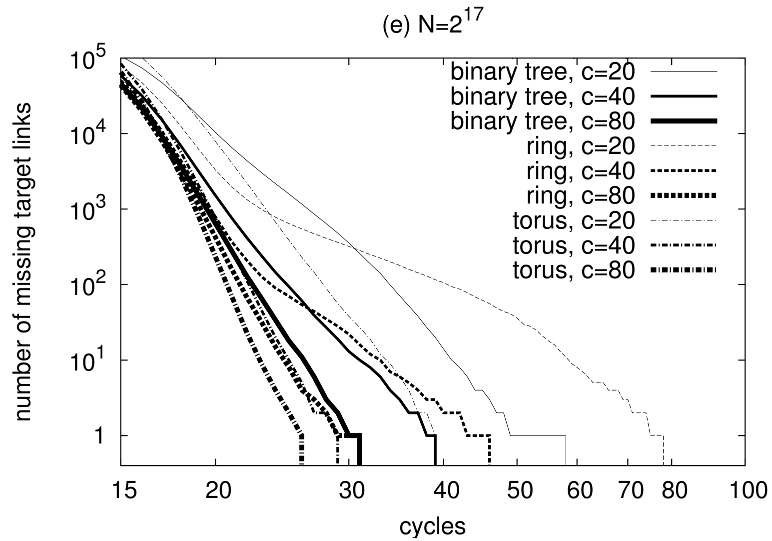
Torus example



Sorting example

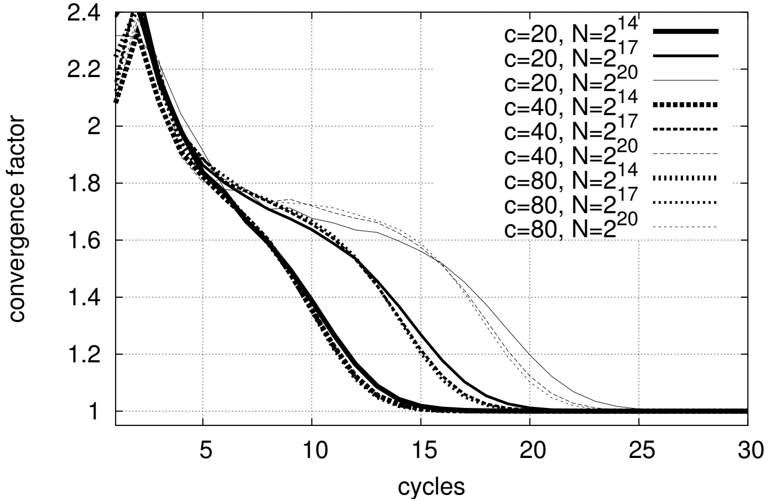


Exponential convergence - time



Exponential convergence - network size

(a) ring



#2
Heartbeat Synchronization

Synchrony in nature

- Nature displays astonishing examples of synchrony among independent actors
  - Heart pacemaker cells
  - Chirping crickets
  - Menstrual cycle of women living together
  - Flashing of fireflies
  - Clapping of an audience at a concert
- Actors may belong to the same organism or they may be parts of different organisms

Coupled oscillators

- The "Coupled oscillator" model can be used to explain the phenomenon of "self-synchronization"
- Each actor is an independent "oscillator", like a pendulum
- Oscillators coupled through their environment
  - Mechanical vibrations
  - Air pressure
  - Visual clues
  - Olfactory signals
- They influence each other, causing minor local adjustments that result in global synchrony

Fireflies

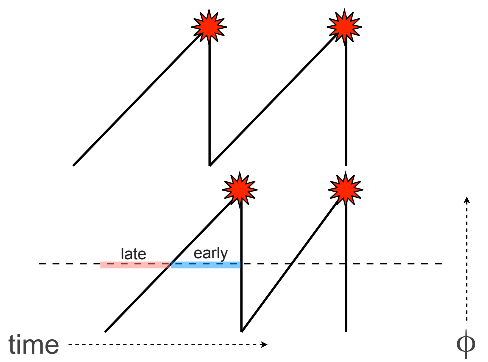
- Certain species of (male) fireflies (e.g., *luciola pupilla*) are known to synchronize their flashes despite:
  - Small connectivity (each firefly has a small number of “neighbors”)
  - Communication not instantaneous
  - Independent local “oscillators” with random initial periods

Gossip framework instantiation

- Style of interaction: push
- Local state ***S***: Current phase ϕ and period Δ of local oscillator
- Method **SelectPeer()**: (small) set of random neighbors
- Method **Update()**: Function to reset the local oscillator based on the phase of arriving flash

The Ermentrout model

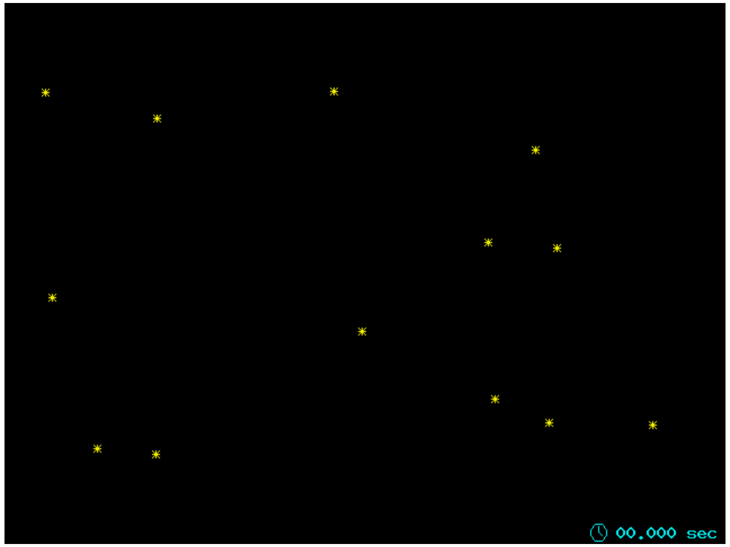
- Modify the local oscillator period based on when flash arrives:
  - if “too late” (ϕ <½), then “slow down” (increase period Δ)
  - if “too early” (ϕ >½), then “speed up” (decrease period Δ)



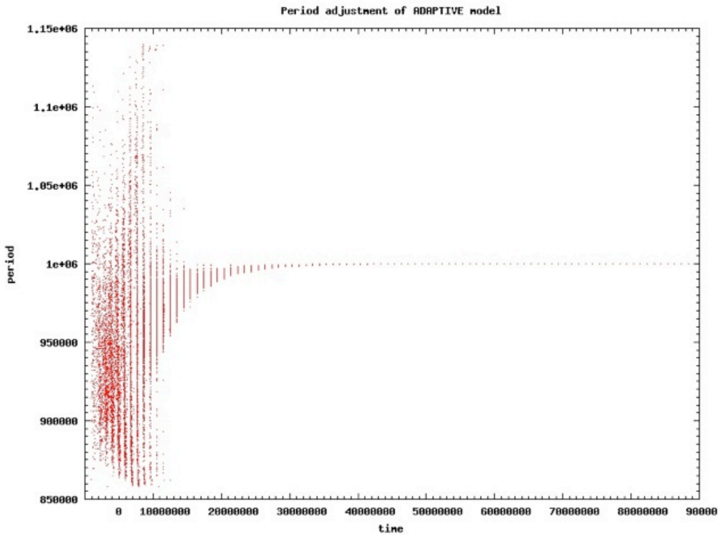
Experimental results

- Network size 2^{10} nodes
- View size of 10
- Initial periods selected uniformly and randomly in the interval [0.85 - 1.15] seconds
- Message latency uniformly and randomly distributed in the interval [1 - 200] ms

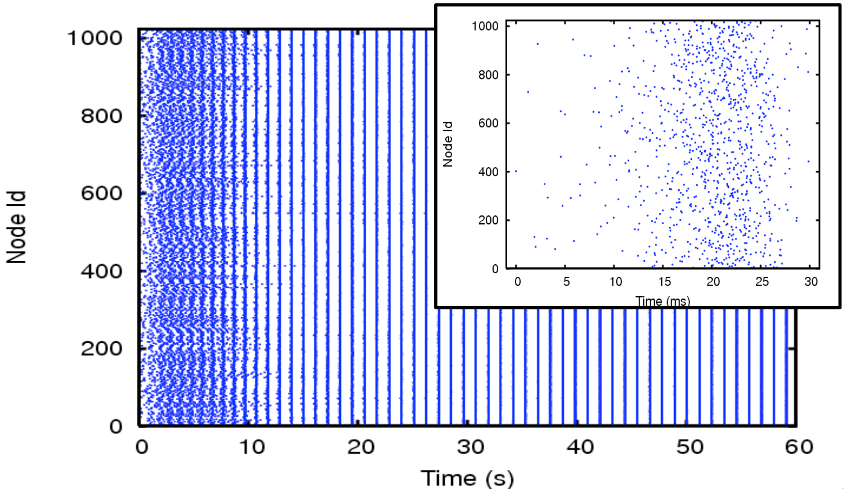
Experimental results



Convergence of periods



Chaos to coherent emissions



#3

Formation Creation in Ubiquitous Computing

Ubiquitous computing

- Large number of mobile “smart” physical objects with identification, sensing and computing power
  - Rescue teams with mobile devices
  - Automobiles in vehicular networks
  - Swarms of robots
  - Satellites in earth orbit
- Ad hoc wireless communication network (no fixed infrastructure) based on physical proximity or contact
- Broadcast or multicast communication model rather than unicast (which would require a routing service)

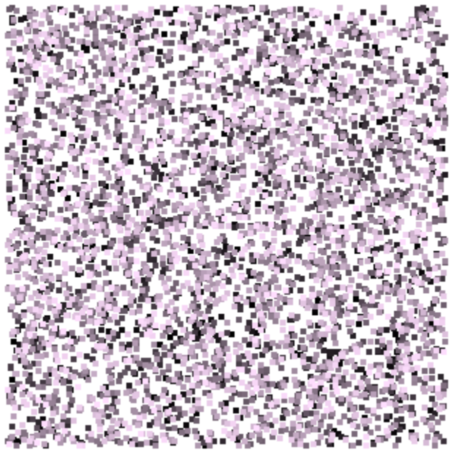
Formation creation

- Dynamic collection of agents that can move in any direction
- Each agent has a unique ID and can determine the relative position of any other agent
- Agents have access to a peer sampling service (can be trivially implemented if the broadcast range of agents covers the entire physical space)
- Devise a protocol such that mobile agents self organize into pre-specified global formations in a totally decentralized manner

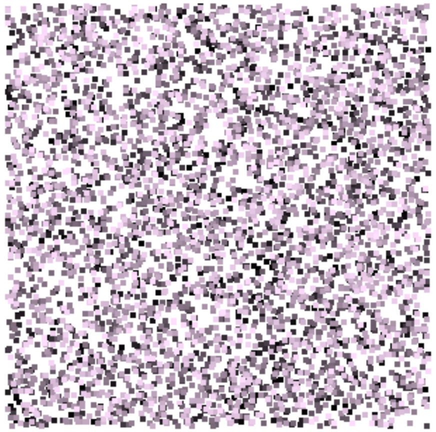
Gossip framework instantiation

- Style of interaction: pull
- Local state ***S***: Current physical position and motion vector
- Method **SelectPeer()**: *k* random samples from population
- Method **Update()**: Compute motion vector based on positions of *most* and *least* preferred neighbor (defined in a manner similar to the *ranking function* of overlay topology creation inspired by *differential cell adhesion*)

Simulation: Ring formation

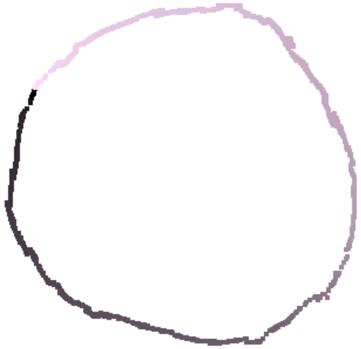


Simulation: Cross formation



Simulation: Self-healing ring

Starting formation: ring of 5000 nodes



Simulation: Self-healing ring

80% of the 5000 nodes are removed

Simulation: Self-healing ring

Remaining 1000 nodes reform the ring

#4 Cooperation in Selfish Environments

ALMA MATER STUDIORUM — UNIVERSITA' DI BOLOGNA

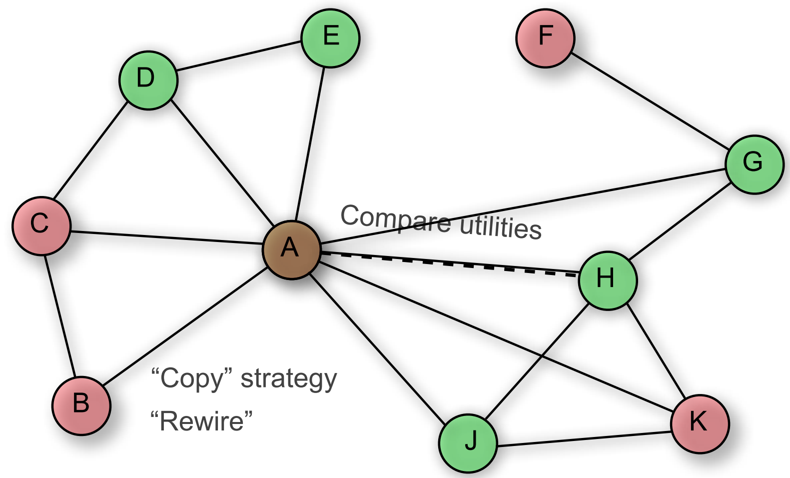
Outline

- P2P networks are usually open systems
  - Possibility to free-ride
  - High levels of free-riding can seriously degrade global performance
- A gossip-based algorithm can be used to sustain high levels of cooperation despite selfish nodes
- Based on simple "copy" and "rewire" operations

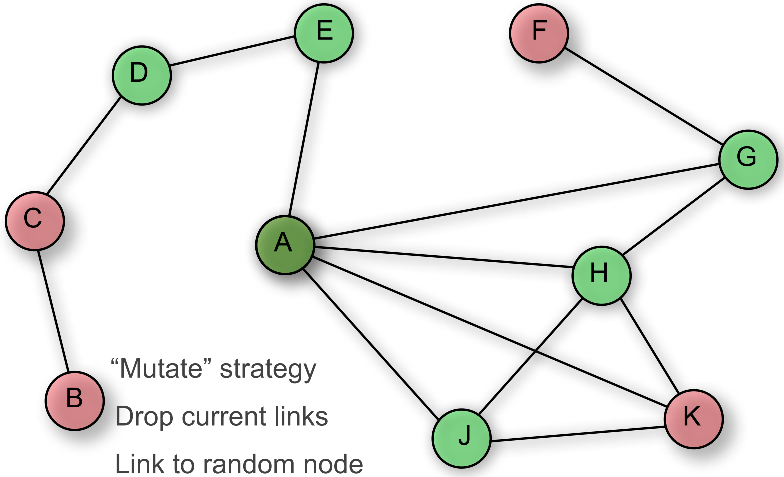
Gossip framework instantiation

- Style of interaction: pull
- Local state *S*: Current *utility*, *strategy* and *neighborhood* within an *interaction* network
- Method **SelectPeer()**: Single random sample
- Method **Update()**: Copy strategy and neighborhood if the peer is achieving better utility

SLAC Algorithm: "Copy and Rewire"



SLAC Algorithm: “Mutate”



Prisoner’s Dilemma

- We test SLAC with Prisoner’s Dilemma (PD)
  - Captures the conflict between “individual rationality” and “common good”
  - Defection (*D*) leads to higher *individual* utility
  - Cooperation (*C*) leads to higher *global* utility

	C	**D**
C	*R,R*	*S,T*
D	*T,S*	*P,P*

- $T > R > P > S$ and $2R > T+S$

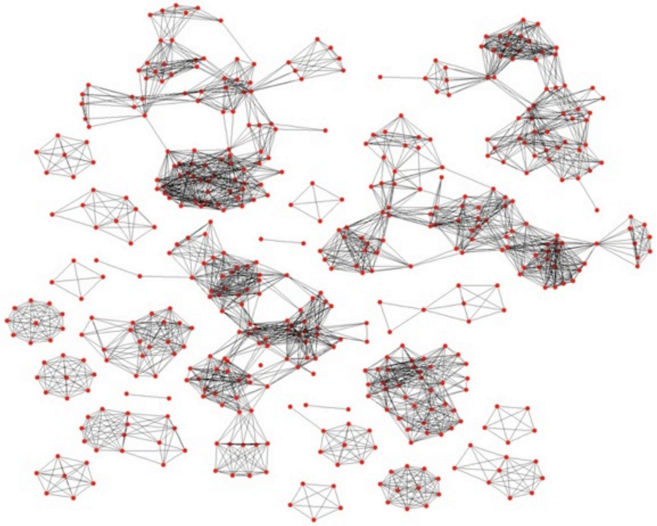
Prisoner’s Dilemma

- Prisoner’s Dilemma in SLAC
  - Nodes play PD with neighbors chosen randomly in the interaction network
  - Only pure strategies (always C or always D)
  - Strategy mutation: flip current strategy
  - Utility: average payoff achieved

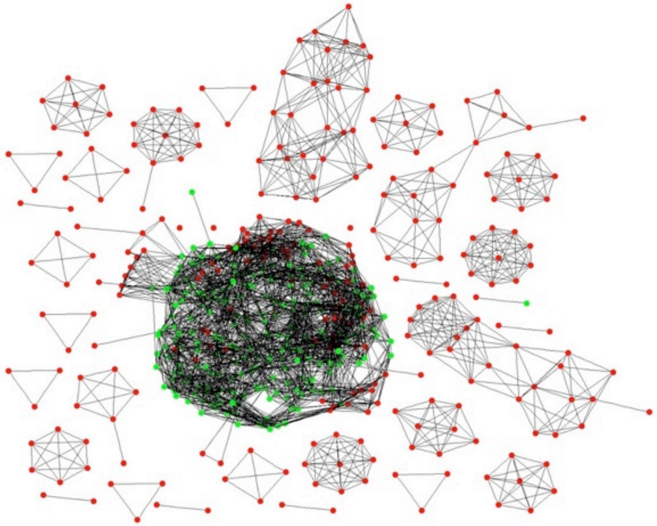
Example

- 500 nodes
- Initial state:
  - All defectors
  - Random interaction network

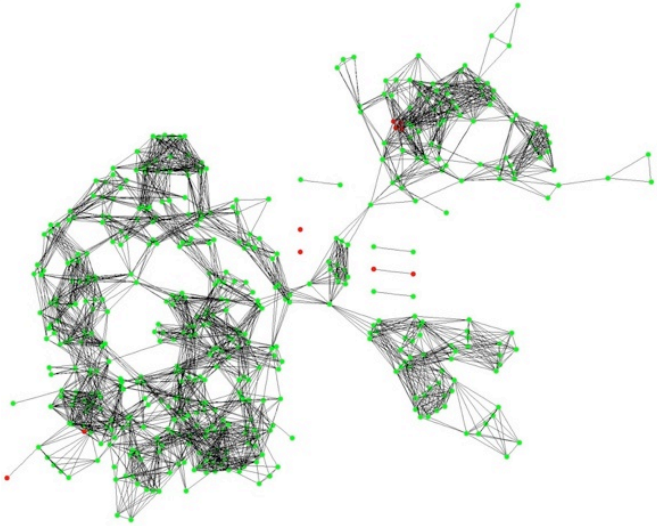
Cycle 180: Small defective clusters



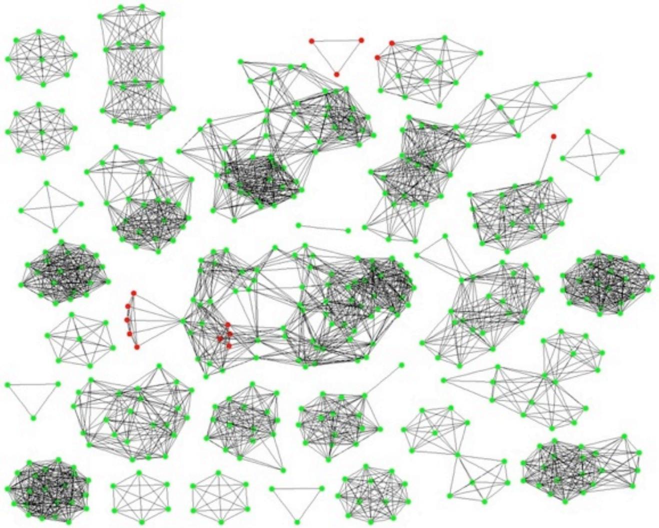
Cycle 220: Cooperation emerges



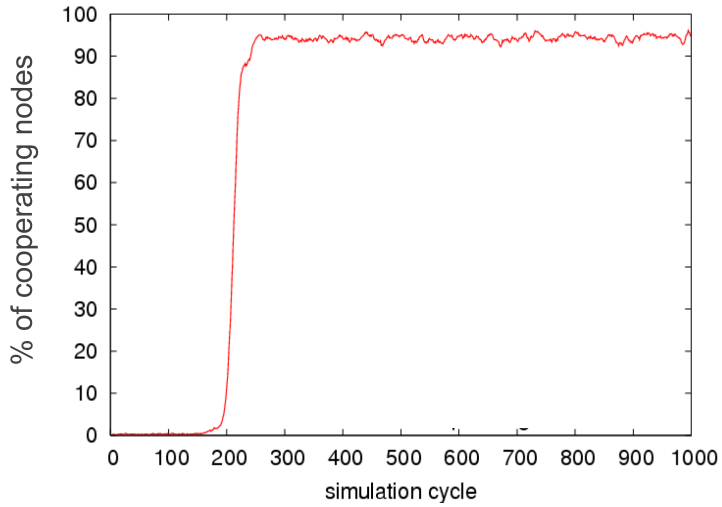
Cycle 230: Cooperating cluster starts to break apart



Cycle 300: Defective nodes isolated, small cooperative clusters formed



Phase transition of cooperation



Summary

- "Complex systems" approach to autonomic computing and self-management
- Advocated the "grassroots" approach
- Self-organization as a key concept
- Emergence as a key mechanism
- Simple, localized, random interactions — gossiping
- Interesting instances that prove to be extremely robust and scalable

Collaborators

- Mark Jelasity
- Alberto Montresor
- David Hales
- Tony Binci
- Stefano Arteconi