Emergent Group-Like Selection in a Peer-to-Peer Network

David Hales
University of Bologna, Italy

www.davidhales.com

This work partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

ECCS Conference
Paris, Nov. 16th, 2005
Consider a P2P overlay network in which each node:
- Offers a service (e.g. storage, processing etc.)
- Receives jobs submitted by users
- May ask neighbour nodes to help complete jobs
- May complete jobs for neighbours
- May move in the network by making and breaking links
- Uses local information only
- Behaves in a selfish way (boundedly rational)
- May compare its performance to other nodes
- May copy links, and behaviours of other nodes

We want a scalable, robust, light-weight decentralised algorithm that self-organises network to maximise system level performance.
SLAC = Selfish Link-based Adaptation for Cooperation

Demonstrated to be effective in P2P networks when:
- Peers play the Prisoner’s Dilemma with neighbours (ESOA’04)
- Peers answer queries and share files (IEEE TSMC’05)

But in these previous scenarios:
- Nodes provided an identical service
- Cooperation resulted from all nodes behaving identically

This new problem requires specialists nodes to work together

In order to maximise system level performance nodes need to do different things, not identical things

This work therefore tests if SLAC can support inter-node specialisation
To test this produced a simulation model called SkillWorld

The population consists of N nodes (fixed)

Each node has the following state:
- A single skill from a set $S \in \{1,2,3,4,5\}$ (the service provided)
- An altruism flag $A \in \{0,1\}$ (indicates if node helps neighbours)
- A utility $U \in \mathbb{R}$ (a performance measure)
- Some set of links to other nodes (max. of 20)

Each node asynchronously receives and attempts to complete jobs
- Each job is marked with a single skill # (randomly chosen)
- Job must be processed by a node with matching skill
- If receiving node i has req. skill, job is completed $U_i = U_i + 1$
- If node i does not have req. skill it asks its neighbours
- If a neighbour j is found with $A = 1$ and matching skill then:
  - Job is completed, $U_i = U_i + 1$, but, $U_j = U_j - 0.25$
Choose Your Tribe! Evolution at the next level in a P2P network

The Simulation - SkillWorld

- = altruist
○ = selfish

Job(1)
Job(3)

0
4

2
1
3

-0.25
+1
+1

1
1
1

Dynamically Evolving, Large-scale Information Systems
David Hales, University of Bologna, www.davidhales.com

ESOA2005, Utrecht, July 2005
Choose Your Tribe! Evolution at the next level in a P2P network
SLAC follows a kind of evolutionary process

Periodically each node:
- Engages in application level activity producing utility (SkillWorld)
- Compares its utility to another randomly chosen node
- If the other node has higher utility then
- Copy links and some behaviour of other node
- With low probability “mutate” links and behaviour
### The SLAC algorithm in SkillWorld

<table>
<thead>
<tr>
<th>Active thread:</th>
<th>Passive thread:</th>
</tr>
</thead>
<tbody>
<tr>
<td>do forever:</td>
<td>do forever:</td>
</tr>
<tr>
<td>sleep for some short time period</td>
<td>j ← this node</td>
</tr>
<tr>
<td>i ← this node</td>
<td>GetState(i):</td>
</tr>
<tr>
<td>with prob. P reproduce:</td>
<td>Send j.Utility to i</td>
</tr>
<tr>
<td>j ← SelectRandomNode()</td>
<td>Send j.Links to i</td>
</tr>
<tr>
<td>j.GetState(i)</td>
<td>Send j.AltruismFlag to i</td>
</tr>
<tr>
<td>if i.Utility ≤ j.Utility</td>
<td></td>
</tr>
<tr>
<td>i ← CopyStatePartial(j)</td>
<td></td>
</tr>
<tr>
<td>Mutate(i)</td>
<td></td>
</tr>
</tbody>
</table>

**Function CopyStatePartial(j):**

i.AltruismFlag ← j.AltruismFlag

drop all links from i

i.Links ← j.Links

**Function Mutate(i):**

with prob. M mutate i.AltruismFlag

with prob. MR mutate i.Links:

don't mutate all links from i

i.Links ← SelectRandomNode()
In each cycle, 10N jobs are submitted to randomly selected nodes
Each job is given a randomly selected skill requirement
Nodes initialised with random skills and links (random network)
Initial topology of network made little difference to results
Compared initialisation of altruism flag randomly and all selfish
Compared different network sizes N
Measured proportion of completed jobs (CJ) in each cycle
Mutation values (M = 0.001, MR = 0.01)
If a node reaches its max. links (20) then a random link is discarded if a new link is required
Utility for each node = CJ - total help cost
Ran simulations until 90% of jobs completed
Choose Your Tribe! Evolution at the next level in a P2P network

Simulation Results

Number of cycles to high performance for different N
When PCJ > 90% over 90% of all jobs submitted to nodes are completed.
Nodes are initialised at random

ESOA2005, Utrecht, July 2005  Choose Your Tribe! Evolution at the next level in a P2P network
Number of cycles to high performance *when all nodes are initialised selfish*

Note that there is a reverse scaling cost here.
Typical single run (N=1000) from random initialisation.
selfish = proportion non-altruists, C = clustering coefficient,
comps = components in the population (normalised by dividing by 60),
conprob = average probability that a route exists between any two nodes.
Choose Your Tribe! Evolution at the next level in a P2P network
Simulation Results

Choose Your Tribe! Evolution at the next level in a P2P network

Cycle 20

Cycle 30
Simulation Results Summary

- From individual bounded utility maximising behaviour (SLAC)
- Altruistic “tribes” emerge with internal specialisation
- Tribes that do well - collectively - tend to recruit new nodes
- Tribes that perform badly - collectively - tend to lose nodes
- Hence productive tribes prosper, defective tribes “die”
- This is a kind of “tribe selection” via recruitment and retention
- By giving nodes the ability to choose their tribes a kind of tribe level evolution happens - evolution at the next level
Issues, on-going and future work

- But SLAC produces extreme tribalism with disconnected components => SLACER (on-going work)
- SLAC assumes honest passing of info and utility comparison => Greedy Cheating Liars (on-going work)
- The SkillWorld task is an “easy” test => more realistic scenarios
- System performance does not attain more than about 93%
- If Skill mutated then can adjust to different job task loadings
- But if Skill is copied like the AltruismFlag then fails to converge, yet in similar scenarios it does
- Tribe recruitment is the key idea => (on-going work)
- Future work could drop utilities and move to satisficing where aspiration level is a kind of “required service level”
Choose Your Tribe! Evolution at the next level in a P2P network
Choose Your Tribe! Evolution at the next level in a P2P network