DELIS

Dynamically Evolving, Large-scale Information Systems

# Towards a group selection design pattern

David Hales*, Stefano Arteconi**, Andrea Marcozzi**, Isaac Chao***

*Technical University of Delft, Netherlands
formally **University of Bologna
*** UPC

DELIS Final Workshop, Feb 28th 2008, Barcelona

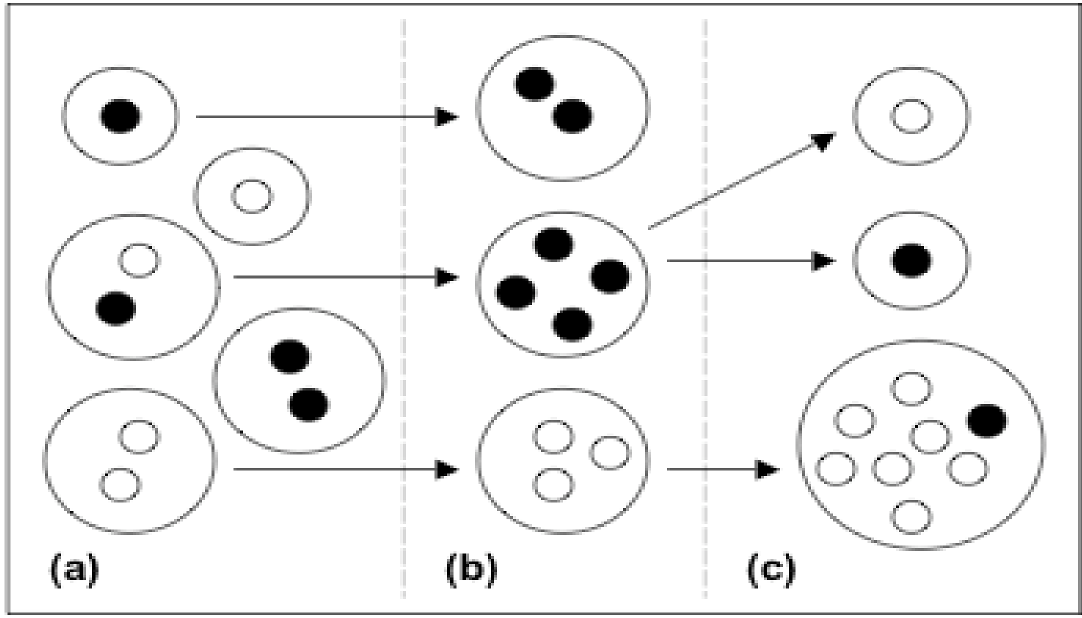
Information Society Technologies

# Group Selection Design Pattern

- A number of novel "group selection" models are coming from theoretical biology and computational social science
- We give initial work towards a "group selection" design pattern or "approach" for creating cooperative distributed systems
- We present some previous simulation models that use the approach in the form of a design template
- There are still open issues

# Group Selection Models

- Recent models of “group selection”
- Based on individual selection
- Producing dynamic social structures
- Limit free-riding
- Increasingly group-level performance
- Don’t require reciprocity
- Could be very useful in P2P



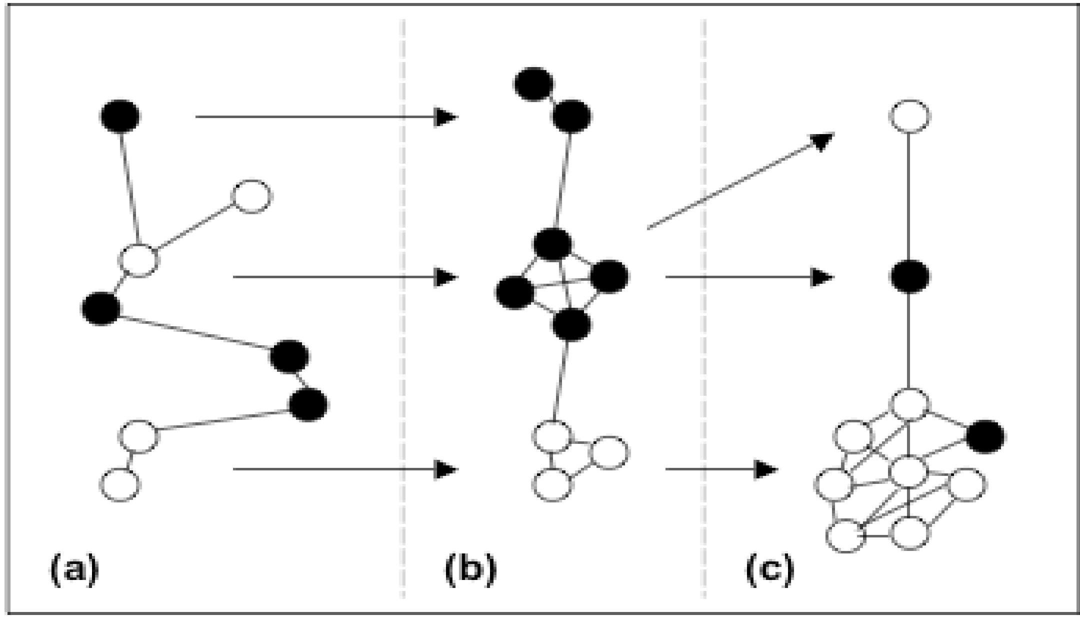

**Outline algorithm for tag model:**

for each generation loop
  interaction within groups (obtain fitness)
  reproduce individuals based on fitness
  with *Prob(mt)* individuals form new group
  with *Prob(ms)* individuals flip strategy
end generation loop

*Group boundary: tag stored by each individual defines group membership*
*Group formation and migration: probabilistic mutation of tag*

Schematic of the evolution of groups in the tag model. Three generations (a-c) are shown. White individuals are pro-social (altruistic), black are selfish. Individuals sharing the same tag are shown clustered and bounded by large circles. Arrows indicate group linage. When ***b*** is the benefit a pro-social agent can confer on another and ***c*** is the cost to that agent then the condition for group selection of pro-social groups is: ***b > c and mt >> ms***

Riolo, Axelrod, Cohen, Holland, Hales, Edmonds…




**Outline algorithm for network model:**

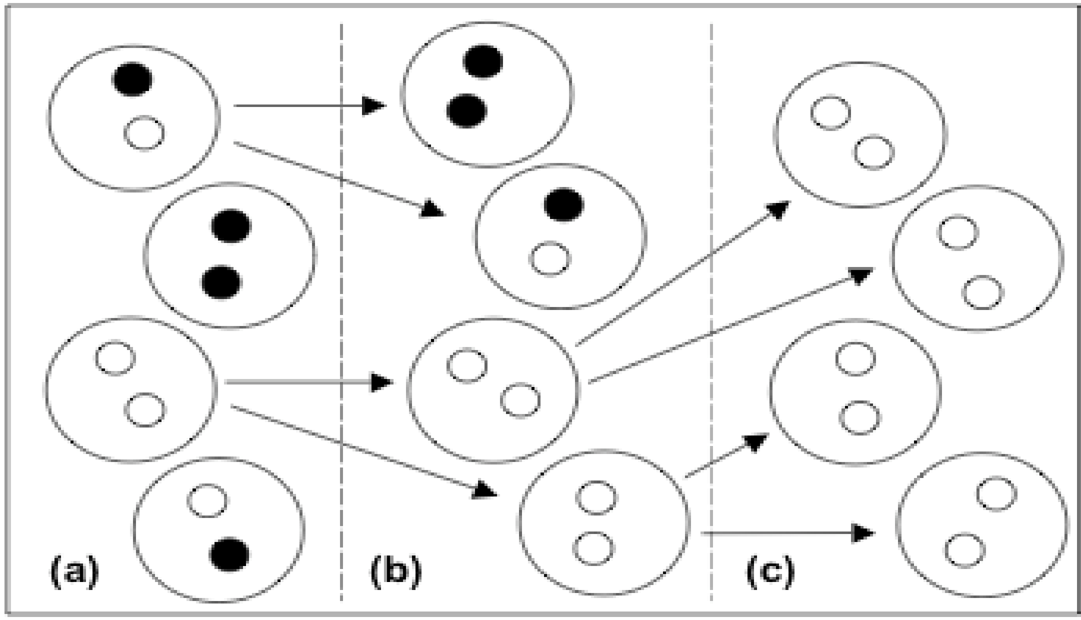
for each generation loop
  interaction within groups (obtain fitness)
  reproduce individuals based on fitness
  with *Prob(t) copy new links*
  with *Prob(mt)* individuals form new group
  with *Prob(ms)* individuals flip strategy
end generation loop

*Group boundary: individuals directly linked in the network*
*Group formation and migration:copying of links* probabilistically

Schematic of the evolution of groups in the network-rewire model. Three generations (a-c) are shown. Altruism selected when:***b > c and mt >> ms.*** When ***t = 1***, get disconnected components, when ***1 > t > 0.5***, get small-world networks

*Hales, D. & Arteconi, S. (2006) Article: SLACER: A Self-Organizing Protocol for Coordination in P2P Networks. IEEE Intelligent Systems, 21(2):29-35*

*Santos F. C., Pacheco J. M., Lenaerts T. (2006) Cooperation prevails when individuals adjust their social ties. PLoS Comput Biol 2(10)*




**Outline algorithm for split model:**

for each generation loop
  interaction in *m* groups (obtain fitness)
  reproduce individuals based on fitness
  with *Prob(q) split any group > n in size*
        *eliminate random group*
end generation loop

*Group boundary: individuals exogenously given group membership*
*Group formation and migration: splitting of group when size > n*

Schematic of the evolution of in the group-splitting model. Three generations (a-c) are shown. Altruism is selected if the population is partitioned into ***m*** groups of maximum size ***n*** and ***b / c > 1 + n / m.***

*Traulsen, A. & Nowak, M. A. (2006). Evolution of cooperation by multilevel selection. Proceedings of the National Academy of Sciences 130(29):10952-10955.*

# Group Selection Design Pattern

- Assumptions:
  - A system is composed of individual entities that can benefit from interaction with other entities
  - The population of entities is partitioned into groups such that interaction is mainly limited to entities within the same group
  - Entities measure their own performance periodically producing a utility value
  - Entities may spontaneously change their behavior and group membership
  - Entities may view and copy some state of other entities
  - Entities desire to increase their performance (utility)

# Group Selection Design Pattern

- Key Aspects:
  - Collective **Goal** - A desirable goal that the population of entities should attain.
  - **Group** Boundary Mechanism - How an entity can locate and communicate with in-group members.
  - Intra-Group **Interaction** - What kinds of utility effecting interactions an entity participates in with other in-group members.
  - **Utility** Calculation Metric - How an entity calculates a utility value based on its individual goal and in-group interactions.
  - Group **Migration** Mechanism - How migration between groups is performed.

# Group Selection Design Pattern

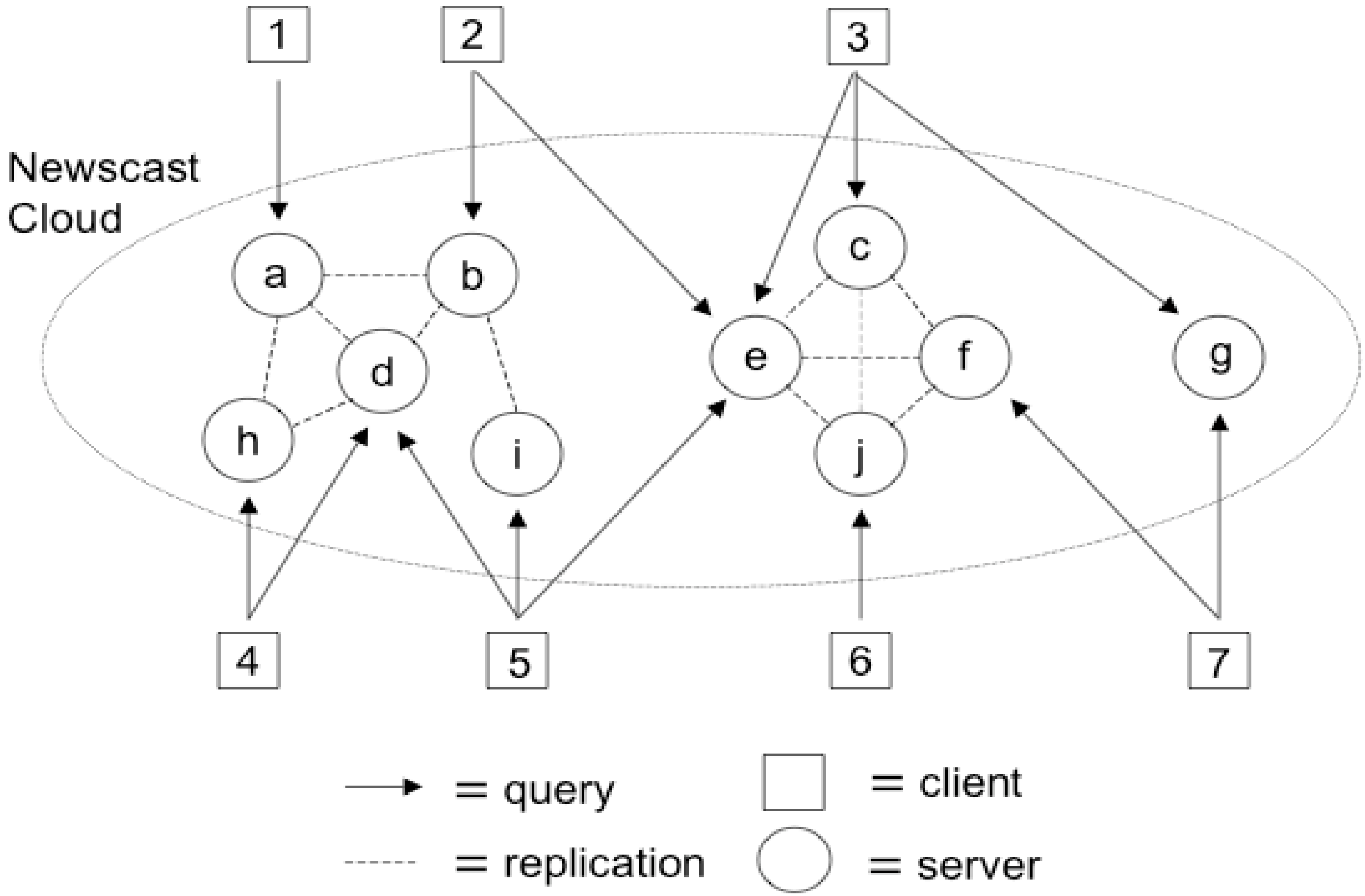
- Emergent Process:
  - Entities are grouped in some initially arbitrary way
  - Interactions between entities within groups determine entity utilities
  - Based on utility comparisons between entities, and possibly randomized change, group memberships and interaction behavior (strategy) change over time
  - Groups which produce high utility for their members tend to grow and persist as entities join
  - Groups which produce low utility for their members tend to disperse as entities leave
  - Hence group beneficial behavior tends to be selected

# CacheWorld

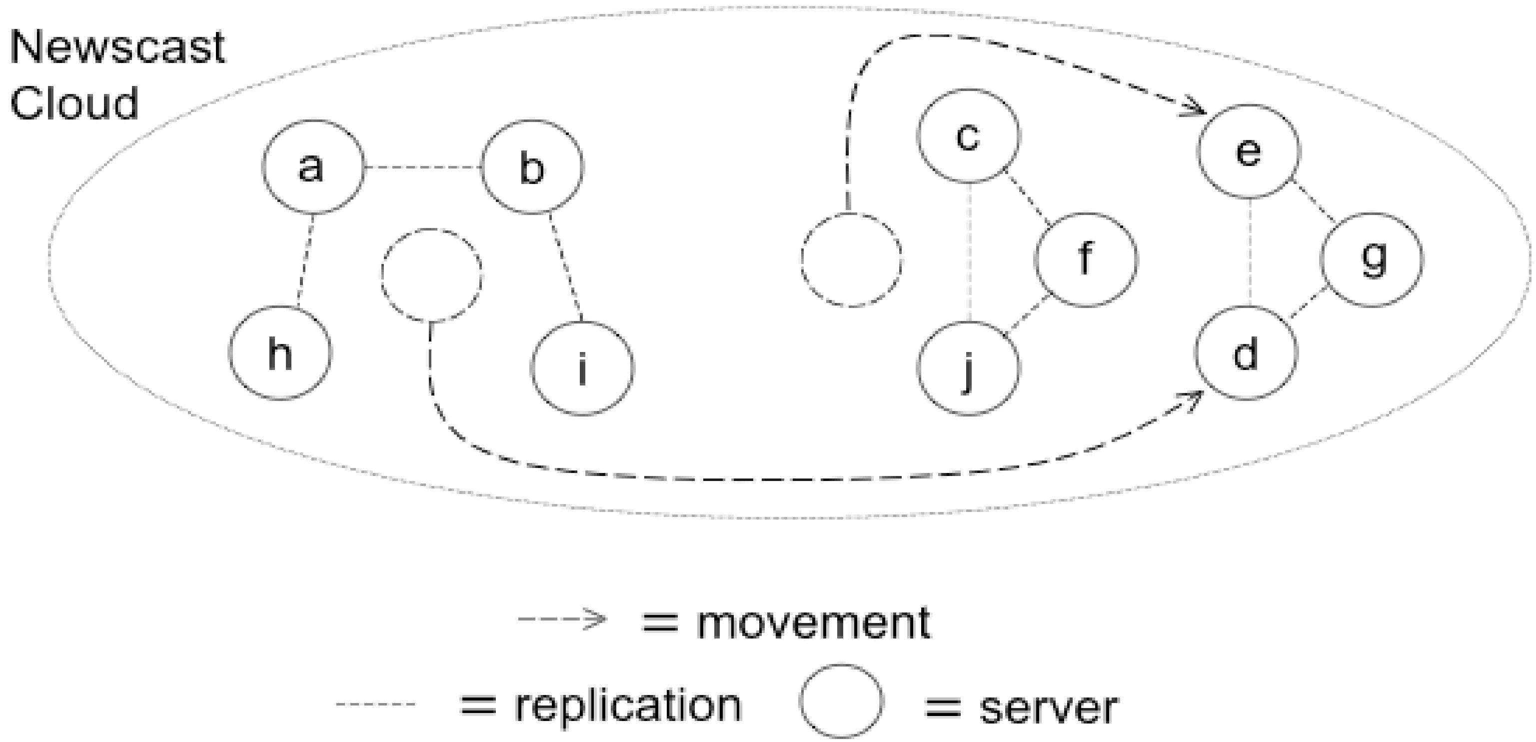
| | |
|---|---|
| Collective Goal | Maximise the total number of queries served by harnessing unused capacity in underloaded nodes. |
| Entity | Peer node - a node in a peer-to-peer overlay network with the ability to receive and serve queries, for a content item, from clients external to the overlay network. Each node has a maximum capaciy limiting number of queries serviceable over a time period. Each node can be thought of as a web server, for example, and stores is own content item and a replicated copy of each of its neighbours content items. |
| Group | The neighbour list (or view) of a node defines its group |
| Interaction | Receiving redirected queries from overloaded nodes or conversely redirecting queries to a random neighbour when overloaded. When a node makes a connection to a new neighbour both nodes mutually replicated their contents. |
| Utility | A simple binary satisfaction function: if all queries received by a node are eventually served then the node is satisfied otherwise it is unsatisfied. |
| Migration | Periodically, unsatisfied nodes move randomly in the network. But a node will only accept an incoming connection from a moving node if it is in a receptive state. A node is only receptive if it has spare capacity or is itself unsatisfied. |

**Figure 19. Key aspects for the CacheWorld model.**

# CacheWorld


1
2
3
Newscast
Cloud
a
b
c
d
e
f
g
h
i
j
4
5
6
7
= query
= replication
= client
= server

# CacheWorld



# CacheWorld Outline Algorithm

- capacity and load for each node specify different scenarios
- maximum number of neighbours (k) currently fixed
- nodes “satisfied” if all queries submitted to them are answered (over a given period - the load cycle)
- nodes associated with single unique content item replicated between linked neighbours
- nodes are “receptive” if they have spare capacity or are not satisfied

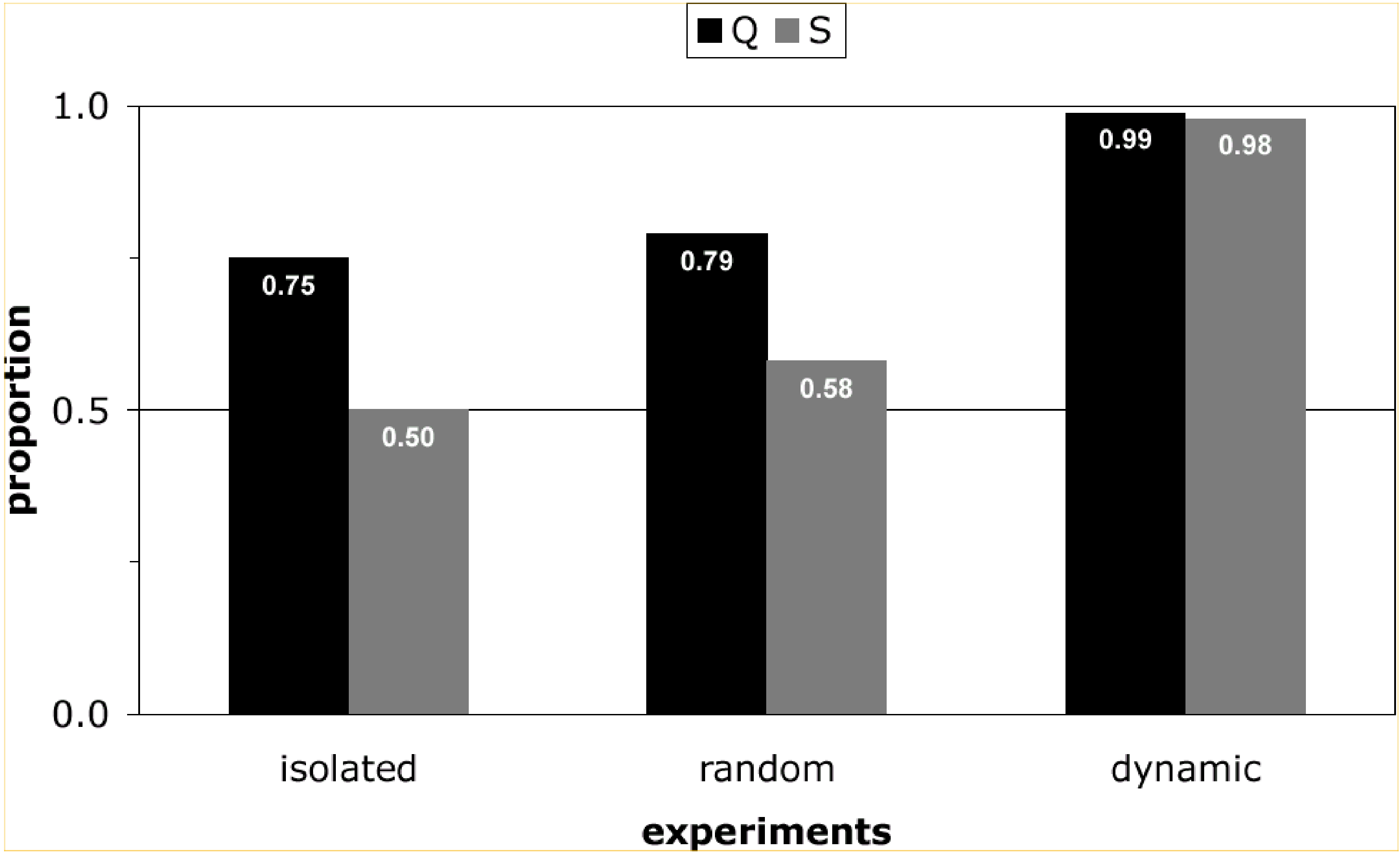
*Passive thread*

```
on receiving a query q, node i:
    if not overloaded, service q directly
    else if neighbors > 0 and q is not
    already a redirected query
        j ← selectRandomNeighbor()
        redirect q to j
    end if
```

*Active thread*

```
periodically each node i:
    if not satisfied
        drop all neighbor links
        j ← selectRandomPeer()
        if j is receptive then link to j
    end if
```

# CacheWorld




Q = queries answered, S = satisfied nodes

(very simple scenario, half nodes underloaded, half overloaded, k = 1)

| | |
|---|---|
| Collective Goal | Maximize the total number of query hits in the file-sharing network as-a-whole. |
| Entity | Peer Node - a node in a peer-to-peer overlay network with inter-agent communications infrastructure. Each node stores a neighbor list (or view) and an Altruism Level (see below). Periodically agents may randomly change their neighbor list and altruism level. |
| Group | View - nodes store a list of links to other peers called their view. All peers within the view are considered to be within the same group. Hence producing an overlapping network of groups. |
| Interaction | Sending and serving queries for files. Peers have a fixed capacity determining how many query messages they can handle in a given period. Nodes store an "altruism level" [0..1] which specifies the proportion of capacity devoted to serving requests from others as opposed to sending their own requests. |
| Utility | Query Hits - Peers generate their own queries with the capacity left over after processing others queries. Utility is the number of such answered queries (or hits) over a given period. |
| Migration | Copying peers with higher utility - Peers periodically select another peer randomly from the entire population (which may include peers outside the in-group). If the utility of this other node is higher then the peer copies its View and Altruism Level (overwriting previous values). By copying the View the agent migrates to the group of the copied node. |

**Figure 8. Key aspects for the FileWorld model.**

# FileWorld

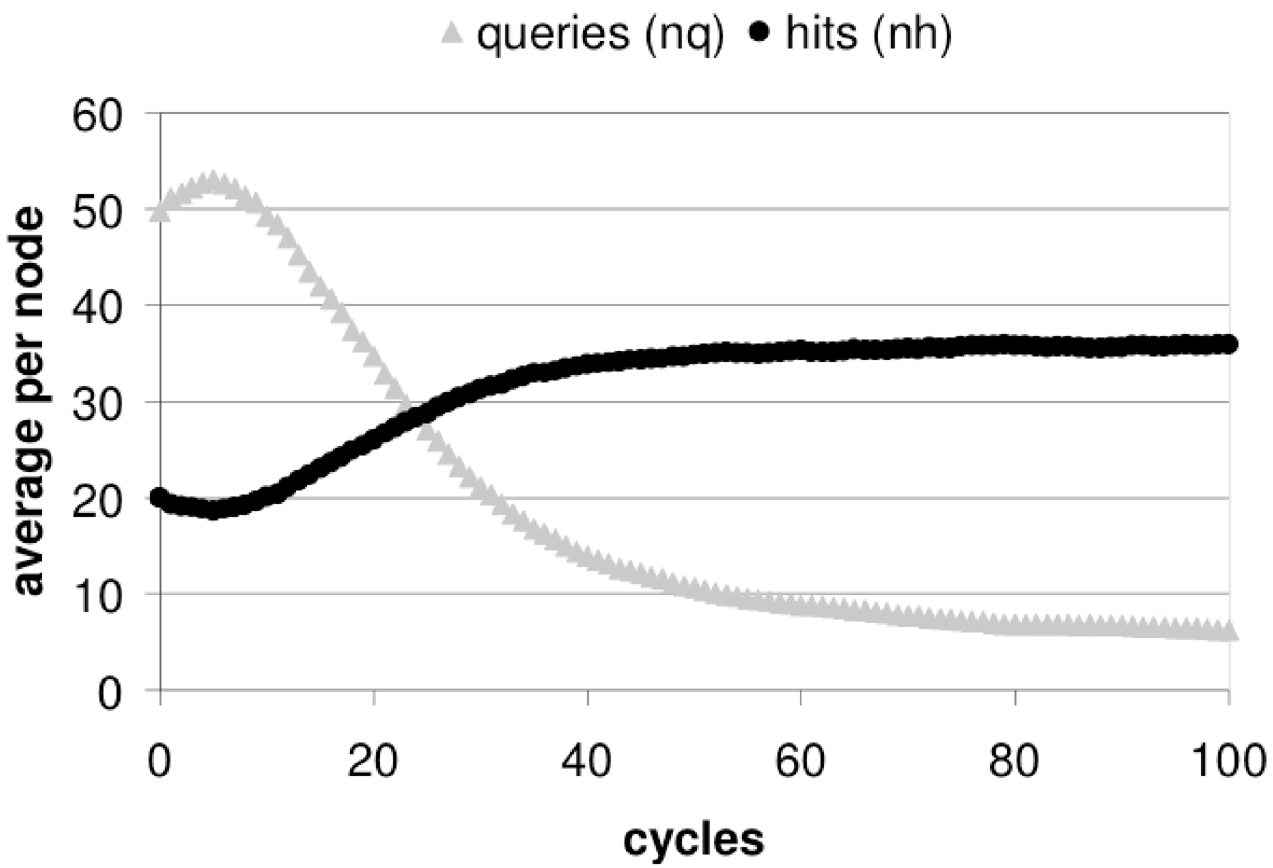
```
LOOP some number of cycles
    Initialise all node capacities and utilities
    LOOP some number of node firings (a time period)
        Select a random node (a) from the population
        IF node (a) has capacity to generate queries
            Decrease capacity by one query
            Generate query and pass to all neighbours (see below)
            Accumulate number of hits (utility)
        END IF
    END LOOP
    LOOP some number of times
        Select randomly a pair of nodes from the population
        Copy view and altruism value from higher to lower utility node
        Apply mutation with low probability to view and altruism value
    END LOOP
END LOOP

When node (b) receives a query:

IF node (b) has capacity to answer queries
    Decrease capacity by one query
    With ``Answering Power'' probability produce a ``hit''
    IF no hit produced and query TTL > 0
        Reduce TTL
        Pass query to all neighbours
    END IF
ELSE
    Ignore query
END IF
```

**Figure 9. Pseudo-code for the FileWorld simulation model.**

# FileWorld


queries (nq)
hits (nh)
60
50
40
30
20
10
0
average per node
0
20
40
60
80
100
cycles

# Conclusion

- Still many open issues for malicious behaviour
- More of an “approach” than a “pattern” ?
- Can this approach be used for real deployed systems?
- Perhaps a similar approach can improve BitTorrent?
- Currently based in Delft on P2P-Next project: based on **tribler.org** social bittorrent engine. Includes VTT, BBC, Pioneer, European Broadcasting Union - “EU next generation internet TV standard”
- Trying to apply these group selection ideas in this deployed system

# www.davidhales.com