3rd CoreGRID Workshop on Middleware – June 2008

Scalable concurrency control in a dynamic membership

A. Ciuffoletti * – A. Asim **

* INFN-CNAF, Bologna, Italy
** University of Paris Sud-XI, Paris, France

Problem statement

- A centralized resource
- A large, dynamic number of users
- Each user must be periodically granted access to the resource

A use case

- A centralized registry that must be periodically checked by each user to synchronize a local cache.

A multi-token approach

Given

- The service time (here *token latency*) *t* and
- The number of nodes in the system *N*

We can figure out the number of tokens needed to sustain a certain period *p*:

$$n = \frac{N \quad t}{p}$$

But we need:

- An overlay ring (difficult)
- The size of the network (difficult)

Our solution

Overlay topology: mesh with degree near to N

Token routing: random neighbor

Token number: closed loop over *p*

Problems:

- Security issues (tokens are received from “anybody”)
- Maintain a cache with most of the neighbors
- Closed loop control

Closed loop control

- Measure the time from the last token receive event
- Early tokens are delayed in a short queue (discard on overflow)
- Generate a new token upon timeout

Appears to be quite simplistic: loop control is still to refine

Problem (minor): sensitivity to network performance

Membership maintenance

- Recent updates to the local membership directory are piggybacked to the token
- Limited capacity of the token
- At each step the list of updates is recomputed

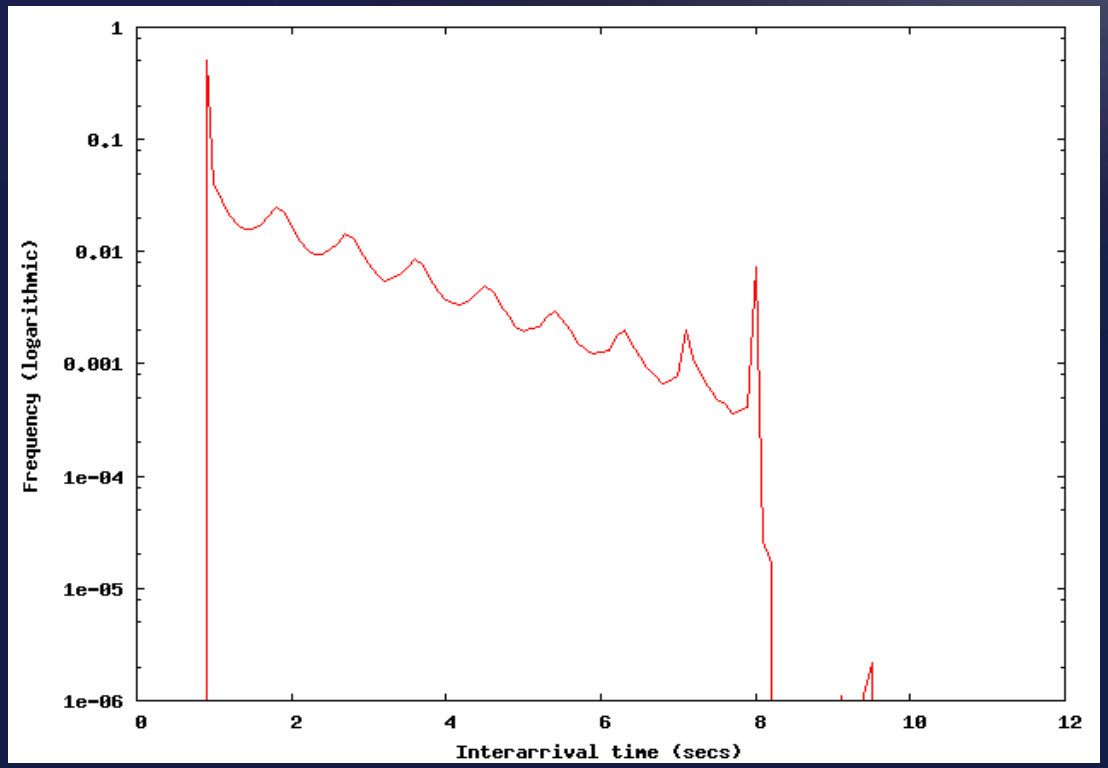
Join event

Create a new token and send it to a known peer

Leave event

File a leave event in the first available token

Experimental results



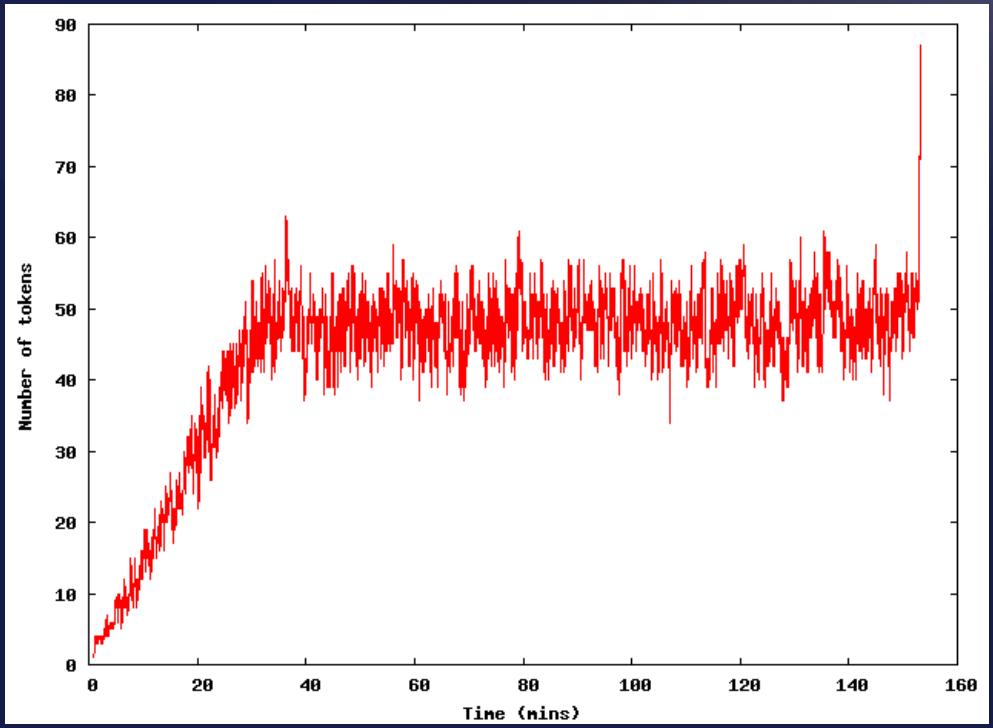
Decrease exponentially (good)

Maximum at the minimum (bad)

Average at 1.7 instead of 2.6 (bad)

Waving? Due to buffering.

Experimental results

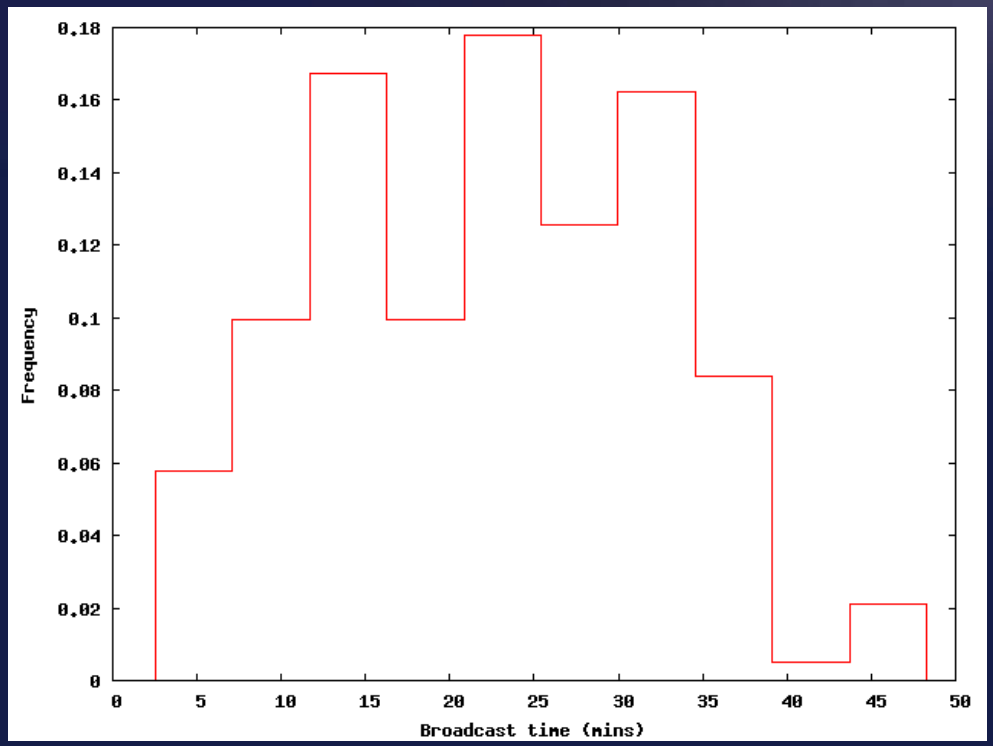


The process converges (good!)

Its dynamics are extremely fast (unexpectedly good)

Number of token exceeds expectation (55 instead of 2!)

Experimental results



All join events eventually received (good)

Takes a long time (expected)

Conclusions

- One of the first experiments of this kind in the Internet scale (instead of theoretical investigation and simulations)
- Mixed results:
  The process stabilizes, local directories eventually converge
  Dynamics are not those expected: too many tokens, short lived, too frequent

Overall: we are on the right way, tuning needed