Scalable Service Deployment using Mobile Agents

Luís Bernardo (lflb@inesc.pt)

Paulo Pinto (paulo.pinto@inesc.pt)

IST / INESC - Portugal

Topics

- System Rational
  - Envisioned Applications
  - Mobile Agent Opportunity
  - Proposed System
- Server Deployment Algorithm
- Simulation Results
  - Time Response
  - Scalability
- Conclusions

Envisioned Applications

- Deployed on large-scale networks with millions of users
- With synchronous patterns on the behaviour of the clients, producing peaks of requests on the servers
- Examples: tele-shopping, real-time sports brokering, stock brokering or applications based on interactive TV interfaces

- Requirements:
  - Dynamic deployment of different types of servers
  - Ubiquitous world-wide platform
  - Localised balanced client-server interactions

Mobile Agents Opportunity

- The dynamic creation or migration of application server´s mobile agents solves:
  - insufficient processing resources
  - insufficient bandwidth
  - unstable connections
- Applicability restrictions:
  - non-mobility requirements (large amounts of data, static resources)
  - strong inter-server requirements (shared variable synchronisation)
- Mobile agents can be used as service proxies (which concentrate requests or caches of information) reducing the load seen by other application components

Proposed System

- Application servers are based (partially or not) on mobile agents
- Application servers specify the range where their service offer is known
- Client´s load is balanced between existing servers by the location service - *Application Names* are resolved to the nearest application server reference
- Application servers are created, migrated or destroyed to adapt its number and location to the client load
- Location servers are also created and destroyed to adapt to the load, but also to the requested service´s range

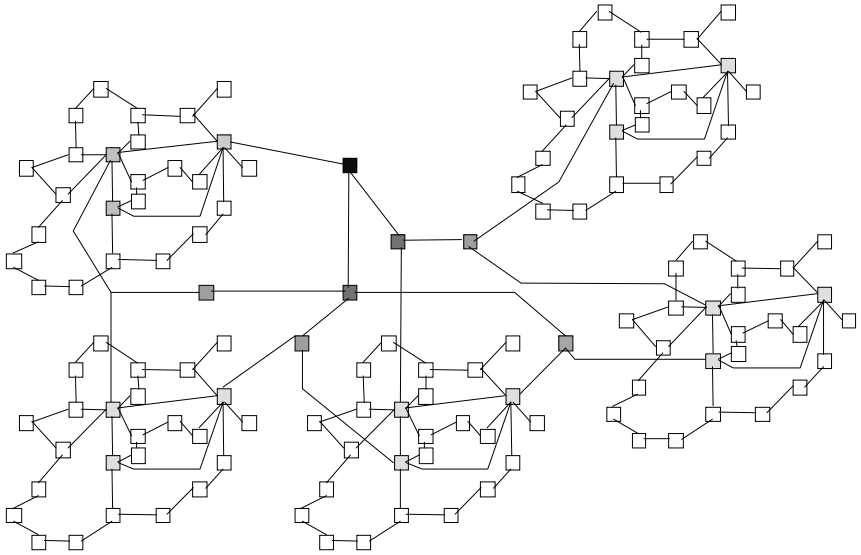
Application Server Deployment I

- Servers measure their local load:
  - server idle time
  - the client input queue length (if possible)
- If (Load > Top threshold(**MaxCliQ**)) then a new server is created using an isolated algorithm :
  - The new server location is selected using the collected client's origin statistics
  - The server creation rate is controlled by the client request's arrival rate
    - the top threshold value is temporarily incremented when a new server clone is created
  - Clients are redistributed between all the available servers after staying bound to a server (without running) more than **Timeout** tics

Application Server Deployment II

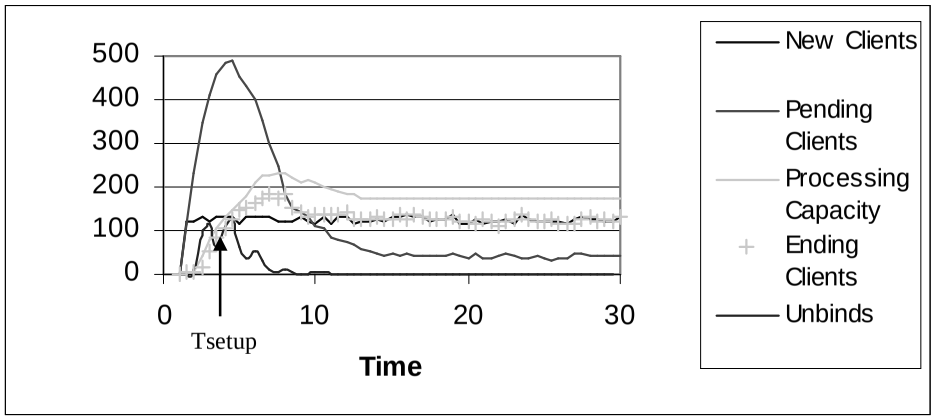
- If (Load < Bottom threshold(**MaxIdleTime**)) then market based algorithm is used to select a server to be destroyed:
  - Servers send “Request for bids” for all the servers within a range, and wait for “Bids” during an interval
  - Servers may bid several requests - the load transfer phase is acknowledged, and may fail if the server “load” is above its acceptance limit
  - The server delegates its clients adjusting the location service information
- The equilibrium between the creation algorithm and the destruction algorithm is set using $0 < \beta \leq 1$ and MaxIdleTime:
  - $Load_{low}(i) = \beta * load\_idleTime\ (i) + (1 - \beta) * Load_{low}(i-1)$
  - $Load_{high}(i)$ = “client request queue length”

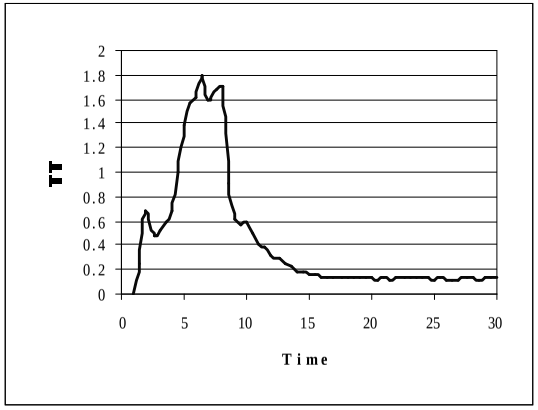
Simulator



- 132 agent systems
- 19 **static** location servers
- ServiceTime= 0.1
- LocServiceTime= 0.001
- ServerCloneSTime= 1
- ClientLoad clients uniform distributed
- Transm. delay= 0.0001

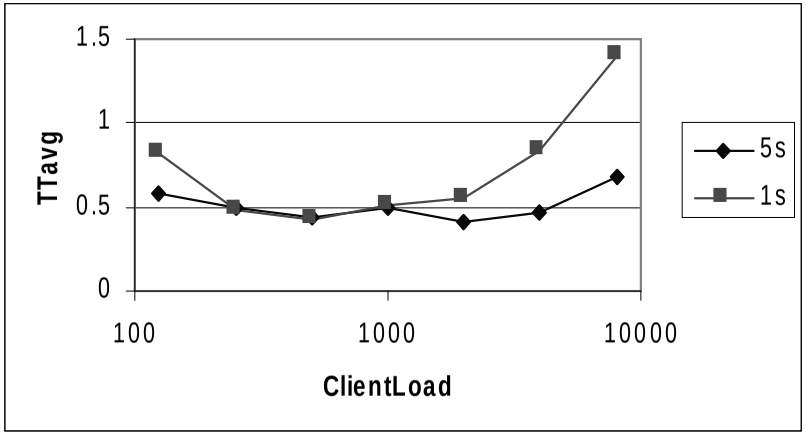
Time Response

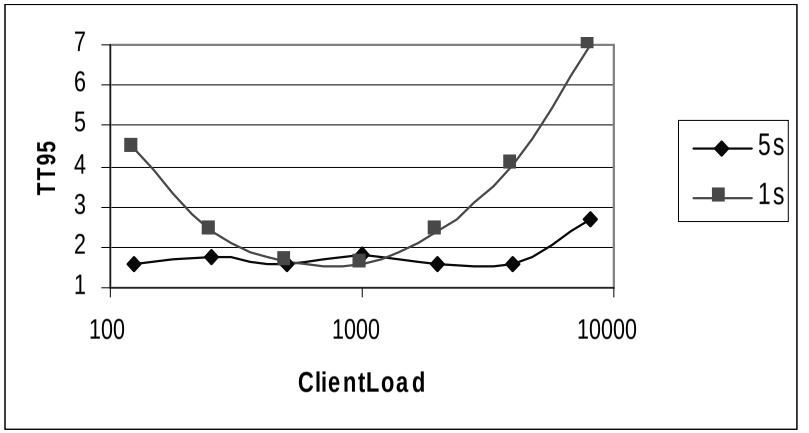


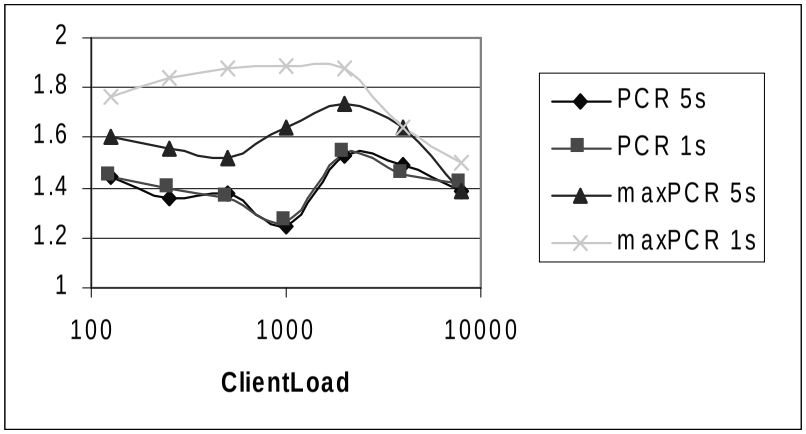


- Other measurements:
  - TTavg, TT95
  - **P**rocessing **C**apacity **R**atio - ratio between the total client processing capacity of available servers and the new client arrival rate

Algorithm Scalability I

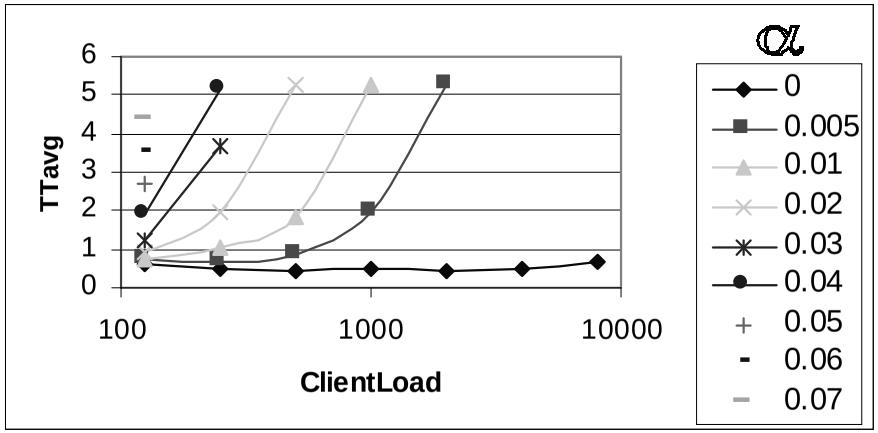


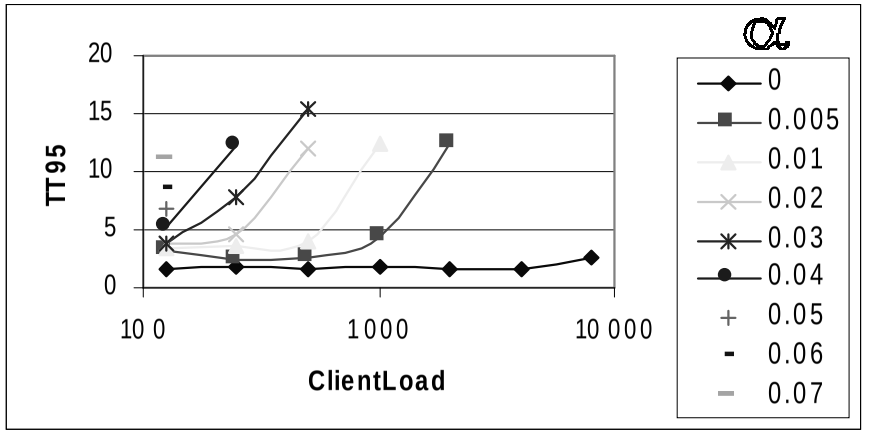


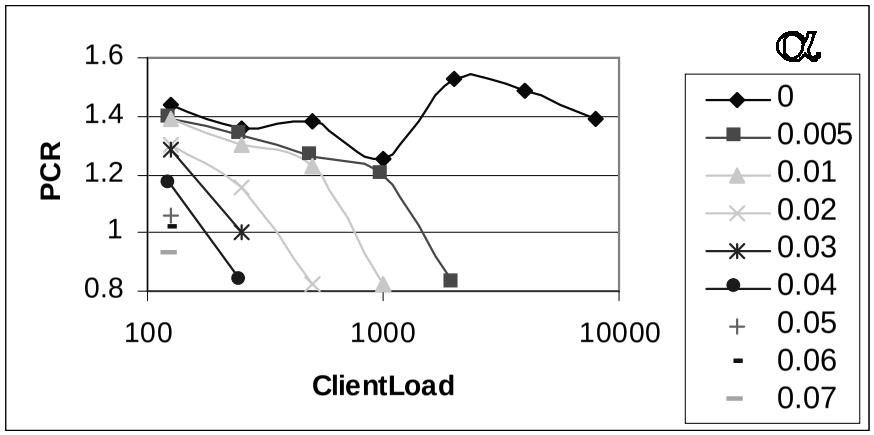


- MaxCliQ= 15, Timeout= 1.5, T_{clone}= 1
- Two visible effects:
  - Location server saturation for high Clientload
  - Slow response for low Clientload

Algorithm Scalability II







- MaxCliQ= 15, Timeout= 1.5, T_{clone}= 1, five initial servers
- Server Inter-synchronization:
  - ServiceTime= 0.1*(1 + α*(N-1))
- Max ClientLoad= 1/(0.1 * α)

Conclusions

- Mobile agent based applications may adapt to peaks of client requests and scale with the client load, and still satisfy some service requirements
- The system support services must also scale well otherwise they become the application’s bottleneck
- The mobile agent semantics provides a sound basis to satisfy most of the system requirements - network centric applications

Next Steps

- Multi-invocation client interactions
- Combined effects of the location server and application server algorithms