Scalable Service Deployment on Highly Populated Networks

Luís Bernardo & Paulo Pinto

IST / INESC

Topics

- Proposed Architecture
  - » System Architecture
  - » Location Service
- Server Deployment Algorithm
- Simulation Results
  - » Time Response
  - » Parameter Settings
  - » Inter-synchronised systems
- 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

System Characteristics

- the application servers must be prepared to operate during peaks of the client load
- new application servers will be deployed when the existing servers are overloaded
- clients may interact with any of the application servers
- servers are implemented using mobile agents

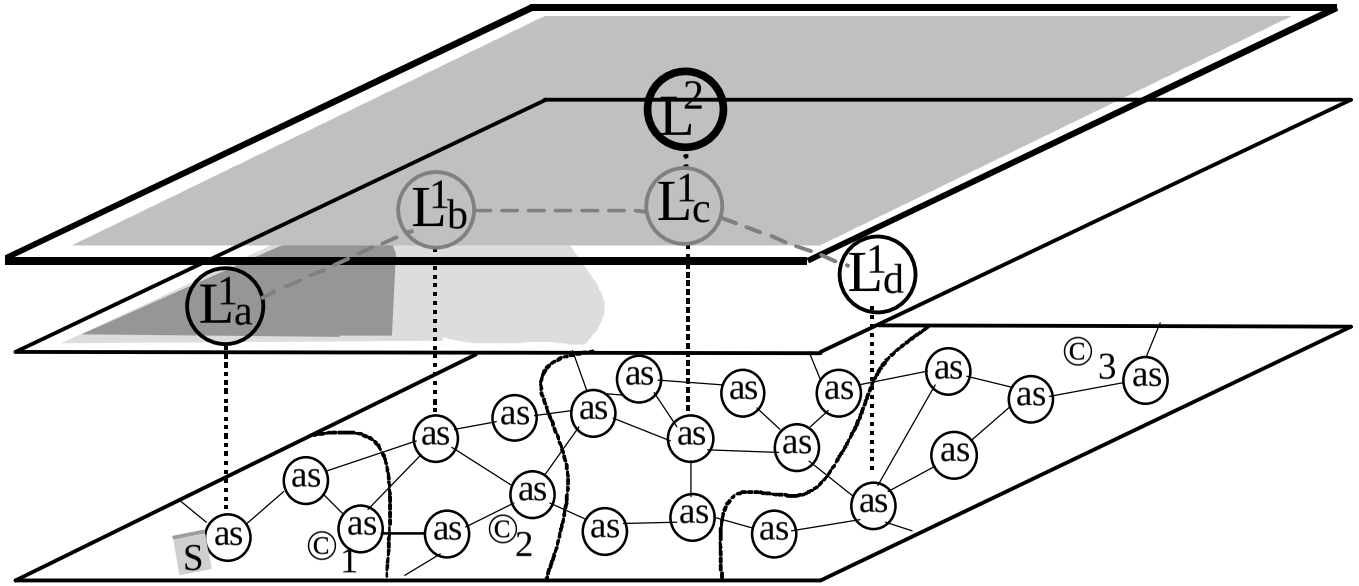
Why mobile agents ?

- Remote creation of server agents of new service types (service provider specialisation)
- Provide a ubiquitous platform of systems deployed world-wide where any client or server agent may run
- **Problems:**
  - » some applications cannot be implemented (large amounts of data or use static resources)
  - » mobile agent systems are still under development

Location service - objectives

- Resolves an application name to the nearest server reference
- Balance load between servers
- Provides updated information : server information values are not cached
- Scalability is achieved by:
  - » hierarchy of meshed levels of location servers
  - » dynamic hierarchical levels and number of L-servers

Location service - example



Application Server Deployment

- Algorithms control the number of servers and their location
- Servers measure their local load:
  - » server idle time
  - » the client input queue length (if possible)
- Top threshold (**MaxCliQ**) >> server creation
- Bottom threshold >> server destruction

Server Creation Algorithm I

- Is 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

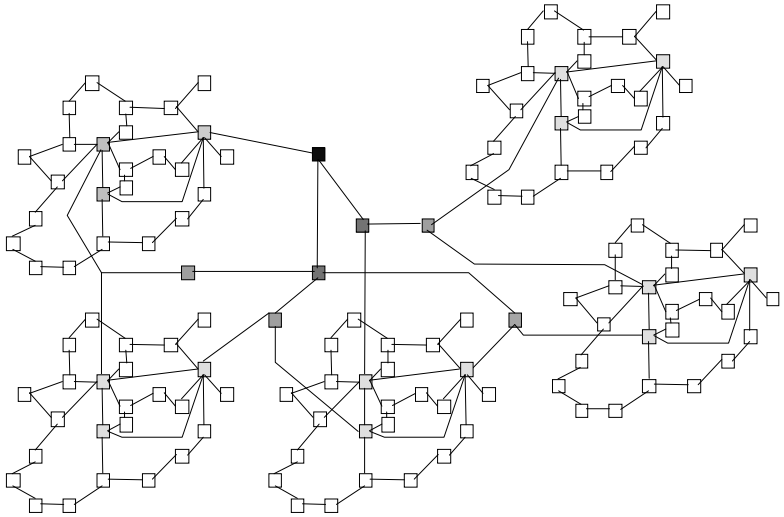
Server Creation Algorithm II

- If a server is crowded with binded clients, the clients must be redistributed between all the available servers:
  - Partial unbinding - clients are removed after staying binded to a server more than **Timeout** tics
  - Total unbinding - servers create a new interface if the queue length is higher than a threshold value

Server Destruction Algorithm

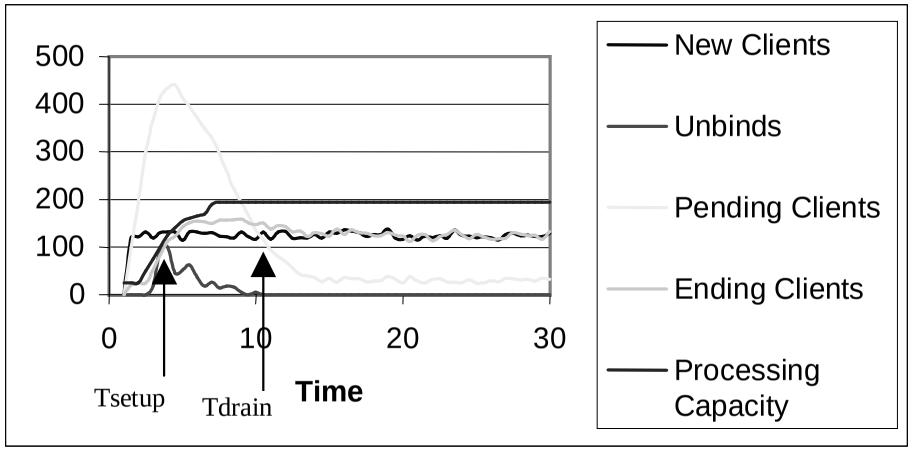
- Application servers are destructed using a market-based control technique
- Servers send “Request for bids” for all the servers within a range, and wait for “Bids” during an interval
- The server delegates its clients adjusting the location service information

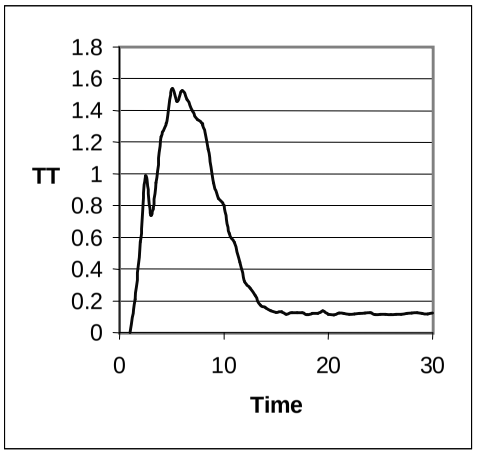
Simulator



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

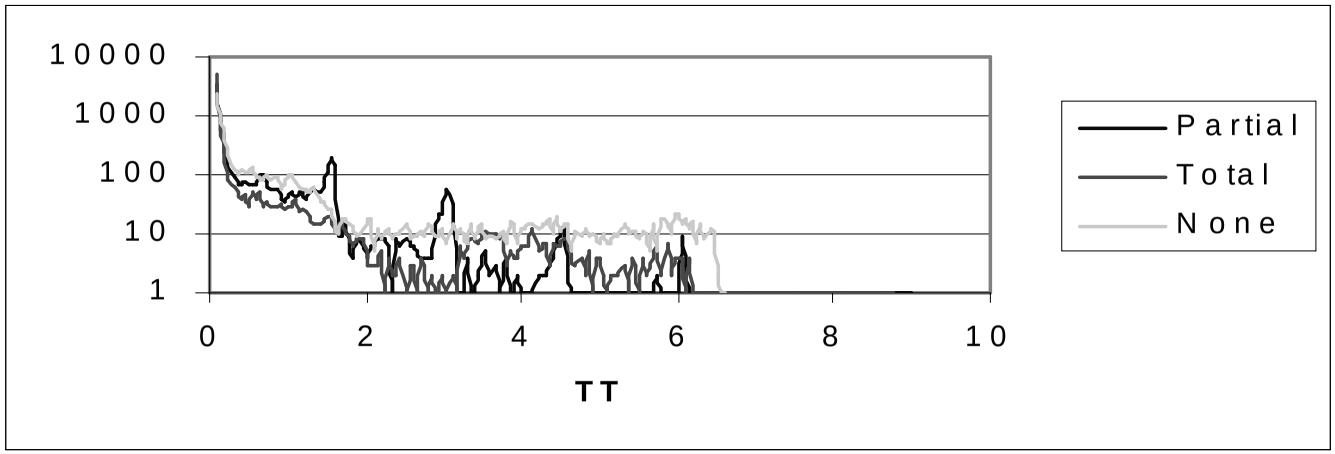
Time Response





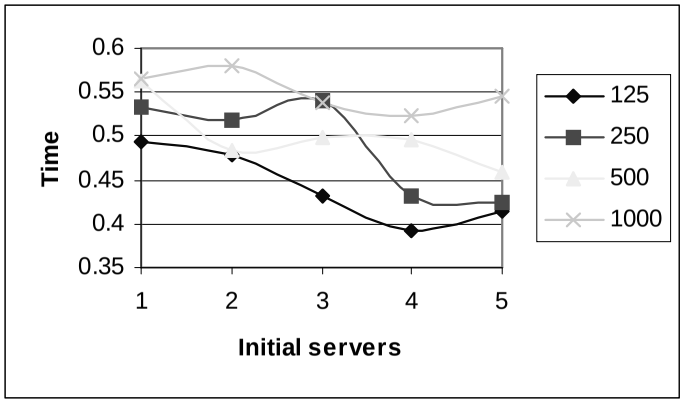
- Other measurements:
  - » TTavg, TT95
  - » **P**rocessing **C**apacity **R**atio - ratio between the total client processing capacity of available servers and the new client arrival rate

Client Redistribution Effect

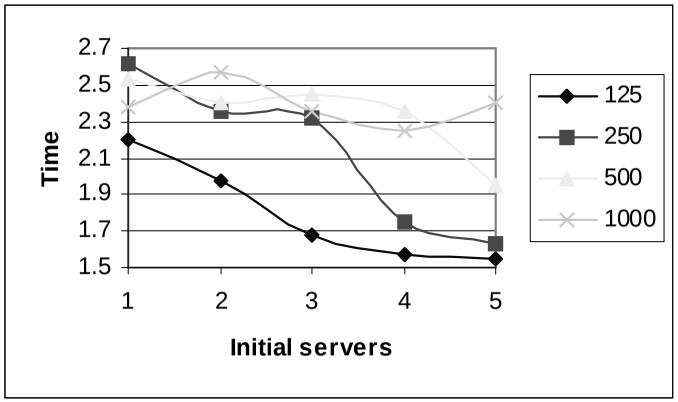


- Partial - client timeout - TTavg= 0.52; PCR= 1.64
- Total - interface reset - TTavg= 0.42; PCR= 3.86
- None - no redistribution - TTavg= 0.93; PCR= 1.38

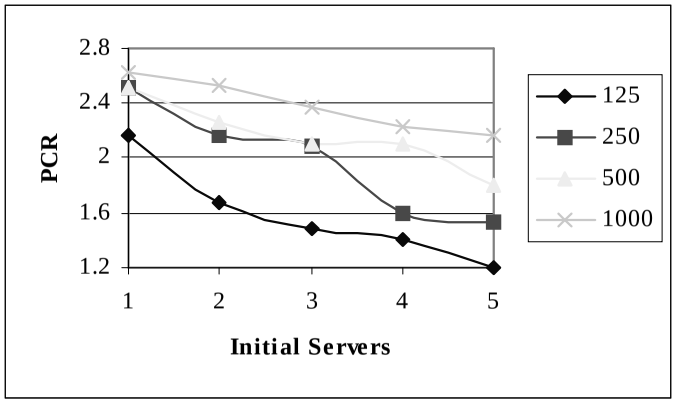
Algorithm Scalability



TTavg



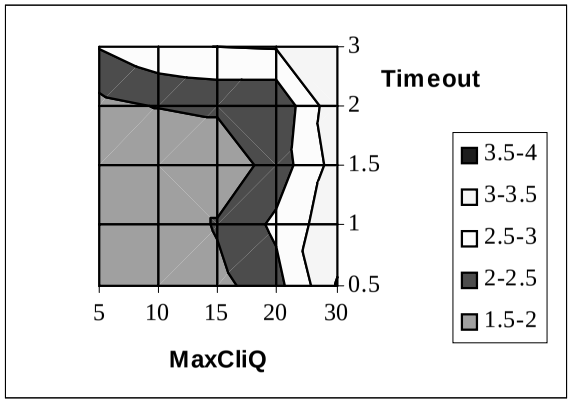
TT95



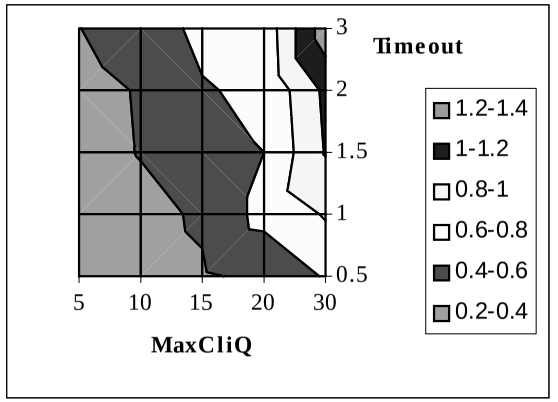
PCR

- MaxCliQ= 15, Timeout= 1.5, T_{clone}= 1

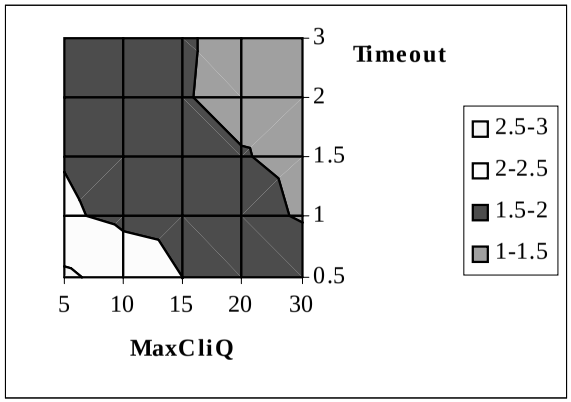
Algorithm Tuning Parameters



TT95



TT_{avg}



PCR

- ClientLoad= 250; 5 initial servers
- Trade-off:
  PCR vs. TT95,TTavg

Synchronised Systems

- Extra service specific synchronisation may be needed (e.g. to maintain consistency of shared data)
- The application service time will be higher, and dependent on the number of servers
- With small modifications the algorithm can be used to control the server deployment keeping TT_{avg} , TT95 and PCR bounded

Conclusions

- Applications may adapt to peaks of client requests and still satisfy some service requirements
- The parameters of the server deployment algorithm and the initial number of servers can be tuned for a given set of requirements, if some system parameters are known (application server response time, clone creation time, etc)

Next Steps

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