Scalability Issues in Telecommunication Services

Luis Bernardo[1], Daniel Crespo[2], António Marques[2], Paulo Pinto[3]

[1]IST - Instituto Superior Técnico
Inesc, R. Alves Redol 9
1000 Lisboa, Portugal
lflb@inesc.pt
Voice: 351-1-3100345
Fax: 351-1-31458434

[2]IST - Instituto Superior Técnico
Inesc, R. Alves Redol 9
1000 Lisboa, Portugal
{drcr,amsm}@cher.inesc.pt
Voice: 351-1-3100342
Fax: 351-1-31458434

[3]Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2825 Monte da Caparica, Portugal
pfp@uninova.pt
Voice: 351-1-2948515
Fax: 351-1-2954461

Abstract

The development of the network technology is allowing the introduction of complex interactive services. The implementation of complex services available to millions of users on very large networks introduces strong scalability requirements, which are not addressed by today's technology. This paper describes some of the problems, and presents an overview of a possible solution, based on the use of mobile agents. A location service model and a partial implementation are presented, which scale to large number of clients and still allow a high rate of updates. The chosen application is a scalable audio on demand service. Audio servers are dynamically deployed in response to the degradation of quality of the audio received by the clients. In result, the service is able to adapt to overloaded servers and link congestion.

I. INTRODUCTION

The scalability of a service implementation to large number of users is related to the system ability to provide a guaranteed quality of service to an unpredictable huge number of clients in an affordable way. This ability is not only related to the limited number of service servers, but depends on the bandwidth and system limitations on overall server support as well. An expensive solution would be to create a very high number of service servers, capable of handling the maximum possible number of clients, and distribute them in the network to avoid overloaded network regions with service interaction messages. However, this is not affordable to all services. For instance, for services which have very large (unpredictable) peaks of requests on concentrated intervals of time (e.g. real-time sports brokering). A common solution is to prepare the system to handle a static maximum number of clients, above which, the service may fail. For instance, implementations based on magic routers [1] or Transaction Processing monitors (for instance [2]) rely on a predefined set of machines where tasks can be processed, and on load distribution facilities. Besides the maximum processing limit, the previous examples have limitations related to the use of "system" generic load balancing facilities, not adapted to some classes of services. For instance, when multimedia streams (or other network resources) are used, the location of the servers must take into account both the bandwidth needed (or the network resources location), and the points in time when such servers can be relocated without degrading the quality of service measured by the clients.

However, new technologies such as mobile agent systems [3] and active networks [4] bring the possibility of creating and destroying servers in real-time, at any node on the network. The servers are able to collect information about the network status, and adapt to machine and network load dynamically. A new solution was proposed on [5][6]. It implements service servers with mobile agents, and controls their deployment using the client load information. By measuring their load, and by storing a statistic map of the client's origins, the service servers are able to decide when and where they create new servers, or when they destroy them. Simulation results proved the algorithm scalability, with bounded client service delay and atomic client-server interactions. This paper extends the algorithm to handle session oriented client-server interactions.

This paper is organised as follows: Section II presents an overview of the model of service platform. Section III and IV present respectively a prototype of the service platform and of an audio on demand service that was recently implemented in Java, using the Voyager platform [7]. It takes into account not only the scalability problems referred above and in [5][6], but also the Quality of Service aspect of interactive multimedia.

II. SERVICE PLATFORM

New system support services are needed to implement the proposed service implementation. A ubiquitous platform of agents systems (agent virtual machines) must be available, to allow the distribution of service servers anywhere on the network.

The service name resolution service (called here location service) is one of the most important system support services. It resolves service unique names to the nearest server references, and balances the clients amongst all the available nearby servers. It also helps balancing the service servers by all the available agent systems (agent virtual machines) in a network region, by providing a view of the agent systems and network state to the servers.

On the proposed service implementation the location service has strong scalability requirements, not supported by most of the present name and directory services. When service servers are being created, their reference must be considered for load balancing on a very short time. Also, the location service must be able to respond to peaks of client searches and service server updates. In consequence, it must not rely on caches of information, because of the costs of cache updating. A new approach is needed, which scales with the number of searches, has a good update performance and is more robust (does not rely on a single primary name server).

A solution was proposed [8], based on the use of self-configuring location servers (L-servers), which implement a dynamic location network (network of L-servers). The service unique names (SUI) are resolved by performing searches on a path of L-servers, from the local L-server until an L-server that has a complete reference to a service server. The routing of the search on the location network is supported by the service routing information disseminated on the network. Each service routing entry includes incomplete service information which routes to a neighbour L-server with more information, or a complete service information with a server interface reference.

The location network has a hierarchical structure. However, at each hierarchical level, L-servers are connected on a complementary flat network. The horizontal service information dissemination is controlled dynamically. Whenever the client search load overloads the higher hierarchical level L-servers, the horizontal service information dissemination is increased incorporating more lower level L-servers, to create alternative paths to resolve the SUI. The server interface information can also be replicated on the nearby L-servers, to adapt to overloaded local L-servers. Additionally, L-servers can be created or destroyed when their load goes above or below the maximum and minimum threshold values, after the service dissemination setting.

The range where each service server is relevant is service dependent. For a local service (e.g. a car parking information service) or for a global service which serves a huge number of simultaneous clients (e.g. an election information service), each server will be responsible for a restricted region of the network (the server domain). On the second case, the global coverage would result from the overlapping of all the service server's domains. Servers control the service information dissemination range. In result, the location service scalability with the number of updates is improved by reducing the dissemination costs, specially on the top hierarchical level on the location network, which will be used mainly by the global, less used, services. The use of lazy complete updates, possible by the creation of temporary service path chains, also improves the location service ability to scale with the number of updates.

The structure of the location network is dynamic. It changes not only with the L-server load, but also with the server domain ranges, the network conditions and with the density of agent systems. The number of hierarchical levels depends on the largest service domain requested and on the service load. The network at top hierarchical level supports the maximum requested server domain range. When a service becomes overloaded, the widespread creation of service servers on the network, and the horizontal dissemination of service information at the lower hierarchical levels (induced by the client searches) will concentrate the service name resolutions at these lower levels.

The present prototype includes a partial implementation of the algorithm. It implements a flat structure (one hierarchical level). For services implemented using the server dynamic deployment algorithm, used on limited networks it presents a good performance. However, this implementation is inefficient for global services on huge networks implemented with a low number of servers, because of the service information dissemination costs.

III. SERVICE PLATFORM IMPLEMENTATION

The prototype was implemented in Java, using the Voyager platform. Voyager offers a distributed application environment, which includes an ORB (Object Request Broker), a set of interaction protocols (including RPC, events, etc), the support of object mobility, CORBA interoperability, etc. The Voyager platform was extended with the location service support.

The location service is implemented using two kinds of components: the Network Service Proxies and the Location servers (L-servers). The Network Service Proxy objects run on every agent system and provide an interface to the global location service. The L-servers are implemented using mobile agents, and run on the agent systems in parallel with the other service agents. L-servers keep all the server information from a set of network service proxies (the L-server domain) and additional information forwarded from external L-servers.

The network service proxies define a reference network, which is configured by the system manager. Figure 1 shows an example. This network changes when a partition occurs on physical network links, or when proxies are added or destroyed. The link's "distance" between network service proxies defines the service metric, usually associated with bandwidth of the link or geographical proximity (but may have other purposes like, for instance, political proximity between two filial companies).

The location network is created dynamically following the network service proxy links. When a new network service proxy starts, it tries to connect to its neighbours (specified in a configuration file), getting a state report from the active ones. L-servers are created by the network service proxies when all registrations from a network service proxy to the existing L-servers fail (an L-server cannot support more than a maximum number of network service proxies, within a maximum distance). The network service proxy registration procedure is also ran when a physical network link or a L-server fails, originating the reconfiguration of the location network, with the possible creation of new L-servers. After a location network reconfiguration takes place, service information is re-disseminated to restore the information coherence.

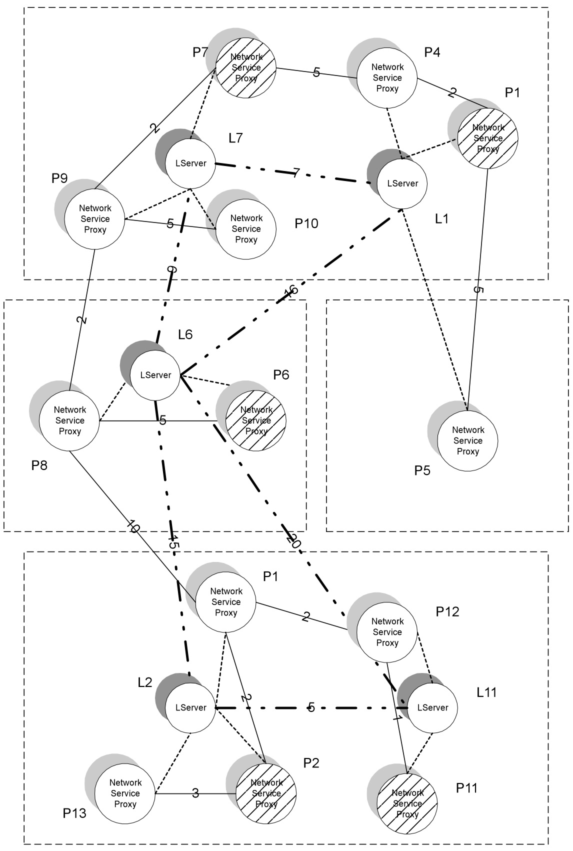

Fig. 1 Service platform structure

Servers register their service interfaces explicitly with their local network service proxy. Clients also register with their local network service proxy, to query the system. If the service request cannot be satisfied on the local network service proxy (on a server running on the same agent system), the local L-server is searched. Depending on the local L-server's information, the search can fail (if no reference to the service is found), the network service proxy can forward the search to another L-server, or the L-server may have an interface reference of a service server.

A monitoring tool was implemented to inspect the network service proxy network and the location network.

Figure 2 shows a screen with: a sub-set of the L-servers running, the network service proxies in the L-server's domains, the service servers registered on each proxy, and the clients using it. Additionally, it can be used to enquire the links of the network service proxies (represented on fig. 2 for proxy 9), the L-server links, and additional client and server information.

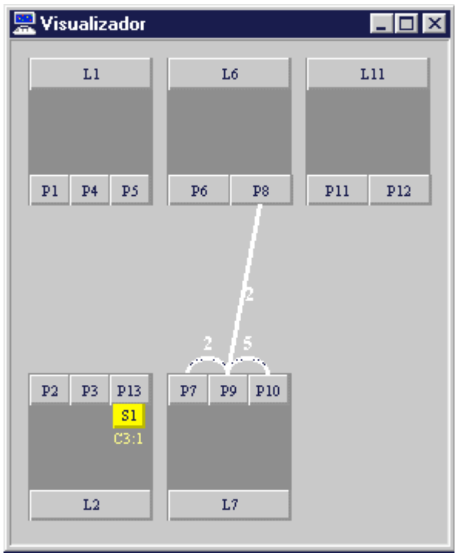

Fig. 2 Service network monitor

IV. Audio on Demand Service

A distributed interactive audio on demand service was implemented using the service platform presented above. The use of session oriented client-server interactions introduce some modification in relation to atomic interactions. The location service load-balancing feature is still needed, to distribute the servers on the network, and the clients amongst the existing servers. However, during a session, the network conditions and agent system load may change in a way that violates the agreed quality of service for the service. In result, a client may have to be connected to a new server, to resume the session with the required quality. Also, for the audio on demand service, an improved server selection algorithm is needed, which takes into account the distribution of the audio files.

The service is implemented using two kinds of components: the audio server agents (ASA) and the clients (C). The audio server agents offer the audio stream and control service interface and keep an audio database. There are two kinds of audio servers: the primary servers are created by the service providers in the locations where the main audio databases are; the secondary servers are deployed dynamically by the server deployment control algorithm, and keep temporary audio databases. The servers support an additional interface for distributing the audio files on the network.

The audio server agents run a co-ordination protocol to control the client distribution amongst them. Clients log into the audio on demand service by resolving the "Audio on Demand" name using the location service, and connecting to the returned server interface. From it, the client gets the initial information from the service, including a list of music available. When the client selects an audio to play, this audio server is responsible for selecting an audio server from where the client may get the requested media. If this server is already overloaded, and cannot support another audio stream, it selects another server or, if no one is available, it starts a new one. A modified contract-net protocol [9] is used to select the server. The contractor server sends a "request for bids" (which includes the audio file name) to the group of audio servers within a maximum range, and waits during an interval for "Bids". All the receivers with enough resources to support another client answer with a "Bid", stating if they have the file and their load, and incrementing their load meter (one extra client) during the request for bids validity time. After the interval, the contractor server will select the nearest audio server (with the audio file if available). If the selected audio server does not have the file, it downloads it from another audio server, and stores it temporarily in its database. If the original audio server agent can still support another audio stream but does not have the audio file, it will first search for another audio server within its local location server, for a free server with the audio file, before downloading it. When no alternative server replies to the request for bids, the contractor server starts the creation of a new audio server. The contractor asks the client's location server for the best agent system (the less loaded) on its region, and creates an audio server agent replica there, initialised for downloading the audio file. After an idle period, the contractor server will rerun the co-ordination algorithm, with the new audio server running, which will be selected. The client is transparently handoff between servers, using control message sent from the contractor audio server. Figure 3 shows an example of a possible service deployment, where a client as been handoff (1→2) to a further audio server in result of the co-ordination protocol. Secondary audio servers are destroyed when they are idle for more than a threshold time.

The service will adapt to network or machine overload. When the reported client's Quality of Service faults are above the maximum limit (in result of extra load on the audio server's machine or bandwidth limitations), the service servers will reconfigure the control and multimedia connections. The server selection procedure is run, and some clients might be transparently handoff to other servers (possibly to new ones). The handoff is controlled by the overloaded server, which continues to play the audio track. When it gets a reference of another server prepared for transmitting the audio, it sends a control message to the new audio server, to resume playing at the same position. After detecting a quality violation, the client will try to raise the receiving buffer occupancy, to have a lower probability of disrupting the presentation. However, this approach may fail and some disruption may be noticeable in the client if the origin of the problem is network congestion. That may also happen if the audio stored in the receiving buffer is shorter than the new server starting delay (message control delay plus the time to start sending the audio packets plus the network delay until they reach the client).

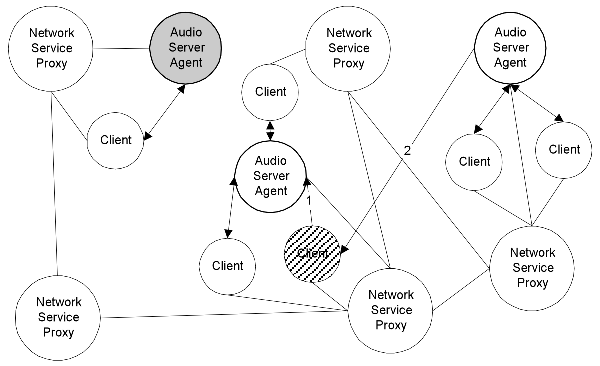

Fig. 3 Service deployment on the network

A co-ordination algorithm is also used to distribute the audio files on the network. When an audio server needs an audio file, it sends a "request for audio" message to all the audio servers, which will answer if they have the requested file. The audio file is downloaded from the nearest audio server (the fastest answer). As fig. 4 shows, a distributed audio database is dynamically created in result of the audio server creation and destruction.

The client application offers a graphical interface to the service. It may run as a local application or as an applet. However, the second approach presents today a severe performance penalty due to the need of sending all messages from the client to the server over a communication gateway at the web server's machine, to overcome the browsers' applet security restrictions. The present prototype still needs the configuration of the network proxy server and the monitoring client, although it can be initialised automatically. It allows the user to browse the list of audio files, and play or stop a presentation.

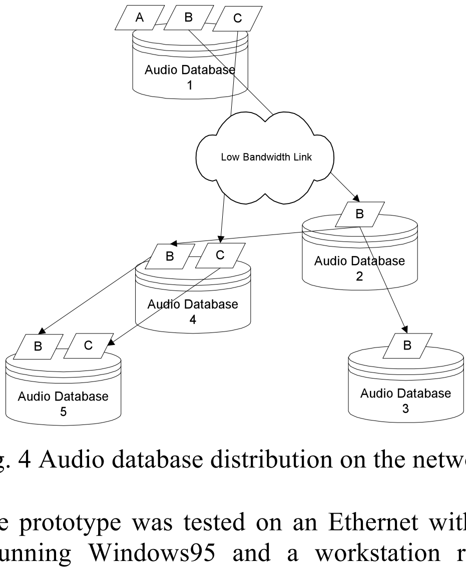

Fig. 4 Audio database distribution on the network

The prototype was tested on an Ethernet with three PCs running Windows95 and a workstation running Solaris. Tests with the set-up of figure 1 using normal clients and "silent clients" (which do not play the music) showed that more than a hundred clients can be supported. The length of the audio server's class is 66

Kbytes. The average time measured for the creation of an audio server agent was 150 ms (including the loading of the server's class and the server agent activation) plus the audio file transfer time. For an audio file size of 3 MBytes, the total audio server creation time was in average 7 seconds.

V. Conclusions and Future Work

This paper presents an implementation of an audio on demand service, where servers are dynamically deployed. The resulting dynamic structure of the audio servers is able to adapt to failures, and to reconfigure the service transparently, as long as the primary audio server agents, with the main databases, remain operational.

Presently, one of the limitations for using the presented system on a large-scale network such as Internet is the available bandwidth. However, it is expected that the future global networks will have a much higher bandwidth available. Some trial networks, such as Internet 2 [10], already go in that direction. Although, they still lack a new set of improved protocols and services, like the location service proposed on this paper, to make a more effective use of the bandwidth.

The current prototype is still crude, and has some limitations, which should be improved on next versions. The location service should implement the entire model, with multiple hierarchical levels, to improve the overall scalability. The algorithms were already constructed for the simulation implementation reported in [5], [6] and [8]. Several improvements can also be done, at the audio on demand service. The present version sends the audio data in μ law audio format. The compression of the audio data (using for instance, the MPEG audio layer 3 [11], which compresses 1/12 for CD quality) would reduce the bandwidth used for the same audio quality. The bandwidth limitations could be overcome by an improved audio file distribution algorithm, which autonomously creates audio database replicas and audio servers using network idle time or in response to bandwidth limitation's information, instead of waiting for client requests. The audio server creation time could be substantially reduced.

VI. References

[1] K. Delgadillo, "Cisco DistributedDirector", Cisco White Paper, 1999. http://www-europe.cisco.com/ /warp/public/751/distdir/dd_wp.htm

[2] BEA, "TUXEDO White Paper", 1996. http://www.beasys.com/Product/tuxwp1.htm

[3] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Onu, M. Oshima, C. Tham, S. Virdhagriswaran and J. White, "MASIF: The OMG Mobile Agent System Interoperability Facility", *Proceedings of the 2nd International Workshop on Mobile Agents (MA'98)*, Stuttgart, Germany, Springer LNCS Vol. 1477, pp. 50-67, September 1998.

[4] D. Tannenhouse and D. Wetherall, "Towards an Active Network Architecture", *ACM Computer Communication Review Vol. 26 N^o 2*, pp. 5-18, April 1996.

[5] L. Bernardo and P. Pinto, "Scalable Service Deployment on Highly Populated Networks", *Proceedings of the International Workshop on Intelligent Agents for Telecommunication Applications (IATA'98)*, Paris, France, Springer LNAI Vol.1437, pp. 29-44, July 1998.

[6] L. Bernardo and P. Pinto, "Scalable Service Deployment using Mobile Agents", *Proceedings of the 2nd International Workshop on Mobile Agents (MA'98)*, Stuttgart, Germany, Springer LNCS Vol. 1477, pp. 261-272, September 1998.

[7] Voyager home page: http://www.objectspace.com/ products/voyager/index.html

[8] L. Bernardo and P. Pinto, "A Scalable Location Service with Fast Update Responses", *Proceedings of IEEE Globecom'98*, Sydney, Australia, pp. 2876-2881, November 1998.

[9] R. Smith and R. Davis, "Frameworks for cooperation in distributed problem solving", *IEEE Transaction on Systems, Man and Cybernetics* SMC-11, pp. 61-70, 1981.

[10] Internet 2 Project, Preliminary Engineering Report, January 1997. http://www.internet2.edu/html/ engineering.html

[11] ISO/IEC 11172-3, Information Technology – Coding of Moving Pictures and Associated audio for digital storage media at up to about 1.5 Mbps – Part 3: Audio