A SCALABLE LOCATION SERVICE WITH FAST UPDATE RESPONSES

Luis Bernardo and Paulo Pinto
IST - Instituto Superior Técnico, Lisboa Portugal
Inesc, R. Alves Redol, 9 P-1000 Lisboa Portugal
{lflb,paulo.pinto}@inesc.pt

ABSTRACT

This paper addresses the issue of defining a location service suitable for very dynamic and highly populated networks (millions of users), where services might experience highly correlated peaks of traffic or synchronized access to specific servers. Mobile agent technology is flexible enough to solve the major problems, allowing the dynamic deployment of new application servers when needed. But it requires an adequate location service to increase client adaptability. This paper describes a very dynamic location service that can adapt both to server and client needs and to the load on the system. Sets of simulations were performed to study the effect of the location network structure on the load distribution.

1. INTRODUCTION

Most of the current distributed applications rely on a location service to match the client with the server objects. Most of the common solutions for location services respond to increases on the load by system administrator reconfiguration of hierarchical divisions or by replication. These techniques are not adequate for large networks, specially if their applications can produce peaks of traffic. A common characteristic of some applications can be the possibility of generating highly correlated peaks of traffic due to client interaction with the servers. Examples are easy to envision: applications based on interactive TV interfaces, where contests, promotional prices announcements or audience queries may synchronise the sending of requests to particular servers; real-time sport brokering; teleshopping; etc. Current location service solutions do not apply properly to large networks with billions of users and millions of services for two main reasons: the application technology will most likely rely on highly variable server groups to adapt to client load peaks, producing a non-static environment; and the location service itself might suffer from overload and become a critical point on the system.

Our proposal for an architecture for this kind of networks relies on the server ability to monitor client load and deploy server clones before an expected service peak, or as a response to a unpredictable overload condition. The deployment will be done to a region close to the majority of the clients. Servers are implemented using mobile agents. The application server deployment algorithm is presented in [3][4]. Clients look for a precise service based on identification and not on characteristics [9] (price, availability, etc.). They use the location service to resolve unique application names to single server references.

On systems such as ours, the location service itself must use the same technique to adapt to its own load. Moreover, its major task must be efficient: it will deal with frequent updates of server offers and possibly bursts of updates or lookups. The updates must be made available to the clients very swiftly and in a coherent manner to allow an overall load balance on the system. These requirements imply a very dynamic structure of servers and information within the location service and are the focus of this paper.

2. LOCATION SERVICE

The network provides a ubiquitous platform of agent systems, in which any mobile agent (server or client) can run. Each agent system is tied to a location server (L-server) (running locally or on a nearby system), where all the interfaces of the local agents are registered. The L-servers are connected to others to offer a global location service.

The location service supports application server replication by resolving globally unique application names to server interface references. Application names form a flat name space, just like in [14]. When a client searches for an application name, the location service helps in the binding process (the association to a server) directing it to the nearest server. If the L-server knows more than one server, it will split the client traffic. If it knows that a new, and closer, server was created it will start using the new one, and propagates this information. When a client comes for resolution, it will get the best answer for that moment.

2.1. EXISTING LOCATION SERVICES

The location service must scale to a large number of clients. It must deal with a large number of updates resulting from server clone creation and server

mobility. Such requirements invalidate some of the current technical solutions, based on static hierarchical systems. Present solutions fall in two general categories: home tracking node approach and a distributed tracking approach.

On the first approach, L-servers are statically associated (universally known) with the tracking of each entity (service, agent, etc). Notorious examples are the classical name (DNS [2][5]) or directory (X.500 [6][10]) distributed systems and trading services (RM-ODP Trading function [7]). DNS completely tights the identification of the information with the identification of the L-server that has it (domain-dot-domain). It provides both divisions on the information space and a path to reach the L-server. Others have less tight links between the information and the L-server location but need extra information to locate the L-server (with schemes such as distinguished name mapping, federation of subspaces, etc). Most of the mobile communication networks (GSM, IS-41 [1]) also fall into this first group. The use of this first approach has several drawbacks: a "home" L-server may create an access bottleneck, a single point of failure and may derive in bad bandwidth usage. To avoid frequent remote update of server locations, the home L-server solution might be extended using "chained forward pointers" [1][13] to define a path from the home L-server to the server location (when the agent moves, a record is kept at the previous L-server with a pointer to the current one). On this last case, the use of more localised update operations (during application server updates) results in a higher susceptibility to node and link failures. Checkpoint techniques with the home L-server might reduce the problem but have costs.

The second approach, the distributed tracking approach, relies on a search mechanism among the L-servers to locate the information. Information is kept at the server's nearest L-server. Some routing-like information must be disseminated to limit the search space. WhoIs++ [15] supports a yellow pages service of users based on a hierarchical structure of L-servers (index servers), where each L-server forwards a list of attributes and the associated set of values towards its higher hierarchical L-servers. The lists are used during searches to exclude subtrees. Further improvements are presented on Globe [14]: a hierarchical structure is proposed where the higher-level L-servers have pointers to the entire space of offers (offers use a flat name space for groups of objects), allowing the selection of the nearest application server. But this solution still has some scale limitations, even considering the splitting of the offer spaces by several L-servers and by filtering the void updates from lower levels. The huge information space and the size of the network may turn the task of managing the root L-servers and the hierarchical subdivisions into an almost impossible task.

A common limitation to most of the previous approaches is the use of static, administratively defined, location server networks which define a static number of L-servers and the links between them. Apart from node failure susceptibility considerations, it limits the scale to which the location service may operate due to bandwidth or processing limits.

In both categories, caching schemes can be used to limit client accesses to the L-servers. However, they are inefficient for our requirements. If the values change very rapidly, and in unpredictable ways, the cached values will have a very short life, or may hide the application server's reconfiguration from the clients. The use of cache invalidation methods [8] would introduce non-scalable world-wide updates.

2.2. LOCATION SERVICE MODEL

Our proposed system is based on a dynamic location network, which follows a distributed tracking approach. The client performs searches on a step-by-step basis, through a sequence of L-servers. The routing information for the path is based on service hints. Hints are either the full application name or incomplete information about the application just to direct the search to another L-server. The objective is to keep hints small and easy to update.

The location network is structured as a mixture of a meshed and a hierarchical structure where L-servers at each hierarchical level interact with some of the others at that level and (possibly) with one above. Higher hierarchical levels always have incomplete information about the available services. Additionally, the hierarchical structure and the range of the mesh change dynamically according to the load of the system, and to the size of the "server domains" (see below). The cost of both decentralising the search (not using a unique root) and having partial information is to have a longer client search if servers are far away (with the possibility of failure). However, within a limited range, it simplifies the search algorithm, and the overall system is simpler and scales better.

The location service uses two kinds of components: the Local Location Proxy (LLP) objects run on every agent system (AS) as an interface to the global location service; and the Location Servers (L-servers), which are mobile agents, run on some agent system and keep the service information.

LLPs provide an indirection level, which hides the location network dynamism from the clients. LLP objects support the location-agent system network represented on the lowest plane in figure 1. Their knowledge of the topology is limited to their neighbours, and to the conditions of each link. This network is static and only changes by system administrator intervention, or when a new agent system is connected (or disconnected). Each link can mean geographical proximity, enterprise proximity (e.g. linking two multinational delegations), or other proximity. The number of agent systems between L-servers defines the link ”distance”, thus defining a network metric.

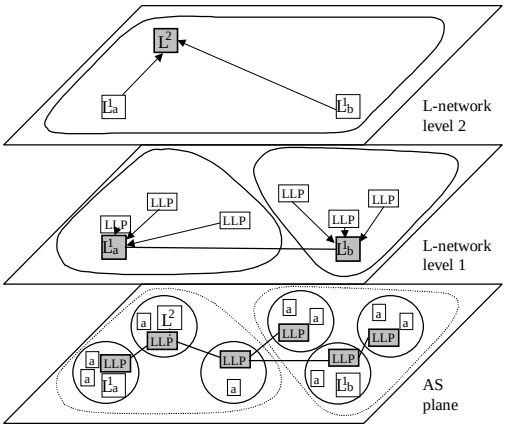

Figure 1: Network Model

The L-server network (L-network) is constructed on top of the agent systems’ network using a topology protocol. The resulting network is dynamic (in number of L-servers and in network structure) adapting itself to the load conditions and possible failures of nodes and links. The service trading protocols use the entire L-network, providing support for service hints dissemination and information queries. Several hierarchical levels of meshed networks might exist to deal with different ranges of service offers.

LLPs exchange identity information, and the identity of the L-servers they know, with other LLPs within a certain range, using a kind of limited range “link state” protocol [11]. Each LLP choses a preferred L-server. LLP objects keep a record of the neighbour L-servers, selecting an alternative one in case the preferred fails. To reduce the overhead of the protocol, the L-server names are valid on a limited range of the network. New LLPs join the system by exchanging initialisation packets with the neighbours, and binding to the nearest L-server. The neighbours may be defined by the system manager (typically on a fixed network), or by a specific resolution server on the network.

Each L-server knows all of its LLPs (or descendant L-servers), and may instruct some or all of them to join another L-server, if for example, its domain gets too big. L-servers get to know their neighbours both by receiving the information forwarded by the LLPs, and by running a similar “link state” protocol at their level (which is also used to feed information one level upwards). The division of L-server domains (or merge, or move of L-servers) will involve only surrounding L-servers. Information messages are exchanged locally, to update the L-server identities. The old identity is kept until the propagation of the new identities to all the surrounding L-servers, to guarantee the overall system coherence.

2.3. TRADING

During an offer, registration servers specify the range where the service will be known (which is related to the relevance of the service). For instance, a city’s transit information is relevant only for the drivers in that city. A weekly lotto broker service can be offered by a single server, known to a broader region of the network during quiet days, and a myriad of servers known very locally on the day just before the draw. Pricing schemes could be a deterrent to artificially large domains. The metric used to define the range is the distance in number of agent system hops. This approach reduces the amount of information that needs to be known at large areas, producing a more scalable approach. However, only clients who search for the service within the server range will be able to follow the path to it.

2.3.1. SERVICE ROUTING

Routing information (service hints) is disseminated between L-servers horizontally (possibly at more than one hierarchical level) and vertically to higher hierarchical levels.

Horizontal dissemination involves L-servers sending information packets to their neighbours. The receivers process the packet, discard the elements within the offer list which are out of range, and forward it to the next L-server further away from the origin if the previous information recorded was modified. Service hints on the forward information packets are successively simplified from a complete information (with the application name and server interface reference); to the application name and a L-server name; to a hash value of the application name and a L-server name. Service hints are removed from the information packet when they reach the range limit, or, do not modify the information at the L-server. The hash function

introduces the loss of uniqueness between the space of application names and the smaller hashed space. This is not a major problem because the interface reference was dropped and the path had to be followed anyway. However, it has failure implications because a client could have been misled if it was looking for another service that generated the same hash result. The distance where the collapse takes place is service dependent and is regulated by the server.

A L-server forwards information packets to its L-server one layer above (if it exists) using an identical simplification process. For each service, the maximum possible hierarchical level required is defined by the requested service range.

Clients control their search range when they lookup a service. If they are near the server's L-server, the information is kept complete, allowing clients to interact directly with the server. Further away it gets simplified and clients have to follow hints, or move in that direction, with a probability of failure.

The information in L-servers at higher hierarchical levels will always be composed of incomplete service hints (identified by hash values) to reduce the update rate to the minimum value (only new or ending service hints are disseminated to this level). They offer a broader but less detailed vision of the services available, which acts as a distributed index service. The location service does not provide a "root" service, which has complete knowledge of the system. In fact, if the client search range and the server offer range does not intersect, clients may fail to run the service. If an absolute reliable resolution is required, a new service (external to the location service) could be provided to search all top-level servers at that moment to look for the service network-wide.

The server offer information dissemination overhead is lowered by creating temporary chained forward service hints after a server migration. The service hint update information packets can then be buffered and multiplexed over a single information packet, reducing the bandwidth usage, but maintaining the coherence of the service hint's path. Priority packets (like service cancellations) override the buffering and force the sending of the updates, to minimise the period of time with incoherent information (paths to a non-existing server).

Figure 2 shows an example of the location service. The lowest plane, the agent systems plane, has sets of agent systems forming meshes. The second and third planes show the first and second hierarchical levels of the L-network (with meshes in each one, and upward connections not designed). In this particular case, a reference to the server 'S' is known completely on all agent systems at the server location's domain (darker grey area). The lighter grey areas represent the scopes where incomplete information is known (reference to the L-server associated with the agent system where the server is running). The resolution path for clients ©$_1$, ©$_2$ and ©$_3$ would be respectively L^1_a; L^1_b - L^1_a; and L^1_d - L^2 - L^1_a.

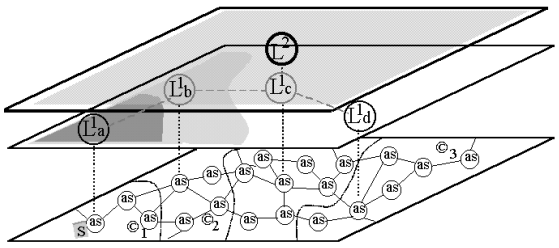

Figure 2: Location Service Network Model

2.3.2. DYNAMIC HIERARCHIES

The hierarchical structure of the L-network is dynamic and depends upon the number of active application servers, the requested service ranges and the number of client requests. It works as a scale mechanism to both balance the L-servers load and reduce the service hint dissemination and lookup overheads.

The overhead of the horizontal dissemination algorithm limits the maximum range supported at each hierarchical level (the maximum number of L-servers involved). The structure of the location service (hierarchical levels and meshes) will vary to gather the necessary number of agent systems the application servers want in their domains. A new hierarchical level is created if the existing meshes do not cover the new requested range.

Other adaptation mechanisms occur when L-servers become overloaded. L-servers measure the number of server hints registered locally and the client load. The overload might result from a restricted set of highly demanded services, or from serving too many descendants agent systems or lower level L-servers (in result of hosting new mobile servers or peaks of requests). When a restricted set of services is involved, the proper setting of the horizontal service information dissemination at levels below the maximum can be used to lower the L-server's client load. This is so, because client load on L-servers at higher hierarchical levels is due to requests not answered at the lower levels. Disseminating server offers horizontally at L-servers in lower hierarchical levels can lower this load, at the cost of higher dissemination overhead. If the overload is indefinite, then L-server replicas are

created to split the load. New intermediate hierarchical levels might also be created, if several L-servers at the same level detect the same problem, resulting on too high horizontal disseminating costs. Temporary "chained service hints" are widely used, to allow low priority service hint updates. The inverse may also occur: L-servers may be destroyed, to reduce the service dissemination costs. The descendants (L-servers or LLPs) are informed of the reconfiguration by the original L-server.

3. SIMULATION RESULTS

A simulator was developed using the "Discrete Event" model on Ptolemy [12] system, which implements the location service protocols. The simulation results presented in this paper compare the efficiency of three static structures for the location service: a pure hierarchical structure with three levels where the topmost one is a single root node; a flat one-level meshed structure; and the multi-level model proposed on this paper (with two levels). The dynamic reconfiguration of the location service (creation and destruction of L-servers) was disabled during the simulations to focus on the information dissemination process - its study was left for a future paper. Client load was constant and symmetrical. When a client is born, it starts searching for an application server, and lives until it can make an interaction. Our main results are the number of service information packets exchanged between L-servers and the number of client lookups per L-server.

All simulations were conducted with a network of 625 agent systems, where each agent system has an average of 3 connections to its neighbours, with a maximum distance between agent systems of 24 (agent system) hops. Two servers were placed on the network (clone creation was disabled - see [4] for this aspect). Servers registered their offers on a range of 12 hops (covering the entire network), and have complete service information on a single L-server. The time to process location service requests and service requests was respectively 1000 and 8000 tics. The transmission time between nodes was set to 1 tic. During a simulation time of 20 million tics 4000 clients were created at all 625 nodes with a uniform distribution of the inter-client creation time on the interval [0, 10000]. The first hierarchical level of L-server network has a total of 125 L-servers, with an average of 8.3 agent systems per L-server. The second hierarchical level (used in the multi-level and hierarchical experiments) has five L-servers. The third hierarchical level (only used in the hierarchical experiment) has a single root L-server.

Figures 3 and 4 show respectively the average number of client lookups per L-server during the experiments (for a total of 4000 clients) and the average number of service information packets exchanged (to the neighbours or upper level) by each L-server per server registered. Multi-level experiment uses a dissemination range at the first hierarchical level of 6 hops.

	Level 1	Level 2	Level 3
Pure Hierarchical	105.0	1279.6	2514.0
Multi-level	123.2	257.6	-
Flat	157.2	-	-

Figure 3: Number of client lookups per L-server for 4000 clients

	Level 1	Level 2
Pure Hierarchical	0.013	0.20
Multi-level	1.72	0.50
Flat	8.80	-

Figure 4: Average number of service information packets exchanged per L-server for each new application server

Figure 3 shows that a pure hierarchy scales poorly. The root L-server receives lookups from more than 50% of the clients and each L-server at the second hierarchical level receives a request from more than 25% of the clients in average. The use of horizontal dissemination in the multi-level and flat experiments allowed a reduction on the number of client lookups per L-server. The disadvantages are the overhead paid in the number of service information packets exchanged (shown in figure 4), and a longer L-server resolution path that will increase for bigger networks (the resolution path in pure hierarchical structures is independent from the network size).

The peak value for L-server lookups happens at L-servers where application servers registered their offers (with about 2000 client lookups). This value can be lowered if the number of L-servers with complete service hints is increased.

The multi-level structure offers an intermediate performance knob, which can be tuned by setting the "spread" of service hints at the first hierarchical level. By changing the value from 0 to 12 LLPs covered, the system behaviour changes from a hierarchical structure (with two levels and a meshed root) to a pure flat system. Figure 5 shows the percentage of clients requests answered by the first level L-server that needed one lookup to the second level.

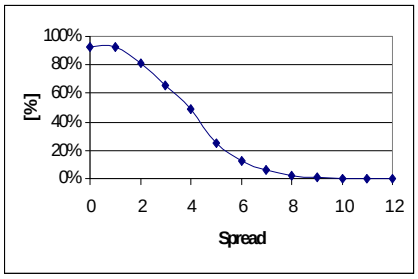

Figure 5: Client requests directed to L-servers of second hierarchical level.

4. CONCLUSIONS AND FURTHER WORK

This paper presents a scalable location service architecture suitable to very large systems with mobility and the possibility of handling correlated peaks of traffic. We argue that a static approach is inappropriate and the inner structure of the service must be dynamic to adapt to the conditions of the moment. The trade-off is the need to exchange information inside the service (a feature not seen on classical name servers). However, this is not so critical because application servers tend to adapt to client load and most of the relevant lookups tend to be local. The simulation results presented prove the importance of the location network structure to the L-servers load, and to the overall performance of the location service.

On-going work includes (a) a thorough study about the changes on the hierarchy structure, (b) the trade-off between application server clone creation and the cost of maintaining consistency of application data ("low-cost" inter-replica state synchronisation techniques), and (c) more complex interactions between clients and servers, such as connection oriented ones in the presence of mobility, to include multimedia, for instance.

ACKNOWLEDGEMENTS

This research has been partially supported by the PRAXIS XXI program, under contract 2/2.1/TIT/1633/95.

REFERENCES

[1] I. Akyildiz and J. Ho, "On Location Management for Personal Communications Networks", *IEEE Comunications*, pp. 138-145, September 1996.

[2] P. Albitz and C. Liu, "DNS & BIND", O'Reilly &Associates Inc., 1992.

[3] 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.

[4] 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, September 1998.

[5] A. Gulbrandsen and P. Vixie, "DNS RR for specifying the location of services (DNS SRV)", *IETF RFC 2052*, October 1996.

[6] ISO/IEC, "Information technology - Open Distributed Processing - The Directory - Overview of concepts, models, and services", ISO/IEC DIS 9594-1, ITU-T Rec. X.500, November 1993.

[7] ISO/IEC, "Information technology - Open Distributed Processing - Trading function: Specification", ISO/IEC DIS 13235-1, ITU-T Rec. X.950, Editors Draft COM 7-51, March 1997.

[8] J. Jing, A. Elmagarmid, A. Helal and R. Alonso, "Bit-Sequences: An adaptive cache invalidation method in mobile client/server environments", *Mobile Networks and Applications* 2, pp. 115-127, 1997.

[9] C. Lynch, "Network Information Resource Discovery: An Overview of Current Issues", *IEEE Journal on Selected Areas in Communications*, Vol. 13 N. 8, pp. 1505-1522, October 1995.

[10] N. Meulemans, "A Yellow Pages service based on X.500", *Computer Networks and ISDN Systems* 28 , pp. 1939-1946, November 1996.

[11] J. Moy, "OSPF version 2" *IETF RFC 2178*, July 1997.

[12] Ptolemy project home page. http://ptolemy.eecs.berkeley.edu/

[13] M. Shapiro, P. Dickman and D. Plainfossé, "SSP Chains: Robust, Distributed References Supporting Acyclic Garbage Collection", Tech. Rep. 1799, INRIA, Rocquencourt, Nov. 1992.

[14] M. van Steen, F. Hauck, A. Tannenbaum, "A Model for Worldwide Tracking of Distributed Objects", Proceedings TINA '96 Conference, Heidelberg, Germany, pp. 203-212, September 1996.

[15] C. Weider, J. Fulton and S. Spero, "Architecture of the Whois++ Index Service", *IETF RFC 1913*, February 1996.