Transparent and Scalable Client-side Server Selection using Netlets

Kalaiarul Dharmalingam and Martin Collier
Research Institute in Networks and Communications Engineering
School of Electronic Engineering
Dublin City University, Republic of Ireland
E-mail: {arul, collierm}@eeng.dcu.ie

Abstract— **Replication of web content in the Internet has been found to improve service response time, performance and reliability offered by web services. When working with such distributed server systems, the location of servers with respect to client nodes is found to affect service response time perceived by clients in addition to server load conditions. This is due to the characteristics of the network path segments through which client requests get routed. Hence, a number of researchers have advocated making server selection decisions at the client-side of the network.**

In this paper, we present a transparent approach for client-side server selection in the Internet using Netlet services. Netlets are autonomous, nomadic mobile software components which persist and roam in the network independently, providing predefined network services. In this application, Netlet based services embedded with intelligence to support server selection are deployed by servers close to potential client communities to setup dynamic service decision points within the network. An anycast address is used to identify available distributed decision points in the network. Each service decision point transparently directs client requests to the best performing server based on its in-built intelligence supported by real-time measurements from probes sent by the Netlet to each server. It is shown that the resulting system provides a client-side server selection solution which is server-customisable, scalable and fault transparent.

I. Introduction

With the user population of the Internet continuously on the rise, the demand for web based services is also witnessing a corresponding exponential rise in demand. Content replication at multiple locations in the network has been identified as a scalable means to provide clients with improved service response time, reliability and performance levels. When replicated servers are available at multiple locations in the network, clients are presented with the problem of dynamically choosing the best or the optimum performing server for service provisioning. Most server selection methods [1–6] proposed to date work to distribute load across servers.

Techniques that perform load distribution across servers were traditionally designed for load balancing across server clusters. In such approaches, the selection decision is purely based on the server load. However, when working to assign client requests to distributed server systems, the location of servers with respect to client nodes has been found to affect the service response time perceived by clients [7, 8]. This is due to the characteristics of the network path segments through which requests get routed. Hence, making server selection decisions based on the client's view of the network and server conditions is appropriate.

To facilitate this, web servers hosting replicas at multiple locations in the Internet provide users with the address list of servers available for service provisioning. Current approaches followed by clients for selecting servers from a replicated set include: (a) random selection; (b) directing requests always to a fixed server; or (c) to choose the closest server according to geographical proximity. However, the above mentioned approaches have proved to offer poor server selection solutions [7–9]. In some cases, clients also try parallel downloads of the same the document from multiple servers. In this approach, once a server accomplishes the requested task, the other requests are terminated. This approach generates redundant network traffic thus consuming excess bandwidth.

Research efforts have been made to identify client-side server selection metrics that support efficient server selection in the Internet [7–9]. It is proposed that the clients themselves would perform the requisite measurements and make the selection decisions. There are two major shortcoming of such techniques: (i) such solutions are not scalable because every client on network will use measurement probes that will consume network resources; and (ii) the servers are unable to influence selection decisions, so that it is not possible to support request distribution across the available servers.

Our solution is intended to meet the following goals:

- **Load Distribution:** it provides a mechanism to distribute client requests for content among multiple, possibly geographically dispersed, servers;
- **Client-side based Service Decision:** assigning requests to a specific server occurs close to a client, to maximise service responsiveness;
- **Server Customised Selection Techniques:** the selection of a server is based on metrics supplied by the servers, allowing eg., load balancing or link bandwidth probing to be performed;
- **Scalability:** avoiding the use of measurement probes generated by individual clients, and employing aggregated set of measurements that can be used for client communities;
- **Demand-based Service Support:** providing selection services at those locations where potential client communities for the service exist;
- **Service Location Transparency:** clients request content from a single address, so that the operation of scheme is completely transparent to the client; and
- **Fault Transparency:** the solution is robust, with no single point of failure.

The requirement to allow server access based on a single address may be met in one of two ways: (i) redirection from a primary server; or (ii) use of anycast addresses. Neither solution can meet all of the above requirements. Our solution is to deploy a virtual primary server across multiple nodes of the network, which is identified using an anycast address. This virtual primary server is implemented using Netlets, a mobile agent based network service component [10, 11].

The rest of the paper is organised as follows. In section.II we describe the related work. Section.III presents the solution overview. We describe the mechanism of transparent server selection using Netlets in sections IV and V. Supporting architectural design features for the Netlets based approach is presented in.VI. Section VIII presents experimental results and section IX concludes the article.

II. Related Work

Approaches proposed to solve the task of selecting the best performing server from a set of geographically dispersed replicated servers can be categorised based on the location at which the selection decision is performed. Popular techniques available for server selection in the Internet are: (a) server-side [2]; (b) network supportive [1,3–6]; and (c) client-side [12].

The commonly used server-side technique in the Internet is the HTTP Redirect [2] scheme. In this, a busy server redirects the client to resubmit the request to another server. This approach generates additional network traffic and causes increased latency.

In the network supportive schemes, the network nodes (for example DNS servers, router, active nodes) act as the service distribution points for client requests. The DNS aliasing scheme [1] stores multiple IP addresses for a single site in its local DNS table and directs requests in a round-robin fashion. The major problem with this scheme is that it is not capable of determining the availability of a given server. This is because intermediate name servers cache the resolved name-to-IP-address mapping thus allowing changes in DNS information to propagate slowly through the network. Consequently, failed servers continue to receive requests, resulting in e.g., HTTP failures to end-users. Furthermore, if in case of a DNS failure the replicas become inaccessible.

In [13], Moore et al. described the SONAR approach for server selection. In this, clients when presented with a list of server replicas consults the nearby SONAR server for knowing the closest replica. The closest replica is selected based on the proximity of the replicas from the SONAR server. The disadvantages with this approach are: (i) client nodes must be aware of the SONAR protocol; and (ii) increased server selection latency due to additional interaction between the client and the SONAR server. This approach is similar to the DNS based approach, hence the cost of setting up SONAR servers purely for proximity measurement in the Internet is not justified. Furthermore, the HOP counts based approaches for server selection has proved to provide poor selection performance [7, 9].

Anycast based schemes were proposed to provide network supportive server selection techniques. Anycasting is defined in [3] as: "a service which provides a stateless best effort delivery of an anycast datagram to at least one host, and preferably only one host, which serves the anycast address". IP anycast [3] is a network service that allows a client to connect to the nearest of the receivers that share the same anycast address. "Nearest" is defined in terms of network distance metrics.

The plain anycast protocol was used by Engel et al. [4] to demonstrate server selection based on network distance metrics, typically HOP counts. However, it has been proved that metrics based on network distances has less correlation with server response time and network delay [7, 9].

A variant of the plain anycast approach, referred to as the "active anycast" [5], employed active nodes as routers to perform load distribution across replicated servers. A limitation of this approach is that every active node must collect metrics from each replicated server for selection decisions. Furthermore, in the case where there are no active nodes along the path of a flow, requests are automatically routed to the closest server of the group as in the plain anycast scheme.

A modified DNS based approach was described in [6], where anycast resolvers present at DNS nodes are used to perform load distribution across replicated servers that share an anycast address. The selection decision was based on locally maintained performance data about individual replicas. This approach will not scale to a large set of anycast services, because DNS servers must collect performance data about each potential server groups in the Internet.

All the above mentioned approaches serve to balance the load across servers and do not take into account the location of clients and network conditions. However, when working with distributed server systems, the location of servers with respect to client nodes have been found to affect the service response times perceived by clients [7, 8, 12]. In [12], a modified web browser referred to as the smart client, was used to perform server selection. This client software downloads an applet supplied by the service provider to realise service-specific routing. This approach creates increased network traffic due to applet downloads and the corresponding communications which ensue between the applet and the servers.

III. Solution Overview

A solution based on Netlets which meets the goals listed in section I will now be presented. We use the Netlet services [10, 11] for providing the service decision points, referred to collectively as the Virtual Primary Server, within the network. The location of decision points is shown in Fig. 1 for our approach and the approaches of [1–6].

Netlets are autonomous, nomadic mobile software components which persist and roam in the network independently, providing predefined network services. The Netlets network uses the mobile agent paradigm to realise an Active Network architecture. The Netlet Node offers support for composition and execution of Netlet based network services. A more detailed discussion of the Netlets approach and the architecture of the Netlet node can be found in [10, 11].

The reference architecture that we employ to demonstrate our solution is shown in Fig. 2. We assume a het-

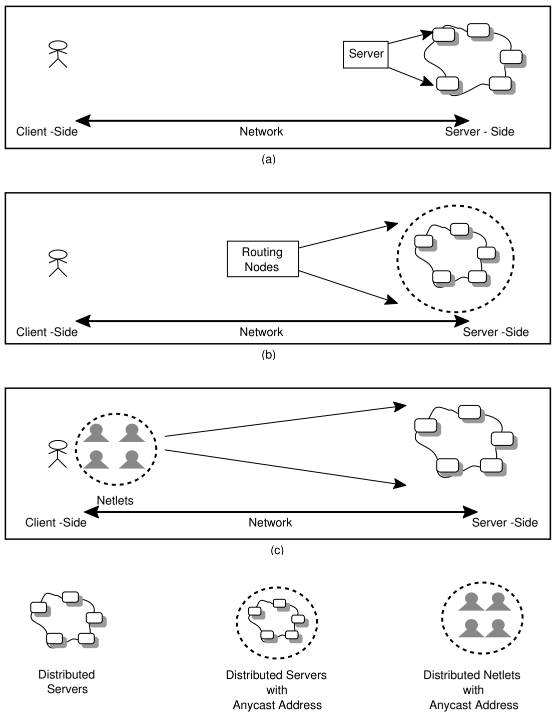


Fig. 1: Server Selection Approaches

erogeneous network environment in which both active and legacy routing nodes exist. Netlet based services embedded with intelligence to support server selection are deployed by servers close to potential client communities to setup dynamic service decision points within the network. We refer to those network services that support server selection as the *director services*. Each service decision point transparently directs client requests to the best performing server based on its in-built intelligence supported by real-time measurements that are performed between itself and server replicas. We propose to deploy director services based on user demand. The exact location and the number of director services present in a network is dictated by the location of the relevant communities of interest in the network.

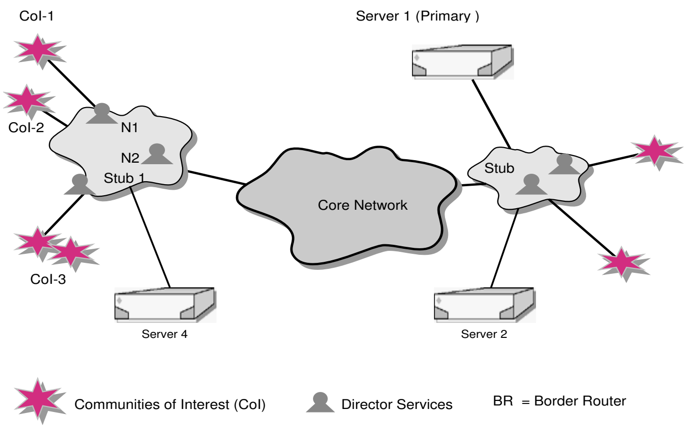


Fig. 2: Replicated Servers and Communities of Interest

Note that the Netlets architecture has a wider set of applications than the problem of transparent server selection [14–17]. Hence, the justification for installing Netlet support at network nodes is not just to provide network services for server selection, but to host other mobile agent or active network based solutions.

IV. Dynamic Setup Of Virtual Primary Server

A. *Anycast based Director Services*

Anycast addresses are used to represent distributed server groups in the Internet. Web sites hosting replicated servers advertise an anycast address instead of the individual IP addresses that correspond to server replicas. This approach has been adapted by anycast based server selection schemes [3–5].

In the Netlets based approach to server selection, we propose to share an anycast address among the Netlet based director services (i.e., inherently the Netlet node at which the service operates) that act as service decision points and with the primary content server. Clients are only aware of the director service location rather than the individual server replicas. Consequently, client requests that correspond to anycast addresses are automatically routed to the closest service decision point rather than directly to a server.

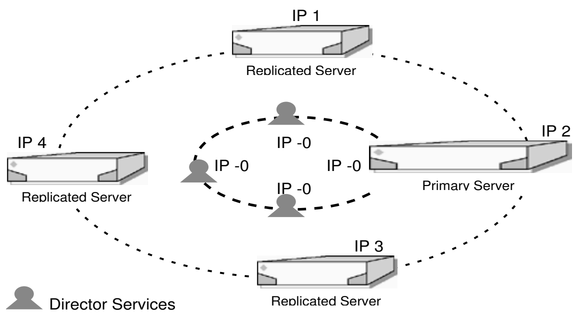


Fig. 3: Address Sharing in the Netlet Scheme

The representation of address sharing in the Netlet based scheme is shown in Fig. 3. The director services share an anycast address, while the replicated servers have distinct IP addresses. The primary server shares the anycast address with director services and also has a distinct IP address. This feature of binding two addresses to the primary server, allows a client request to get automatically routed to a server if no service decision points exist close to its location.

B. *Service Deployment*

The mobile and autonomous property of service code in the Netlets architecture avoids manual intervention for service deployment. To introduce service selection support at multiple points in the network, the director service is informed with the address list of nodes requiring service. This Netlet then autonomously migrates to each node and installs service thus avoiding centralised deployment schemes and generating less network traffic. Methods to find exact locations to providing director service support and the scheme to discover active nodes at those locations are presented in section. VI.

C. *Registration of Director Services at Netlet Nodes*

When a director service is deployed at a Netlet node, this service requests the local node: (i) to register for receiving client requests that correspond to the anycast address for which the Netlet holds the permission; and (ii) to advertise routes for the anycast address.

The concept of virtual host and interfaces used by IP aliasing can be used to register director services at a Netlet node. IP Aliasing is simply a mechanism that enables a single physical or virtual network port to assume responsibility for more than one IP address. For example, in a linux based router, a simple command such as, *ifconfig eth0:<virtual interface number > <anycast ip> <netmask>* can be used for this purpose. By using this feature a Netlet node will be able to support multiple director services simultaneously. New routes to anycast addresses can be advertised as part of normal routing table updates.

V. Server Selection using Director Services

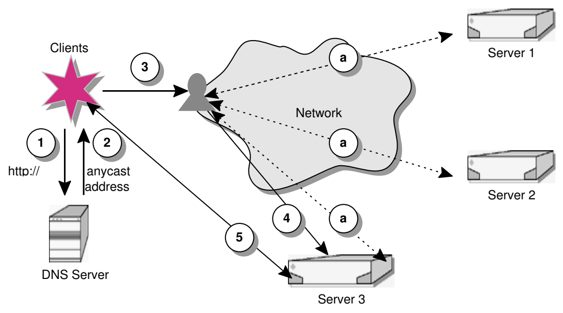


Fig. 4: Transparent Server Selection using Director Services

Below we describe the mechanism used to transparently direct client requests to the optimal server. For the example below, we assume that TCP is used as the transport protocol.

When a client wants to connect to a server, the client performs a name resolution query to the DNS server (step 1). The reply from the DNS node consists of an anycast address (step 2) which refers to the distributed server group.

The client sends a TCP SYN packet to this address to initiate a connection. This packet is automatically routed to the closest service decision point (i.e., director service) that corresponds to the server group (step3 in Fig. 4). On receiving the request, the director service selects the optimum server (step 4) based on selection metrics (step a). The selection metrics are described in section.VIII. Hence, the request is directed to the chosen server (for example server 3 as in Fig. 4).

Note that the SYN packet from the client has the anycast address as its destination address. Hence, a mechanism is required to direct the SYN packet transparently to the chosen server. One solution to this problem is to encapsulate the SYN packet within a unicast packet destined to the unicast address of the selected server. Unique protocol identifiers can be used to identify such encapsulated packets at the server end. The server on receiving the SYN packet replies with the SYN-ACK packet directly to the client based on the available destination address.

Stateful connections (step 5) can then be maintained with the selected server using the approach described in [4]. In this approach, the server receiving the anycast packet pins the route for future packets originating from the client during that session to pass through the unicast address of the selected server. With modifications performed at the TCP/IP control blocks at the server side, when such packets are received, the IP block passes it to the request processing daemon. Stateful connections may be alternatively maintained over UDP [4].

VI. Architectural Design Features

In this section we discuss the architectural features required to support the Netlets based approach to server selection. This feature set includes: (i) a method to support discovery of locations requiring director service support; (ii) discovery of active nodes at those locations; and (iii) scalable routing for anycast addresses using unicast routing protocols.

A. Communities of Interest

A deployment scheme is required for distributing director services within the network. Analysis of access logs of various web servers have shown the existence of *communities of interest* in the Internet [18–20]. These are groups of clients which are responsible for generating a high proportion of the workload on servers and which are geographically close or under common administrative control. Servers should deploy director services close to such communities.

In [18], a network-aware method based on prefixes and netmask information gathered from Border Gateway Protocol (BGP) routing table snapshots was used to identify client clusters (referred as communities of interest in this paper) in the Internet. The authors validated the BGP based technique to locate communities of interest by employing two approaches based on "domain name" and "traceroute". This technique gave good performance even when used with historical snapshot data.

The results from [18] based on globally collected web server logs show that 90% of communities have 100% of their clients topologically close to each other. It was also reported that around 5% of communities accounted for the majority of the clients and for generating a high percentage of the workload on the web server (see Fig. 5). This confirms earlier studies [19] that claim the existence of Zipf-like distributions in a variety of web measurements.

By being able to locate communities of interest, servers will be able to provide transparent selection support to the majority of the client population that use the services. Remaining clients are served directly by the primary server. Since there are relatively few clients outside the communities of interest, this does not represent a major burden on the primary server. We adapt the above described BGP based technique to locate communities of interest present in the network to support director service deployment.

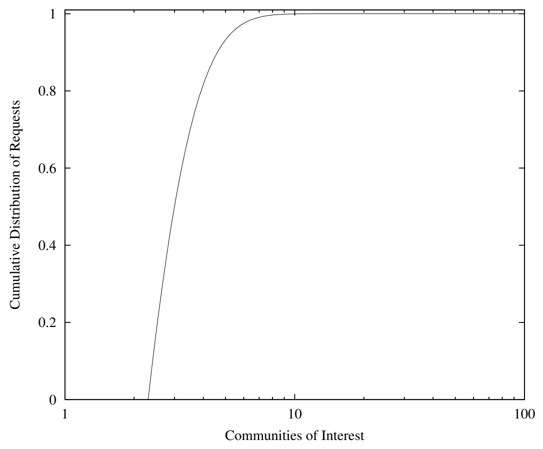


Fig. 5: Cumulative Request Distribution Across Communities of Interest

B. Locating Hot Spot Nodes

On locating communities of interest in the network, a discovery service is required to identify appropriate operation locations for the director services which utilises network resources efficiently. Due to the heterogeneous nature of the Internet, not all nodes in the Internet will be active. Thus both active and non-active nodes are expected to coexist in the future. Hence, a discovery scheme to locate active nodes in the Internet is mandatory.

Currently available service discovery protocols such as Jini [21], SLP [22] have been developed for working with small network topologies, such as LAN networks. The DNS based scheme proposed below can be used to locate active network nodes present in the Internet.

DNS-based Discovery Scheme

Domain Name System (DNS) servers support features to list existing hosts located within a domain. This feature can be exploited to discover active nodes present in a network domain. The set of active network nodes that fall under a common administrative control can be listed in the node reachability information list of their corresponding domain servers. By using existing DNS query tools such as nslookup, dig, the host list of a domain can be retrieved. The information retrieved includes both host names and corresponding addresses of the nodes. By making the host names self-descriptive with standard prefix formats, the list of active nodes present in a network domain can be extracted.

For example, a name such as *active.netlet-node-32.dcu.ie* can be used to represent an active node of type Netlet present in the dcu.ie domain. If, by co-incidence, a passive node matches the prefix format, this scheme will include it in the list of active nodes. Valid active nodes can be identified by exchanging hello messages between the server and the nodes on the list. Further work is necessary to study scalability issues when using such confirmation messages.

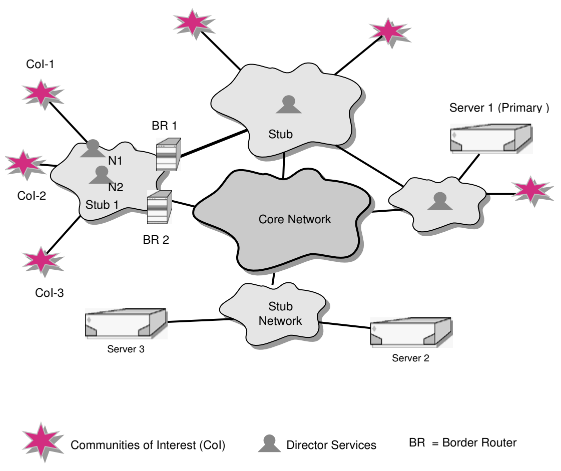
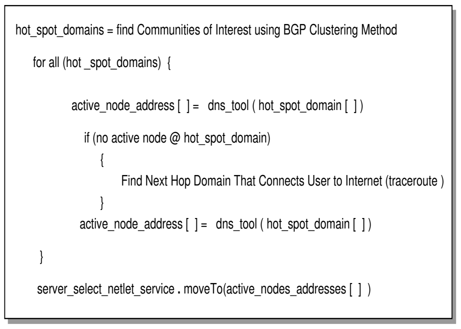


Fig. 6: Communities of Interest and Director Services

Hot Spot Nodes

A suitable location for director service operation is at the ingress/internal routing nodes of the stub network (such as N1 and N2 in Fig. 6) through which users connect to the Internet. This is a consequence of the feature of route aggregation present in the Internet [23].

For example in Fig. 6, let the director service, N2 direct requests to server S1 (IP addresses 192.15.36.12) and S3 (136.10.1.2) based on some predefined selection metric. Suppose the border router BR1 aggregate routing entries for destination IP addresses starting with 192 while BR2 serves for IP addresses with 136 as the start. In this example, the route over which N2 communicates with the servers will share many links with the corresponding routes for clients in the community CoI_1 which accesses the Internet via *stubnetwork*1. We refer to those Netlet nodes that act as judicious points for deployment of director services as "hot spot nodes" and their addresses as "hot spot addresses"(e.g., N1 is the hot spot location for users from CoI_1 while N2 is for clients present in communities CoI_2 and CoI_3 in Fig. 6).



```
hot_spot_domains = find Communities of Interest using BGP Clustering Method
    for all (hot _spot_domains) {
            active_node_address [ ] =  dns_tool ( hot_spot_domain [ ] )
              if (no active node @ hot_spot_domain)
                 {
                      Find Next Hop Domain That Connects User to Internet (traceroute )
                 }
             active_node_address [ ] =  dns_tool ( hot_spot_domain [ ] )
     }
     server_select_netlet_service . moveTo(active_nodes_addresses [ ] )
```

Fig. 7: Algorithm for Active Node Discovery and Service Deployment

The algorithm for locating and deploying director services in the network is presented in Fig. 7. The BGP based scheme [18] (section.VI-A) automatically generates the list of hot spot addresses in the Internet. Using the above described DNS based approach, we will be able to discover corresponding hot spot nodes and their addresses.

If the domain that holds the client group fails to contain active nodes, the next hop domain within the stub network connecting the clients to the Internet is queried. Locating the name of the second domain can be performed using traditional network tools such as traceroute.

C. *Scalability of Unicast Routing Protocol for Anycast Addresses*

Netlet based director services employ global anycast addresses to seamlessly integrate the dynamically constructed service decision points with the client-server based web communication model. When director services are deployed within stub networks they behave as local anycast groups to the corresponding stub domain. Due to this specific nature of the Netlets approach, conventional intra-domain routing protocols will be sufficient to route packets destined to anycast receivers local to the domain. For example, distance-vector algorithms, such as RIP inherently provide support for anycast service [4].

Employing unicast protocols for anycast services causes each service decision point present within a stub network to take up an entry in the internal routing table. However, this approach is scalable because: (i) the number of service decision points within a network is driven by user demand local to that domain; and (ii) routing nodes present in stub networks has more free memory resources and CPU cycles when compared to routing nodes present in core networks.

Recall that when no service decision points exist within a domain, the anycast packets are routed to the primary content server which shares the same anycast address with director services (Fig. 3). The inter-domain routing can be implemented in a scalable manner using the method of Global IP Anycast (GIA) [24]. GIA uses the notion of popular and unpopular anycast groups in the Internet. The popular groups refers to those sets of anycast addresses that are often accessed by users from a particular domain. However, for unpopular groups (here, those groups which are routed to the primary server), packets are routed to a default unicast address that is encoded within the 32-bit anycast address.

VII. Benefits of Employing Director Service Netlets

Temporal Shifts in User Demand Across Communities of Interest: Analysis of commercial web server logs [20] have proved the existence of demand shifting across communities of interest in the Internet. The authors of this paper propose to allocate distributed resources on demand near client locations to support such variations. Complementarily, using the Netlets approach, director services will be able move in accordance with demand to support server selection. The intelligence to support such feature can be embedded in the director services themselves.

Scalability and Knowledge Sharing: Selection techniques based on using measurement probes by each client for server selection will not scale for large networks such as the Internet. The Netlet scheme offers to implement scalable server-customised probing techniques. For example, director services that belong to the same server group and operating in close vicinity (eg., the same stub domain) will be able to share measurement probes. Furthermore, director services can be scoped to probe only a reduced set of servers when the replica set comprises a large server group.

Support for Wireless Network nodes: Wireless network nodes have constraints on the availability of local resources and power. Hence, supporting server selection software at such nodes will be inefficient. Furthermore, wireless nodes will be unable to participate in continuous communication with server groups to perform selection decisions. The Netlets scheme readily offers support for wireless nodes by implementing the decision procedure in the network rather than on the client nodes.

VIII. Experiments

Server Selection Metrics

The service response time perceived by a client can be formulated as:

$$ServiceTime = T_{Locate} + T_{Connect} + T_{Serve}$$

where, T_{Locate} refers to the time taken to locate a server; $T_{Connect}$ is the time required to establish the connection; and T_{Serve} is the remaining time taken to serve the request.

The T_{Locate} and $T_{Connect}$ components are dependent on the prevailing network and server conditions. The T_{Serve} component is largely dependent on the request type but also depends on the server load. Hence, selection metrics that use server load and network parameters to make server selection decisions will be able to control the effect of these components on the total service response time perceived by the client. In this paper, we compare the performance of the following server selection metrics when employed by the director services: (i) server load; (ii) network latency; (iii) random selection and (iii) end-to-end processing delay.

A. Server Load Distribution

The first set of experiments includes: (i) the implementation of load distribution across servers using director services; and (ii) a study of the impact of server load on client perceived service response time. This set of experiments were conducted in a LAN environment with a pair of Apache servers [25] (S_1, S_2) running on Linux machines. Server S_1 was configured the closest to the client node (a single HOP away) while server S_2 was configured with 2 HOPS as the distance metric.

These servers were configured to accept a maximum of 150 connections simultaneously. The mod-status module present in the Apache server was configured to monitor the load condition on the servers. Httperf [26], a HTTP traffic generation tool was used to generate background traffic on servers. Client requests to servers were modelled with exponential inter-arrival times. The goal the director service had to accomplish was: *to route client requests to the best performing server based on load conditions (obtained using the mod-status module of apache), thus achieving load distribution across the servers*. The maximum load threshold at the servers was defined as 80%. The director services worked with this value for server selection.

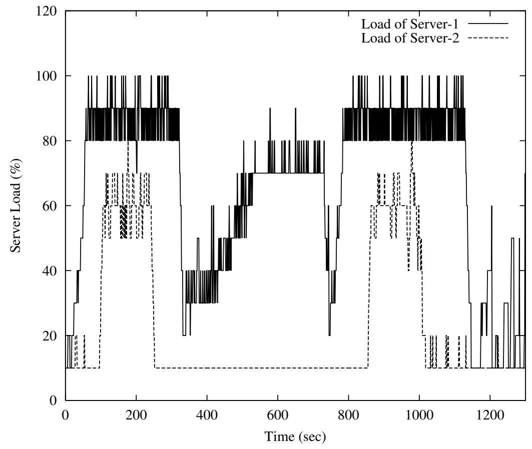


Fig. 8: Load Based Server Selection

The httperf tool generated background traffic to S_1, constituting around 90% load on the server for the first 350 seconds (see Fig. 8). During this period of time, the director service routed requests to server S_2. When the background traffic was removed from server S_1, the Netlet service directed requests to the closest best performing server, S_1. This corresponds to the period from 350 to 750 seconds in Fig. 8. When the background traffic was introduced back on server S_1, the director service routed requests to S_2, thus accomplishing load distribution.

To study the impact of server load on service response time, average download latency for files from the two servers were analysed. File sizes used in our tests varied from 500K to 5000K. Files were downloaded when server S_1 was operating at 80% load and when server S_2 was having 40% load imposed. This corresponds to load conditions at different instants of time in Fig. 8. The average download latency experienced for each file at both servers are shown in Fig. 9.

The average download latency offered by server S_2 is 2 to 3 times less than that of the closest server S_1. Thus, we can conclude that the server load affects the response time perceived by clients. Furthermore, approaches (e.g., [4]) based on locating closest server replicas nodes for serving

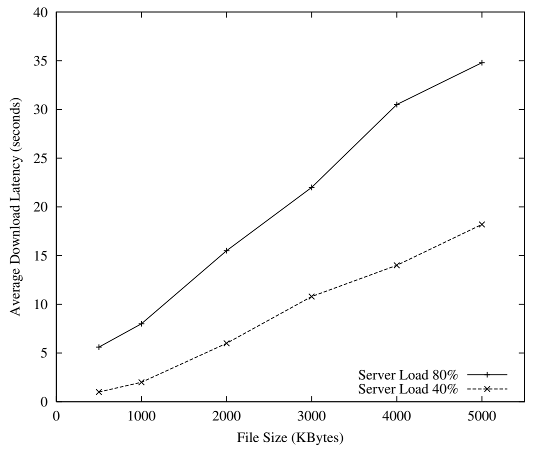


Fig. 9: Impact of Server Load on Service Response

client requests thus does not always provide an accurate server selection technique.

B. Server Selection in the Internet

The next set of experiments evaluated and compared three different metrics for server selection in the Internet. For this purpose, we used a set of 10 mirror servers (www.*.kernel.org) [27] present at different geographical locations in the Internet. File sizes used for testing the average download latencies experienced by clients varied from 500K to 15000K. The total set of measurements spanned across a 10 day period at different times of the day so as to minimise effects of caching and time-of-day effects.

1) Random Selection: For this metric, the director services implemented the random server selection strategy popularly followed in the Internet. Random number generators were used to decide the server to be selected from the replica set. Average download latency for each file (500K to 15000K) was recorded (see Fig. 10).

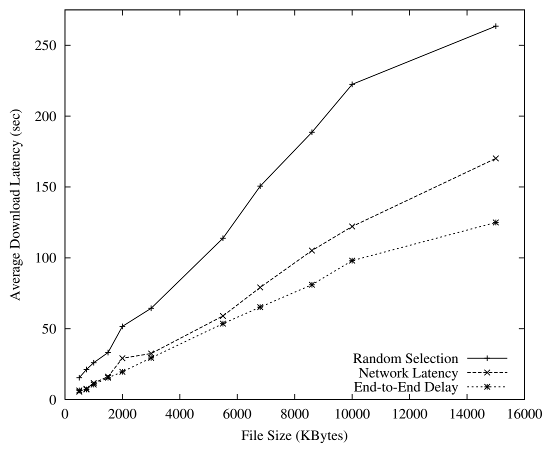


Fig. 10: Comparison of Server Selection Metrics

2) Network Latency: In the second server selection technique, the Netlet services used network latency as the parameter for deciding the optimum performing server. The average round-trip time (RTT) was measured for each server by the Netlet service. This measurement was carried out using ping probes to each server.

The selection decision at the Netlet service was either made on past probe measurement results or on new measurements that were made prior to assigning the requests to a server. We selected the timeout period for past measurement results as 90 seconds. The timeout value was arbitrarily chosen to reduce frequent probing. The probability of selecting a server, S_j, from a replicated set of N server replicas was calculated using the following equation:

$$Prob\ (s_j) = \frac{\sum_{j=0}^{N} 1/RTT_{S_j}}{\sum \sum_{j=0}^{N} 1/RTT_{S_j}}$$

where RTT_{s_j} is the average round-trip time that corresponds to server S_j from the director service.

Based on the measurements, the server with the highest probability was selected. The average download latency for each file was recorded (see Fig. 10).

3) End-to-End Latency: The selection metric based on network latency does not account for the load condition at server nodes. This is because the ping responses from servers are handled by server daemons other than the web server daemons. Applications that are both CPU intensive and delay sensitive will require both server and network load parameters to be involved in deciding the best performing server. A solution to support such decisions will be to check the end-to-end latency perceived by the client. The end-to-end latency, L_{S_j}, can be formulated as

$$L_{S_j} = RTT_{s_j} + Pdelay_{S_j}$$

where, RTT_{s_j} is the average round-trip time to server S_j from the Netlet service and $Pdelay_{S_j}$ is the request processing delay at the server. The end-to-end latency can be measured by downloading a small test file from all server replicas.

A 100K file was used to measure the end-to-end latency in our experiments. We selected the timeout period for past measurement results as 90 seconds. The probability of selecting a server, S_j, from a replicated set of N server replicas was calculated using the following equation:

$$Prob\ (s_j) = \frac{\sum_{j=0}^{N} 1/L_{S_j}}{\sum \sum_{j=0}^{N} 1/L_{S_j}}$$

The average download latency for each file from the group was recorded (see Fig. 10).

Metric Comparison

The end-to-end latency technique performs the best among all the three schemes discussed (see Fig.10). This finding is consistent with results presented in [8]. The random selection technique popularly followed in the Internet offered the worst performance. The average download latency offered by this technique was 3 times more than the end-to-end latency and almost twice that of the network latency based approach. The technique of downloading small test files to measure end-to-end latency will not scale for servers containing large set of replicas in the Internet. A scalable approach as described in section VII can be adapted.

IX. Conclusions and Future Work

In this paper, we presented a novel technique to support transparent and flexible server selection in the Internet. The Netlets based approach provides a client-side server selection solution which is server-customisable, scalable and fault transparent. Our approach combines the benefits of anycast addressing with a mechanism allowing the adoption of any server selection algorithm. By using mobile agent technology in the form of Netlets, service decision points can be deployed dynamically to the locations in the network where they can most efficiently serve a large number of clients. This approach makes our solution inherently scalable, since it minimises the amount of overhead generated by measurement probes.

Future work will extend the server selection mechanism to be content-aware. This will allow the scheme to automatically select routes to servers which provide a level of QoS support relevant to the type of content.

References

[1] Thomas P. Brisco. DNS support for load balancing. *RFC 1794:*, http://info.internet.isi.edu/innotes /rfc/files/rfc1794.txt April 1995.

[2] D.Andresen et al. SWEB: Towards a Scalable World Wide Web Server on Multicomputers. *Proc. of 10th IEEE International Syrup. on Parallel Processing*, IPPS April 1996.

[3] T. Mendez C. Partrige and M. Miliken. Host anycasting service. *RFC 1546*, Nov. 1993.

[4] R. Engel et al. Using ip anycast for load distribution and server location. *In Proc. Third Global Internet Conference*, 1998.

[5] M. Yamamoto et.al H. Miura. Server load balancing with network support: Active anycast. *The Second International Working Conference on Active Networks(IWAN)*, pages 371–384, 2000.

[6] Ellen W. Zegura, Mostafa H. Ammar, Zongming Fei, and Samrat Bhattacharjee. Application-layer anycasting: a server selection architecture and use in a replicated Web service. *IEEE/ACM Transactions on Networking*, 8(4):455–466, 2000.

[7] M. Sayal and Y. Breitbart. Selection algorithms for replicated web servers. *Proceedings of the Workshop on Internet Server Performance*, June 98.

[8] Sandra G. Dykes, Kay A. Robbins, and Clinton L. Jeffery. An empirical evaluation of client-side server selection algorithms. In *INFOCOM (3)*, pages 1361–1370, 2000.

[9] R.L. Carter and M.E. Crovella. Server selection using dynamic path characaterization in wide-area networks. *Proceedings of the IEEE INFOCOM*, 97.

[10] Martin Collier. Netlets: The Future of Networking? *IEEE OpenArch*, April 1998.

[11] Kalaiarul Dharmalingam and Martin Collier. Netlets: A New Active Network Architecture. *IEE/IEI Symposium on Telecommunications Systems Research, Dublin, Ireland*, Nov 2001.

[12] Chad Yoshikawa and Brent Chun. Using smart clients to build scalable services. In *Proceedings of the USENIX 1997 Annual Technical Conference*. USENIX Association, 1997.

[13] Keith Moore, Jason, Cox and Stan Green. SONAR: A Network Proximity Service. *http://www.globecom.net/ietf/draft/draft-moore-sonar-03.html*, 1998.

[14] Kalaiarul Dharmalingam and Martin Collier. Transparent QoS Support For Network Applications using Netlets. *IEEE MATA 2002, Lecture Notes in Computer Science ISBN 3-540-00021-6*, Oct 2002.

[15] Kalaiarul Dharmalingam and Martin Collier. RSVP Reservation Gaps: Problems and Solutions. *IEEE International Conference on Communications, ICC*, May 2003.

[16] Kalaiarul Dharmalingam and Martin Collier. An Active Network Solution to the Problem of RSVP Reservation Gaps. *IEE Proceedings of London Communication Sympoisum, ISBN 0-9538863-2-8*, Sept. 2002.

[17] K. Psounis. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, First Quarter 1999.

[18] B. Krishnamurthy and J. Wang. On Network-aware Clustering of Web Clients. *In Proceedings of ACM SIGCOMM*, 2000.

[19] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM (1)*, pages 126–134, 1999.

[20] Jose Santos Renato et al. Understanding Service Demand for Adaptive Allocation of Distributed. *IEEE Globecom*, 2002.

[21] Jim Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, 1999.

[22] Erik Guttman. Service location protocol: Automatic discovery of IP network services. *IEEE Internet Computing*, 3(4):71–80, 1999.

[23] S.V.Fuller, T.Li et al. Classless Inter-Domain Routng (CIDR): An Address Assignment and Aggregation. *RFC 1519*, 1993.

[24] Dina Katabi and John Wroclawski. A framework for scalable global IP-anycast (GIA). In *SIGCOMM*, pages 3–15, 2000.

[25] Apache Group. http://www.apache.org.

[26] David Mosberger and Tai Jin. httperf: A tool for measuring web server performance. In *First Workshop on Internet Server Performance*, pages 59—67. ACM, June 1998.

[27] Linux Kernel. http://www.kernel.org.