PROXY SELECTION AND PERFORMANCE ANALYSIS OF A DYNAMIC PROXY FRAMEWORK

Choong Khong Neng[1*], Borhanuddin Mohd Ali[1], Veeraraghavan Prakash[1], Elok Robert Tee[2], and Yee Yoke Chek[3]

Recived: Feb 10, 2004; Revised: Oct 13, 2004; Accepted: Oct 22, 2004

Abstract

The benchmark of a proxy-based service creation and content delivery framework called the Chek Proxy Framework (CPF) was described. Unlike conventional proxy-based systems, CPF was an application level approach that provoked the use of client machines to host at runtime an intermediate object called the Dynamic Application Proxy Server (DAPS) based on the designed clustering policy. We conduced performance analysis of CPF in a video broadcasting environment based on two clustering policies, namely the *Neighbor* and *Regional* clustering policy, under the context of two proxy selection algorithms, i.e. the conservative and greedy algorithms. The results showed that the conservative algorithm works well at low request loads while the greedy algorithm performed better at higher request loads. Compared to the *Neighbor* policy, applying the combined policy (*Neighbor* and *Regional*) with the greedy algorithm further decreased the server workload by 11.38%, and enhanced the average client receiving rate and network throughput by at least 100% and 35%, respectively.

Keywords: Dynamic proxy, application level active network, chek proxy framework, contents delivery

Introduction

CPF (Choong *et al.*, 2003) is an application level approach that addresses the delivery of Internet contents with active services, i.e. by deploying intermediate object called DAPS into the network. DAPS is regarded as the centerpiece of CPF, and hence choosing the right and optimal client machine to host DAPS becomes a critical factor that affects the overall performance of the CPF-enabled systems. This paper focuses on the discussions of two CPF proxy selection algorithms, i.e. the conservative and greedy algorithms, based on two CPF clustering policies, namely the *Neighbor* and *Regional* clustering, in a video broadcasting application scenario. These algorithms are augmented with various settings such as the user-awareness, selection threshold and resource utilisation threshold, to further benchmark the effect of allocating DAPS. A performance analysis of CPF is conducted using simulation, to observe,

[1] *Department of Computer and Communication, Faculty of Engineering, University Putra Malaysia, 43400 UPM Serdang, Malaysia, E-mail: choongkn@hotmail.com or choongkn@yahoo.com*

[2] *Department of Communication Technology and Networking, Faculty of Computer Science, University Putra Malaysia, 43400 UPM Serdang, Malaysia*

[3] *Motorola Multimedia Sdn Bhd, Global Software Group, 16th Floor Menara Luxor, 6B Jalan Persiaran Tropicana, 47410 Petaling Jaya, Selangor, Malaysia*

[*] *Corresponding author*

benchmark, and compare the average WAN throughput, workload reduction of the central server, and average data receiving rate of the clients between the two proxy selection algorithms under the above two different clustering policies.

This paper is organised as follows. Section 2 provides an introduction of the clustering policies and group formation concept of CPF, followed by various proxy selection factors and two proxy selection algorithms. Section 3 describes the simulation methodology, which includes the network and simulation model, simulation parameters, assumptions and performance metric. The detailed discussions on the simulation results are provided in section 4. Section 5 summarises the paper and spells out possible future works of CPF.

Proxy Selection Strategies

CPF Clustering Policy

The CPF works around the concept of clustering, similar to those in Knutsson and Peterson (2001); NTT PF Labs (2003); Chu *et al*. (2000); Pendarakis *et al*. (2001); Erilsson (1994). In addition, it advocates that early clustering at the lowest network hierarchy, i.e. the ISP and enterprise network, serves as a prevention approach to solve congestion. This is because once the traffic has entered the interior of the public Internet, it becomes much more difficult to track, monitor and optimise (CiteSeer, 2004). Furthermore, since most end-hosts (clients) tend to be on the access links, i.e. ISP access network rather than at network backbone, it is desirable to confine traffics on the access links rather than on the network backbone (Pendarakis *et al*., 2001).

In CPF, a cluster is defined as a collection of clients that are located within the same network segment. The scope of a segment is bounded by the number of hops (network distance) between the clients. Each segment is governed by a type of clustering policy, and served by at least one DAPS. CPF performs clustering based on the divide-and-conquer approach and works in a bottom-up manner, i.e. from the lowest subnetwork level to the enterprise network level, before reaching the ISP network level. The three levels of clustering in CPF are:

- *Local* clustering: *Local* clustering focuses on the lowest network level, where clustered clients are within the same subnetwork or LAN. The performance of *Local* clustering has been evaluated with prototype implementation, and reported in our earlier paper (Choong *et al*., 2003).
- *Neighbor* clustering: This clustering technique is applied on clients that are of two network hops distant away, i.e. separated by a common local gateway. Such a technique is based on identifying the similarity of the network prefix or domain name of the clients, as suggested by Krishnamurthy and Wang (2000). *Neighbor* clustering is feasible only if clients that are physically co-located within the network segment share the same network prefix in their IP addresses. Both the *Local* and *Neighbor* clustering techniques are applicable to the campus and enterprise network, depending on how each individual LAN is physically linked and configured.
- *Regional* Clustering: *Regional* clustering concentrates on the ISP network. This clustering technique finds the topological relationship between clients as their requests traverse from the sources, pass through the ISP region, to the central server. A tree or mesh topology is to be constructed to promote an efficient application-level routing. Similar to Bestavros and Cunha (1996); Krishnan *et al*. (2000), CPF uses the traceroute mechanism to form a tree topology for both data and control streams. Any cluster formed at the network levels lower than the ISP region is channeled to separate DAPS(s) in the ISP region.

Group Formation

Within each segment that is governed by a CPF clustering policy, clients can be further gathered as application groups according to some application-level relations, such as application preferences, trust, QoS provisions, reliability,

and machine capacity. Each group is served by one proxy as the leader and its size depends on the serving capacity of the proxy. A new group is formed dynamically at runtime in segment with no existing group, or where the existing groups are functionally expired, i.e. either overloaded or terminated

Basically, for the sake of bandwidth conservation, grouping is formed only when there are at least two clients demonstrating similarities in terms of application relations. Given the presence of the first group, the creation of subsequent groups can be driven by a number of factors. These factors define when the first proxy should stop serving future client requests and when a new group should be allocated. These factors include:

- Number of Clients: This is the simplest way to infer proxy overloading based on a counter-based approach, wherein the proxy workload is directly deduced from the number of clients served. It is easy to implement, and involves less state management (bandwidth) overheads. It uses a threshold that is defined in finite scales based on the capacity of the proxy-hosting machines, to induce the respective maximum serving capacity of various proxy-hosting machines. As an example, by scaling the machine capacity into 5 levels, level-0 denotes machine with the highest capacity while level-4 represents the opposite extent. This is known as the selection threshold of DAPS. The ranking of machine capacity could be judged on the CPU processing power, or a combination of CPU processing power, amount of memory and secondary storage capacity, of the proxy-hosting machine. This approach is feasible only with an assumption that no third party programs (user activated desktop applications) are competing for local resources during the proxy-hosting session.
- Resource Utilisation Level: This is basically a status-monitoring approach where the proxies periodically acknowledge the central server about their resource utilisations. This allows the server to determine the availability of the proxies to serve more clients. Monitoring and updating the status information of distributed proxies is an expensive process, which consumes both the network and proxy resources. Besides, configuring an optimal status-updating interval is also crucial. A small interval could result in frequent updates at the waste of bandwidth, whereas a larger interval (which could be due to highly congested network) could render an inaccurate proxy status to be reported to the central server.
- Operational Time: This is a time-based approach whereby the proxy is limited to work only for a fixed period of time, e.g. one hour at lunchtime, or during off-office hours. The principle adopted here is that the computer resources are volunteered only when there are no major computing tasks in actions. However, DAPS services may be highly demanded during office hours, particularly in streaming lecturing materials or clips in either the academic or business environments, and barely even acquired at some other intervals. Another obstacle of this approach is the absence of a cost factor, such as workload or resource utilisation. As an example, ignoring these costs would result the proxy to work in a highly overloaded manner.

In practice, a combination of the above factors should be considered to ensure smooth application sessions. In our simulation, we have taken the number of clients as the overloading threshold of proxy for its simplicity and practicability.

Selection Factor

In CPF, the two main factors that influence proxy selection are user awareness and resource contribution mode. The user awareness is divided into both user-aware and user-unaware. In the user-aware arrangement, users are asked for the permissions to serve their machines as proxies. This is the case where users are in full control over their machines, e.g. a personally-used machine. In the user-unaware environment, CPF makes implicit utilisation of the user machines without acknowledging the users

individually[4]. The user-unaware technique serves as a comparison template against the user-aware technique.

The resource contribution mode is also divided into two types, namely the dedicated mode and the shared mode. The dedicated mode simply means that the proxy is granted to consume all the resources of the client machine. The shared mode states that only a portion of the resources (e.g. 50% or 75%) should be used. This is to leave room for the executions of other desktop applications. Based on the explanation above, a logical selection preference of the given factors would be alphabetically labeled as shown in Table 1.

Table 1. Selection preferences.

	Dedicated	Shared
User-aware	A	-
User-unaware	C	B

The most effective CPF session would be to allocate DAPS on a dedicated and user-aware machine (A). Running on a dedicated machine allows DAPS to make full utilisation of the local resources and thus subject to less sudden disruptions such as scarcity of resources due to the execution of other applications, or machine shutdown. Being user-aware should not restrict the DAPS from making total utilisation of local resources. Hence, the user-aware and shared column is considered logically irrelevant. The user-unaware and shared column deserve the second priority (B) because it is practically more feasible to allow DAPS to run in a user-unaware environment to tap into only 50-75% of resources, instead of 100% as denoted by (C).

It is unknown how much resource should be tapped in the user-unaware approach. However, to be conservative, we believe that exploiting only 50-75% of resources is logical, while leaving an extra quarter for other computing purposes, such as web browsing and text editing. A 100% utilisation of resources could cause significant interruptions to any ongoing computing work. Our simulation studies the effect of choosing various percentage of resource utilisation for the user-unaware approach in relation to a range of selection thresholds, and makes reasonable conclusions based on the simulation results.

Selection Algorithms

Two proxy selection algorithms have been proposed in CPF, namely the conservative and greedy algorithms. In the conservative algorithm, the best machine is selected among a group of active clients. The next machine is not selected until the current proxy machine is overloaded or failed. The selected machine continues to serve as the proxy even if more capable machines are available within the same cluster. Due to its conservative characteristic, this algorithm does not incur extra communication and processing overheads on monitoring the status of machines. Unfortunately, for the same reason, it has a tendency of not utilising the best available machine to host the proxy.

Contrasting to the conservative algorithm, the greedy algorithm always hunts for a better machine to host the proxy, similar to the Bully election algorithm (Coulouris *et al*., 2001). Being highly dynamic, this algorithm trades-off the extra overheads of status monitoring for (possibly) better proxy performance. Whenever a better machine is encountered, a proxy handover shall be initiated. This involves a series of steps on shifting the workloads from the current proxy to the more capable proxy. In cases where the total number of clients (which includes both 2-tier clients and clients served by the current proxy) exceeds the serving capability of the new proxy, priority shall be given to the 2-tier clients. In such case, 2 proxies will be allocated to serve requests. Assuming all clients have been successfully shifted to the new proxy, the current proxy shall be configured into an inactive mode, waiting to be invoked again.

[4] *This is the scenario of a network where machines are centrally managed by a common administrator, who wishes to contribute a pool of possibly spare machine resources (frequently less loaded) to improve the overall system performance where possible. Typical examples include the voluntary computing (Sarmenta and Hirano, 1999), grid computing and potentially the Storage Area Network.*

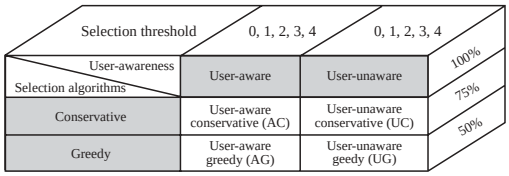

Figure 1. The CPF proxy selection algorithms.

Enriching the above 2 algorithms with the user-awareness factors results in 4 different algorithms, denoted as AC, UC, AG and UG as shown in Figure 1. By governing these 4 algorithms with the selection thresholds, which is scaled into 5 levels (0 - best, 4 - worst), and 3 levels of resource utilisations (100, 75 and 50%), these algorithms are given more options.

Simulation Methodology

We study the performance of CPF by using simulation as a complement effort to the prototype implementation in Choong *et al*. (2003). Doing this allows us to examine the effect of CPF on large networks with different scenarios, and enables us to evaluate different proxy selection algorithms, choices of configuration thresholds, and their impacts on different performance parameters, before applying them into our future CPF system.

Network Model

The campus or enterprise network is considered in this simulation study. Separate links of similar bandwidth connect all clients over a common router. This means the distance between each client is at least 2 hops apart. Each client is assumed to represent a subnetwork, where more clients were attached. Further, it is assumed that no multicast-capable switch or router exists in the network. In this paper, we start by studying the effect of the *Neighbor* clustering policy. From the results collected, we further evaluate the *Regional* clustering policy based on the preferred algorithms and selection thresholds.

Simulation Model

This simulation is based on the event-driven model. Events are generated according to the standard traffic distributions, i.e. the Poisson and 2-state distributions. The simulation models all events occurring in the running of the CPF system where possible, to closely adhere to the CPF hand-shaking setup protocol (Choong *et al*., 2003).

The proxy delivery mechanism is as follows. Upon receiving a stream of video data, the proxy distributes the data to a list of clients in a round-robin fashion. In a LAN, such a distribution could take less than 1 second (at a certain type of video resolution) with the speed of 100 Mbps and the availability of Ethernet-based IP multicasting facility. In a network without multicasting facility, disseminating contents within 1 second is impossible, particularly when the number of clients is large. This is because the interval of the subsequent distribution cycle has a linear relation to the client size. To simulate such a distribution, an equation of the *transfer_rate* has been derived as follows:

transfer_rate = bandwidth_LOCAL _NETWORK/
(clients.size() * bandwidth_VIDEO_TRANSMIT *
PROXY_RELAY_COST* *getRelayDelay(proxy.*
client.machineType) * CLIENT_RECEIVE_COST*
getReceiveDelay(client.machineType)); (1)

This equation is composed according to a list of cost factors as follows:

- Client size: It has a direct impact on the delivery rate.
- Bandwidth requirement of the video: A constant value.
- Proxy relaying weight: A constant value for each proxy, set to 1.
- proxy machine types: The capability of the proxy-hosting machine.
- Client receiving weight: A constant value for each client, set to 1.
- Client machine types: The capability of the client machine.

By multiplying the *transfer_rate* with the bandwidth of LAN, the amount of data received is given by the following equation:

data_received = bandwidth_LOCAL _NETWORK *
transfer_rate (2)

Simulation Parameters

To study the effect of proxy on a typical campus network environment, some simulation studies have been conducted with synthetic workloads. The trace-driven simulation is not conducted for two reasons. First, most server traces available on the web (CS Department, 2003) are of HTTP-traffic types; hence they are not compatible to the CPF underlying transport protocols. Second, these traces are generally collected from globally sparse populated clients instead of those originated from the same subnetwork or enterprise network regions, on which the CPF clustering policies are focusing. Furthermore, the concept of voluntary computing is still new to the Internet user community, and we do not have sufficient statistical logs or data to justify a trace-driven simulation. However, using synthetic workloads allows us to demonstrate that the behaviour of the framework is consistent with our intuitions, and are useful for studying the sensitivity of the framework and selection algorithms to various networking and server parameters.

Several simulation parameters have been synthetically defined as follows:

- Number of clusters: 10. This is an arbitrary value to denote the number of faculties in the campus.
- Total simulation time is 150 slotted time. This value is chosen because the most significant proxy allocation activities happen only at the very beginning of the simulation, and the overall traffics would then be stabilised once sufficient proxies are allocated. Therefore, running the simulation beyond 150 slotted time exhibits little workload difference on the bandwidth and server.
- Client arrival period is restricted to the first 30 slotted time, to study the effect of proxy on network and server workload conservation with large clients arriving within a short period of time.
- Number of clients arriving per simulation slotted time has been assigned values from 2, 4, 6, 8 to 10, for 5 different runnings. This means that there are a maximum of 10 similar requests originating from the campus network to the central server. To study the effect of proxy on different network topology, 20 clients are assigned per slotted time to simulate larger and busier campus network.
- Distribution of machine capacity, scaling at 5 levels with the following ratio: 2 : 10 : 38 : 25 : 25. The ordering is arranged in such a way where the most capable machines occupy only 12% (the first 2 ratio values) of the total machines, average capacity machine at 38%, and the remaining 50% for lower capacity machines.
- Types of client contribution modes: dedicated, shared and ordinary clients (as explained in earlier section).
- Distribution of client contribution (user-aware computing) is of the following ratio: 10, 20 and 70 (as respective to the above client contribution modes). The ratio is conservatively assigned, i.e. only 10% of the clients are fully sharing their machine resources (dedicated volunteers), followed by 20% and 70% as shared and ordinary clients, respectively.
- Distribution of client contribution (user-unaware computing): 10, 90 and 0. In this case, the resource of every client machine is assumed to be usable. However, the possible amount of accessible resources is limited to 50%, 75% and 100%.
- Traffic arrival pattern: Poisson distribution represents the traffic with some bursty nature, and hence it is being used to simulate the client inter-arrival time, with a mean value of 3.3 (IPAM, 2004). The transition between various event stages simulates a series of network connection (on) and disconnection (off); hence represents the 2-state traffic pattern (Liu *et al*., 2000; Moore, 2002).
- Packet size is 32 bytes.
- Initial client request uses 1 packet.
- The size of both proxy and client application object: 320 packets. This comes to 10,240 bytes (10 KB), the exact file size of the customised video proxy and client programs.
- Bandwidth requirement of the video broadcasting system is 320 packets per second

(pps). This value is adopted from the RealVideo standard in delivering a "talking head with motion" video with the total bitrate conservatively coded at 80 Kbps (Technology College Software, 2003).

- Backbone bandwidth (from the local router to the server) is 6,144 packets per second, representing a T1 connection, i.e. 1.5 Mbps.
- Local network bandwidth is assumed to be 50% of a 100 Mbps Ethernet, where the remaining 50% is assumed to be occupied by other applications.

Assumptions

To clearly observe the effect of proxy allocation, the simulation starts with a clean network with no users. For conservative reasons, the most capable clients do not arrive in the first 20 simulation time so as to study the effect of different selection thresholds and the outcome of the greedy algorithm. Further, 3 out of 10 clusters are assumed to have no proxy volunteers. It is also assumed that no delivery interruptions such as proxy failure occur in the simulation. Lastly, no clients quit the system during the simulation to maintain the workload of the network.

Performance Metrics

This simulation collects 4 main performance metrics as listed below:

- Average Client Receiving Rate: This refers to the number of packets received by each client in a second. It is measured as the Total Packets Received (TPR) divided by the Total Connection Time (TCT) of each client. The Client Receiving Rate (CRR), is given as CRR = TPR / TCT. The average client receiving rate can be obtained by accumulating each client receiving rate, divided by the total client size (x), i.e. $\sum (CRR_i) / x$.
- Network Throughput: This refers to the maximum amount of bandwidth allocated to either the proxy or 2-tier client at a specific time. The throughput here is measured as the data rate of 1.5 Mbps divided by the Total Number of Connections (TNC), for the duration of the simulation. The Network Throughput (NTP) is given as NTP = 1.5 Mbps / TNC.
- Number of WAN Connections: This refers to the number of WAN connections made from the client to the server. It consists of both the 3-tier and 2-tier connections.
- Server Utilisation: This refers to the percentage of the resources used for handling client requests. It is based on the number of server threads created to serve the client requests. It is measured as the Number of Threads (NT) times the Unit of resource consumed (U), divided by the Total Resource (TR) for the duration of the simulation. In short, the Server Utilisation (SU) is given as SU = ((NT * U) / TR) * 100%.

Results and Discussions

General Analysis

Table 2 summarises the performance results of both the conservative and greedy algorithms running with different user-awareness settings and selection thresholds (Shaded columns are identified for further studies). The table also shows the effect of selection threshold, n, from 1 to 4. The threshold of zero is not considered because the probability of getting a voluntary machine that satisfies such requirement is rather impossible, given the distributions listed in earlier sections. The numbers next to the name of the algorithm denote the resource utilisation level, e.g. UC-75 refers to the conservative algorithm running in user-unaware mode, which utilises 75% of the computing resources.

When the selection threshold is 1 (n = 1), the proxy selection became highly restrictive, i.e. only machines with capacity level of 1 could be chosen. In the case of AC algorithm, there were only 2 proxies (out of 101) allocated to serve 227 clients. Hence, the total WAN connections made were over 100, where 90% were of 2-tiers. Although it is expected that clients served by the most capable proxy machine should gain much favorable performance, the large number of 2-tier connections had quickly degraded the overall

content delivery performance. As a result, no clients received at 320 pps. The same explanation applies to the other algorithms as well. With the selection threshold of 1, the receiving rates achieved by all algorithms were below 60 pps.

The proxy selection scope becomes wider as *n* is set above 2. As the number of proxy increases (with higher *n* values), the total of WAN connections established is further reduced as shown in Table 2. The number of WAN connections has been reduced from more than 100 down to 11-14 (n = 2), 13-22 (n = 3) and 12-26 (n = 4).

Both the user-aware and user-unaware (with 100% resource utilisation) approaches achieved comparable results when *n* is 2. However, when the *n* becomes larger, the performance difference of both approaches becomes more significant. This is because apart from being able to make full resource utilisation, the user-unaware approach has higher machine availability (as all machines are assumed to be transparently usable). However, as explained earlier, exploiting full resources of client machine transparently is practically infeasible; hence, the results of user-unaware with 100% resource utilisation serve merely for comparison purposes, and are not included for detailed study.

From Table 2, it can be concluded that setting the selection threshold to 2 yields the most optimal performance, i.e. allowing 81% of the total clients to receive at 320 pps, in the case of the AC algorithm. It is also noticed that in general, the AC algorithm performed slight better than the AG algorithm. This could be due to the simplicity of AC algorithm in serving requests, unlike the AG that involves some overheads in status monitoring, and conducting proxy handover.

Performance Comparisons

After excluding the infeasible approaches, and approaches that produce unfavorable results, only 4 algorithms are left for further studies (shaded columns in Table 2), i.e. the user-aware, and user-unaware approach running with 75% resource utilisation. These remaining algorithms are further investigated in terms of the average client receiving rates, bandwidth utilisation of the WAN and the performance of proxy. The effect of proxy on the server workload is discussed in details in section 4.3 under the study of different network topology. The following comparison study is based on the results of setting the selection threshold to 2. Thresholds of 3 and 4 are excluded because they allow only

Table 2. Performance of algorithms.

	AC	AG	UC-100	UC-75	UC-50	UG-100	UG-75	UG-50
***n* = 1**								
Total WAN links	101	101	102	128	131	101	128	131
> = 320 pps	0%	0%	0%	0%	0%	0%	0%	0%
Average Client Receiving Rate	57	56	55	48	47	57	48	48
***n* = 2**								
Total WAN links	11	12	11	12	15	11	14	18
> = 320 pps	81%	71%	75%	76%	17%	69%	67%	6%
Average Client Receiving Rate	404	357	394	380	267	257	349	257
***n* = 3**								
Total WAN links	14	17	13	17	22	19	19	22
> = 320 pps	52%	50%	59%	28%	19%	32%	48%	20%
Average Client Receiving Rate	313	325	326	299	271	298	331	275
***n* = 4**								
Total WAN links	14	19	12	17	26	18	21	26
> = 320 pps	47%	39%	72%	36%	5%	44%	34%	11%
Average Client Receiving Rate	316	315	363	322	238	328	294	249

about 50% of the total clients to receive at 320 pps (Table 2).

Average Client Receiving Rate

Figure 2 depicts the Cumulative Density Function (CDF) of these 4 algorithms with Table 3 detailing the respective actual performance figures. It shows that the AC algorithm outperformed the others significantly, achieved nearly 80% CDF at 320 pps. AC further allows 19% (which is 10% more than the others) of CDF to receive at 450 pps. The next most capable algorithm is also the conservative algorithm running in user-unaware approach, UC. As higher video bandwidth requirements (450 pps) are imposed, most algorithms (except AC) achieved similar performance, which is less than 10% of CDF.

The greedy algorithms seem to contribute little effort although it has a property of always ensuring that the best machine is selected to host proxy. The two most likely reasons of such observation are as follows. Firstly, there are some overheads in evaluating and monitoring the machine capability upon the arrival of a new client. The cost could also come from the time spent on performing proxy handover. Secondly, the client size may be too small to justify the merits of the greedy nature of the algorithm. Thus, we shall further investigate the behaviors of both conservative and greedy algorithms with a larger client size in the later section.

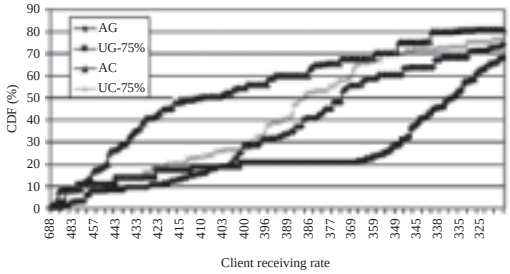

Figure 2. CDF of the selected algorithms with selection ceshold 2.

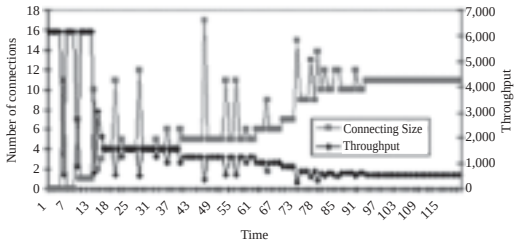

Figure 3. Bandwidth utilisation of WAN.

Table 3 shows that the AC utilised only 11 backbone connections (n = 2), which is the lowest among the 4 algorithms. Consequently, it enables more clients (19%) to receive at higher rates, at an average of 404 pps.

Based on the simulation results, it is concluded that the AC algorithm serves the most optimal performance in terms of client receiving rate and on selecting proxy candidates, considering the settings of user-awareness, resource utilisation, and selection threshold as highlighted earlier.

Bandwidth Utilisation of WAN

Figure 3 shows the effect of the number of connections and the related throughput on the network backbone when dynamic proxy is employed. In general, the throughput increases as the number of connections drops. During the first 80 simulation time, the network throughput highly fluctuated due to the initial direct connections that each client established to the central server. As more proxies were allocated, the network throughput gradually stabilised (beyond simulation time 90). The average throughput was then sustained at about 560 pps.

Performance of Proxy

This section evaluates the performance of the proxy, based on observations of the proxy

Table 3. Performance of selected algorithms with selection threshold 2.

n = 2	AC	AG	UC-75	UG-75
2-tier links	0	0	0	0
3-tier links	11	12	12	14
Total links	11	12	12	14
> = 320 pps	81%	71%	76%	67%
> = 450 pps	19%	8%	10%	11%
Client Avg. receiving rate	404	357	380	349

machine capacity, client heterogeneity, and availability of local proxy. Table 4 shows the detailed performance figures of the AC algorithm (highlighted rows are quoted examples for discussions). As indicated in Table 4, there were only 7 clusters that have proxy volunteers. Hence, it is expected that some proxies may need to serve non-local clients originated from the remaining 3 clusters.

In general, the performance of proxy is not directly affected by the load incurred as the result of serving more clients. However, this is true only if the client size is under a certain threshold, i.e. based on the serving capacity of the proxy. The last 3 rows of Table 4 imply such observations, where the first proxy (proxyID = 0) is capable of achieving slightly higher receiving rate than the less loaded proxyID = 7 although it served twice or more clients. However, the heterogeneity of client machine does affect the overall receiving rate. As an example, there is a difference of 152.5 (494.8 - 342.3) pps, in terms of receiving rate between the first proxy (proxyID = 0) and the second proxy (proxyID = 1), although both are in the same cluster.

Performance figures of cluster 6 (first row of Table 4) reveal the performance impact of serving clients from non-local clusters. Although nearly 42% of its client machines were of capacity levels 1 and 2, the receiving rate of the proxy (proxyID = 3) in cluster 6 is lower than the first proxy (proxyID = 0) in cluster 0. This is due to the fact that about 32% of the total clients in cluster 6 originated from the neighbor clusters with no local proxies.

Nevertheless, from the statistical perspective, the early proxy allocation does help clients to achieve higher average receiving rate. As an example, the first proxy (proxyID = 0) allocated in cluster 0, achieves the highest rate among all proxies, whereas the later proxies (such as proxyID = 3, 4 and 7) achieved slightly lower receiving rate. The most capable proxy machine (proxyID = 4) achieved 63.6 pps (494.8 - 431.2) lower than the first proxy (proxyID = 0) because the first proxy started at the time when the number of WAN and LAN connections were insignificant.

Every network object has a performance limit. As more clients join the network, the delivery bottleneck is expected to shift from the proxy machine to the backbone, reflecting again the problems of the 2-tier scenario, i.e. too many proxies forming direct connections to the central server. Hence, it is important to derive a better scheme to scale CPF further, as will be explained in the next section.

Effect of Network Topology

As explained earlier, allocating proxies as more clients arrived is only a temporary solution. For better scalability, efforts must be made on arranging proxies into some structures,

Table 4. Performance figures of the AC algorithm.

Cluster ID	Proxy ID	Receive rate	Client size	Distribution of client machine types				
				0	1	2	3	4
6	3	394.6	34	0	2	12	11	9
5	6	458.2	6	0	1	1	3	1
4	5	423.4	7	0	0	1	3	3
3	9	446.5	5	0	0	3	2	0
2	4	431.2	10	0	0	6	1	3
1	2	349.6	40	0	0	2	16	22
1	11	429.5	3	0	0	0	1	2
0	0	494.8	40	0	0	2	37	1
0	1	342.3	40	0	0	0	5	35
0	7	421.2	15	0	0	6	3	6

e.g. a hierarchical organisation. Incorporating the combination of both the *Regional* and *Neighbor* clustering algorithms serve this purpose, i.e. by allocating proxies at the ISP network to serve all the campus- or enterprise-level proxies in a cascaded manner. This section investigates both the AC and AG algorithms (with selection threshold of 2) in such a context with a larger client size than earlier studies. Out of 600 total clients, 30 were allocated in the ISP network region while the remaining scattered among the lower network hierarchy. Both algorithms are compared in terms of the client receiving rate. The algorithm that achieves more significant results shall further be studied in terms of the network throughput and server workloads.

Figure 4 shows the CDF of client receiving rate of both the AC and AG algorithms, running with the combined clustering policy. By allocating proxies to function in a cascaded manner, both algorithms are capable of allowing an extremely high receiving rate. This is because only the proxies at the ISP network are permitted to form direct connections to the server. It could be seen that the AG algorithm slightly outperforms the AC algorithm, with an average difference of 5.9%. It is also noted that the combined clustering policy has enhanced the receiving rate by more than 100%, over the performance generated by the *Neighbor* clustering policy (Figure 2).

In the next section, the AG algorithm is analysed to study the performance difference in terms of network throughput and server workload, before and after the use of the combined clustering policy. From Figure 5, it is noticed that enforcing the combined clustering policy yields significant network performance differences, from time 78 onwards. The throughput oscillates centered at the maximum throughput of 6,144 pps. The frequent but brief flops of throughput from time 78 onwards are caused by the setup connections of new clients to the central server. It is also noticed that the throughput gets stabilised towards the last 20 simulation time. Overall, the network throughput has increased by 35% by adopting the combined clustering policy than without, at an average of 3,419.56 pps.

The combined clustering policy also incurs substantial impacts on the server workload as depicted in Figure 6. The workload here is measured by the number of server threads allocated to serve the client requests. Workloads were gradually accumulated on the first half of the simulation run. However, they were actively distributed in the second half of the simulation run, as shown by the presence of brief workload spikes. The workload difference on the second half simulation period, before and after the use of the combined clustering policy is about 11.38%.

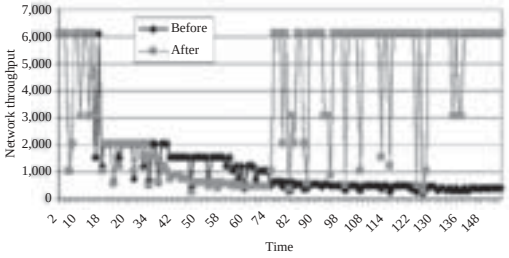

Figure 5. Effects of the combined clustering policy on network throughput.

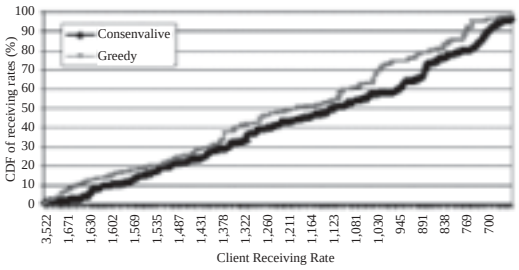

Figure 4. CDF of client receiving rate of the AC and AG algorithms.

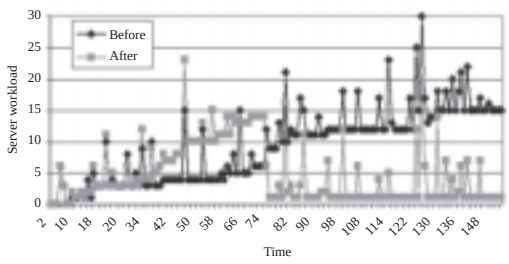

Figure 6. Effects of the combined clustering policy on effects on server workloads.

Conclusion

This paper has studied the overall performance of the CPF system formulated under the context of proxy selection. The study focuses on three main areas: the client receiving rate, network utilisation, and server workload. Two contrasting selection algorithms, namely the conservative and greedy algorithms, have been proposed. These algorithms are augmented with various settings such as the user-awareness, selection threshold and resource utilisation threshold, to further benchmark the effect of dynamic proxy allocation.

In general, the simulations show that the conservative algorithm works well at low request loads while the greedy algorithm operates better at higher request loads. Further, running the algorithm in a user-aware manner gains more favorable results than the opposite case, besides being practically more acceptable. While deploying dynamic proxies at the enterprise network level improves the overall networking performance, introducing them at the higher ISP network level to work in a cooperative manner further scales the overall performance. As examples, the average client receiving rate has been enhanced by at least 100%, the network throughput has increased by 35% and the server workload has been reduced by 11.38%.

Improvements on the proxy selection algorithm could be achieved by deriving an adaptive algorithm that uses either the conservative or greedy algorithms in a selective manner according to the client size. In addition, a scheme is also required to define the selection threshold of proxy-hosting candidatures dynamically based on the distribution of client machine capacity coupled with the voluntary momentum in a given network segment.

References

Bestavros, A., and Cunha, C. (1996). Server-initiated document dissemination for the WWW. IEEE Data Engineering Bulletin, 19(3):8.

Choong, K.N., Mohd, A.B., Prakash, V., Tee, E.R., and Yee, Y.C. (2003). The framework of a dynamic proxy system. Suranaree Journal of Science and Technology, 10(1):7-18.

Chu, Y., Rao, S., and Zhang, H. (2000). A case for endsystem multicast. Proceedings of ACM Sigmetrics; June, 2000; Santa Clara, CA, USA, 12 p.

CiteSeer. (2004). Versatile primitives for application-level multicasting. CiteSeer. Available from: www. citeseer.ist.psu.edu/526529.html. Accessed Feb 14, 2004.

Coulouris, G., Dollimore, J., and Kindberg, T. (2001). Distributed systems: concepts and design. 3rd edition, Addison-Wesley, USA, 772 p.

CS Department. (2003). Web Server Traces. University of Wisconsin, Madison: CS Department. Available from: www.cs.wisc.edu/~cao/icache/trace.html. Accessed Oct 14, 2003.

Erilsson, H. (1994). MBone: The multicast backbone. Communications of ACM, 37(8):6.

IPAM. (2004). Active measurements on the AT&T IP Backbone. University of California, LA: IPAM. Available from: www.ipam.ucla.edu/publications/cntop2002/cntop2002_gramachandran.ppt. Accessed Aug 10, 2004.

Knutsson, B., and Peterson, L. (2001). Transparent proxy signaling. Journal of Communications and Networks, Korean Institute of Communication Sciences, 3(2):10.

Krishnamurthy, B., and Wang, J. (2000). On network-aware clustering of web clients. Proceedings of ACM SIGCOMM; Aug, 2000; Stockholm, Sweden, 14 p.

Krishnan, P., Raz, D., and Shavitt, Y. (2000). The cache location problem. IEEE/ACM Transactions on Networking, 8(5):14.

Liu, E., Cuthbert, L.G., Schormans, J.A., and Stoneley, G. (2000). Neural network in fast simulation modeling. IEEE_INNS_ENNS Proceedings of International Joint Conference on Neural Networks (IJCNN); July 24-27, 2000; Como, Italy, 5 p.

Moore, A.W. (2002). Measurement-based management of network resources.

Computer Laboratory (UCAM-CL-TR-528), University of Cambridge. Technical Report No. 528. 273 p.

NTT PF Labs. (2003). Yallcast Architecture Overview. Japan: NTT PF Labs. Available from www.yallcast.com. Accessed Oct 27, 2003.

Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M. (2001). ALMI: An application level multicast infrastructure. Proceedings of the 3rd Usenix Symposium on Internet Technologies & Systems (USITS); March 26-28, 2001; San Francisco, CA, USA, 17 p.

Sarmenta, L.F.G., and Hirano, S. (1999). Bayanihan: Building and studying web-based volunteer computing systems using Java. Future Generation Computer Systems Special Issues on Metacomputing, Elsevier Publication, 15(5/6):11.

Technology College Software. (2003). Real Audio/Video - Test Page. Technology University of North Carolina, Wilmington: Technology College Software. Available from: www.uncwil.edu/tc/real. Accessed Sept 20, 2003.