Master Project/Thesis Proposal

Compare different approaches of TCP Delayed Binding in a Content Switch in addition to emphasis on Load Balancing and Fault Tolerance

Chandra Prakash

1 Committee Members and Signatures:

Approved by Date

_____ _____
Advisor: Dr. Edward Chow

_____ _____
Committee member: Dr. Jugal Kalita

_____ _____
Committee member: Dr. Bill Ayen

2 Introduction

The tremendous growth in World Wide Web usage has become a double-edged sword for operators of large web sites. On the one hand, increases in request volume translate into increased subscription, advertising, and hosting revenue. On the other hand, scaling web sites to meet this increased demand has become more and more difficult as the number of requests for content exceed a particular server's ability to respond. In the best case, users will experience degraded service, in the worst case the server can be driven to collapse resulting in a complete loss of service.

One approach to alleviate handling of large volume of requests is to distribute their load among a group of servers. A master controller, that can be a dedicated host or a process, first receives the requests and delegates it to the appropriate real server. This describes a typical load balancing system. A content switch (CS) is such a load balancing system that distributes load based on the content of the received requests.

There are conventional ways of load balancing at the transport layer (Layer 4 of TCP/IP[1]). One of them is to use the port number of the incoming request and direct it to a real server responsible for handling the response for that specific port. For example, if the port number in the incoming request is 21 it can be routed to machine catering FTP requests and if the port number is 80 it would be routed to a host running the HTTP server. This mechanism cannot differentiate between requests for different content on a single web site as shown in Table 1.

Scope of Layer 3 Switch <---------- >	
Scope of Layer 4 Switch <---------- >	
Scope of Layer 5 Switch <---------- >	

[1] Since WWW systems primarily use HTTP protocol which is built on top of TCP/IP, when we refer to a protocol layer mean it in the context of TCP/IP (which has 5 layers) and not OSI (which has seven layers) unless explicitly stated.

IP (Layer 3 Switch)	Transport (Layer 4 Switch)	Content (Layer 5 Switch)
Src 01/Dest 02	Port 80	http://www.yeehoo.com/news
Src 01/Dest 02	Port 80	http://www.yeehoo.com/sports
Src 01/Dest 02	Port 80	http://www.yeehoo.com/books
Src 01/Dest 02	Port 80	http://www.yeehoo.com/movies

3 Thesis Plan

The largest deliverable of this project will be the a working software implementation of the content switch [1] whose architecture is as shown in the figure 1, which appears in Dr. Chow's introduction to content switch presentation [10] below:

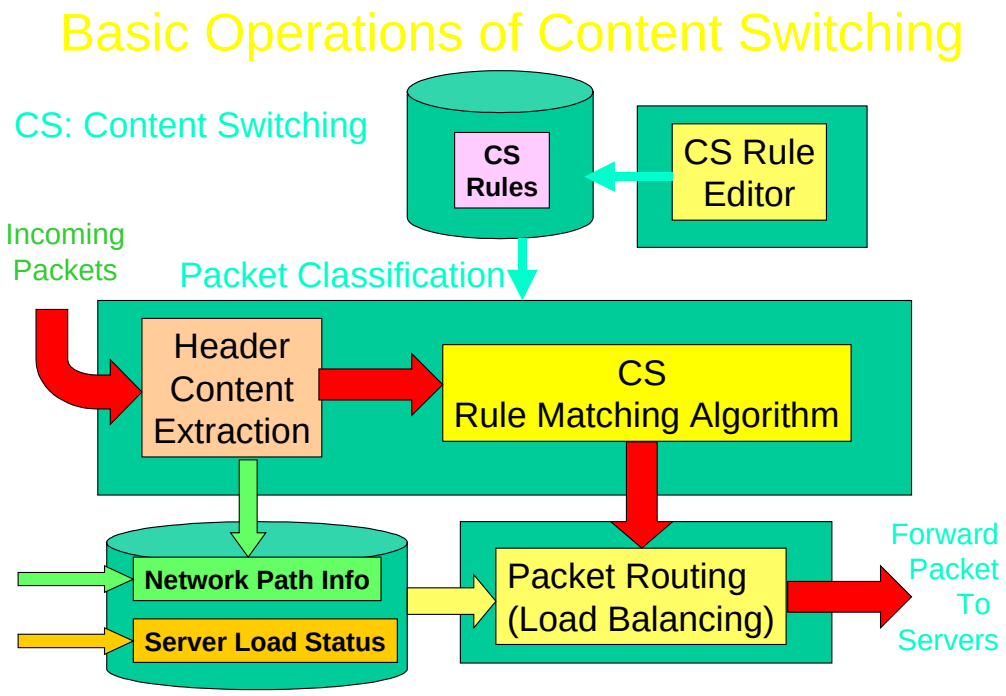

Figure 1: *High Level Architecture of Content Switch*

Here is a brief description of key components shown in Figure 1:

1. CS Rule Editor: CS Rule editor allows user to specify the set of rules for the content switch.
2. CS Rules: Each rule can consist one or more boolean conditions on fields extracted from the incoming request. If all conditions of a rule are satisfied the request is routed to the target server as specified by the rule. Here is a set of sample rules:

```
R1: if (xml.purchase/totalAmount > 52000) { routeTo(server1, STICKY); }
R2: if (xml.purchase/customerName == "CCL") {
      routeTo(server2, NONSTICKY); }
R3: if (strcmp(url, "gif$") == 0) { routeTo(server3, NONSTICKY); }
R4: if (srcip == "128.198.60.1" && dstip == "128.198.192.192" &&   dstport == 80) {
        routeTo(LBServerGroup, STICKY); }
```

3. Header Content Extraction: This component is responsible for extracting the headers from the incoming requests and parsing out fields with their values that comprise the boolean conditions of the CS rules.
4. CS Rule Matching: The input to this component will consist of fields extracted by the content extraction module. This component will use efficient algorithms using input fields values to select the (set of) real server(s) that can handle the request.
5. Load Balancing: As the name suggests this component is responsible for selecting the real server with minimal load among the set of possible servers that can handle the request. The decision will be based on dynamic load balancing policies. A few potential approaches for load balancing are discussed in sections 4.2 and 4.3.

4 Tasks

4.1 Improve TCP Delay Binding

In TCP, the client will not deliver the upper layer request content until it finishes the three way handshake with the server. The content switch needs to act on behalf on the real server to commit the sequence number in the SYN/ACK packet. After receiving the upper layer request content, the content switch establishes another three way handshake with the real server and serves a bridge for relaying the data packet between the client and the real server. This is called ***TCP delay binding***. Since the sequence number committed by the real server and the content switch will be different, it is required to modify the sequence number on both direction. This is so-called ***TCP delay binding problem***. This is shown in figure 2, which appears in Dr. Chow's introduction to content switch presentation [10].

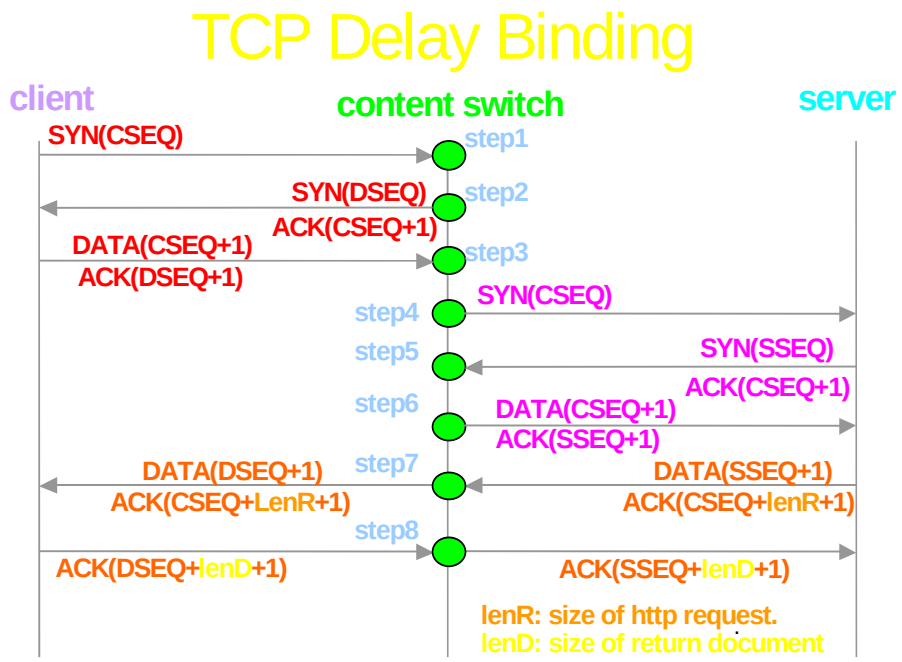

Figure 2: *An illustration of TCP Delayed Binding in Content Switch*

4.1.1 Investigation of Moving the Sequence Number Translation Logic to the Real Server

The content switch after negotiating the initial sequence number with client and selecting the real server can send the sequence number translation rule to the real server. The real server will directly reply to the client with the correct translated sequence numbers in the outgoing response. The real server also needs to spoof the source address in the response to that of the content switch so that the client is not confused as it originally sent the request at (Virtual IP Address) VIP which is the content switch IP address. The benefit of this approach is processing at the content switch will be reduced and its chief task will be only to select the best server using rule matching.

4.1.2 Comparison of Techniques for Improving TCP Delay Binding

There are also pre-allocate server scheme and filtering scheme [10] for improving the TCP delay binding processing in content switching. We will implement these three schemes in a network prototype and compare their performance with the basic TCP delay binding.

4.1.3 Solution to "Potential" Problems resulting from Keep-Alive HTTP Session

Even though we have observed using a sample html document containing few jpeg images in both Netscape Navigator 4.5 and IE 5 browsers that in a Keep-Alive HTTP Session subsequent request is sent only when the reply for the previous request is received in the "same" TCP/IP connection, the HTTP/1.1 RFC relaxes this assumption. It specifically mentions that in a Keep-Alive (or persistent) HTTP connection requests can be pipelined in one connection without waiting for the results of previous requests. May be true request pipelining is used when the requested HTML document is very large and has many embedded links. This is yet to be ascertained. If true request pipelining is really present, we need to multiplex the responses from possibly multiple real servers into one consolidated response for the client with the right set of sequence numbers. Following are two suggested approaches to handle this situation.

1. Cache the responses at the content switch and deliver it to the client as and when the response completes. This will allow segments for one response to be assigned one contiguous series of sequence numbers.
2. Buffer all requests at the CS in case of multiple Keep-Alive requests. Send only one request at a time to the real server. As soon as its response is complete submit the next request in the input queue and so on. In this manner only request need to be buffered (which are usually small in size) as opposed to responses in case 1 above. Possibly this is the way that an HTTP server handles multiple responses in a Keep-Alive Session with multiple requests.
3. Keep-Alive option is only present in HTTP/1.1. The server can deny such requests and force the client to resubmit that request and hence the client will learn to submit different requests in different connections subsequently. This mechanism is provided in HTTP for backward compatibility.

4.1.4 Detailed Investigation of Using Direct Routing and IP Tunneling instead of NAT for Content Switch

The literature for DR and IP tunneling is still under study [14].

4.1.5 Connection Reuse

To remove the overhead associated with connection establishment and tear-down, a set of connections between content switch and real servers can be set up ahead of time and reused. This has definite advantages especially when the response size is very small and majority of packet traffic is that of SYNs/ SYN ACKS and FINS. Also it is seen that due to TCP/IP slow start algorithm the bandwidth used around initiation of a TCP/IP connection is very little as compared to what can be achieved during steady state.

4.2 Load Balancing Information Gathering

Indira Semwal in her thesis report an architecture for load balancing a web clusters [13]. In her work she mentions about using both the server load and characteristics of path leading to the chosen server. This work is useful when

the chosen server can exists in a different network as that of the CS. The following are two suggestions on load balancing information gathering for content switch:

1. The load balancing information can be obtained during real server startup as described before. During steady state, the CS can proactively send requests to real servers for their load at configurable heartbeat intervals. The real server will send replies containing their load information to CS for such requests. What exactly will comprise the load of the server is undecided at this stage and needs to be investigated. One potential way to obtain this information is use information provided system monitoring tools like vmstat or system activity report (sar).
2. The load balancing information is collected at the CS itself that can be used to estimate primarily the network activity of the real server. The information will essentially be the number of active connections to the real server or the overall bandwidth to/from it at that instant. This can be coupled with the static system profile (obtained initially during registration) to come up with an index that determines most suitable server during dynamic server selection.

A third approach which is a combination of schemes 1 and 2 discussed above can be used.

4.3 Dynamic Server Selection

The incoming request from the client will be parsed and a broad classification of the request type will be obtained by the CS Rule Matching algorithms [12]. Once the type of request is determined it can be mapped to a logical cluster that can serve this request. The elected cluster name will be used by the CS, load balancing module to pick the optimal choice among the members of the chosen cluster. The IP address information of the final selected real server can obtained from a name to IP address mapping table. This table can be either in memory or possibly stored in a database. Note that this mapping information is populated from the registration request sent by real servers.
This is borrowed from ASAP/ENRP scheme [7][8] of server IP address resolution from its logical name.

4.4 Dynamic Cluster Maintenance

Each real server will be part of one or cluster groups. Each cluster will have a unique logical name assigned to it. During initial set up the real server will register with the content switch. In the registration request the real server will provide the cluster name it belongs to and the set of IP Addresses at which it can be contacted, if it is multihomed. In addition to the cluster specific information the real server will also supply its load balancing policy. Since present CS rules mandate fixed set of clusters, a real server will be allowed to be registered if its cluster name is one among the fixed set of cluster group. This mechanism allows dynamic addition and removal (via de-registration request) of a real server from a cluster group. This solution is borrowed from ENRP [8] server registration scheme.

4.5 Fault Tolerance

If during steady state the connection to the selected real server breaks down due to network breakdown or host crash, the content switch can redirect the traffic to the next available member of that cluster. If the failure that resulted in connection breakdown was network related the CS can initiate the connection to an alternate IP address of the real server, if it is multi-homed, without client even knowing about it. The issue here might be to recover the state of the transaction exactly from a stage just before the breakdown. If the state somehow cannot be recovered the best solution will be to report failure to the client and let it retry. I believe there are headers in HTTP responses that specifically tell the client to retry. This scheme is borrowed from SCTP /ASAP [6][7] fault tolerance mechanism.

Heartbeat, Mon, and Ldirector are used in high available Linux Virtual Server system [14][15]. We will explore the use of two content switches to achieve a fault-tolerant highly available content switch system.

4.6 Miscellaneous Findings and Suggestions

- Possible use of redirect feature of HTTP servers. What this means is if server is not able to handle the request it can redirect request to a different server that the client will retry the request. In this scenario content switch only serves to find the best server and send a redirect response to client for that real server. The possible drawback is

problem with Keep-Alive connection would not be efficiently used as client has to reopen the connection to the redirected server.

- Possible use of caching techniques at CS. The cache can be validated using proper cache coherency algorithms as suggested in the RFC of HTTP/1.1[4]. This can significantly reduce the response time and server processing overhead for idempotent requests like GET and HEAD. The cache need not be very large. A size of around a few megabytes or few 100 kilobytes can perform well using good caching algorithms.
- Other issues like security, best rule matching algorithms etc., can also be looked into depending on the thesis progress and time constraints.

5 Deliverables

- A working software implementation of content switch.
- A thesis report documenting the design, implementation and performance benchmarks of Content Switch. In addition a description on related technologies and scope for future improvements.

6 Timeline of Thesis work

Based on a first hand estimate of effort involved in the identified tasks, here is the tentative schedule of work to be done in the thesis:

- Improve TCP Delay Binding: February-20 - March 15.
- Load Balancing Information Gathering: March-16 - April 15.
- Dynamic Server Selection: April 16 - May 15.
- Dynamic Cluster Maintenance: May 16 - June 15.
- Fault Tolerance, Miscellaneous Findings and Suggestions: June 16 - July 15.
- Thesis Report Writing: July 16 - July 30.

7 References

1. George Apostolopoulos, David Aubespin, Vinod Peris, Prashant Pradhan, Debanjan Saha; "Design, Implementation and Performance of a Content-Based Switch." http://cs.uccs.edu/~chow/pub/contentsw/paper/contentsw/apostolopoulosum.pdf
2. RFC 793, Transmission Control Protocol (TCP/IP), ftp://ftp.isi.edu/in-notes/rfc793.txt.
3. RFC 1945, Hypertext Transmission Protocol (HTTP) / 1.0, ftp://ftp.isi.edu/in-notes/rfc1945.txt
4. RFC 2068, Hypertext Transmission Protocol (HTTP) / 1.1, ftp://ftp.isi.edu/in-notes/rfc2068.txt
5. Jeffery C. Mogul; The Case for Persistent Connection HTTP, Digital Western Research Laboratory, http://www.research.compaq.com/wrl/publications/abstracts/95.4.html.
6. RFC 2096, Stream Control Transmission Protocol, IETF proposed standard, ftp://ftp.isi.edu/in-notes/rfc2960.txt.
7. Aggregate Server Access Protocol (ASAP), IETF draft standard for highly available data transfer mechanism in pool of servers using a name based communication model, http://www.ietf.org/internet-drafts/draft-xie-rserpool-asap-01.txt.
8. Endpoint Name Resolution Protocol (ENRP), IETF draft standard to provide a fully distributed fault-tolerant real-time translation service that maps a name to a set of transport addresses pointing to a specific group of networked communication endpoints registered under that name, http://www.ietf.org/internet-drafts/draft-xie-rserpool-enrp-01.txt.
9. Douglas E. Comer. Internetworking with TCP/IP, Principles, Protocols and Architecture.
10. C. Edward Chow, "Introduction to Content Switch," http://cs.uccs.edu/~cs622/doc/contentsw.ppt.
11. C. Edward Chow and Weihong Wang, "Design and Implementation of a Linux-based Content Switch," UCCS Tech Report EAS-CS-2001-3, submitted to the Second International Conference on Parallel and Distributed Computing, Applications, and Techniques, http://cs.uccs.edu/~chow/pub/contentsw/status/chow1.doc.
12. C. Edward Chow Ganesh Godavari, and Jianhua Xie, "Content Switch Rules and their conflict Detection," UCCS Tech Report EAS-CS-2001-4, submitted to the Second International Conference on Parallel and Distributed Computing, Applications, and Techniques, http://cs.uccs.edu/~chow/pub/contentsw/status/chow2.doc.
13. Indira, Semwal, Improving web server cluster performance using load balancing agents.
14. Linux Virtual Server (LVS) documentation, http://linuxvirtualserver.org/Documents.html

15. High Availability Linux Project, http://www.linux-ha.org/.