# URL-BASED SWITCHING

# WHITE PAPER

## ISSUE 1: FEBRUARY 2001

## CONTENTS

1 PURPOSE AND SCOPE OF THIS DOCUMENT ..... 1
2 URL-BASED SWITCHING ..... 2
3 APPLICATION EXAMPLES ..... 5
4 APPENDIX A ..... 7
4.1 USING TCP TERMINATION FOR URL SWITCHING ..... 7
4.2 EVENTS IN A TCP TERMINATOR IN THE LIFE OF A HTTP SESSION ..... 7
4.3 TCP REQUIREMENTS ON TCPT DEVICES ..... 8
4.4 ACTIVITIES FOR THE INBOUND PACKETS FROM WAN SIDE ..... 10
4.5 'SLOWER' PATH ACTIVITIES ..... 11
4.6 'SLOW PATH' AT TCPT ..... 12

## LIST OF FIGURES

FIGURE 1 WEB CACHING ..... 5

FIGURE 2 WEB SWITCHING USING PACKET CONTENT (URL) ..... 6

FIGURE 3 URL SWITCHING WITH TCP TERMINATION ..... 8

# 1 PURPOSE AND SCOPE OF THIS DOCUMENT

This document investigates Layer 7 (OSI reference model) URL-Based packet switching.

# 2 URL-BASED SWITCHING

Switching packets based on Layer 7 information content, for example URL data, allows better management of server based applications and improves the performance and reliability of Internet services. URL switches allow traffic to be managed on a per request for data instead of on a per connection basis.

URL switches optimize web traffic by using URLs in addition to IP addresses to make switching decisions. Unlike conventional routers / switches that direct content requests based on TCP or UDP port numbers, URL-based switches direct content requests based on URLs or even cookies.

In addition to URLs, cookies can help manage web content flexibly and give still greater control of the traffic. For example, cookies facilitate customized web applications and user tracked content as well as directing HTTP requests to specific servers. Cookie cutting is the ability to capture, parse and switch traffic based on HTTP cookies. Commonly used in search engines, shopping carts, web-based e-mail and secure e-commerce applications, cookies help identify and classify traffic by individual user. Cookies enable web applications that require session persistence. Persistence refers to transactions from a client that must be processed by the same server to ensure the integrity of each transaction. Common applications are the maintenance of items added to an online shopping cart and the checkout transaction where credit card processing occurs.

URL switching will enable the building of more efficient web server farms. URL-based switches can redirect requests based on content type, such as image or video, to servers that are optimized for that particular content type. They can also direct requests for dynamic content to live servers and requests for static information to caches. In a web server farm, content is replicated among servers for load balancing. Fault tolerance URL switching makes it possible to use partial replication of web server contents, thus simplifying capacity planning. Servers with hot content can easily be segregated.

Internet service providers would like to provide service differentiation for commercial applications. Some beneficial commercial application services are:

- Firewalls – blocking traffic sent by insecure sites
- Resource reservation – preferential treatment for premium traffic
- QoS routing – routing based on traffic type and source.

Content-based networking would allow companies to prioritize and align network resources.

Content-based switching is becoming increasingly important as web server farms and web caches become more widespread. Devices that only provide Layer 4 services can

use TCP or UDP transport layer information, for example port numbers found in TCP/UDP headers, to make packet-forwarding decisions. Content-based switches can use application-layer information, such as packet payload.

Layer 4 switches direct all traffic with a particular destination TCP port to a particular network port. For example, a Layer 4 switch might direct all traffic going to port 80, the HTTP traffic port, to a particular port on the switch where a web cache is attached. This may be done to provide transparent web caching. More sophisticated Layer 4 switches might provide additional functionality such as Network Address Translation (NAT), load balancing among servers or caches, server fault tolerance, and configurable quality of service for different kinds of traffic. Basic Layer 4 switches, which provide functionality such as load balancing and support for transparent caching, perform TCP traffic redirection by redirecting the initial SYN packet from the client to the chosen destination and redirecting all subsequent packets on the connection to the same destination. To accomplish such redirections, the switch only needs to look at the IP and TCP headers for connection information such as IP address and TCP port numbers. Basic Layer 4 switches redirect IP TCP traffic at a switch by looking at SYN packets. Redirecting TCP traffic based on content or application level (Layer 7) information is not as simple. For TCP transactions, application level information is not available until the TCP connection establishment phase is completed.

Content-based switching refers to the capability of a switch, located in front of clients or servers, to redirect HTTP requests to servers based on the URL specified by clients' GET requests. When a user enters a URL into a browser, the browser constructs an HTTP GET request that contains the URL and other HTTP client header information. With content-based switching, a switch located on the path between the client and the servers will intercept the GET request and make a decision about the server to which the request should be directed.

The TCP connection request from the client needs to be terminated at the switch and the connection established between the client and the switch before any application level information can be received. Once the application layer information is received, it is parsed to determine which back-end server or cache should receive the request before the request is redirected. A second TCP connection is established between the switch and the backend server and the client request is passed to the backend server through this connection. The response is received at the switch from the backend server on the second connection and transferred through the first connection to the client. This happens transparently to the client.

Let us consider a specific URL switch example. In this instance, an HTTP request from a client to a server is transparently inspected by a content-based switch that forwards it, depending on the object being requested, to either a server or to a cache. For example, if the request is for a non-cacheable item, it will be forwarded to a server.

Normally, when a client makes an HTTP request, a TCP connection is first established with the server. The client then sends information pertaining to the object requested as a part of a GET request. The server parses this request and returns the object to the

client. When a URL switch is introduced, assuming the destination is port 80, the switch transparently intercepts the connection request packets from the client and sends them up to a local application level proxy that understands HTTP. A TCP connection is established between the client and the proxy. The proxy masquerades as the original server to which the client request was made. The client then sends the GET request to the proxy. The proxy determines the destination based on the requested object. The proxy establishes a TCP connection with the destination, forwards the GET request on this connection, receives the response, and transfers it back to the client.

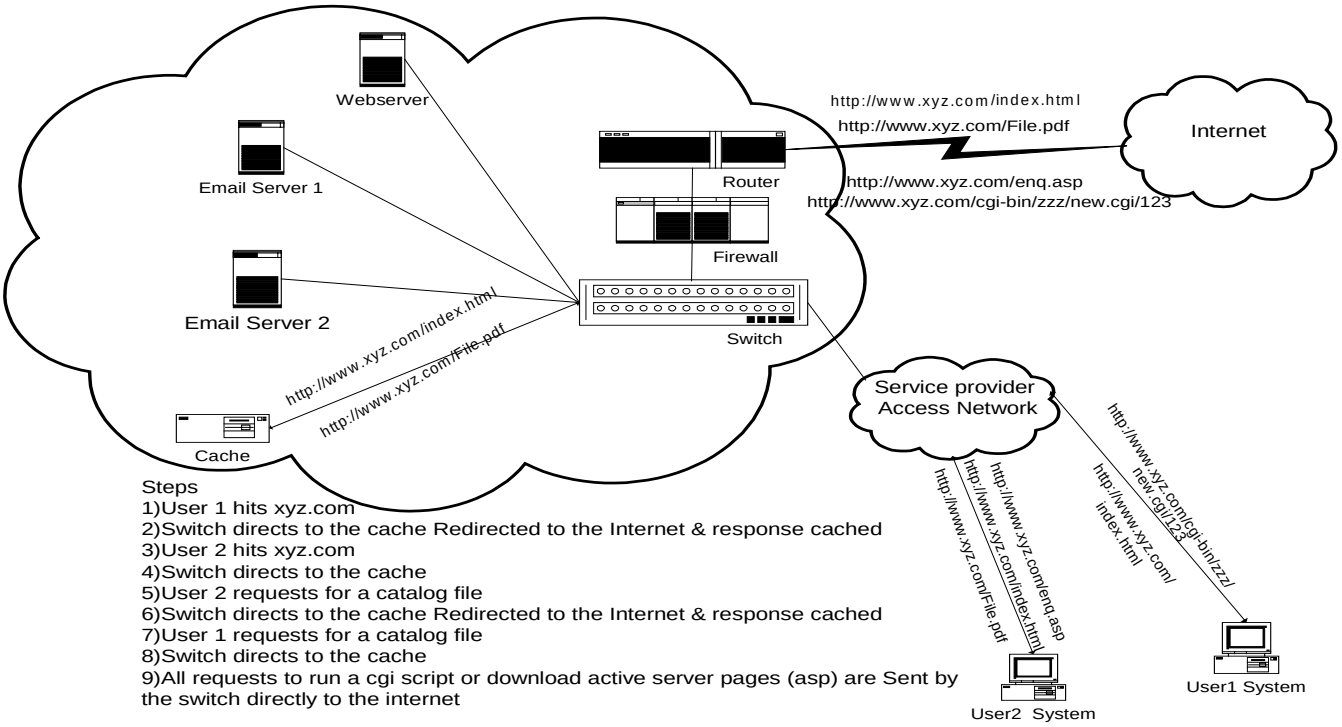
Content-based switches use the information found in the payload of packets to provide even more sophisticated capabilities. For example, the URL found in HTTP GET requests can be examined to determine whether an image is being requested. If an image is requested, all packets belonging to the TCP connection corresponding to this request can be switched to a server that is optimized to deliver images. Cache efficiency can be improved by parsing the content of HTTP requests at the switch. Requests for un-cacheable content, for example the results of a CGI script, will be sent to a web server instead of a web cache, thus eliminating unnecessary load on the cache. Content-based switches are also capable of converting complete URLs to absolute URLs in GET requests to support transparent web caching with standard proxy caches. Looking beyond the URL, it is also possible to parse cookie information that might be carried as part of the HTTP header and to make switching decisions based on this information.

Content-based switches perform tasks that are related to specific applications. Hence, the switches have to evolve with the applications and the ways in which the applications are used. The ability to execute multiple programmable and conditional classification sequences becomes extremely important. As a result of the need to support more complex and rapidly evolving content-based switching capabilities, new switch architectures are appearing that include powerful network processors / CPUs. Some of these new switches contain network processors / CPUs dedicated to ports on the main switch for rapid packet manipulation, a main CPU for management and configuration, and various hardware components for accelerating common functionalities such as table lookup and packet queuing. Network classification processors can provide hardware acceleration to off-load these switch specific processors and ensure better performance.

In addition to facing hardware challenges, the developer of content-based switches must devise software architecture that will facilitate the implementation of the complex functionality of the switch. Content and policy SDK will ease this implementation.

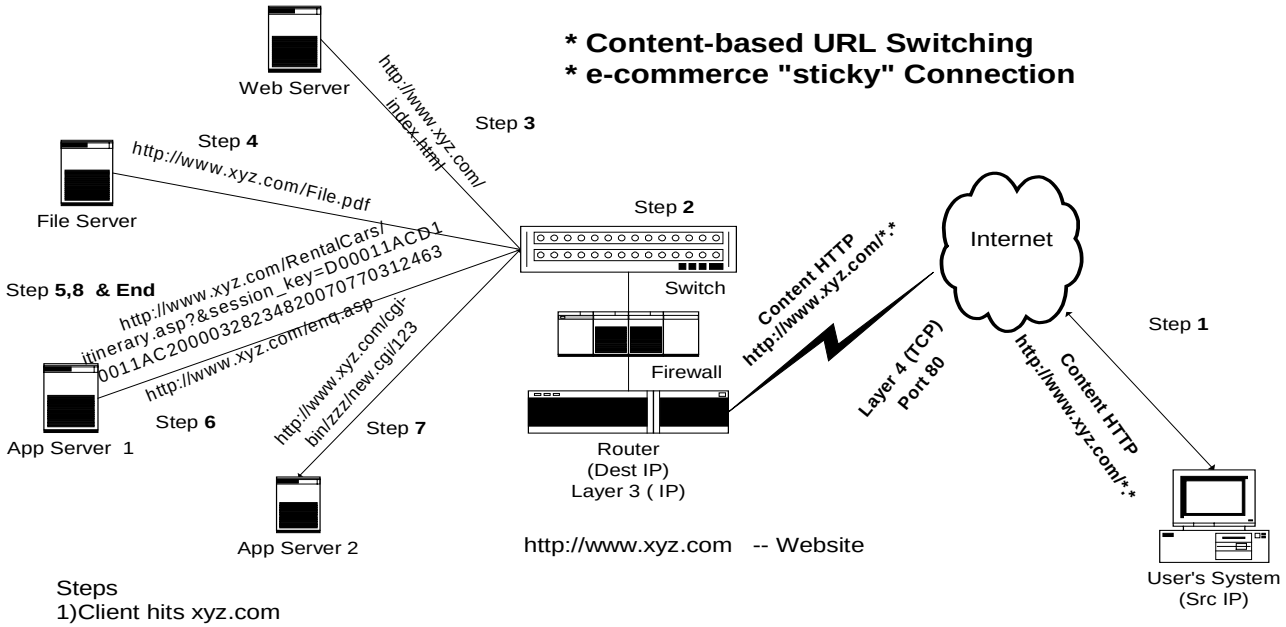
# 3 APPLICATION EXAMPLES

## Figure 1 Web Caching




Steps
1)User 1 hits xyz.com
2)Switch directs to the cache Redirected to the Internet & response cached
3)User 2 hits xyz.com
4)Switch directs to the cache
5)User 2 requests for a catalog file
6)Switch directs to the cache Redirected to the Internet & response cached
7)User 1 requests for a catalog file
8)Switch directs to the cache
9)All requests to run a cgi script or download active server pages (asp) are Sent by the switch directly to the internet

## Figure 2 Web Switching Using Packet Content (URL)




Steps
1)Client hits xyz.com
2)gets connected to the site
3)gets switched to the Webserver
4)Requests for a catalog
5)hits rentalcars& session key allocated
6)makes enquiry
7)clicks an option which runs a cgi script
8)Completes the rental transaction
Session ends

# 4 APPENDIX A

## 4.1 Using TCP Termination for URL Switching

Appendix A discusses the requirements for performing URL switching with the help of TCP termination performed using ClassiPI. Other networking devices that require TCP termination are Web (URL) switch, Web Cache, Web Balancer and Web Director etc. Hereafter, these devices are referred as TCP terminators.

## 4.2 Events in a TCP Terminator in the Life of a HTTP session

When a session starts from the client site with a request to a web server, the following events would happen at the TCP terminator (TCPT).

1. The TCPT would spoof as a web server. This requires maintenance of a partial TCP state machine that would take a connection to the established state.

2. Once the connection is established, TCPT would track the HTTP request over this connection from the client.

3. An Access Control List (ACL) policy to determine original server, load balancing algorithms and any other HTTP specific processing would be applied to the contents extracted from the HTTP request packet.

4. To spoof the client side, a new TCP connection would be set up with the original server based on the rule match. Thus for a single session, TCPT would be handling two TCP connections, one in the role of a server and the other in the role of client.

5. For each packet received over a connection from the client, Address Translation would be applied to pass a packet to the actual server.

6. A similar process would occur on the reverse path of the packet (i.e. from the server via TCPT to the client). No string (URL) search is required on the reverse path.

7. Both TCP connections would be removed from the TCPT based on the connection termination requests on either side.

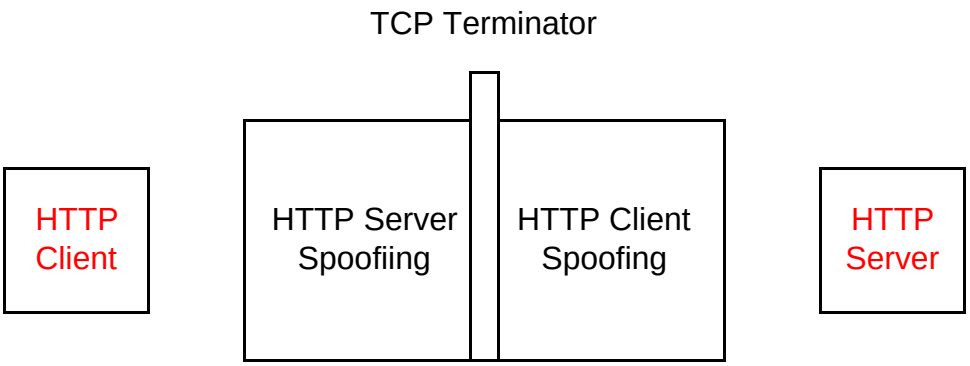
Some other possible flows are:

1. Due to authentication requirements, a session may not be terminated at TCPT. A session termination request would be forwarded to the server.

2. Similarly, due to other requirements, a session might be supposed to go to the server directly and hence packets over this session are only forwarded from Inside to Outside and from Outside to Inside.

These other possible flows require that new rules be added to the Access Control List that instruct an incoming session simply to be forwarded, or bypassed, to the server without TCP termination. It might be that, for authentication purposes, the client would be required to have a session with the original server. If this authentication is at application level, and for security reasons server addresses are not available outside and only virtual addresses are known outside, then address translation might always be required.

**Figure 3 URL Switching with TCP Termination**




## 4.3 TCP Requirements on TCPT Devices

TCPT implementation would require the following TCP code:

1. TCP connection establishment mechanism
2. TCP connection close mechanism
3. Global sequence number and TCP port numbers
4. Handling of acknowledgement numbers

### Different Paths

It is assumed here that various activities required to be done at TCPT can be grouped together. The grouping could be based on the time taken by those activities, its need and its impact on performance to process packets at wire speed. One schema has been suggested here. With multiple tasks running on a network processor, these activities could be done as separate tasks running at different priorities. Following are the paths and activities that are performed in these paths.

## Data Structures

Following are the data structures required.

## Some global variables

1. TCP sequence number
2. TCP port numbers

## TCP Connection Table

1. SIP
2. SP
3. DIP
4. DP
5. TCP flags
6. TCP state
7. Initial send sequence number
8. Initial receive sequence number
9. Send window size
10. TCP send sequence variables
    a. Send and unacknowledged
    b. Send next
11. TCP receive sequence variables
    a. Receive window
    b. Receive next
12. TCP acknowledge number

### Accepting the Data Packets over a Connection:

Each data packet over a connection from a client to TCPT would be searched for the presence of one string amongst a set of strings within the URL. Here it is assumed that the first data packet would contain the URL string. (With HTTP1.0 this would be true?)

### Select the application server based on matched pattern/ACL:

When data packets from the client to the server match a URL or an ACL, the application server address is selected on that basis.

### During a connection setup with the original server:

When managing acknowledgements and sequence numbers between a pair of connections, each pair would contain one connection with the client and one connection with the original server.

## 4.4 Activities for the Inbound Packets from WAN Side

Following are the assumptions about the fast path.

Rules in the ClassiPI are available for a connection. A connection entry already exists in the connection table. The corresponding connection with the application server also exists. The following are the activities that would occur per packet:

- Load an incoming packet to the ClassiPI. (SIP, SP and payload of minimum128 bytes)
- Process the packet after the ClassiPI results are available.

1. If a new URL rule is matching, branch to the slower path, else go to step 2.
2. Fetch the corresponding connection table entry.
3. If FIN or ACK for FIN is present, branch to the slower path, else go to step 4.
4. Modify SIP/SP.
5. Modify DIP/DP.
6. Modify acknowledge number.
7. Modify sequence number.
8. Calculate incremental TCP checksum.

9. Calculate incremental IP checksum.

## 4.5 Slower Path Activities

### 1. Complete a connection setup process that has been initiated by the client (Spoofing and accepting TCP connection from the client)

The following processing is required in this case:

1. A TCP packet with SYN/NO ACK would be received by the TCPT.
2. If a connection already exists in the connection table then it is a retransmission of the SYN. A response with ACK/SYN would be sent out. This could be done by packet on the fly modification to the incoming packet.
3. Otherwise, this is a new connection request. Create a connection table entry and send a packet with SYN/ACK to the client.
4. If an ACK has been received (i.e. three-way handshake is over) the connection state is marked as established. Modify the connection state information in the connection table.
5. Add a rule to the ClassiPI for this connection.

### 2. Setup a new connection with the original server

The following processing would happen:

1. Send a SYN/NO ACK packet to the selected original server.
2. If a SYN/ACK has been received from the original server, send ACK. A connection to the original server is complete. Add required information in the connection table. These connections would require a temporary database until a connection is setup. The final table index for this connection would be linked with ClassiPI rules.
3. If ACK is not received for a SYN, send a SYN/NOACK again after timeout.
4. If SYN/ACK is received again for this connection from the original server, send ACK.

### 3. Tear down the connection with the original server

Following are the steps for tearing down the connection:

1. Send FIN to the original server; update the state in the connection table entry.
2. As per TCP state machine either ACK or FIN/ACK would be received.

3. Resend the ACK or FIN if required.

4. Start the 2*MSL timer and then remove this entry from the connection table.

**4. Tear down the old connection with one original server and setup a new connection with another original server**

A new URL might be in the session that would be served by the other original server. This would be a mix of activities listed in 2 and 3 above.

## 4.6 Slow Path at TCPT

1. Do minimum packet buffering on the client side connection while the original server side connection is being setup.

2. Perform the retransmission if required on the either side of the connection

NOTE: The issue of retransmission is currently out of scope but if required then it would be in a slow path. As mentioned earlier, slow path would also execute on the network processor.

## CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information: document@pmc-sierra.com
Corporate Information: info@pmc-sierra.com
Application Information: apps@pmc-sierra.com
(604) 415-4533
Web Site: http://www.pmc-sierra.com

None of the information contained in this document constitutes an express or implied warranty by PMC-Sierra, Inc. as to the sufficiency, fitness or suitability for a particular purpose of any such information or the fitness, or suitability for a particular purpose, merchantability, performance, compatibility with other parts or systems, of any of the products of PMC-Sierra, Inc., or any portion thereof, referred to in this document. PMC-Sierra, Inc. expressly disclaims all representations and warranties of any kind regarding the contents or use of the information, including, but not limited to, express and implied warranties of accuracy, completeness, merchantability, fitness for a particular use, or non-infringement.

In no event will PMC-Sierra, Inc. be liable for any direct, indirect, special, incidental or consequential damages, including, but not limited to, lost profits, lost business or lost data resulting from any use of or reliance upon the information, whether or not PMC-Sierra, Inc. has been advised of the possibility of such damage.

© 2001 PMC-Sierra, Inc.

PMC2002232 (R1) Issue date: February 2001