Reconstructing HTTP/1.1 Traces

Balachander Krishnamurthy and Jennifer Rexford
AT&T Labs–Research
{bala,jrex}@research.att.com

Yun Fu
Duke University
fu@cs.duke.edu

Motivation

Why HTTP/1.1?

- The Web is the dominant Internet application
- HTTP/1.1 is the latest version of the protocol
- The performance of HTTP/1.1 features is not well understood

Why packet monitoring?

- Full HTTP request and response headers (plus body!)
- Detailed timing information at the TCP/IP level

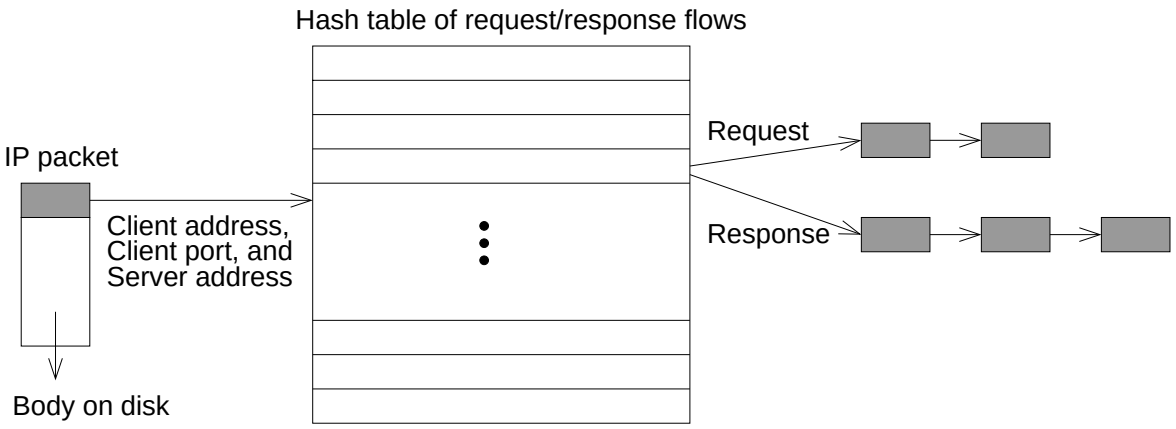
Steps in Reconstruction

- Tapping one or more links carrying IP packets
- Capturing the packets associated with HTTP transfers
- Demultiplexing the packets into TCP connections
- Reconstructing the ordered stream of bytes
- **Extracting HTTP messages from the byte stream**
- Generating a detailed log of the HTTP messages

Modified tcpdump

- Monitor multiple interfaces at a time (e.g., two FDDI cards)
- Capture full contents of port-80 TCP packets (HTTP traffic)
- Populate a C structure with key IP and TCP fields
- Write the C structure and the TCP segment to separate files
- Generate a new pair of files every n packets

Demultiplexing Packets



Reconstructing the Ordered Byte Stream

Splay tree for each flow

- Use TCP sequence numbers to keep ordered list of packets in flow
- Store only IP/TCP information (and pointer to TCP segment on disk)
- Remove portions of packets that overlap (e.g., duplicates)

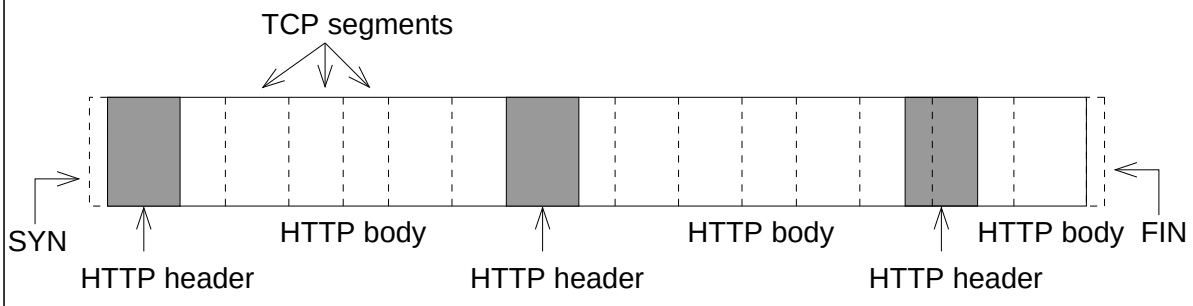
Deciding when to process the reconstructed flow

- Complete when SYN/FIN sequence numbers agree with number of bytes
- Or, timeout of idle TCP flow after some period of inactivity

Dealing With Missing Packets

- The monitor may miss occasional packets during transient overload
- This can result in undersampling of long-lived transfers
- Fortunately, loss *inside* a request/response body is not a disaster
- ... as long as you can successfully get to the next header/body
- Cannot "read" across a missing packet, but can do a "seek"
- Loss of SYN packet or part of HTTP header requires skipping rest of flow

Multiple HTTP Messages in a Flow



- A single message may span multiple packets
- A single packet may have portions of multiple messages
- HTTP/1.1 defines numerous ways to delimit messages

Extracting HTTP Messages – High-Level View

- Search for double CRLF to find the end of HTTP response header
- Log the HTTP response header (or desired fields)
- **Determine the end of the response body (tricky!)**
- Log information about body (e.g., checksum, timestamps, HREFs, etc.)
- Match with corresponding HTTP request header in request flow
- Continue with the next response in the TCP connection

Finding the End of the Response Body

1. Not HTTP/1.1 and no `Connection: Keep-Alive` header — body proceeds to end of flow
2. Response code 1xx, 204, or 304 — no body
3. `HEAD` request — no body
4. `Connection: close` header — body proceeds to end of flow
5. `Transfer-Encoding: chunked` header — body divided into chunks
6. `Content-Length` header — header value represents length
7. `Content-Type: multipart` header — self-delimiting media type

Chunked Transfer Coding in HTTP/1.1

```
HTTP/1.1 200 OK
Server: Apache/1.2.7-dev
Date: Tue, 07 Jul 1998 18:21:41 GMT
Transfer-Encoding: chunked
Content-Type: text/html
691
        <...1681 bytes of chunk data...>
76
        <...118 bytes of chunk data...>
0
```


Multipart/Byteranges in HTTP/1.1

```
HTTP/1.1 206 Partial Content
Date: Thu, 10 Feb 2000 20:25:23 GMT
Server: Apache/1.2.6 Red Hat
ETag: cc678-12d12-66394036
Content-type: multipart/byteranges; boundary=----ROPE----

----ROPE----
Content-Type: text/html
Content-Range: bytes 0-100/15044
...the first 101 bytes of the resource...
----ROPE----
Content-Type: text/html
Content-Range: bytes 9600-15043/15044
...all bytes after byte 9600 of the resource...
----ROPE------
```

Non-Compliant Web Clients and Servers

- Incorrect `Content-Length` value
- Presence of body in a 1xx, 204, and 304 response
- Presence of body in response to a `HEAD` request
- Additional message(s) after a `Connection: close`

Errors detected by checking for valid beginning of next HTTP header

See "PRO-COW: Protocol Compliance on the Web—A Longitudinal Study" Balachander Krishnamurthy and Martin Arlitt, USITS, March 2001, `http://www.research.att.com/~bala/papers/usits01.ps.gz`

Log Files – TCP/IP Fields

- Client and server IP addresses
- Number of packets and bytes transmitted
- Timestamp for beginning of TCP connection (SYN)
- Timestamp for ending of TCP connection (FIN/RST)
- Termination method for flow (FIN vs. RST)
- Errors (missing packet within flow)

Log Files – HTTP Level

- HTTP request header
- HTTP response header
- Timestamp for beginning of HTTP header
- Timestamp for beginning of HTTP body
- Timestamp for ending of HTTP body
- Errors (invalid HTTP message)
- (Optionally) entity body, checksum, or embedded HREFs/IMGREFs

Status

Software

- Modified tcpdump source
- 6000 lines of C code for reconstruction software
- Extensive use of libraries (sfio, hash, splay tree)

Collecting traces

- Software run on 10 hours of packet traces from AT&T IP Backbone
- Successfully reconstructed over three GB of packet traces

Many more details :)

Web Protocols and Practice
HTTP/1.1, Network Protocols, Caching, and Traffic Measurement

Balachander Krishnamurthy
&
Jennifer Rexford

Hardback, 650 pp, Addison-Wesley, Spring 2001,