Load Balancing and Sessions

Goals

• Scalability
  • multiple servers
• Availability
  • server fails → do not route to it
• Manageability
  • take servers on/off line
  • decouple application from server
  • partition content across servers
• Security
  • NAT & firewall functions
• QoS
  • e.g. identify high priority clients
DNS Load Balancing

Ordinary DNS Lookup

User enters www.loadbalancedsite.com in the address box of the web browser.

Give me the IP address for www.loadbalancedsite.com

www.loadbalancedsite.com

IP address is 203.24.23.3

GET 203.24.23.3

Internet

Internet
DNS Load Balancing

Round-Robin DNS Lookup

Cluster

203.34.23.3

www.loadbalancedsite.com

Round Robin DNS

203.34.23.4

203.34.23.5
DNS Load Balancing - Drawbacks

- DNS has no access to server state
- balances hits, not server load
- no fault tolerance
- DNS client behavior (caches)?
Dedicated Load Balancer - Basic Structure

- Server farm has a Virtual IP address (VIP)
- VIP is routed to load balancer
- Load balancer forwards to Real Servers
Which Layer to Switch?

- Layer 7  Application
- Layer 6  Presentation HTTP etc
- Layer 5  Session
- Layer 4  Transport TCP/UDP
- Layer 3  Network IP
- Layer 2  Data Link MAC
- Layer 1  Physical

Switching can be done in layer 2, 2/3, 4/7
VIP to Real Server Mappings

VIP=123.122.121.1

Load Balancer

Servers

RS1: 10.10.10.1
Web+FTP

RS2: 10.10.10.2
Web+SMTP

RS3: 10.10.10.2
SMTP

VIP:80 (HTTP) → RS1:80 + RS2:80

VIP:21 (FTP) → RS1:21

VIP:82 (SMTP) → RS2:82 + RS3:82

NAT here is subtle here
NAT Issues

A: ClientIP:ClientPort → VIP:80
B: XXX → 10.10.10.2:80
C: 10.10.10.2:80 → 10.10.10.1
D: VIP:80 → ClientIP:ClientPort

• I don’t care what XXX is, but ...
• Client requires VIP:80 as source of packet D
• NAT (Load Balancer) must translate?
Direct Server Return

10.10.10.1

A: ClientIP:ClientPort → VIP:80
B: ClientIP:ClientPort[M1] → VIP:80[M2]
C: ClientIP:ClientPort[M1] → VIP:80[M3]
D: VIP:80 → ClientIP:ClientPort

• Configure VIP as loopback address on server
• Balancer translates MAC address but not IP
• Reply need not go through Balancer!
  • reply is usually bigger than request

10.10.10.2[M3]
VIP is loopback

(switched) local network
Port-Address Translation

- Map to multiple ports on same real server
  - VIP:p $\rightarrow$ RS1:p1 + RS1:p2 + ... + RS1:p17
  - Run 17 instances of service on RS1
  - Some services work better that way!
Session Persistence Techniques

- They interact with load balancing ...
TCP Protocol

client

SYN

SYN ACK

ACK

Data Exchange

FIN

FIN ACK

server
HTTP Transaction

Newest version of HTTP allows (but does not require) clients to keep TCP connection open for multiple request/response.

So session != TCP connection

No guarantee all TCP connections in a session will map to the same server.
Balancer has two kinds of information:

- server conditions - load, crashed, ...
- client data - information in the SYN packet
SYN Based Session Identification

- All connections with “similar” information in the SYN are part of same session, subject to “session persistence timeout”

- Source IP
  - all activity from a given host
- Source IP + VIP + Port
  - all activity from given host to given service
- Source IP + VIP
  - allows switch HTTP to SSL
SYN Based Session Identification - II

- **Port Grouping**
  - first connection to given VIP and port group choses based on server load
  - subsequent connections to same VIP and port group respect existing session
- **Concurrent connections**
  - e.g. passive FTP
• Proxy servers introduce two problems:
  • load balancing
  • session persistence
• All client connections appear to come from the same host - the proxy server itself!
Session Maintenance with Proxies

- Client connections in same session may come from different proxy servers
How Serious is Proxy Problem?

- Requires
  - many clients
  - behind small number of proxies
  - all accessing my site
• Suppose load balancer wants to use information from the data?
  • e.g. the URL in the HTTP header
  • Balancer must “peek” into the TCP data
TPC Sequence Numbers

- SYN [seq 100]
- SYN ACK [seq 500, ack 101]
- ACK [seq 101, ack 501]
- data [seq 102, ack 501]
- data [seq 501, ack 103]
- data [seq 103, ack 502]
- data [seq 502, ack 104]

- sequence number in one direction is initialized by the server
- so how can the Balancer look at the data before choosing the server?
TPC Sequence Numbers - II

- Balancer must adjust seq numbers in every TCP packet!

→ SYN [seq 100]
← SYN ACK [seq 500, ack 101]
→ ACK [seq 101, ack 501]
→ data [seq 102, ack 501]

→ SYN [seq 100]
← SYN ACK [seq 600, ack 101]
→ ACK [seq 101, ack 601]
→ data [seq 102, ack 601]
← data [seq 601, ack 103]

← data [seq 501, ack 103]
→ data [seq 103, ack 502]
← data [seq 502, ack 104]
→ data [seq 103, ack 602]
← data [seq 602, ack 104]
Cookies

- We all know what cookies are ...

HTTP GET request

HTTP GET reply contains cookie (server=2)

next HTTP GET request includes cookie (server=2)
Cookie-Read Technique

- Each server creates “server=myID” cookie at start of session
-Balancer uses delayed binding to read the cookies and respect existing sessions

Note: this requires changes to the server apps!
Cookie-Insert Technique

• Balancer inserts a “server=17” cookie in the reply stream when server 17 replies to a request that creates a new session
• Balancer uses delayed binding on reverse TCP connection
• Server applications completely unchanged!

• Disadvantage: Balancer must rewrite the TCP data stream, very expensive
Cookie-Rewrite Technique

- Each server creates “server=XXX” cookie at start of session
- Balancer uses delayed binding to XXX by actual server ID in reply
- Balancer uses delayed binding to read cookies in requests
- Minimal change to server application
- Balancer does not need to change length of the reply stream
Encryption?

Must decrypt to interpret session data
This requires a delayed binding implementation
• SSL V 3.0 and above supports session IDs in the protocol

SSL initiation request with optional SSL session ID

Server response includes SSL session ID

Encrypted HTTP request

Encrypted HTTP reply

But: there is no precise specification of how clients use the SSL session ID, when it expires, ...
Sessions and Failures

• If an application server fails, how much session state will be lost?
• Obviously, we want to minimize this
• But at reasonable expense
Approach 0 - State in Cookies

- Pass cookies back and forth between client and server encoding all session state.
- Cookies can hold only a limited amount of information.
- But this makes the sessions stateless at the application servers.
Approach 1 - Brute Force

- Keep session state in the database...
- Commit frequently.
- Not widely used because of expense of storing session state to disk
Approach 2 - Group Communication

- Multicast session state to all (or a group of) application servers
  - e.g. use Spread toolkit
- Can switch between application servers without loss of session state
- Application servers keep session state in volatile memory
  - loss of all app servers results in loss of session data - unlikely
Approach 3 - Compromise

• Commit session state to the data base at the important points
  • after significant changes
  • HTTP to HTTPS change
  • hope “important” is “infrequent”
URL Switching

• An obvious technique we should mention ...

• Balancer can use delayed binding to
  • examine the URL in an HTTP request header
  • choose the server accordingly

• commercial products support pattern-matching languages to specify this
URL Switching Caveat

- HTTP 1.1 protocol supports more than one request on same TCP connection ...
- What if two successive HTTP requests on the same TCP connection should map to different servers?
- this gets real hard
Single Points of Failure

Router

Load Balancer

Load Balancer

L2 switch

here

here
Single Points of Failure

- Router
- Load Balancer
- L2 switch
- Router
- Load Balancer
- L2 switch

Network topology diagram showing a redundant system with routers, load balancers, and L2 switches to illustrate single points of failure.