SMTP, Porcupine

Amin Vahdat
CSE 123b
May 4, 2006
Annoucements

- Midterm: May 9
- Second assignment due May 15
Load Balancing Switches
F5 Networks 3DNS

1. User queries local DNS to resolve domain
   - Local DNS queries 3-DNS acting as primary DNS - resolving both content and country origin
2. 3-DNS triggers a metric query for BIG-IP Controller at each location or individual servers
   - Query responses are collected and the "best" server determination is identified
   - Frequency of metric queries are user definable
3. 3-DNS responds with the "best" IP address
4. Traffic flows between user and remote site
Network Service Architecture

- WAN Router
- Firewall
- Load Balancing Switch
- Ethernet LAN Switches

Integrate?
Network Service Architecture

- Load Balancing Switch
- Ethernet LAN Switches
- Application Servers
- HTTP (front end)
- Storage Tier (Fileserver/Database)
Network Service Architecture

Load Balancing Switch

HTTP (front end)

Virtual Cluster 1 (site1.com)

Virtual Cluster 2 (site2.com)

Load balancing switch maps external virtual IP addresses to physical IP addresses in internal network
F5 “Big-IP”
Big IP: Cookie Encryption

- rule when HTTP_REQUEST {
    #
    # Decrypt the cookie on its way to the server.
    #
    HTTP::cookie decrypt "cookie_name" "password-key"
}

rule when HTTP_RESPONSE {
    #
    # Encrypt the cookie on its way to the client.
    #
    HTTP::cookie encrypt "cookie_name" "password-key"
}
Big IP: Sanitize HTTP Headers From All Apps

- rule when HTTP_RESPONSE {
  
  #
  # Remove all but the given headers.
  #
  HTTP::header sanitize "ETag" "Content-Type"
  "Connection"

  }

Big IP: Remove SSN from Responses

rule ssn_rule {
    when HTTP_RESPONSE_DATA {
        set payload [HTTP::payload [HTTP::payload length]]
        set ssnx "xxx-xxx-xxxx"

        # Find the SSN numbers
        regsub -all {\d{3}-\d{2}-\d{4}} $payload $ssnx new_response

        # Replace the content if there was a match
        ...
    }
}
Redline Networks

- Performance
- Security
- Availability
- Management
  - Statistics, event-logging, etc.
- Flexibility
  - In-line deployment, or single connection w/redirect
  - Multi-function
Compression Policy Engine
- Automatically compress content to reduce bandwidth consumption
- Optional compression of MIME types: doc, xls, ppt, flash, etc.
- Web services compression for server-to-server protocols including SOAP

TCP Connection Management
- Terminates and persistently maintains client and server connections
- Servers typically see 1 connection for every 500 – 1,000 clients

Transaction Brokering
- All requests upgraded to 1.1, full support for chunking and pipelining

Never-Close Client Connections
- Eliminates unnecessary TCP slow-start for faster delivery
- Reduces client connection set-up time for lower latency
Redline Security

- **Protocol Security**
  - Protocol scrubbing never passes packet fragments, ensures only valid, well-formed requests reach servers
  - End-to-end and one-way SSL with accelerated download of secure content

- **Network Security**
  - Defend against SYNFlood and DoS attacks of 100Mbps+
  - Support SYN cache and SYN cookie

- **Launch-Point Security**
  - No UNIX shell – no "root" access
  - Cannot install or run shell scripts, arbitrary programs, Trojan horses, etc.

- **Deep Packet Inspection**
  - Block, log or re-write bad URLs, malicious requests
  - Buffer overflow inspection + protection
Redline Availability

- **Active-N High Availability**
  - Self-healing mesh of up to 64 boxes actively processing traffic to one or more VIPS with cascading failover
  - Linear scaling – only deploy one box rather than a pair for additional capacity
  - Also provide Active-Active and Active-Standby high-availability

- **Server Load Balancing**
  - Full layer 4-7 server load balancing including HTTP/S, TCP, UDP, and FTP protocols
  - Fewest Outstanding Requests, round-robin, least connections, fail-over chaining, etc.
  - Cookie sticky with state shared across up to 64 Redline devices for HTTP or HTTPS (SSL) traffic
  - IP sticky for Intranet or Internet IP addresses
Email

Unattributed Internet Source
**Terminology**

- **User Agent**: end-user mail program
- **Message Transfer Agent**: responsible for communicating with remote hosts and transmitting/receiving email
  - Both client and server
- **Mail Exchanger**: host that takes care of email for a domain
SMTP

Used to exchange mail messages between mail servers (Message Transfer Agents)
SMTP Protocol

- SMTP sender is the client
- SMTP receiver is the server
- Alternating dialogue:
  - Client sends command and server responds with command status message
  - Order of the commands is important
  - Status messages include ASCII encoded numeric status code (like HTTP, FTP) and text string
SMTP Commands

- **HELO** - identifies sender
- **MAIL FROM:** - starts a mail transaction and identifies the mail originator
- **RCPT TO:** - identifies individual recipient. There may be multiple **RCPT TO:** commands
- **DATA** - sender ready to transmit a series of lines of text, each ends with \r\n. A line containing only a period ‘.’ indicates the end of the data
Data Format

- ASCII only- must convert binary to an ASCII representation to send via email
- What if we want to send a line containing only a period?
  - Sender prepends a period to any line staring with a period (in the message)
  - Receiver strips the leading period in any line that starts with a period and has more stuff
Typical Exchange

[vahdat@ramp vahdat] $ telnet fast.ucsd.edu 25
Trying 132.239.15.4...
Connected to fast.ucsd.edu.
Escape character is '^]'.
220 fast.ucsd.edu ESMTP
HELO ramp.ucsd.edu
250 fast.ucsd.edu Hello ramp.ucsd.edu [137.110.222.239], pleased
to meet you
MAIL FROM: <vahdat@cs.ucsd.edu>
250 2.1.0 <vahdat@cs.ucsd.edu>... Sender ok
RCPT TO: vahdat
250 2.1.5 vahdat... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
This is a test message.
More text
.
250 2.0.0 j3QKGnsR029250 Message accepted for delivery
Leading Period

DATA
354 Enter mail, end with "." on a line by itself
Hi dave - this message is a test of SMTP
.
..foo
.
.
250 VAA0771 Message accepted for delivery

Resulting Message:

Hi dave - this message is a test of SMTP
.
..foo
.
.
Other SMTP Commands

**VRFY** - confirm that a name is a valid recipient

**EXPN** - expand an alias (group email address)

**TURN** - switch roles (sender <=> receiver)
more Commands

**SOML** - Send Or Mail
  
  if recipient is logged in, display message on terminal, otherwise email.

**SAML** - Send and Mail

**NOOP** - send back a positive reply code

**RSET** - abort current transaction
Mail Headers

- Email messages contain many headers
  - Some headers are created by UA
  - Some automatically added by MTA
- Every MTA adds (at least) a "Received:" header
  - Spamassassin?
- Some of the headers are read by (parsed) intermediate MTAs, but the content is ignored and passed on transparently
POP – *Post Office Protocol*

- Used to transfer mail from a mail server to a UA.
**POP (version 3)**

- Similar to SMTP command/reply lockstep protocol
- Used to retrieve mail for a single user
  - requires authentication
- Commands and replies are ASCII lines
  - Replies start with **"+OK"** or **"-ERR"**.
  - Replies may contain multiple lines
POP-3 Commands

- **USER** - specify username
- **PASS** - specify password
- **STAT** - get mailbox status
  - number of messages in the mailbox
- **LIST** - get a list of messages and sizes
  - One per line, termination line contains ‘.’ only
- **RETR** - retrieve a message
More POP-3 Commands

- **DELE** - mark a message for deletion from the mailbox
- **NOOP** - send back positive reply
- **RSET** - reset. All deletion marks are unmarked
- **QUIT** - remove marked messages and close the (TCP) connection
Optional Commands

- **TOP** - send header lines from messages
- **APOP** - alternative authentication
  - Message digest based on opening greeting sent from POP server
  - Requires shared secret
  - No cleartext password on network
  - Does not authenticate the server
A Pop3 Exchange

[vahdat@ramp vahdat]$ telnet fast pop3
Trying 132.239.15.4...
Connected to fast.
Escape character is '^]'.
+OK QPOP (version 3.1.2) at fast.ucsd.edu starting.
<263.1114547001@fast.ucsd.edu>
user vahdat
+OK Password required for vahdat.
pass mypassw
+OK vahdat has 1 visible message (0 hidden) in 1761 octets.
Stat
+OK 1 1761
list
+OK 1 visible messages (1761 octets)
1 1761
.
retr 1
+OK 1761 octets
Received: from mailbox7.ucsd.edu (mailbox7.ucsd.edu [132.239.1.59])
Message-Id: <200504262023.j3QKNATg015405@smtp.ucsd.edu>
Reply-To: <vahdat@cs.ucsd.edu>
From: "Amin Vahdat" <vahdat@cs.ucsd.edu>
To: <vahdat@cs.ucsd.edu>
Subject: test
Date: Tue, 26 Apr 2005 13:23:08 -0700
MIME-Version: 1.0
Content-Type: text/plain;charset="US-ASCII"
X-Greylisting: NO DELAY (Trusted relay host);
   processed by UCSD_GL-v1.2 on mailbox7.ucsd.edu;
X-MailScanner: PASSED (v1.2.8 34463 j3QKNA8g038256 mailbox7.ucsd.edu)
X-Spam-Flag: Spam NO
X-Scanned-By: milter-spamc/0.15.245 (fast.ucsd.edu [132.239.15.4]); pass=YES;
   Tue, 26 Apr 2005 13:23:12 -0700
X-Spam-Status: NO, hits=-4.90 required=5.00

test
MIME

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789
--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain
Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data ..... 
........................
......base64 encoded data
--98766789--

(Kevin Jeffay)
Functionally Homogeneous Clustering: A New Architecture for Scalable Data-intensive Internet Services

Yasushi Saito
Goals

- Use cheap, unreliable hardware components to build scalable data-intensive Internet services.
  - Data-intensive Internet services: email, BBS, calendar, etc.
- Three facets of scalability ...
  - *Performance*: linear increase with system size
  - *Manageability*: react to changes automatically
  - *Availability*: survive failures gracefully
Contributions

- **Functional homogeneous clustering:**
  - Dynamic data and function distribution
  - Exploitation of application semantics

- Three techniques:
  - Naming and automatic recovery
  - High-throughput optimistic replication
  - Load balancing

- Email as target application

- Evaluation of the architecture using *Porcupine*
Data-intensive Internet services

- Examples: Email, Usenet, BBS, calendar, Internet collaboration (photobook, equill.com, crit.org)
- Growing rapidly as demand for personal services grows

- High update frequency
- Low access locality

⇒ Web techniques (caching, stateless data transformation) not effective

- Weak data consistency requirements
- Well-defined & structured data access path
- Embarrassingly parallel
Rationale for Email

- Email as target application.
  - Most important among data-intensive services
  - Service concentration (Hotmail, Gmail, AOL,, ...).

- Practical demands
  - The most update-intensive
  - No access locality
Conventional Solutions: Big Iron

- Just buy a big machine
- + Easy deployment
- + Easy management

- Limited scalability
- Single failure domain
- Really expensive
Conventional Solutions: Clustering

- Connect many small machines
- + Cheap
- + Incremental scalability
- + Natural failure boundary

- Software & management complexity
Existing Cluster Solutions

- **Static partitioning**: assign data & function to nodes statically
- Management problems:
  - Manual data partition
- Performance problems:
  - No dynamic load balancing
- Availability problems:
  - Limited fault tolerance
Functionally Homogeneous Clustering

- Static function and data partitioning leads to problems
- So, make everything *dynamic*:
  - Any node can handle any task (client interaction, user management, etc)
  - Any node can store any piece of data (email messages, user profile)
Advantages

Advantages:
• Better load balance, hot spot dispersion
• Support for heterogeneous clusters
• Automatic reconfiguration and task re-distribution upon node failure/recovery
• Easy node addition/retirement

Results:
• Better Performance
• Better Manageability
• Better Availability
Challenges

- Dynamic function distribution:
  - Solution: run every function on every node.

- Dynamic data distribution:
  - How is data named and located?
  - How is data placed?
  - How does data survive failures?
Key Techniques and Relationships

Functional Homogeneity

- Load Balancing
- Name DB w/ Reconfiguration
- Replication

Framework

Techniques

Goals

Performance

Manageability

Availability
Overview: Porcupine
Receiving Email in Porcupine

1. “send mail to bob”
2. Who manages bob? ⇒ A
3. “Verify bob”
4. “OK, bob has msgs on C, D, & E
5. Pick the best nodes to store new msg ⇒ {C,D}
6. “Store msg”
7. “Store msg”
Basic Data Structures

User map

Mail map / user profile

Mailbox storage

- hash("bob") = 2

- Bob’s MSGs
- Suzy’s MSGs
- Ann’s MSGs
- Joe’s MSGs

A

B

C
Scaling Performance

- User map distributes user management responsibility evenly to nodes
- Load balancing distributes data storage responsibility evenly to nodes
- Workload is very parallel

⇒ Scalable performance
Measurement Environment

- Porcupine email server
  - Linux-2.2.7+glibc-2.1.1+ext2
  - 50,000 lines of C++ code
- 30 node cluster of not-quite-all-identical PCs
  - 100Mb/s Ethernet + 1Gb/s hubs
  - Performance disk-bound
  - Homogeneous configuration
- Synthetic load
  - Modeled after UWCSE server
  - Mixture of SMTP and POP sessions
POP performance, no email replication

- **Porcupine**
  - 25m/day
  - 68m/day

- **sendmail+popd**
  - 25m/day
  - 68m/day
How Do Computers Fail?

- Large clusters are unreliable
- **Assumption:** live nodes respond correctly in bounded time most of the time
  - Network can partition
  - Nodes can become very slow temporarily
  - Nodes can fail (and may never recover)
  - Byzantine failures excluded
Recovery: Goals and Strategies

- **Goals:**
  - Maintain function after unusual failures
  - React to changes quickly
  - Graceful performance degradation / improvement

- **Strategy:** Two complementary mechanisms
  - Make data “soft” as much as possible
  - **Hard state:** email messages, user profile
    ⇒ Optimistic fine-grain replication.
  - **Soft state:** user map, mail map
    ⇒ Reconstruction after configuration change
Soft-state Recovery Overview

1. **Membership protocol**
   **Usermap recomputation**

2. **Distributed disk scan**

Timeline
How does Porcupine React to Configuration Changes?

![Graph showing messages per second over time for different node failure scenarios.]

- **No failure**
- **One node failure**
- **Three node failures**
- **Six node failures**

See breakdown