Operating Systems: Distributed Applications

Fabrice Le Fessant and Albert Cohen

INF 552
Operating Systems
Ecole Polytechnique, Palaiseau

13 mars 2007

Plan

Operating Systems

- Lecture on Distributed Systems
    - Just for fun !
- Computer Class Room on Sockets/Threads
    - To continue the *forum* project

Outline

OSI Model

Open Systems Interconnection Basic Reference Model

- Layer 7 : Application layer (FTP, NFS)
- Layer 6 : Presentation layer (XDR)
- Layer 5 : Session layer (TCP)
- Layer 4 : Transport layer (TCP, UDP)
- Layer 3 : Network layer (IP)
- Layer 2 : Data Link layer (Ethernet, ADSL)
- Layer 1 : Physical layer (Cable, Fiber)

In this lecture, we consider only the Application Layer

Byzantine Generals

Consensus in Distributed Systems

- One huge army in the valley
- Two armies on the mountains on both sides
- They can win IF AND ONLY IF they attack together at the same moment
- Their messengers must cross the valley, maybe got killed or delayed
- Can they win ?

- Consensus is not achievable in distributed systems with failures.
- Most protocols are either centralised, or have to refer to a central authority in case of problem.

Finding Information on Distributed Protocols

RFC : Request for Comments

- RFC-Editor : `http://www.rfc-editor.org/rfc.html`
- IETF : Internet Engineering Task Force
- IANA : Internet Assigned Numbers Authority

On your Unix/Linux computer

- `/etc/services` : ports for services

Network Protocol : TCP/IP

- Computers have numeric addresses (IPv4/IPv6) : "129.104.247.1"
- Computers have symbolic names : "kelen.polytechnique.fr" (host name and domain name)
- Some protocols are responsible for routing (BGP) and name translation (DNS)

Application Protocols

- Client-Server Protocols : SMTP, FTP, HTTP
- Distributed File-Systems : NFS, SMBFS, AFS
- Peer-to-Peer Protocols : Kad, Bittorrent, Skype

Plan

DNS : Domain Name Server

Translation of symbolic names to IP addresses

- UDP or TCP on port 53
- Finding the IP address of `kelen.polytechnique.fr` :
  - Query `A IN kelen.polytechnique.fr` to your name server
  - Reply `A IN kelen.polytechnique.fr DA 129.104.247.1`
  - Reply has an authority : `NS IN polytechnique.fr DNAME dns1.polytechnique.fr`
  - Reply may contain more info : `A IN dns1.polytechnique.fr DA 129.104.1.1`
- Known names are stored in `/etc/hosts`
- Local name servers are stored in `/etc/resolv.conf`.

SMTP : Simple Mail Transfer Protocol

Typical Mail Path

- Mailer $\xrightarrow{SMTP}$ SMTP Relay
  The mail is delivered to your Internet Provider SMTP Relay.
- SMTP Relay $\xrightarrow{SMTP}$ SMTP Host
  The mail is delivered to each recipient SMTP Host
- SMTP Host $\xrightarrow{POP/IMAP}$ Mail Reader
  Each recipient downloads the mail locally from the SMTP Host

RFC 821, Aug. 1982

- Many RFCs : 821 (SMTP), 822 (Mail), 1870, 2920, ... (extensions), 2821 (new proposal)
- Port : 25, Protocol : TCP

Components of a Mail

A mail contains

- A sender address
- Some recipients (fully qualified addresses)
  - Not presented in the mail (Bcc)
  - Translated automatically (mailing-lists)
- A content in 7bit format. Other formats are encoded :
  - base64 [a-zA-Z0-9+/]
  - quoted-printable [=00]

What does a mail look like ?

```
Received: by 10.48.157.14 with HTTP; Mon, 9 Jan 2006 02:38:34
Message-ID: <1f4f725e0601090238t2f0f2c11y@mail.gmail.com>
Date: Mon, 9 Jan 2006 11:38:34 +0100
From: Fabrice Le Fessant <fabrissimo@gmail.com>
Reply-To: fabrice@lefessant.net
To: florence@lefessant.net
Subject: nouvelles ?
In-Reply-To: <43C2226C.7060603@sncf.com>
MIME-Version: 1.0
References: <43C2226C.7060603@sncf.com>

Salut,

  Comment tu vas ?

-- Fab
```

Translating email addresses

An example : `fabrice.le_fessant@inria.fr`

- DNS Query : `MX IN inria.fr`
- DNS Reply : list of servers with weight :
  `concorde.inria.fr, 10 ; backup.inria.fr, 5`
- Try to propagate the mail in turn to each server in the list, by decreasing weight.

SMTP Protocol

A Text-Based Protocol

- Human readable
- CR-LF (\r\n) as line terminator

SMTP Commands

- HELO or EHLO : for connection
- MAIL FROM : identifying the sender
- RCPT TO : identifying the recipients
- DATA : the content of the mail (finished by a single . line)
- VRFY : testing an email address
- RSET : resetting an email
- QUIT : for ending the connection

An example of communication with my SMTP relay

```
peeromane:~/C8%  telnet smtp.free.fr 25
220 smtp4-g19.free.fr ESMTP Postfix
EHLO mycomputer.free.fr
250-Hello mycomputer.free.fr from smtp4-g19.free.fr
250-PIPELINING
250-SIZE 100000000
250-VRFY
250-ETRN
250 8BITMIME
MAIL FROM: fabrice@lefessant.net
250 Ok
RCPT TO: fab@lefessant.net
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Coucou, Comment ca va ?
 - Fabrice
.
250 Ok: queued as 72B8567482
```

FTP : File Transfer Protocol

RFC 959 > 765, Oct. 1985

- A protocol to download files from and upload to a server
- The server waits on port 21
- The protocol distinguishes between connections :
  - Control connections for commands and reply statuses
  - Data connections for file content
- A complex protocol for firewalls
  - The firewall must understand the commands, since it may have to forward data connections !

Example session

```
220 Service ready
> USER JohnDoe
331 User name ok, need password
> PASS mumble
230 User logged in
> PORT 192,168,150,80,14,178
200 PORT command successful.
> LIST
150 Opening ASCII mode data connection for file list.
226 Transfer complete.
> CWD /tmp
250 CWD command successful.
> TYPE A
200 Command OK
> RETR test.txt
150 File status ok; about to open data port 3762
226 Closing data connection, transfer successful
> QUIT
221 Goodbye.
```

HTTP : Hyper-Text Transfer Protocol

RFC 2616 (HTTP/1.1), Jun. 1999

- A protocol to download files from a Web-server, on a single connection.
- The server waits on port 80
- Communcations are made of requests, each request containing :
  - A query/reply line
  - A set of headers
  - An empty line for separation
  - A data block
- The data block length is specified either by one of the headers, or by the end of the connection (before HTTP 1.0).
- Different requests : GET, POST, HEAD, ...

Example request

URL `http://www.peerple.net/index.html`

```
GET /index.html HTTP/1.1
Host: www.peerple.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; Ub
Accept: text/xml,application/xml;q=0.9,text/plain;q=0.
Accept-Language: en-US,en;q=0.8,fr;q=0.6,fr-FR;q=0.4,e
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: https://localhost:7481/frames
Cookie: login_RYN4XCHW7HVLZMHCGCKBVLRY6NHMRVDK=5EC16E4
```

Example reply

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 138
Connection: close
Content-Type: text/html; charset=UTF-8

<html><head>
<title> Page du serveur </title>
</head>
<body> Rien a lire </body>
</html>
```

Secure Sockets Layer : SSL

Need for Secured Internet Connections :

- E-Commerce : sending your Credit Card Number
- Privacy : protecting personal data transmitted

Secured Connections offer :

- Authentication : need to identify the other side and trust it
- Encryption : data should not be compromised to a third-party listening to the connection packets (man-in-the-middle attack)

SSL/TLS is easily used everywhere (layer 5)

- HTTPS = HTTP + SSL
- Used also for POP3, IMAP, SMTP, etc...

SSL : Applied Cryptography

How to use Cryptography to secure connections ?

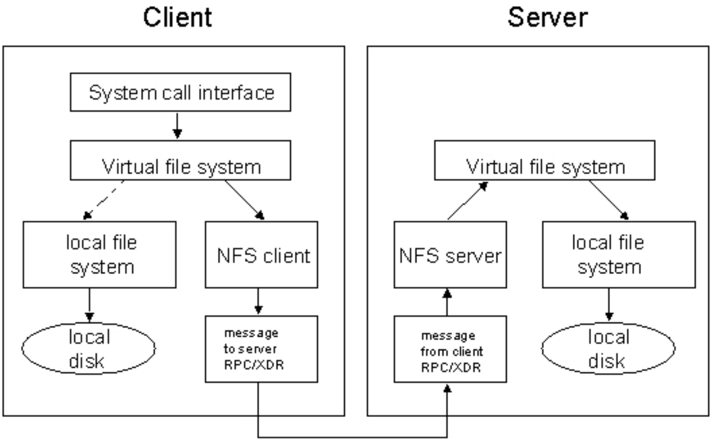
- Each host has a pair of cryptographic keys
  - One key is secrete (nobody else must know it)
  - One key is public (everybody should know it)
- A certificate signed by a trusted party (Verisign...) guarantees the public key owner.
- Asymmetric encryption (slow) is used to exchange a temporary session key
- Symmetric encryption (fast) with the session key is used to exchange data

Plan

Distributed File-System Requirements

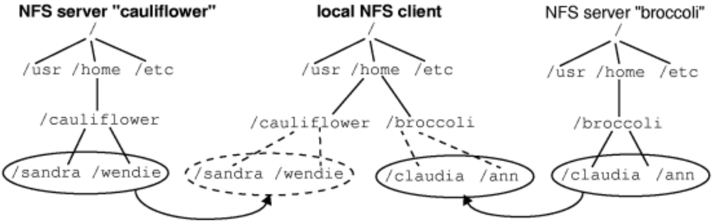
- Transparency
  - Access Transparency
  - Location Transparency
  - Mobility Transparency
  - Performance Transparency
  - Scaling Transparency
- Concurrency control
- Replication
- Fault-Tolerance
- Heterogeneity
- Security
- Consistency

NFS Architecture



Mounting NFS Partitions

- `/etc/exports` specifies which sub-trees are shared (server)
- `/etc/fstab` specifies which partitions are mounted (client)
- Partitions can be mounted, depending on server failures :
  - *hard*, clients are suspended while waiting
  - *soft*, clients are released after a few retries



NFS Server

- Either inside the Kernel, or a daemon process
- Stateless server :
  - Access control on each request
  - No/bad file locking
- Server caching :
  - *Read-ahead*, reads blocks following recently read blocks
  - *Write-Through* or *Write-Back* caching

- What are the problems of each approach ?

NFS Client

- Semantics different from UNIX FS Semantics
- File handles used by Clients :
  - File-System ID on Server
  - i-Node number of File
  - i-Node generation of Server
- SUN RPC translated to XDR, UDP and TCP

Plan

Peer-to-Peer File-Sharing

Definition of a Peer-to-Peer Network

- All computers are *peers*, i.e. both client and server
- Thousands to millions of computers are inter-connected, all over the world
- Computers join and leave the network all the time

File-Sharing Requirements

- *File Discovery :* find the file you are interested in
- *File Localization :* find some peers sharing the file you are interested in
- *Efficient File Download :* download the file as fast as you can
- *Sharing Incentives :* force other peers to share as much as they can

Peer-to-Peer File-Sharing

Definition of a Peer-to-Peer Network

- All computers are *peers*, i.e. both client and server
- Thousands to millions of computers are inter-connected, all over the world
- Computers join and leave the network all the time

File-Sharing Requirements

- *File Discovery :* find the file you are interested in
- *File Localization :* find some peers sharing the file you are interested in
- *Efficient File Download :* download the file as fast as you can
- *Sharing Incentives :* force other peers to share as much as they can

Peer-to-Peer File-Sharing

Examples

- File-Sharing : Kad, Bittorrent, Edonkey (Emule), Fasttrack (Kazaa), ...
- Telephony : Skype, ...

In this lecture :

- How to localize files ?
  - Unstructured networks
  - Structured networks
- How to download files ?

Peer-to-Peer File-Sharing

Examples

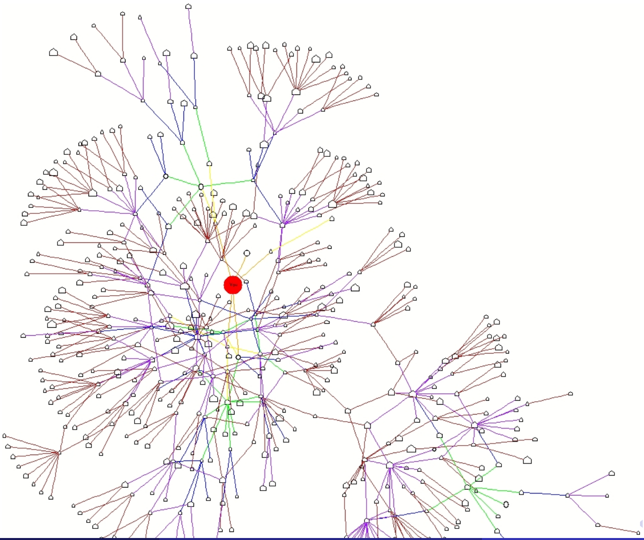
- File-Sharing : Kad, Bittorrent, Edonkey (Emule), Fasttrack (Kazaa), ...
- Telephony : Skype, ...

In this lecture :

- How to localize files ?
  - Unstructured networks
  - Structured networks
- How to download files ?

How to localize a ressource in Unstructured Networks

Diffuse the request to the whole network (Gnutella) ?



Bloom Filters

A Bloom Filter is an array of bits which summarizes which files are stored on a peer

- Compute different hash functions for each file :
  $file \rightarrow int \times int \times int \times \ldots$
- Set one bit per hash function in the filter
- A file can be present *only if* all the bits are set

How to use Bloom Filters to reduce diffusion ?

- Each peer sends its Filter to all its neighbours.
- Filters can be combined (ORed) to summarize which files are stored on a set of peers.
- Filters are easy to compute, most efficient when mostly null, and easy to compress when mostly null !
- Forward a query only if the Filter tells you to do so

Bloom Filters

A Bloom Filter is an array of bits which summarizes which files are stored on a peer

- Compute different hash functions for each file :
  $file \rightarrow int \times int \times int \times \ldots$
- Set one bit per hash function in the filter
- A file can be present *only if* all the bits are set

How to use Bloom Filters to reduce diffusion ?

- Each peer sends its Filter to all its neighbours.
- Filters can be combined (ORed) to summarize which files are stored on a set of peers.
- Filters are easy to compute, most efficient when mostly null, and easy to compress when mostly null !
- Forward a query only if the Filter tells you to do so

How to localize a ressource in Structured Networks

Structured Networks

- Distributed Hash Tables (Chord, Kademlia, Broose,...)
- Skip Lists
- Voronets

Often $O(logN)$ complexity

- For exact queries
- For range queries
- For multiple-criteria queries

Hash Tables

Finding the value associated with an entry

- Compute a key from the entry using a *hash function* (usually, an integer modulo the size of the array)
- Find the node in the hash table associated with that key (usually, a list stored at the key index in an array)
- Find the entry in the list, and the corresponding associated value
- If $|$ array $| \geq |$ values $|$, lookup in $O(1)$

Application to distributed systems ?

Hash Functions

Cryptographic Hash Functions

- MD4, MD5, SHA1, ...
- Transform any string in a short string (16-20 chars)
- Collisions are possible (of course !) but unlikely
- Secure :
  - Hard to reverse a hash
  - Hard to generate a collision

Replace indexes by hashes

- Each peer has a hash (randomly generated)
- Each peer stores a list for a hash
- In fact, each peer stores a list for all hashes *closed* to its hash

Hash Functions

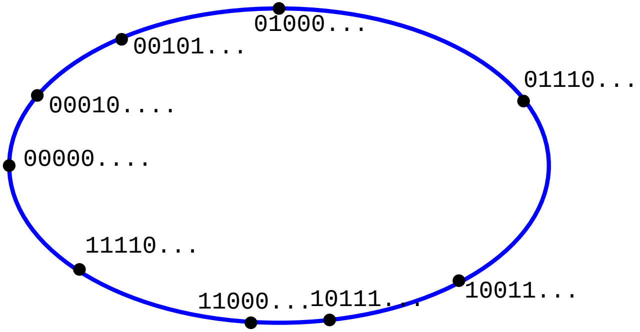
Cryptographic Hash Functions

- MD4, MD5, SHA1, ...
- Transform any string in a short string (16-20 chars)
- Collisions are possible (of course !) but unlikely
- Secure :
  - Hard to reverse a hash
  - Hard to generate a collision

Replace indexes by hashes

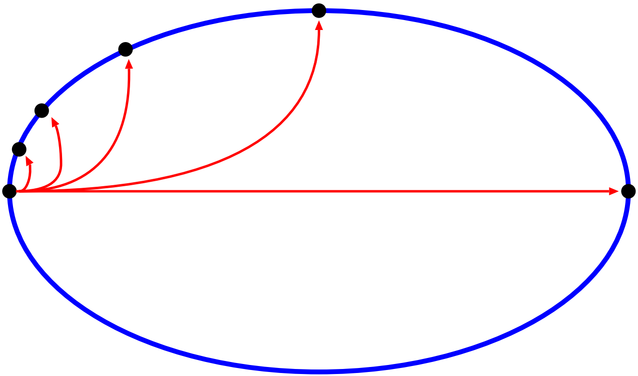
- Each peer has a hash (randomly generated)
- Each peer stores a list for a hash
- In fact, each peer stores a list for all hashes *closed* to its hash

Chord (1)



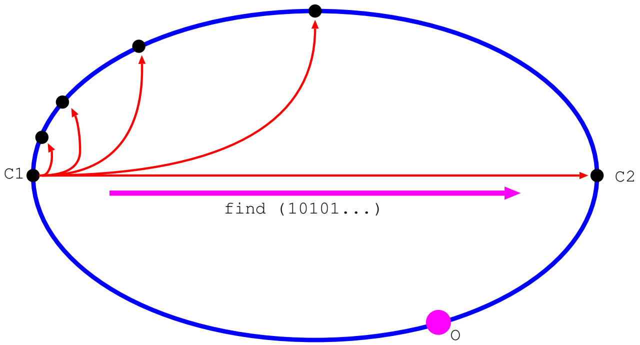
- All clients are put on a logical ring, depending on their random identifier

Chord (2)



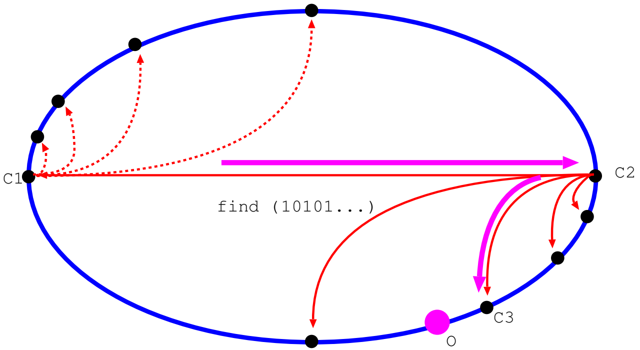
- Each client has a routing table, containing peers (*fingers*) with particular identifiers :
  - Peer *id* has one neighbour in the interval $[id..id+\frac{1}{2}]$, $[id+\frac{1}{2}..id+\frac{1}{4}]$, $[id+\frac{1}{4}..id+\frac{1}{8}]$, $[id+\frac{1}{8}..id+\frac{1}{16}]$,...

Chord (3)



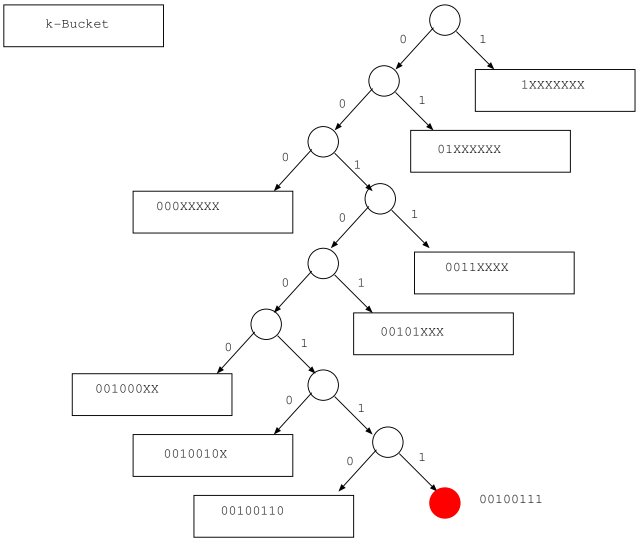
- To find a file with hash *hash*, just forward the query to the peer p_i (finger i) such that $hash \in [id + \frac{1}{2^{(i-1)}}..id + \frac{1}{2^i}]$

Chord (4)



- The message is then forwarded again, with the guarantee that each time, the interval will decrease
- When no interval is found, the peer is responsible for replying

Kademlia : trees as routing tables



Peer-to-Peer Download

How to make peer-to-peer download more efficient than client-server download ?

- A file is located on many peers
- Verify that it is the same file
  - Use a cryptographic hash functions (MD5, SHA1)
- Download the file from as many peers as possible :
  - Split the file in blocks
  - Verify each block after download
  - Share each block as soon as possible

Bittorrent Incentives

- Share a file with only 4 other peers
  - 3 peers are the best uploaders
  - 1 peer is chosen randomly
- Change every 30 seconds

Peer-to-Peer Download

How to make peer-to-peer download more efficient than client-server download ?

- A file is located on many peers
- Verify that it is the same file
  - Use a cryptographic hash functions (MD5, SHA1)
- Download the file from as many peers as possible :
  - Split the file in blocks
  - Verify each block after download
  - Share each block as soon as possible

Bittorrent Incentives

- Share a file with only 4 other peers
  - 3 peers are the best uploaders
  - 1 peer is chosen randomly
- Change every 30 seconds