Architecture of Financial Systems

Prof. Stephen A. Edwards

Columbia University

April 2009
A Billing System

- Four different databases
- Eighteen other processing elements
- Transaction and web inputs; external link to credit card processor

(Courtesy of Steve Bellovin)
Part I

Networking
Ethernet

Started in about 1976 at Xerox PARC

IEEE Standard 802.3

Carrier-sense multiple access/carrier detect protocol:

1. Listen to the cable
2. If nobody’s there, start talking
3. If someone interrupts, stop, and retry after a random time
10Base-5 “Thicknet”

Shared coax bus with “vampire tap” tranceivers

Yellow color suggested by the 802.3 standard

From http://www.turkcenet.org/yerel_htm/10base5.htm
10Base-2 “Thinnet”

50-Ohm coax segments with BNC “T” connectors

Coax invariably black
10Base-T and 100Base-T

Put the shared medium in a hub: a star topology. Everybody uses it now.

From http://www.asante.com/downloads/legacy/fh200bugra.pdf and
http://www.connectworld.net/cables_u/patch-cable-manufacturer.html
100Base-TX wiring (CAT 5)

Pair of twisted pairs, one pair for each direction.

Hub-to-computer cable is straight-through.

Computer-to-computer cable is a “crossover.”

From the Netgear EN104TP 4-port hub manual off of Amazon.com
Encapsulating Data in UDP
### An Ethernet Frame

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>46–1500</td>
<td>4</td>
</tr>
</tbody>
</table>

- **SOF**: Start of Frame
- **Dest.**: Destination address
- **Src.**: Source address
- **Type**: Type of packet or length of data field
  - 0x0800 for IP, 0x0806 for ARP, etc.

Bytes sent LSB first

- Minimum packet length: 64 (6 + 6 + 2 + 46 + 4)
- Lengths > 1500 indicate packet type
Ethernet (MAC) addresses

48 bits ≈ 281 trillion (world population: 6.5 billion)

Bits 48–24: Vendor code

Bit 41: 0=ordinary, 1=group (broadcast) address

Bits 23–0: Serial number

On my desktop:

$ ifconfig eth0
eth0 Ethernet HWaddr 00:08:74:23:CC:AB

OUI (Organizationally Unique Identifier):

00:08:74 is Dell Computer

Address FF:FF:FF:FF:FF:FF is broadcast
## An Ethernet Packet

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00d06269c00</td>
<td>Destination MAC address (router)</td>
</tr>
<tr>
<td>00087423ccab</td>
<td>Source MAC address (my desktop)</td>
</tr>
<tr>
<td>0800</td>
<td>Type = IP packet</td>
</tr>
<tr>
<td>45</td>
<td>IPv4, 5 word (20-byte) header</td>
</tr>
<tr>
<td>00</td>
<td>Normal service</td>
</tr>
<tr>
<td>0028</td>
<td>Total length = 40 bytes</td>
</tr>
<tr>
<td>c31c</td>
<td>Identification (unique)</td>
</tr>
<tr>
<td>4000</td>
<td>“Don’t Fragment”</td>
</tr>
<tr>
<td>40</td>
<td>64 hops to live</td>
</tr>
<tr>
<td>06</td>
<td>TCP protocol</td>
</tr>
<tr>
<td>3ff1</td>
<td>Header checksum (one's complement)</td>
</tr>
<tr>
<td>803b1372</td>
<td>Source IP 128.59.19.114 (desktop)</td>
</tr>
<tr>
<td>40ec6329</td>
<td>Destination IP 64.236.99.41</td>
</tr>
</tbody>
</table>

deac 0050 bf49 9ba6 a1a4 8bed 5010 ffff 1093 0000
## IP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Positions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>31-28</td>
<td>4 for IPv4</td>
</tr>
<tr>
<td>Words in Header</td>
<td>27-24</td>
<td>Number of 32-bit words in the header</td>
</tr>
<tr>
<td>Type of Service</td>
<td>23-16</td>
<td>Typically 0 (data)</td>
</tr>
<tr>
<td>Total number of bytes in the IP packet</td>
<td>15-12</td>
<td>Number of bytes in the IP packet</td>
</tr>
<tr>
<td>Identification Number</td>
<td>15-0</td>
<td>Which packet</td>
</tr>
<tr>
<td>Flags</td>
<td>15</td>
<td>DF: Don't Fragment, MF: More Fragmentation</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>13-12</td>
<td>(which fragment)</td>
</tr>
<tr>
<td>Time-to-Live</td>
<td>16-12</td>
<td>Hops left</td>
</tr>
<tr>
<td>Protocol</td>
<td>16-15</td>
<td>6 = TCP, 17 = UDP</td>
</tr>
<tr>
<td>Header checksum</td>
<td>0</td>
<td>One's complement sum</td>
</tr>
<tr>
<td>Source IP Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options and padding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IP Addresses

32 bits ≈ 4 billion (world population: 6.5 billion)

First $n$ bits indicate network ($n = 8, 16, 24$)

For example, columbia.edu
owns 128.59.0.0 – 128.59.255.255

Magical addresses:

127.0.0.1 “Me”
192.168.x.x Never assigned worldwide
10.x.x.x Never assigned worldwide
255.255.255.255 Broadcast
## UDP Packets

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (bytes)</td>
<td>Checksum or 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Dumb packet protocol: unreliable, danger of out-of-order delivery
Part II

Servers
19" Racks
Single-Core Processors

Intel i386, c. 1985. 275,000 transistors, $1.5\mu = 1500$ nm
Multi-Core Processors

Intel Core 2 Quad, c. 2007. 582,000,000 transistors, 65 nm
Hard Disk Drives
Areal Density Trend

RAID: Redundant Array of Inexpensive Disks

RAID 0
- Striping

RAID 1
- Mirroring

RAID 2
- Error Correction

RAID 3
- Byte Parity

RAID 4
- Block Parity
RAID: Redundant Array of Inexpensive Disks

Distributed Parity
1 may fail safely

Dual Distributed Parity
2 may fail safely
Part III

Operating Systems
Batch Operating Systems

Original computers ran in batch mode:

Submit job & its input

Job runs to completion

Collect output

Submit next job

Processor cycles very expensive at the time

Jobs involved reading, writing data to/from tapes

Costly cycles were being spent waiting for the tape!
Timesharing Operating Systems

Way to spend time while waiting for I/O:
Let another process run

- Store multiple batch jobs in memory at once
- When one is waiting for the tape, run the other one

Basic idea of timesharing systems

Fairness primary goal of timesharing schedulers

- Let no one process consume all the resources
- Make sure every process gets equal running time
Aside: Modern Computer Architectures

Memory latency now becoming an I/O-like time-waster.
CPU speeds now greatly outstrip memory systems.
All big processes use elaborate multi-level caches.

An Alternative:
Many high-end chips now contain two or three contexts (Intel’s “Hyperthreading”). Can switch among them “instantly.”
Idea: while one process blocks on memory, run another.
OSes vs. Virtual Machines
Part IV

Databases
The Relational Model: Raw Data

<table>
<thead>
<tr>
<th>Trans. Code</th>
<th>Date</th>
<th>Destination Code</th>
<th>Destination Name</th>
<th>Product Code</th>
<th>Product Name</th>
<th>Unit Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>3/5</td>
<td>12</td>
<td>Minani</td>
<td>101</td>
<td>Melon</td>
<td>800</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>102</td>
<td>Cherry</td>
<td>150</td>
<td>500</td>
</tr>
<tr>
<td>1102</td>
<td>3/7</td>
<td>23</td>
<td>Alpha</td>
<td>103</td>
<td>Apple</td>
<td>120</td>
<td>1700</td>
</tr>
<tr>
<td>1103</td>
<td>3/8</td>
<td>25</td>
<td>Ritol</td>
<td>104</td>
<td>Lemon</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>1104</td>
<td>3/10</td>
<td>12</td>
<td>Minami</td>
<td>101</td>
<td>Melon</td>
<td>800</td>
<td>2500</td>
</tr>
<tr>
<td>1105</td>
<td>3/12</td>
<td>25</td>
<td>Ritol</td>
<td>103</td>
<td>Apple</td>
<td>120</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>104</td>
<td>Lemon</td>
<td>200</td>
<td>700</td>
</tr>
</tbody>
</table>
First & Second Normal Forms

<table>
<thead>
<tr>
<th>Trans. Code</th>
<th>Date</th>
<th>Destination Code</th>
<th>Destination Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>3/5</td>
<td>12</td>
<td>Minani</td>
</tr>
<tr>
<td>1102</td>
<td>3/7</td>
<td>23</td>
<td>Alpha</td>
</tr>
<tr>
<td>1103</td>
<td>3/8</td>
<td>25</td>
<td>Ritol</td>
</tr>
<tr>
<td>1104</td>
<td>3/10</td>
<td>12</td>
<td>Minami</td>
</tr>
<tr>
<td>1105</td>
<td>3/12</td>
<td>25</td>
<td>Ritol</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trans. Code</th>
<th>Product Code</th>
<th>Product Name</th>
<th>Unit Price</th>
<th>Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>101</td>
<td>Melon</td>
<td>800</td>
<td>1100</td>
</tr>
<tr>
<td>1102</td>
<td>103</td>
<td>Apple</td>
<td>120</td>
<td>1700</td>
</tr>
<tr>
<td>1103</td>
<td>104</td>
<td>Lemon</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>1104</td>
<td>101</td>
<td>Melon</td>
<td>800</td>
<td>2500</td>
</tr>
<tr>
<td>1105</td>
<td>103</td>
<td>Apple</td>
<td>120</td>
<td>2000</td>
</tr>
<tr>
<td>1105</td>
<td>104</td>
<td>Lemon</td>
<td>200</td>
<td>700</td>
</tr>
</tbody>
</table>

- Rows with multiple data split
- Each row has a primary key (one or more columns) that uniquely determine the rest of the row
### Third Normal Form

<table>
<thead>
<tr>
<th>Transactions</th>
<th>Sales</th>
<th>Destinations</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>3/5</td>
<td>12</td>
<td>1101</td>
</tr>
<tr>
<td>1102</td>
<td>3/7</td>
<td>23</td>
<td>1101</td>
</tr>
<tr>
<td>1103</td>
<td>3/8</td>
<td>25</td>
<td>1102</td>
</tr>
<tr>
<td>1104</td>
<td>3/10</td>
<td>12</td>
<td>1103</td>
</tr>
<tr>
<td>1105</td>
<td>3/12</td>
<td>25</td>
<td>1104</td>
</tr>
<tr>
<td>1105</td>
<td>3/12</td>
<td>25</td>
<td>1105</td>
</tr>
<tr>
<td>1105</td>
<td>3/12</td>
<td>25</td>
<td>1105</td>
</tr>
</tbody>
</table>

- Primary key always necessary to determine value in other rows
CREATE TABLE products (  
  prod int NOT NULL,
  prod_name varchar(255),
  price int NOT NULL,
  PRIMARY KEY (prod)
);

INSERT INTO products VALUES(101, 'Melon', 800);
INSERT INTO products VALUES(102, 'Cherry', 150);
INSERT INTO products VALUES(103, 'Apple', 120);
INSERT INTO products VALUES(104, 'Lemon', 200);
CREATE TABLE destinations (  
dest int NOT NULL,  
dest_name varchar(255),  
PRIMARY KEY(dest)  
);

CREATE TABLE products (  
prod int NOT NULL,  
prod_name varchar(255),  
price int NOT NULL,  
PRIMARY KEY(prod)  
);

CREATE TABLE sales (  
trans int NOT NULL,  
prod int NOT NULL,  
quantity int NOT NULL,  
PRIMARY KEY(trans, prod)  
);

CREATE TABLE transactions (  
trans int NOT NULL,  
tdate date NOT NULL,  
dest int NOT NULL,  
PRIMARY KEY(trans)  
);
SQL: A Query

```
SELECT tdate, dest_name, prod_name, price, quantity
FROM sales, transactions, products, destinations
WHERE transactions.trans = sales.trans
  AND transactions.dest = destinations.dest
  AND sales.prod = products.prod;
```

2009-3-5|Minani|Melon|800|1100
2009-3-5|Minani|Cherry|150|500
2009-3-7|Alpha|Apple|120|1700
2009-3-8|Ritol|Lemon|200|500
2009-3-10|Minani|Melon|800|2500
2009-3-12|Ritol|Apple|120|2000
2009-3-12|Ritol|Lemon|200|700
SQL: Reconstructing the original table

```
SELECT transactions.trans, tdate, transactions.dest, dest_name, sales.prod, prod_name, price, quantity
FROM sales, transactions, products, destinations
WHERE transactions.trans = sales.trans AND
  transactions.dest = destinations.dest AND
  sales.prod = products.prod;
```

<table>
<thead>
<tr>
<th>Trans. Code</th>
<th>Date</th>
<th>Destination Code</th>
<th>Destination Name</th>
<th>Product Code</th>
<th>Product Name</th>
<th>Unit Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>2009-3-5</td>
<td>12</td>
<td>Minani</td>
<td>101</td>
<td>Melon</td>
<td>800</td>
<td>1100</td>
</tr>
<tr>
<td>1101</td>
<td>2009-3-5</td>
<td>12</td>
<td>Minani</td>
<td>102</td>
<td>Cherry</td>
<td>150</td>
<td>500</td>
</tr>
<tr>
<td>1102</td>
<td>2009-3-7</td>
<td>23</td>
<td>Alpha</td>
<td>103</td>
<td>Apple</td>
<td>120</td>
<td>1700</td>
</tr>
<tr>
<td>1103</td>
<td>2009-3-8</td>
<td>25</td>
<td>Ritol</td>
<td>104</td>
<td>Lemon</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>1104</td>
<td>2009-3-10</td>
<td>12</td>
<td>Minani</td>
<td>101</td>
<td>Melon</td>
<td>800</td>
<td>2500</td>
</tr>
<tr>
<td>1105</td>
<td>2009-3-12</td>
<td>25</td>
<td>Ritol</td>
<td>103</td>
<td>Apple</td>
<td>120</td>
<td>2000</td>
</tr>
<tr>
<td>1105</td>
<td>2009-3-12</td>
<td>25</td>
<td>Ritol</td>
<td>104</td>
<td>Lemon</td>
<td>200</td>
<td>700</td>
</tr>
</tbody>
</table>
Part V

Websites
<!DOCTYPE html>
<html>
<head>
  <title>Hello HTML</title>
</head>
<body>
<p>Hello World!!</p>

<h1>Paragraphs</h1>
<p>This is the first paragraph. Not too interesting.</p>
<p>This is the second paragraph. Still not too interesting.</p>

<h2>A Table</h2>
<table>
  <tr><th>Date</th><th>Dest</th><th>Product</th><th>Price</th><th>Quantity</th></tr>
  <tr><td>2009-3-5</td><td>Minani</td><td>Melon</td><td>800</td><td>1100</td></tr>
  <!-- rest of the rows -->
</table>
</body>
</html>
Client-side Scripting: Javascript

```html
<!DOCTYPE html PUBLIC>
<html>
<head><title>Program that rolls two dice</title>
<script language = "JavaScript">
  function play() {
    sumOfDice = rollDice();
    if ( sumOfDice == 2 )
      window.alert("Snake Eyes!");
  }
  function rollDice() {
    var die1, die2, sum;
    die1 = Math.floor( 1 + Math.random() * 6 );
    die2 = Math.floor( 1 + Math.random() * 6 );
    sum = die1 + die2;
    dice.firstDie.value = die1;
    dice.secondDie.value = die2; dice.sum.value = sum;
    return sum;
  }
</script>
</head>
<body>
<form name = "dice">
  <table border = "1">
    <tr><td>Die 1</td><td><input name="firstDie" type="text"></td></tr>
    <tr><td>Die 2</td><td><input name="secondDie" type="text"></td></tr>
    <tr><td>Sum</td><td><input name="sum" type="text"></td></tr>
    <tr><td><input type="button" value="Roll Dice" onclick="play()"></td></tr>
  </table>
</form>
</body>
</html>
```
function roll() {
    srand(time());
    for ($i=0; $i < 2; $i++)
        $die[] = rand() % 6 + 1;
    print("\t<tr>\n        \t<td width="33%">\n            \t<center>$die[0]\n                \t</center>\n        \t</td>\n        \t<td width="33%">\n            \t<center>$die[1]\n                \t</center>\n        \t</td>\n        \t</tr>\n    if ($die[0] == 1 && $die[1] == 1)
        print("\t</tr><td colspan="3">\n            \tSnake Eyes!\n        </td></tr>\n    }?
    <table border="1">
        <? roll(); ?>
        <tr>
            <td colspan="3" align="center">
                <form method="POST" action="dice.php">
                    <input type="submit" value="Roll"/>
                </form>
            </td>
        </tr>
        </table>
    </body>
</html>
Part VI

Security
# A Typical Hour For My Firewall

<table>
<thead>
<tr>
<th>Time</th>
<th>Port</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 3 19:41:55</td>
<td>2967</td>
<td>61.147.107.56</td>
</tr>
<tr>
<td>May 3 19:45:03</td>
<td>9090</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 19:45:03</td>
<td>7212</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 19:59:26</td>
<td>9090</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 19:59:26</td>
<td>7212</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 20:18:44</td>
<td>9090</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 20:18:44</td>
<td>7212</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 20:19:08</td>
<td>4899</td>
<td>202.28.25.152</td>
</tr>
<tr>
<td>May 3 20:22:50</td>
<td>55277</td>
<td>59.31.217.194</td>
</tr>
<tr>
<td>May 3 20:25:18</td>
<td>80</td>
<td>92.240.68.152</td>
</tr>
<tr>
<td>May 3 20:33:28</td>
<td>9090</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 20:33:28</td>
<td>7212</td>
<td>221.192.199.36</td>
</tr>
<tr>
<td>May 3 20:35:55</td>
<td>33439</td>
<td>72.34.61.146</td>
</tr>
<tr>
<td>May 3 20:36:04</td>
<td>33440</td>
<td>72.34.61.146</td>
</tr>
<tr>
<td>May 3 20:36:09</td>
<td>33439</td>
<td>72.34.61.146</td>
</tr>
<tr>
<td>May 3 20:36:09</td>
<td>33440</td>
<td>72.34.61.146</td>
</tr>
<tr>
<td>May 3 20:36:14</td>
<td>33439</td>
<td>72.34.61.146</td>
</tr>
<tr>
<td>May 3 20:36:14</td>
<td>33440</td>
<td>72.34.61.146</td>
</tr>
<tr>
<td>May 3 20:36:19</td>
<td>33439</td>
<td>72.34.61.146</td>
</tr>
</tbody>
</table>

80 = HTTP, 2967 = Symantec Antivirus, 4899 = radmin (remote PC software), 7212 = GhostSurf (anonymity), 9090 = WebSM (IBM RS/6000)
SSH Attacks

May 3 07:44:55 Did not receive identification string from 88.191.20.156
May 3 16:06:59 refused connect from 117.21.249.75 (117.21.249.75)
May 3 16:45:27 Did not receive identification string from 216.128.67.231
May 3 17:02:42 Invalid user emma from 216.128.67.231
May 3 17:21:54 Did not receive identification string from 216.128.67.231
May 3 17:45:20 Failed password for root from 216.128.67.231 port 60196 ssh2
May 3 19:08:29 Did not receive identification string from 94.43.103.230
May 3 19:13:15 Failed password for root from 94.43.103.230 port 38393 ssh2
May 3 19:13:18 Failed password for root from 94.43.103.230 port 38429 ssh2
May 3 19:13:21 Failed password for root from 94.43.103.230 port 38518 ssh2
May 3 19:13:24 Failed password for root from 94.43.103.230 port 38604 ssh2
May 3 19:13:28 Failed password for root from 94.43.103.230 port 38726 ssh2
May 3 19:13:31 Failed password for root from 94.43.103.230 port 38849 ssh2
May 3 19:13:32 refused connect from 94.43.103.230 (94.43.103.230)
Vending Machines
<table>
<thead>
<tr>
<th>Source of Loss</th>
<th>Loss</th>
<th>Countermeasure</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theft through front door</td>
<td>$20,000</td>
<td>Better door</td>
<td>$1,000</td>
</tr>
<tr>
<td>Office supplies</td>
<td>$100</td>
<td>Closet with lock</td>
<td>$5,000</td>
</tr>
<tr>
<td>Dishonest waiters</td>
<td>$10,000</td>
<td>Restaurant Guards</td>
<td>$1,000,000</td>
</tr>
</tbody>
</table>
Symmetric-Key Cryptography

- Sender and receiver need to share a secret (the key) and the algorithm.
- Size of keys directly related to security, computational complexity. 128, 192, 256-bit keys now typical.
- Examples: AES (128, 192, or 256 bits), Blowfish (32–448 bits, 128 typ.), DES (56-bit, insecure), Triple DES (168-bit)
Public Key (Asymmetric) Cryptography

E.g., RSA.

Slower than symmetric-key cryptography; often used to exchange keys for symmetric crypto algorithms
TLS aka SSL

**Phase 1**
- Client: Generate random number
- Server: Generate random number

**Phase 2**
- Server: Send certificate
- Client: Check server certificate

**Phase 3**
- Client: Send client certificate
- Server: Check client certificate
- Server: Generate random number pre-master-secret
- Server: Send encrypted message with public key server
- Client: Calculate Master-Secret

**Phase 4**
- Change to encrypted connection with Master-Secret as key
- End SSL handshake
- Change to encrypted connection with Master-Secret as key
What the US Government Assumes

Serial #1 goes to the Kremlin
Security through obscurity guaranteed not to work
Operation Ivy Bells

Submarine Cable Tap

USS Halibut
Never Trust Proprietary Solutions

Incredibly difficult to get right
What are they hiding? Trojan Horses?