Networking in Java

10 March 2011
CMPT166
Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Outline for today

- Sockets: TCP vs. UDP
- TCP client-server in Java
  - Multithreaded server
- UDP communication in Java
- Networking concepts
  - IP addresses
  - NAT
  - IPv6
  - DNS

Sockets

- Sockets are a way for processes to communicate
  - Foundation of the Internet, including HTTP, FTP, IM, streaming media, etc.
- Local or Internet: same host or diff hosts?
- Connection-based or connectionless: must each packet specify destination?
- Packets or streams: message boundaries?
- Reliable or unreliable: Can messages be lost, duplicated, reordered, or corrupted?

TCP vs. UDP

- All data on the Internet is sent via packets conforming to the Internet Protocol (IP)
  - Specify host and port (0-65535)
- Two most common types of packets:
  - TCP: Transmission Control Protocol:
    - Virtual circuit: connection-based
    - Client-server model
  - UDP: User Datagram Protocol:
    - Connectionless: peer-to-peer, less overhead
    - Packets might disappear, or be out of order, or get duplicated

TCP client-server

- TCP is connection-based:
  - Phone analogy
  - Initial setup, but subsequent packets do not need to specify destination again
  - Server: waits, listens for client
  - Client: initiates connection (phone call)
  - Once connection is established, communication may be two-way (send/receive)
  - Either client or server may terminate

Making a TCP Server in Java

- java.net.ServerSocket object

```
server = new ServerSocket( port, maxcl );
```

  - *port:* port to listen on (0-65536, 0-1023 reserved)
  - *maxcl:* queue length (reject extra clients)
  - BindException raised if port invalid or in use
- Bind socket (start listening) (blocking):

```
conn = server.accept();
```

  - Returns a java.net.Socket object
- Communicate via streams:

```
conn.getInputStream();
conn.getOutputStream();
```

TRINITY WESTERN UNIVERSITY

Communicating with streams

- Both client and server may send or receive:

```
conn.getInputStream()
conn.getOutputStream()
```

- Communicate via text streams:

```
new Scanner( conn.getInputStream() );
new PrintWriter( conn.getOutputStream() );
```

- Or object streams:

```
new ObjectInputStream( conn.getInputStream() );
new
ObjectOutputStream( conn.getOutputStream() );
```

How do we accept clients?

- Iterating server: only one client at a time
  - One operator answering phones
  - Simplest to implement
- Forking server:
  - Split off a child thread for each connection
  - Original master thread continues to listen
  - Switchboard

More on forking server

- Multiple threads running concurrently
- Master thread listens on port
- When a client connects, fork off a thread
  - Thread handles communication with that client
- Master thread continues listening for other connections (switchboard)
- Overhead in forking new threads: so keep pool of available threads, and reuse dormant threads

Connectionless client/server

- TCP is connection-oriented
- UDP is connectionless
  - Send data one packet at a time
    - Similar to envelopes through CanadaPost
    - Fragment larger data into multiple packets
  - Packets might:
    - Not arrive at all
    - Arrive out of order
    - Get duplicated
  - Less overhead, better latency and possibly better throughput

Receiving a UDP packet

- Create a DatagramSocket (in java.net):

```
sock = new DatagramSocket( port );
```

- Create a DatagramPacket to store the data:

```
byte payload[] = new byte[ 100 ];
packet = new DatagramPacket(
                        payload, payload.length );
```

- Wait (block) for a packet:

```
sock.receive( packet );
```

- Read info from packet:

```
packet.getData(), .getLength(),
.getAddress(), .getPort()
```

Sending a UDP packet

- Prepare payload:

```
String msg = "Hello, World!";
byte[] payload = msg.getBytes();
```

- Package payload:

```
packet = new DatagramPacket(
    payload, payload.length,
    hostname, port );
```

- Send packet:

```
socket.send( packet );
```

Networking layers

- OSI 7-layer model of networking
- 7: Application (HTTP, SMTP)
- 6: Presentation: data repr., encryption (SSL)
- 5: Session: auth, session checkpointing/restore, stream synchronization (sockets, SSH, RPC)
- 4: Transport: reliability, connection (TCP, UDP)
- 3: Network: routing, addressing (IP)
- 2: Data link: physical address (Ethernet MAC)
- 1: Physical: signals (twisted-pair, fiber, radio)

IP addresses

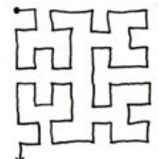
- Every public Internet host has an IP address:
  - Four bytes: e.g., 64.114.134.52
- IP addresses are partitioned into networks (blocks of addresses), via a netmask:
  - e.g., 64.114.134.52 / 255.255.255.0 (or /24) means range: 64.114.134.0 – 64.114.134.255
- Large chunks of the IP address space have been given out to countries, organizations, companies, etc.
  - IBM has 9.0.0.0 / 8 (1/256th of the IP space!)

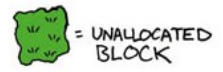
xkcd visualization of IP space

http://xkcd.com/195/

THIS CHART SHOWS THE IP ADDRESS SPACE ON A PLANE USING A FRACTAL MAPPING WHICH PRESERVES GROUPING -- ANY CONSECUTIVE STRING OF IPs WILL TRANSLATE TO A SINGLE COMPACT, CONTIGUOUS REGION ON THE MAP. EACH OF THE 256 NUMBERED BLOCKS REPRESENTS ONE /8 SUBNET (CONTAINING ALL IPs THAT START WITH THAT NUMBER). THE UPPER LEFT SECTION SHOWS THE BLOCKS SOLD DIRECTLY TO CORPORATIONS AND GOVERNMENTS IN THE 1990's BEFORE THE RIRs TOOK OVER ALLOCATION.

0	1	14	15	16	19 →
3	2	13	12	17	18
4	7	8	11		
5	6	9	10		





Running out of IP space: NAT

- Very few public IP addresses left! Solutions?
- NAT (Network Address Translation)
  - LAN goes through router to get to Internet
  - Router gets one public IP address
    - 64.114.134.52 is TWU's
  - LAN gets private IP addresses:
    - 192.168.*/16, 172.16.*/12, 10.*/8
  - Connections mapped to ports on the router
  - How to run public services on a LAN host?

Running out of IP space: IPv6

- Another solution: IPv6
- 128-bit addresses instead of 32-bit
  - Each known star in the sky could get $4.5*10^{15}$ addresses!
- 64 bits to identify the subnet
  - Hierarchy simplifies routing
  - Easier to do multicast, etc.
- 64 bits to identify host uniquely
  - Every network card has a unique 64-bit MAC (media access control) address

Names to numbers: DNS

- Using "twu.ca" instead of 64.114.134.52
- Top-level domains: .com, .org, .ca, etc.
- DNS (Domain Name System):
  - Query local server for host's IP address
    - May return several IP addresses!
    - Also info on mail server, owner, etc.
  - Authoritative for its own domain
  - If it doesn't know, it asks other servers
    - Which may tell it which server to ask
  - Root servers: [a-m].root-servers.net