Networks: DNS, byte ordering

See:
- socket/ example code

20 Mar 2009
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Review last time: networks

- OSI 7-layer model of networks
- IP addresses, subnets
  - NAT
  - IPv6

Reserved IP addresses

- Special reserved IP addresses:
  - Private: 192.168.*/16, 172.16.*/12, 10.*/8
  - Broadcast: send to whole subnet
    - e.g., 192.168.1.255 floods 192.168.1/24
    - 255.255.255.255: limited broadcast to LAN
  - Multicast: 224.0.0.0 – 239.255.255.255
    - Each addr represents a group of listeners
    - Hosts may subscribe to a multicast address
  - Localhost: 127.*/8, e.g., 127.0.0.1
    - Same host that we're running on

From names to numbers: DNS

- Want to say “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

DNS lookup: gethostbyname()

```
#include <netdb.h>
struct hostent* myDNS =
    gethostbyname( "www.twu.ca" );
```

- Returns DNS record as a hostent struct:
  - h_name (char*): the canonical name
  - h_aliases (char**): other aliases (array)
  - h_addrtype (int): AF_INET
  - length (int): 4 bytes for IPv4
  - h_addr_list (struct in_addr**): IP addrs
  - h_addr (in_addr*): pick one IP address
- Only for IPv4! See getaddrinfo() for IPv6

Ports

- A socket needs both an IP address and a port
- Ports 0 – 1023 are reserved for specific uses
  - 53:dns, 80:http, 587:smtp submit, etc.
- Ports 1024 – 49151 are registered but free
  - Open, but register with an ICANN registrar to ensure interoperation with other apps
  - /etc/services for a list
- Ports 49152 – 65535 are dynamic for everyone
  - Often used for outgoing client connections

Network byte order

- Age-old problem: how to store multi-byte nums?
- Little-endian: least-significant byte (LSB) first
  - (hex) 1 E → 1(1) + 14(16) = 225
  - Intel CPUs
- Big-endian: most-significant byte (MSB) first
  - (hex) 1 E → 1(16) + 14(1) = 30
  - Matches how we write numbers
  - PowerPC, Sun Sparc, ARM (configurable)
- Network byte order (e.g., headers) is big-endian

Byte swapping functions

- Functions to convert between:
  host byte ordering and network byte ordering
  - For shorts (16bit) or longs (32bit)
- Host to network, for shorts:
  - uint_16t htons(uint_16t v);
    - serverAddr.sin_port = htons(4410);
- Network to host, for longs:
  - uint_32t ntohl(uint_32t v);
- etc.: htons(), htonl(), ntohs(), ntohl()

String display of IP addresses

- IP addresses are stored as struct in_addr:
  - 4 bytes (1 unsigned long int)
- Not same as the string: “64.114.134.52”!
    - #include <arpa/inet.h>
    - struct in_addr serverIP;
- ASCII string to numeric (struct in_addr):
    - inet_aton(“64.114.134.52”, &serverIP);
  - Beware octal: 226.0.0.037 => 226.0.0.31!
- Numeric to ASCII:
    - cout << inet_ntoa(serverIP);

Sockets and FLTK

- Sockets are straight C and built-in to the OS
  - Cygwin/Linux: g++ client.cpp -o client.exe
- Theoretically, no problems mixing with FLTK
- BUT: synchronous communication blocks while waiting for the other side
  - e.g., server waiting for client to connect
  - e.g., waiting to receive message
- Program will appear to hang, display won't even refresh!
- Solution: multithreading