Sockets, TCP/IP and Bluetooth

Lecture 3
Feb 15, 2006
Larry Rudolph
A little history

- Simple case: two devices, one cable (2 wires)
  - e.g. first telephone
  - one device puts pulses on wire, the other senses them. Went from analog => digital
- Generalize: multiple devices, one wire
  - why do we want this?
Bus

- signal broadcast (all devices “sense” it)
- first broadcast an address
- device with that address receives data
  - and ack’s it
- Who knows the device’s address?
Ethernet

- Ethernet is a type of bus -- easy to add devices
- Every ethernet device has a unique device address
  - how do you know it is unique?
  - a company sells you a unique address
  - this is known as a MAC
  - all devices listen for their mac on bus
- What if devices is not on the bus?
Gateway or Bridge

- Special device that transfers packets from ethernet to the rest of the world
- How does it know where to send it?
  - Another address
  - IP address  aaa.bbb.ccc.ddd
    - 4 fields of 8 bits (256 values) per field
    - last field is local network
- Where is that address specified?
Layers

• Need different layers
  • lowest (bottom) for physical transport
  • higher layers as get more removed
• Which picture makes sense? (first bits on left)
  • (top Header (bot H B T) top Trailer )
  • (bot Header (top H B T) bot Trailer )
Lots of Layers

- seven official layers to ISO, but mostly three:
  - DNS name (domain name system)
    - blah.com blah.edu blah.sg stuff at the end
  - IP address
  - MAC address
What happens at end?

- message arrives at destination machine
- Operating system deals with it -- why?
  - places msg buffer in system memory
- How does OS know the associated app?
  - what are the choices?
    - must specify some type of name
    - done via agreement
Ports

• First allocate **socket**

• Message contains a port number

• Sending message:
  • must specify port in destination address

• Receiving message
  • application tells OS its associated port
    • the **Bind** command
Protocols

- Lots of different protocols -- common two:
  - UDP (unreliable) & TCP (reliable)
- Why different protocols?
- End-to-end argument
  - Don’t put stuff in lower layers that may not be needed by the end-points.
- What is an end-point?
- Specify in `socket` allocation, AF_INET, SOCK_STREAM
Setting things up

• Client & Server agree on Port number

• Server starts up first
  • specifies protocol, IP, port
  • activates the socket
  • waits for a client to connect

• Client starts up second
  • specifies protocol,
  • Connects to IP, Port
Server Code

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.bind( ("0.0.0.0", port) )

s.listen(5)

(client, address) = s.accept()

while True:
    data = client.recv(1024)
    if len(data) == 0: break

client.close()
Client Code

```python
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.connect( (sys.argv[1], int(sys.argv[2]) ) )

while True:
    s.send(question)
    ans = s.recv(max_size)
```
Bluetooth

- Nearly the same

```python
from bluetooth import *

port = 1
server_sock = BluetoothSocket( RFCOMM )
server_sock.bind( "", port )
server_sock.listen(1)

client_sock, client_info = server_sock.accept()

data = client_sock.recv(1024)

client_sock.close(): server_sock.close()
```