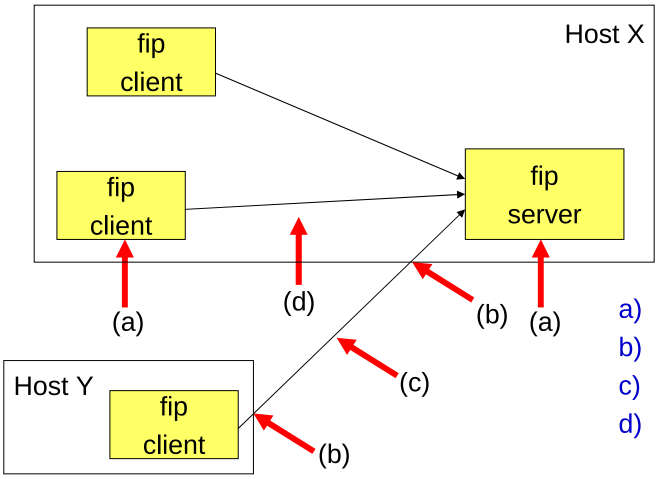
Some Considerations on Protocol Analysis and Debugging

Protocol Analysis and Debugging

- Figuring out why your protocol does not work
- Finding out why it does not interwork with someone else
- Understanding what your implementation actually does
  - What does it send?
  - How does it react when it receives what?
- We focus on the functional aspect for now
- Numerous tools available
  - Support for many standardized protocols
  - Understanding what is going on between third party implementations
  - Understanding whether your protocols sends the right things
- Many tools support performance analysis

Simple Setup: fip



a) In your application
b) In your local system
c) At the link interfaces
d) On the wire/in the air

a) In your application

- Of course, there are gdb, profilers, ...
- Make extensive use of logging
  - Problems may be hard or unpredictable to reproduce
    - Need to live with what you got
  - Use meaningful information, consistent spelling/terminology (for grep(1))
    - Possibly format lines for later processing
  - Include timestamps, sources, destinations
    - You will figure out what you have missed
  - Format for easy subsequent processing (field separators, etc.)
  - May also be helpful for later performance measurements
  - Use command line switches (or config files) to control (the amount of) logging
    - Recompilations without logging (“#ifdef”) may make errors disappear
- Log close to transmission and reception
  - Timestamps are more accurate
  - You cannot have accidentally messed with the buffer

a) In your application (2)

- Hexdumps are useful
  - Gets around internal conversion and interpretation
    - Did you receive the wrong thing or did you interpret it incorrectly

IPv4 UDP UPnP Packet

```
00000380: 4500 00a1 7b90 0000 0111 788b 83f6 5140  E...{.....x...Q@
00000390: efff fffa 0d20 076c 008d 7f2d 4d2d 5345  ..... .l...-M-SE
000003a0: 4152 4348 202a 2048 5454 502f 312e 310d  ARCH * HTTP/1.1.
000003b0: 0a48 6f73 743a 3233 392e 3235 352e 3235  .Host:239.255.25
000003c0: 352e 3235 303a 3139 3030 0d0a 5354 3a75  5.250:1900..ST:u
000003d0: 726e 3a73 6368 656d 6173 2d75 706e 702d  rn:schemas-upnp-
000003e0: 6f72 673a 6465 7669 6365 3a49 6e74 6572  org:device:Inter
000003f0: 6e65 7447 6174 6577 6179 4465 7669 6365  netGatewayDevice
00000400: 3a31 0d0a 4d61 6e3a 2273 7364 703a 6469  :1..Man:"ssdp:di
00000410: 7363 6f76 6572 220d 0a4d 583a 330d 0a0d  scover"..MX:3...
00000420: 0add 9224 420f 3e06 008b 0000 008b 0000  ...$B.>.........
00000430: 0001 005e 7fff fa00 0e35 428a 4d08 00    ...^.....5B.M..
```

b) Local link interface

- Tools for tapping into the packets exchanged on a link
- Tcpdump (www.tcpdump.org)
  - Highly configurable command line tool
  - Capture packets seen by the link interface
    - Builds upon packet capturing library (libpcap)
    - Link interface in promiscuous mode: captures all packet on the wire)
    - Otherwise: only packets anyway received by the node
  - Allows for filtering
  - Stores complete capture, selected packets, or prints summary
  - Allows analysis down to the link layer headers
  - Prerequisite: root access to the system in question
  - Does not work for host local traffic!
  - Numerous tools exist for post-processing

b) Local link interface

- Ethereal (now called Wireshark)
  - www.ethereal.com, www.wireshark.org
  - tcpdump with graphical user interface and built-in analysis tools
  - Broad spectrum of support:
    - Following individual (TCP) connections (including performance analysis)
    - Analyzing message contents (including protocol decoding)
- Obviously does not work if you use security
  - VPN tunnels (IPsec), TLS connections
  - In those cases, you can only analyze their setup

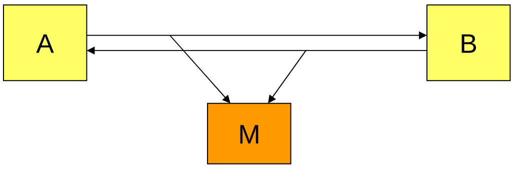
c) Link interface (3rd party monitoring)

- Ethernet: works only with hubs
  - Switches need to be configured to perform snooping on the certain port
- WLAN: promiscuous mode often not supported
  - At least in Windows drivers
  - Does not work with WPA and peerwise negotiated keys
  - AirPcap for wireshark
- Does not work with security (see b)
- In all cases: Respect the privacy of others

d) Local monitoring (1)

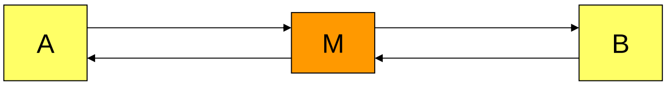
- (Without root permissions)

- UDP: use multicast and write a small protocol monitor
  - Both sides send multicast packets
  - May use the same multicast addresses
    - May need to filter out own ones
  - May use different multicast addresses



d) Local monitoring (2)

- UDP/TCP: build and use a bridge module
  - Forward received data
  - Log the data in arbitrary formats
  - Interpret the protocol as necessary



- strace/trace/truss
  - Monitor system calls executed by the application
  - Essentially works just for simple ones

- Further support may be available from your development tools

d) Local monitoring (3)

- With root permissions and lots of energy :-)
  use/create monitoring inside the kernel

Wireless Networks

- Just for completeness: finding WLANs
  - For configuration purposes or for debugging performance
  - Who is around? And on which channels?
- Network stumbler (www.stumbler.net)
- Kismet (www.kismetwireless.net)
- Sometimes, it is also worthwhile look at the spectrum
  - Microwave ovens, other noise
  - Need specific piece of sensing hardware