Tools/Methods for Protocol Analysis and Debugging

Protocol Analysis & Debugging - Why?

- Multiple implementation of a protocol (different vendors)
  - Verifying interoperability
  - Validating conformance to specification
- Own Protocol Implementation: Figuring out why it does not work?
- Understanding what the implementation actually does?
  - What does it send?
  - How does it react when it receives what?
- There are protocols that come with mandatory and optional features
    - Checking What it supports and what it does not?

Three Approaches Discussed Here

We Focus on analyzing functional aspects of protocol

- Collecting Data From the Application
- Collecting Data From the Link Local Interface
  - Using Tools like wireshark, tcpdump
- Collecting Data by building a bridge module and by using Multicast Address

(Data Refers to Information needed for Analysis and Debugging)

Analyzing Protocol Behavior – (i) From Application

- Make Extensive use of logging (log debug messages)
  - Include Timestamps in Log messages
  - Sender and Destination identifiers
  - Use consistent terminology to classify log messages
  - Use consistent delimiters to separate fields
    - 'grep' can help in analyzing log file
    - subsequent processing shall be easy
  - Use configurable logging depth(amount of)

(i) From Application contd..

- **Logging - Example**

<TimeStamp> <MessageType> <ModuleName or Function Name> <Message>

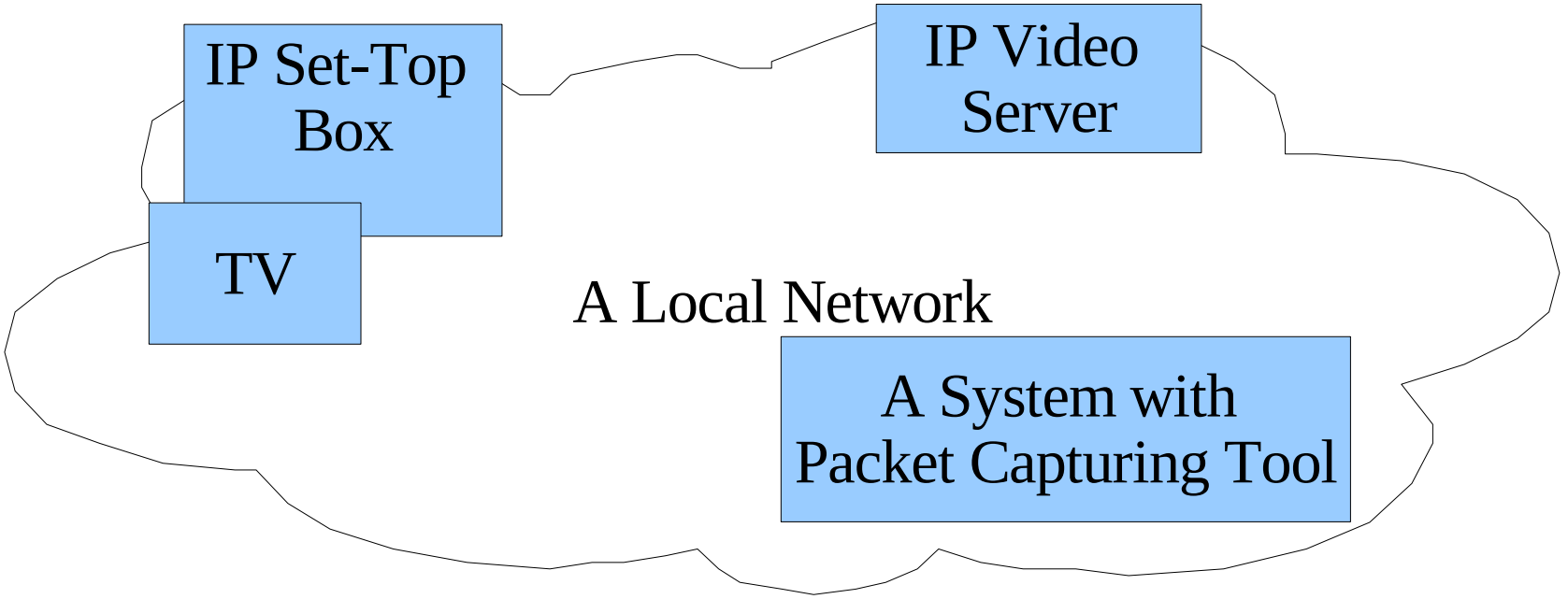
27 Jan 2008 13:25:45	INFO	NetworkStatus	Network is now connected
27 Jan 2008 13:26:05	INFO	ModuleLoader	Loaded 'NetworkManager'
27 Jan 2008 13:26:13	ERROR	ServiceManager	Service Refresh Failed: Failed to parse XML
27 Jan 2008 14:55:43	WARN	ShutdownManager	Preparing to sleep...

- Of course, there are gdb, profilers etc
- Also several tools available detecting memory leaks (Ex: valgrind)
  - Observe strange code behavior ?: Perform Memory checks using tools like valgrind (can detect misuse of allocated memory)

Analyzing Protocol Behavior – (ii) Monitoring Local Link Interface

- As a Participant(as a sender or receiver)
  - Wireshark, tcpdump
    - Supports many standardized protocols
    - Allows filtering based on protocols, addresses etc
    - Possible to build tools to automate the analysis
- Monitoring and Analyzing as a Third Party
  - To analyze the exchanges between two devices
  - Devices may not support running tools like wireshark
- Monitoring Local Link Interface does not work, if encryption is used
  - VPN tunnels (IPsec), TLS connections
  - In those cases, one can only analyze their setup

(i) Monitoring Link Local Interface As a Third Party



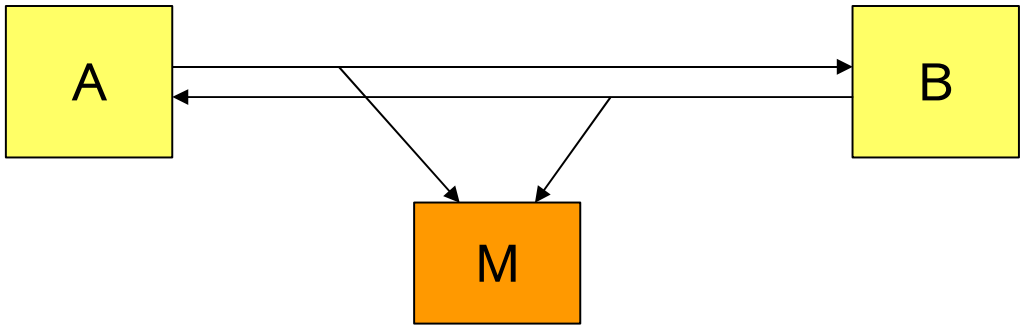
NEED: Message exchanges of IP Set-Top Box and IP Video Server need to be analyzed(to fix/spot an Interoperability Issue)

(ii) Monitoring Link Local Interface As a Third Party

- Feasibility of analyzing the message exchanges depends on the support from underlying Layer-2 device
- **Ethernet:** works only with hubs
  - OR Switches need to be configured to perform snooping on the certain port
- **WLAN:** promiscuous mode requires root privileges
  - AirPcap for wireshark
    - not part of default wireshark package and not FREE

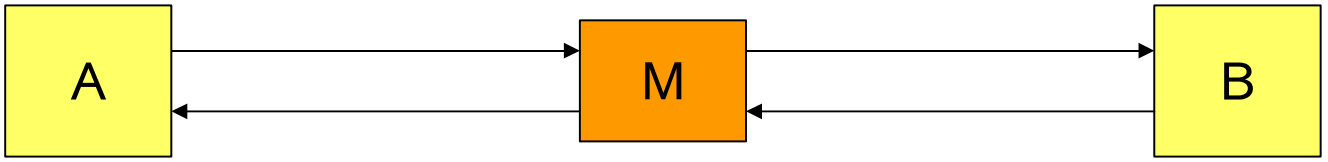
Analyzing Protocol Behavior - (iii) (a)Using Multicast Address

- (Without root permissions)
- Can be used to Analyze protocol operating over UDP
- Use multicast address 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



(iii) (b) Using a bridge Module

- UDP/TCP: build and use a bridge module
  - Forward received data
  - Log the data in arbitrary formats
  - Interpret the protocol as necessary
  - Particularly useful, in the absence of root permissions



Other Relevant Tools/Methods

- Possible to add support for your own protocol in Wireshark
  - http://www.wireshark.org/docs/wsdg_html_chunked/ChDissectAdd.html
- Monitoring WLAN(s) Network
  - For configuration purposes or for debugging performance
  - Who is around? And on which channels?
  - Kismet (www.kismetwireless.net)
- Bridge modules can be prepared to create error scenarios
- Some Linux Utilities that can emulate error conditions
  - NIST Net, Netem (Network Emulator)
  - Can emulate variable delay, loss, duplication and re-ordering
  - http://www.linux-foundation.org/en/Net:Netem
  - http://snad.ncsl.nist.gov/nistnet/