CIS 551 / TCOM 401

Computer and Network Security

Spring 2009

Lecture 5

Announcements

- First project: Due: 6 Feb. 2009 at 11:59 p.m.
- http://www.cis.upenn.edu/~cis551/project1.html
- Group project:
  - 2 or 3 students per group
  - Send e-mail to cis551@seas.upenn.edu with your group

- Plan for Today
  - Worms & Viruses Continued
  - Start of Network Security

Worm Research Sources

- "Inside the Slammer Worm"
  - Moore, Paxson, Savage, Shannon, Staniford, and Weaver
- "How to 0wn the Internet in Your Spare Time"
  - Staniford, Paxson, and Weaver
- "The Top Speed of Flash Worms"
  - Staniford, Moore, Paxson, and Weaver
- "Internet Quarantine: Requirements for Containing Self-Propagating Code"
  - Moore, Shannon, Voelker, and Savage
- "Automated Worm Fingerprinting"
  - Singh, Estan, Varghese, and Savage

- Links on the course web pages.

Analysis: Random Constant Spread Model

- IP address space = 2^{32}
- N = size of the total vulnerable population
- S(t) = susceptible/non-infected hosts at time t
- I(t) = infective/infected hosts at time t
- β = Contact likelihood
- s(t) = S(t)/N   proportion of susceptible population
- i(t) = I(t)/N   proportion of infected population

- Note: S(t) + I(t) = N

Infection rate over time

- Change in infection rate is expressed as:

$$\frac{dI}{dt} = \underbrace{I(t)}_{\text{\# of infected hosts}} * \underbrace{\beta}_{\text{rate of contact}} * \underbrace{s(t)}_{\text{likelihood that contacted hosts is susceptible}}$$

# of infected hosts

rate of contact

likelihood that contacted hosts is susceptible

Rewrite to obtain:

$$\frac{di}{dt} = \beta * i(t) * (1-i(t))$$

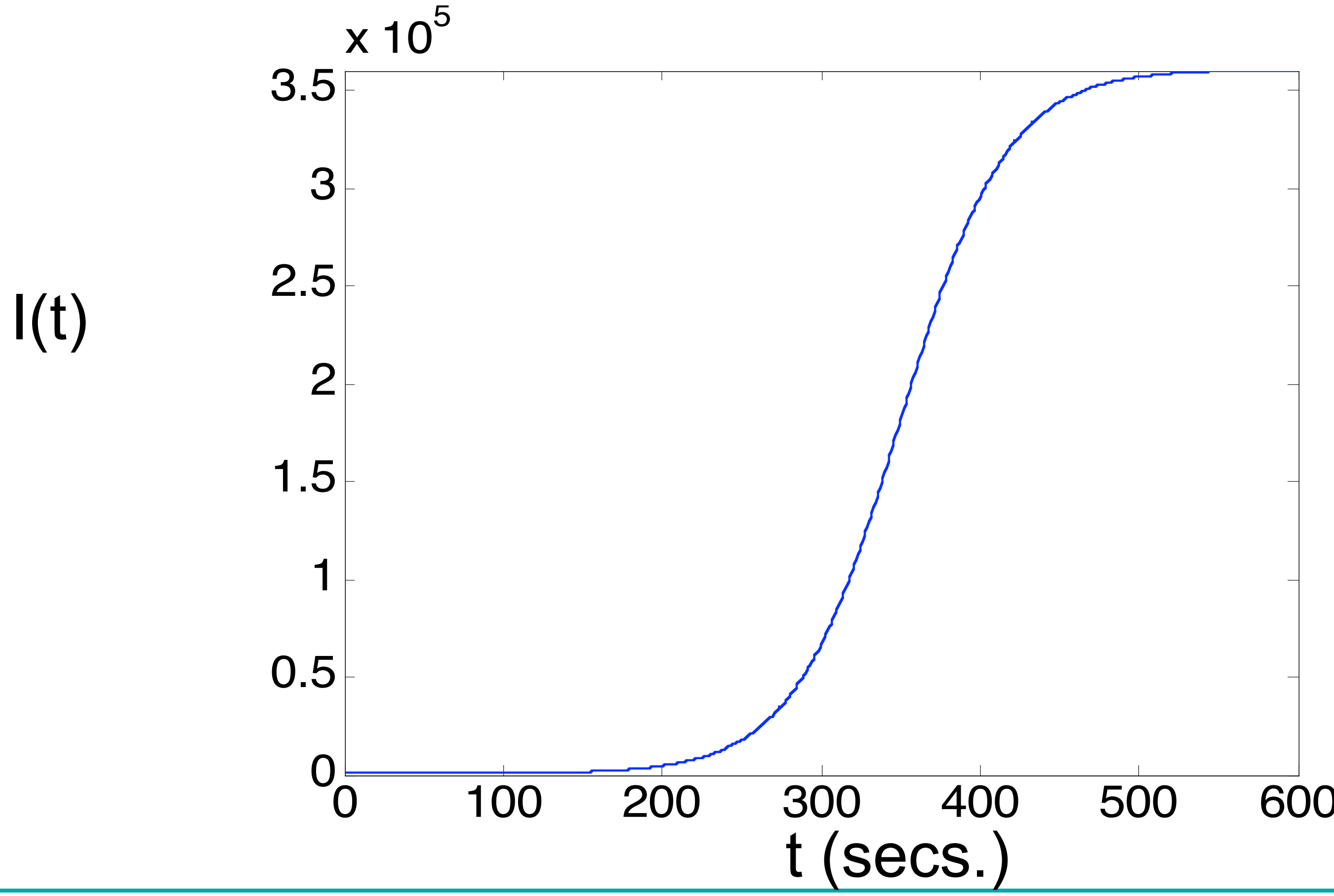
Integrate to get this closed form:

$$i(t) = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}}$$

T = integration constant

Exponential growth, tapers off

- Example curve of I(t) (which is i(t) * N)
- Here, $N = 3.5 \times 10^5$ (β affects steepness of slope)



What can be done?

- Reduce the number of infected hosts
  - **Treatment**, reduce I(t) while I(t) is still small
  - e.g. shut down/repair infected hosts
- Reduce the contact rate
  - **Containment**, reduce ß while I(t) is still small
  - e.g. filter traffic

} Reactive

- Reduce the number of susceptible hosts
  - **Prevention**, reduce S(0)
  - e.g. use type-safe languages

} Proactive

Treatment

- Reduce # of infected hosts

- Disinfect infected hosts
  - Detect infection in real-time
  - Develop specialized “vaccine” in real-time
  - Distribute “patch” more quickly than worm can spread
    - Anti-worm? (CRClean)
    - Bandwidth interference…

Effects of "patching" infected hosts

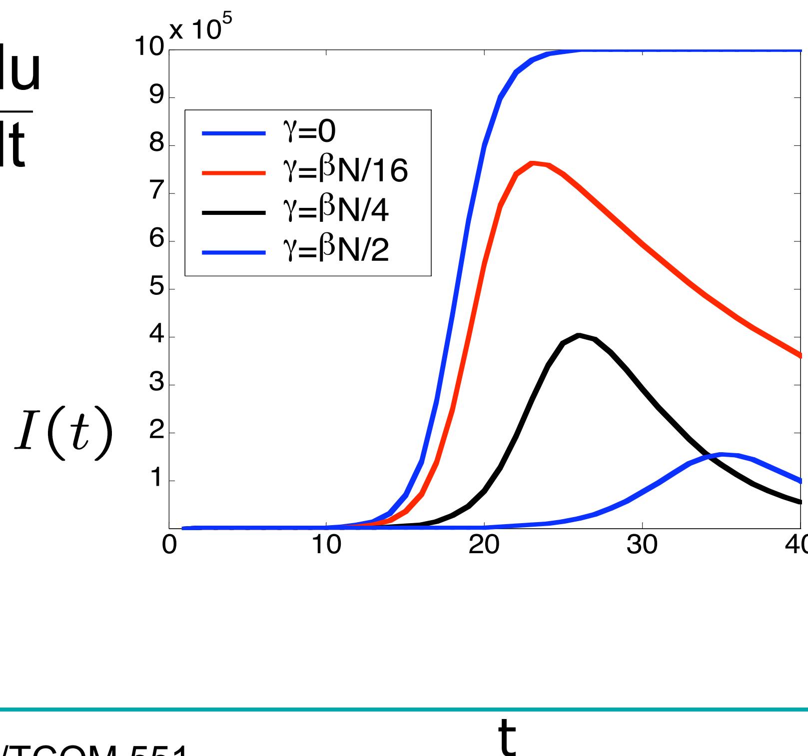
- Kermack-McKendrick Model
- State transition:

U(t) = # of removed from infectious population

γ = removal rate

$$\frac{di}{dt} = \beta * i(t) * (1 - i(t)) - \frac{du}{dt}$$

$$\frac{du}{dt} = \gamma * i(t)$$



Containment

- Reduce contact rate β

- **Oblivious defense**
  - Consume limited worm resources
  - Throttle traffic to slow spread
  - Possibly important capability, but worm still spreads…

- **Targeted defense**
  - Detect and block worm

Design Space

- Design Issues for Reactive Defense
  [Moore et al 03]

- Any reactive defense is defined by:
  - ***Reaction time*** – **how long** to detect, propagate information, and activate response
  - ***Containment strategy*** – **how** malicious behavior is identified and stopped
  - ***Deployment scenario*** - **who** participates in the system

- Savage et al. evaluate the requirements for these parameters to build **any** effective system for worm propagation.

Methodology

- **Moore et al., "Internet Quarantine:…" paper**
- **Simulate spread of worm across Internet topology:**
  - infected hosts *attempt* to spread at a fixed rate (probes/sec)
  - target selection is uniformly random over IPv4 space
- **Simulation of defense:**
  - system detects infection within reaction time
  - subset of network nodes employ a containment strategy
- **Evaluation metric:**
  - % of vulnerable hosts infected in 24 hours
  - 100 runs of each set of parameters (95^{th} percentile taken)
    - Systems must plan for reasonable situations, **not** the average case
- **Source data:**
  - vulnerable hosts: 359,000 IP addresses of CodeRed v2 *victims*
  - Internet topology: AS routing topology derived from RouteViews

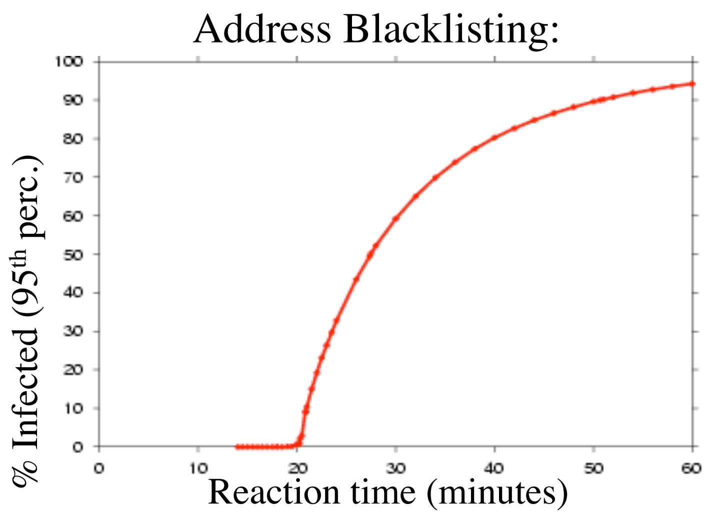
Initial Approach: Universal Deployment

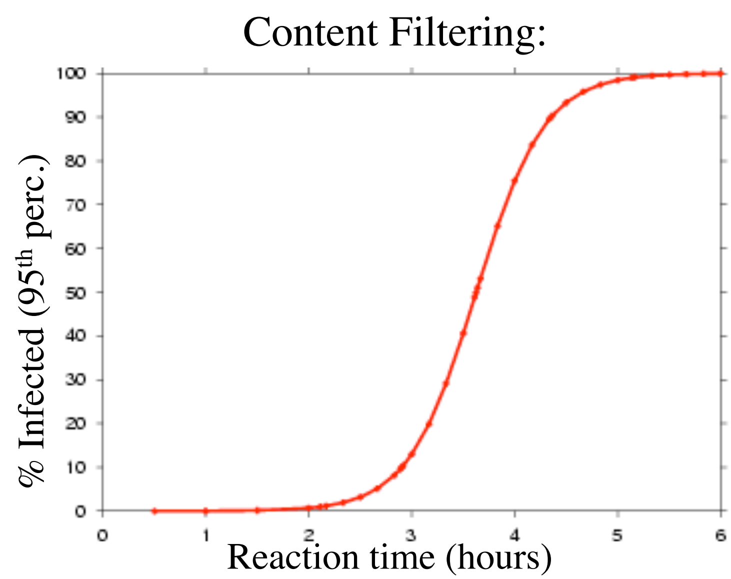
- Assume **every host** employs the containment strategy

- Two containment strategies they tested:
  - ***Address blacklisting:***
    - block traffic from malicious source IP addresses
    - reaction time is relative to each infected host
  - ***Content filtering:***
    - block traffic based on signature of content
    - reaction time is from first infection

- How quickly does each strategy need to react?
- How sensitive is reaction time to worm probe rate?

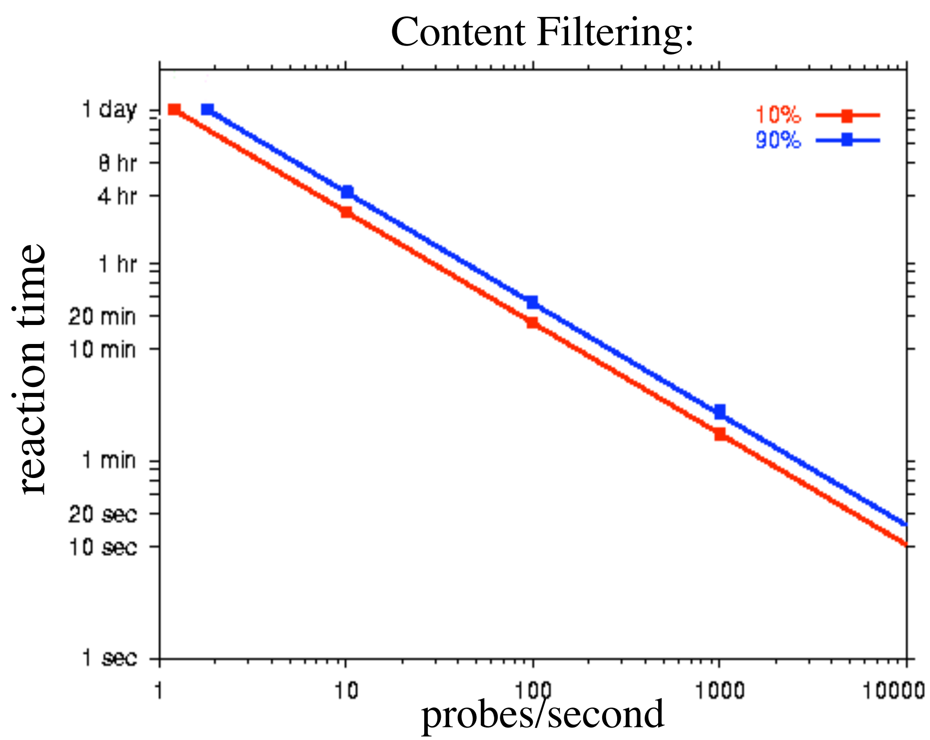
Reaction times?





- To contain worms to 10% of vulnerable hosts after 24 hours of spreading at 10 probes/sec (CodeRed):
  - Address blacklisting: reaction time must be < 25 minutes.
  - Content filtering: reaction time must be < 3 hours

Probe rate vs. Reaction Time



- Reaction times must be fast when probe rates get high:
  - 10 probes/sec: reaction time must be < 3 hours
  - 1000 probes/sec: reaction time must be < 2 minutes

Limited Network Deployment

- Depending on <u>every</u> **host** to implement containment is not feasible:
  - installation and administration costs
  - system communication overhead

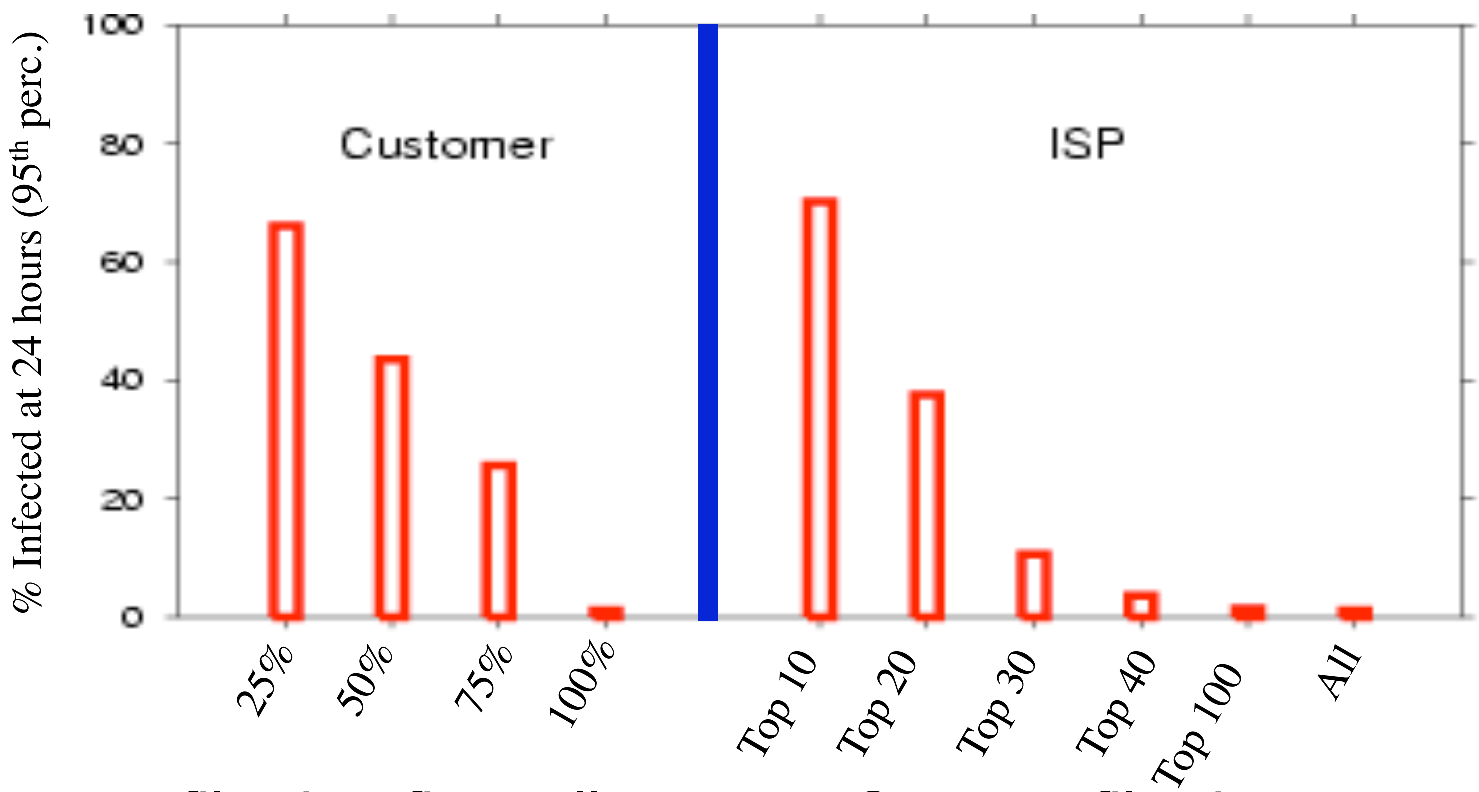
- A more realistic scenario is <u>limited</u> deployment in the **network**:
  - Customer Network: firewall-like inbound filtering of traffic
  - ISP Network: traffic through border routers of large transit ISPs

- How effective are the deployment scenarios?
- How sensitive is reaction time to worm probe rate under limited network deployment?

Deployment Scenario Effectiveness?

Reaction time = 2 hours

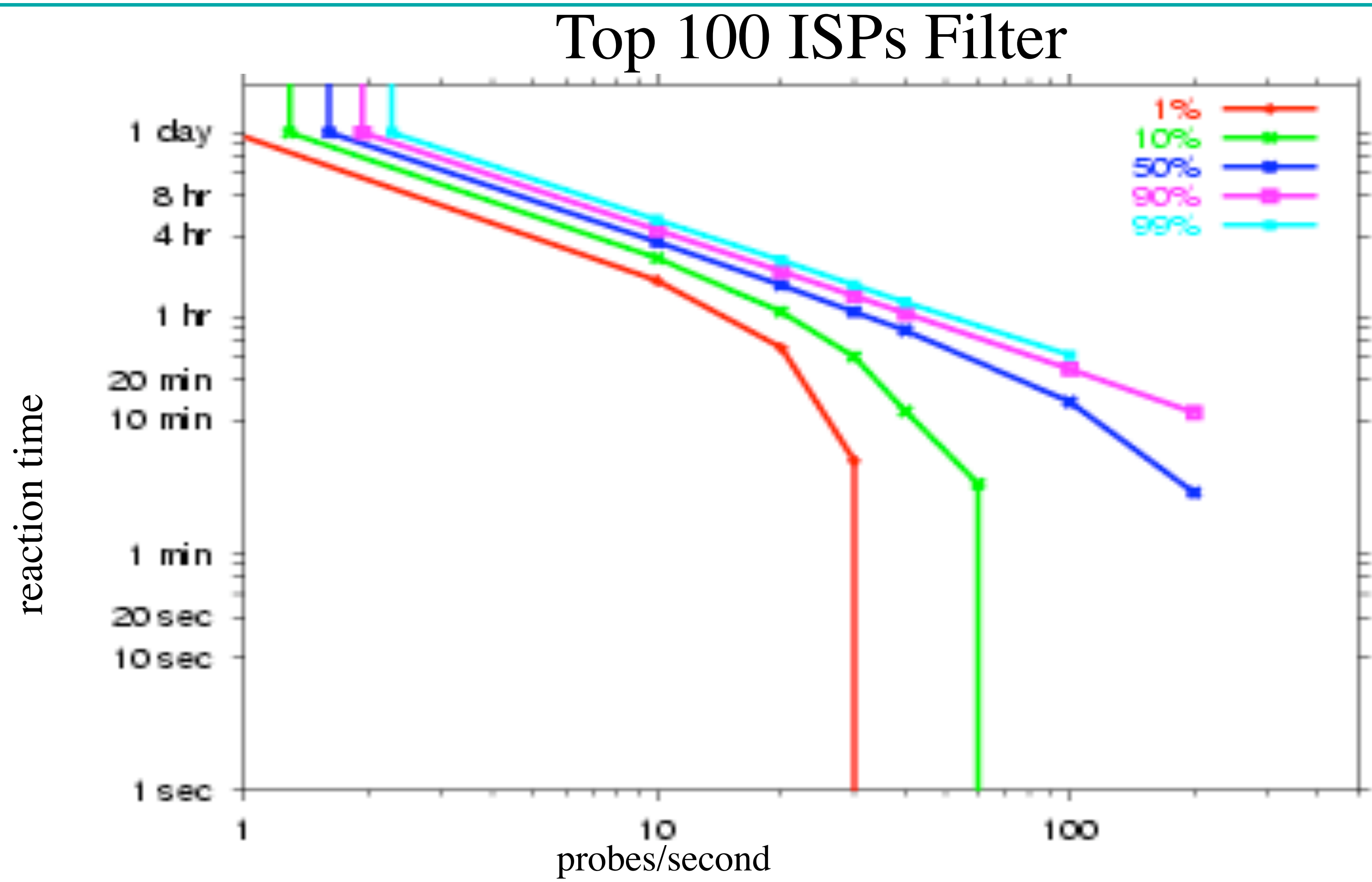
CodeRed-like Worm:



Content filtering firewalls at edge of customer nets.

Content filtering at exchange points in major ISPs.

Reaction Time vs. Probe Rate (II)



- Above 60 probes/sec, containment to 10% hosts within 24 hours is impossible even with *instantaneous* reaction.

Summary: Reactive Defense

- Reaction time:
  - required reaction times are a couple minutes or less (far less for bandwidth-limited scanners)
- Containment strategy:
  - content filtering is more effective than address blacklisting
- Deployment scenarios:
  - need nearly all customer networks to provide containment
  - need at least top 40 ISPs provide containment

Mechanisms to Mitigate Malware

- Network-level defenses:
  - Firewalls
  - Intrusion Detection Systems
  - Content filtering

Next several lectures: networks & network security.

- OS-level defenses:
  - Access controls
  - Authorization
- Software-level defenses:
  - Type safe languages
  - Program verification
  - Software certification

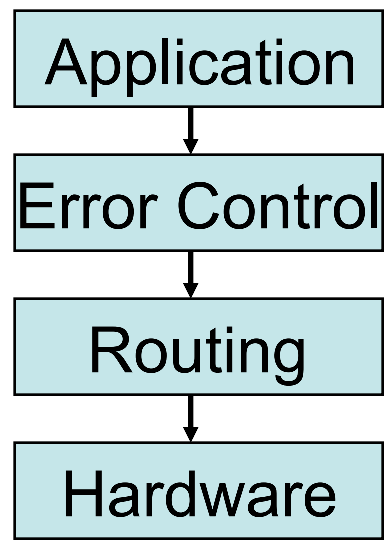
Network Architecture

- General blueprints that guide the design and implementation of networks
- Goal: to deal with the complex requirements of a network
- Use *abstraction* to separate concerns
  - Identify the useful service
  - Specify the interface
  - Hide the implementation

Layering

- A result of abstraction in network design
  - A stack of services (layers)
  - Hardware service at the bottom layer
  - Higher level services are implemented by using services at lower levels

- Advantages
  - Decompose problems
  - Modular changes

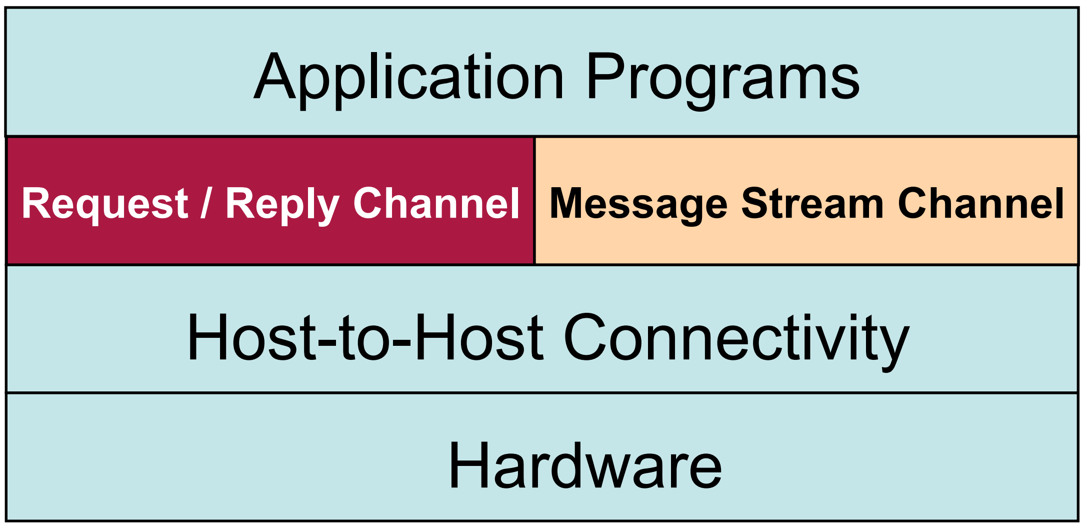


Protocols

- A *protocol* is a specification of an interface between modules (often on different machines)

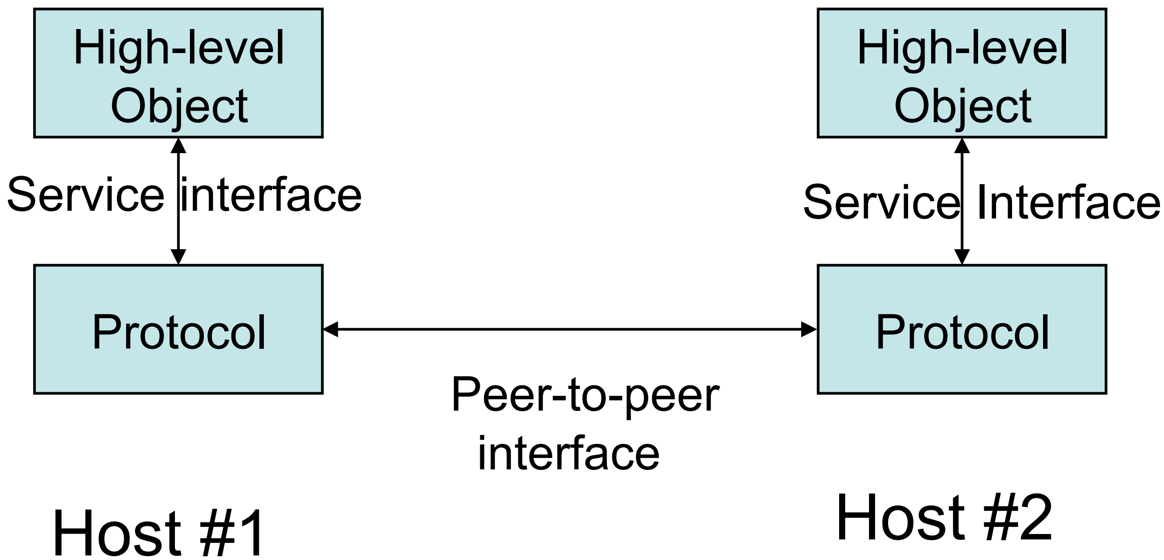
- Sometimes “protocol” is used to mean the implementation of the specification.

Example Protocol Stack

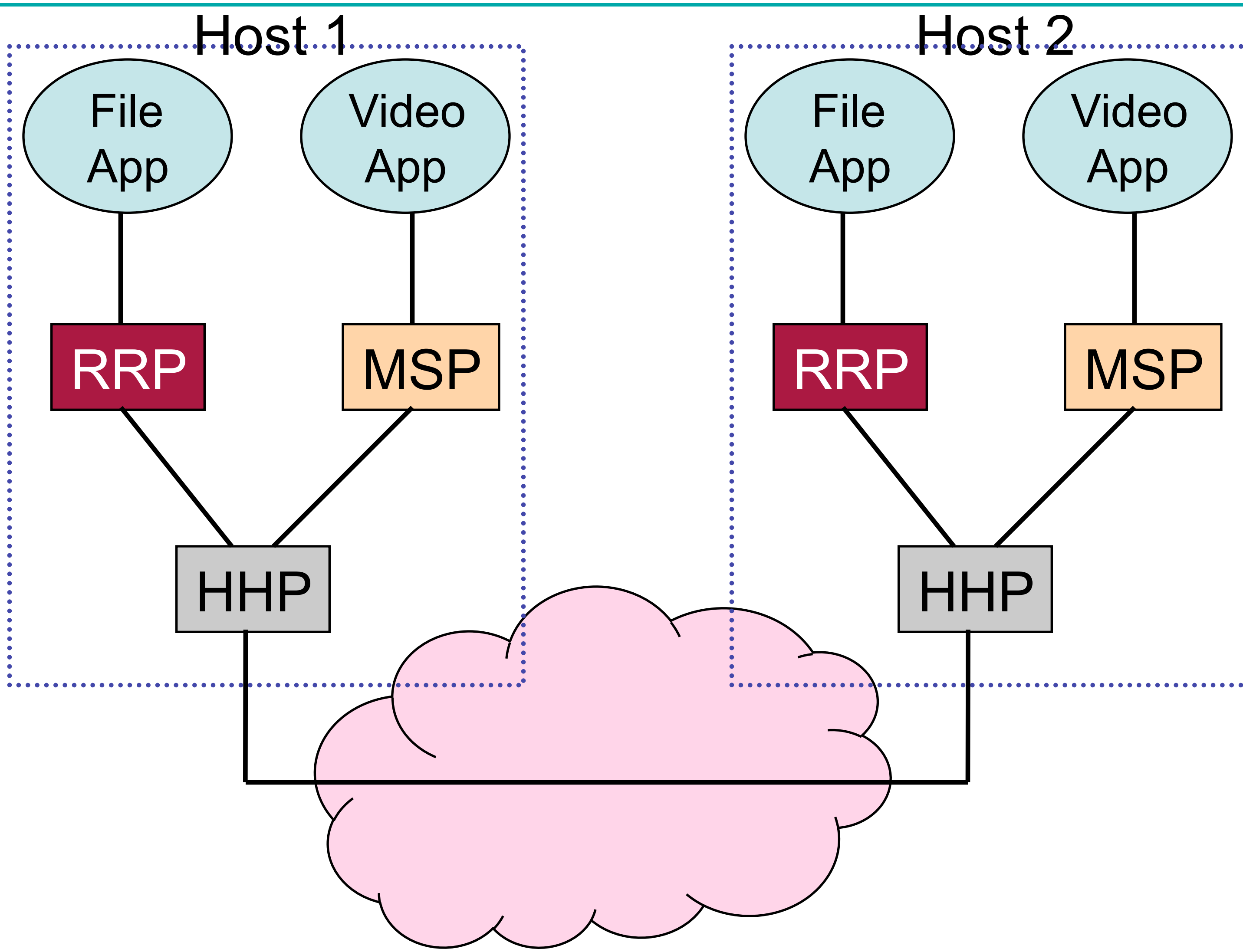


Protocol Interfaces

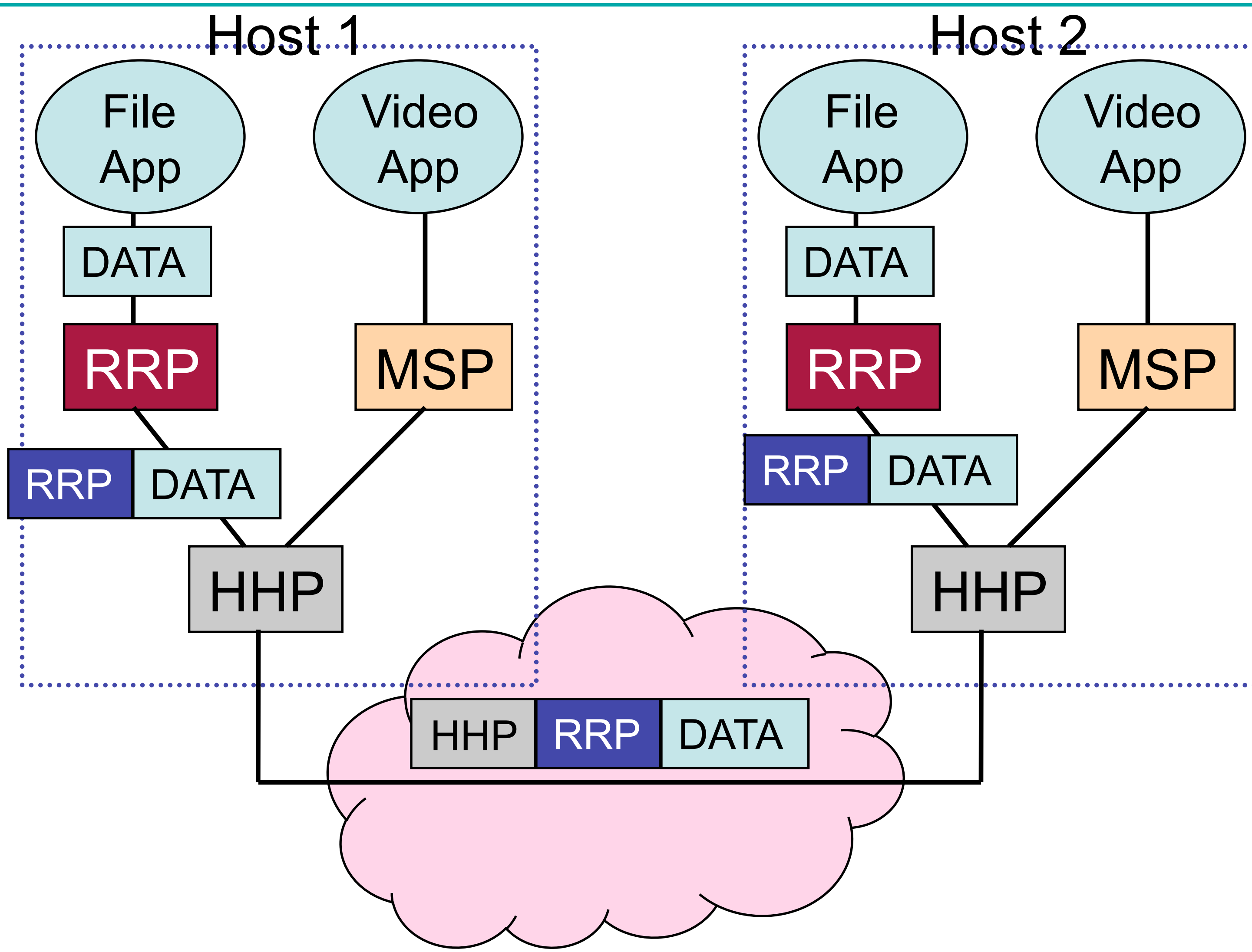
- Service Interfaces
  - Communicate up and down the stack
- Peer Interfaces
  - Communicate to counterpart on another host



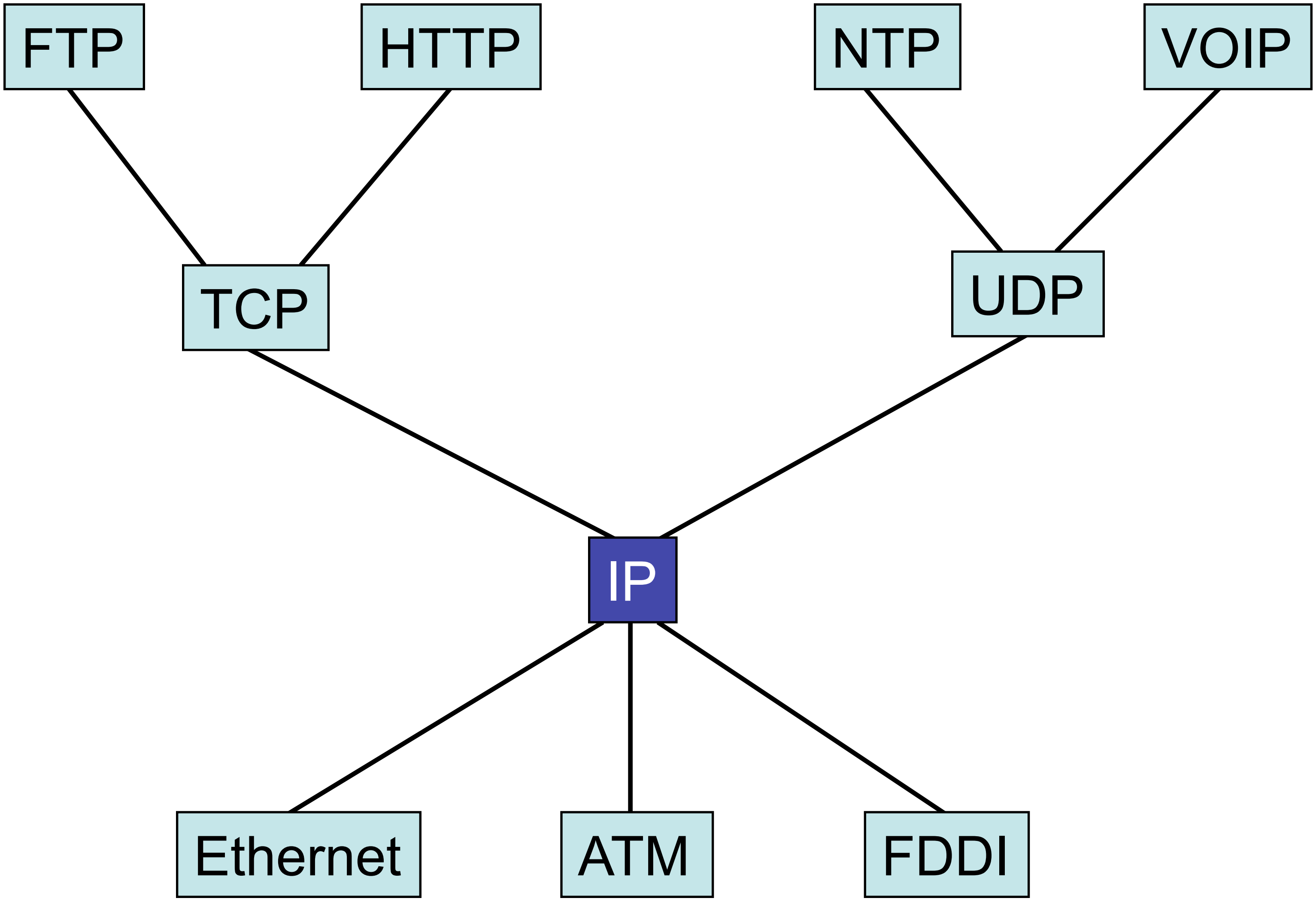
Example Protocol Graph



Encapsulation



Internet Protocol Graph



Open Systems Interconnection (OSI)

End Host	Reference model – not actual implementation.
Application	Transmits *messages* (e.g. FTP or HTTP)
Presentation	Data format issues (e.g. big- vs. little-endian)
Session	Manages multiple streams of data
Transport	Process to process protocols
Network	Routes *packets* among nodes in network
Data Link	Packages bit streams into *frames*
Physical	Transmits raw bits over link