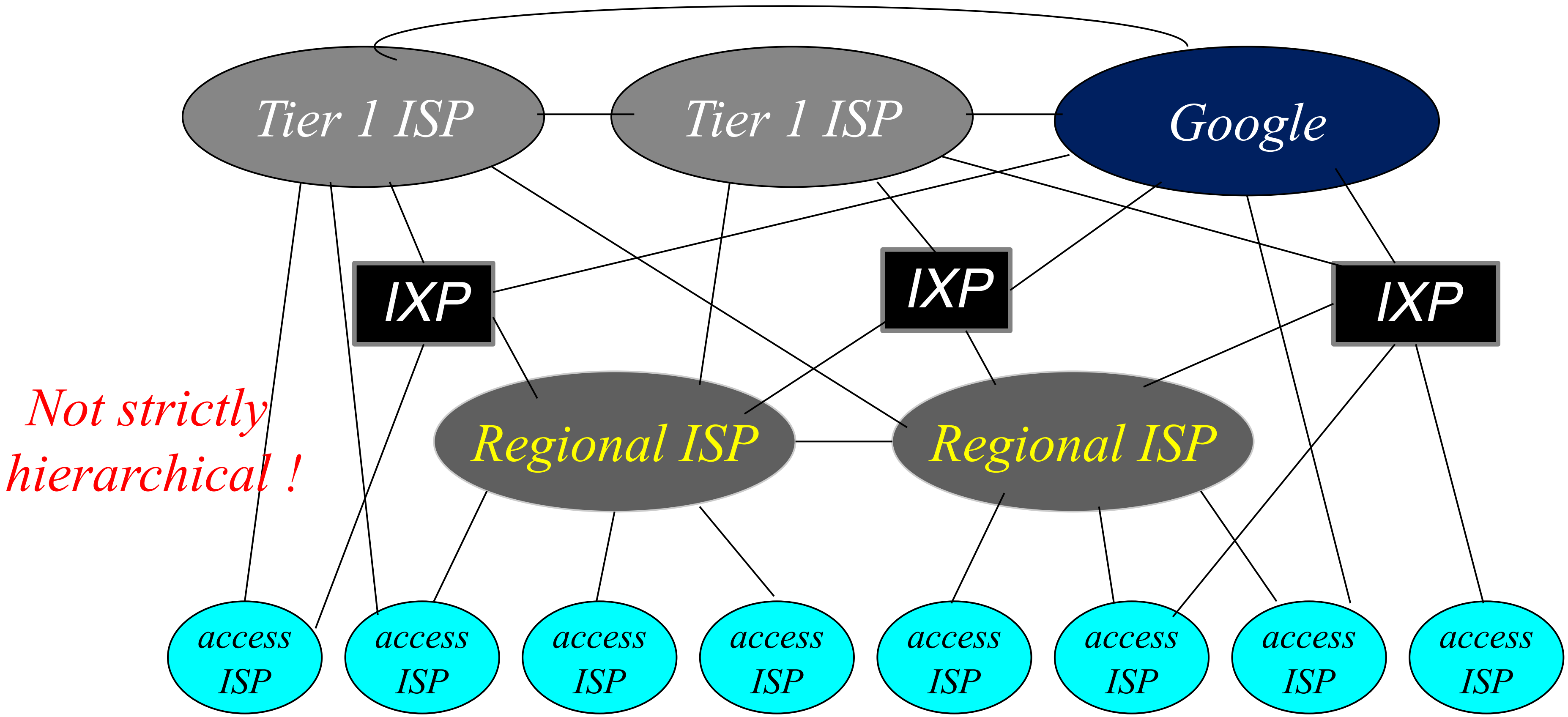
Network Security Basics

Acknowledgements

- Some of the slides used in this course are adapted from the following sources with permission:
  - “An Engineering Approach to Computer Networking” by S. Keshav http://www.cs.cornell.edu/skeshav/book/slides/index.html
  - “Computer Networking: A Top-down approach featuring the Internet” by James F. Kurose and Keith W. Ross 6th Edition, all material copyright 1996-2013, J.F Kurose and K.W. Ross, All Rights Reserved.
  - Nick McKeown, seminar notes, Stanford University.
  - “Computer Networks -- A system approach” by Peterson and Davie
-

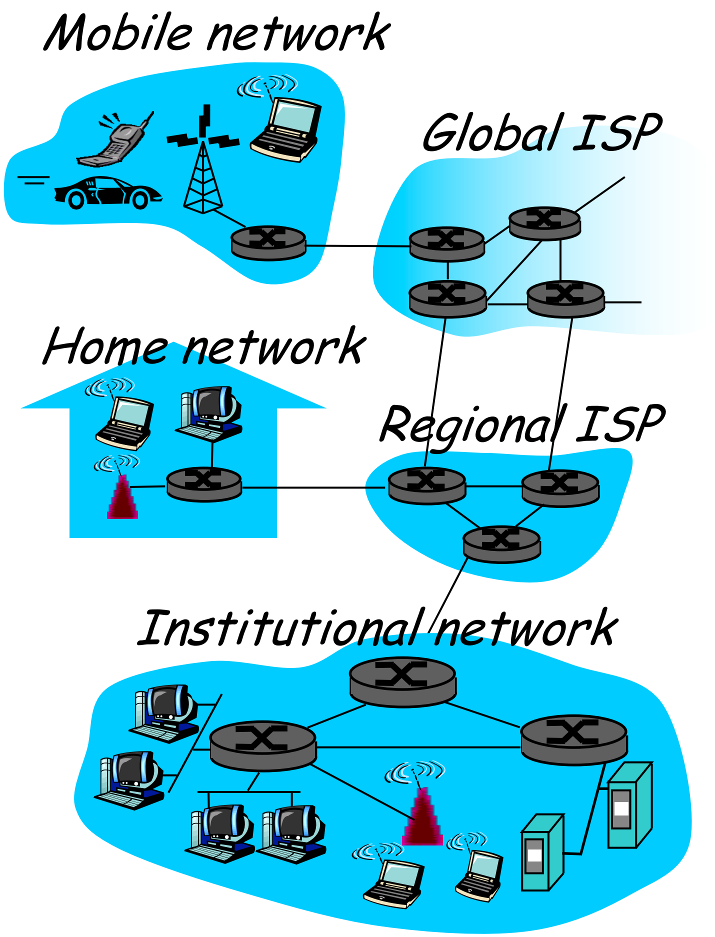
Internet structure: network of networks



- At core: small # of well-connected large "Default-free" networks
  - "Tier-1" commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
  - Content Provider Network (e.g, Google): private network that connects its data centers to Internet, often bypassing tier-1, regional ISPs

What’s the Internet: “nuts and bolts” view

- *Internet:* “network of networks”
  - loosely hierarchical
  - public Internet versus private intranet
- *protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, Ethernet
- Internet standards
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



What does a protocol tell us?

- *Syntax* of a message
  - what fields does it contain?
  - in what format?
- *Semantics* of a message
  - what does a message mean?
  - for example, not-OK message means receiver got a corrupted file
- *Actions* to take on receipt of a message
  - for example, on receiving not-OK message, retransmit the entire file

Protocol layering

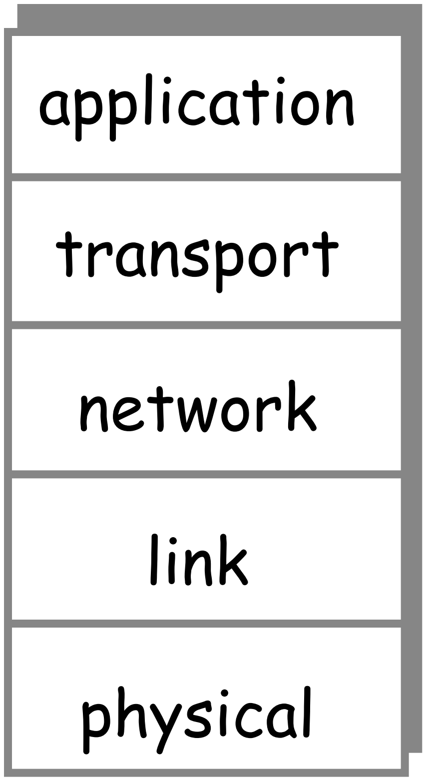
- A network that provides many services needs many protocols
- Turns out that some services are independent
- But others depend on each other
- Protocol A may use protocol B as a *step* in its execution
  - Example 1: the reliable transfer of a file is one step in downloading a webpage which may involve reliable transfer of multiple files
  - Example 2: sending a packet from the source to the destination host is one step in the execution of the example reliable file transfer protocol
  - Example 3: to a send packet along a path consisting of multiple routers from a source and a destination requires packet transfer between directly connected routers along that path.
- This form of dependency is called *layering*
  - reliable file transfer is *layered* above packet transfer protocol
  - network-wide packet delivery is *layered* above packet transfer over a direct link between neighboring routers
  - It is like calling a subroutine/function within a program

Protocol stack

- A set of protocol layers
- Each layer uses the layer below and provides a service to the layer above
- Key idea
  - once we define a service provided by a layer, we DO NOT need to know the details of *how* the layer actually implements the service
  - information hiding
  - decouples changes in different layers

Layering of Internet protocols (aka the protocol stack)

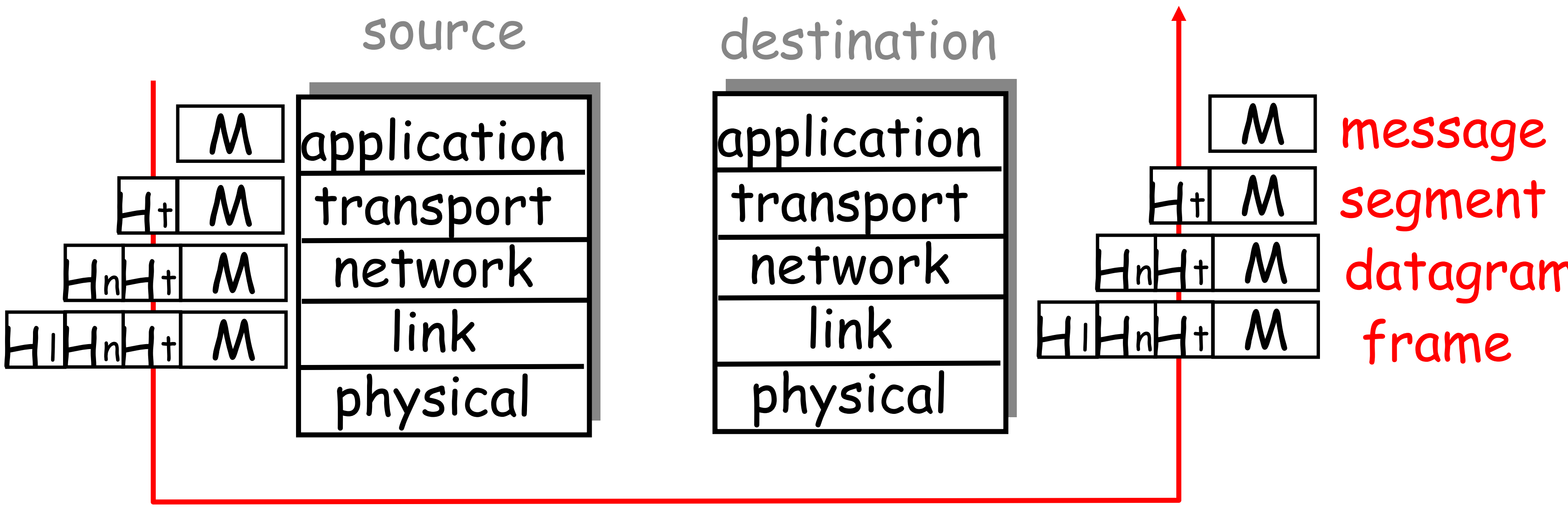
- application: supporting network applications
  - FTP, SMTP, HTTP, TELNET
- transport: process-to-process data transfer (process = a program currently being run on an end-host)
  - TCP, UDP
- network: routing of packets from source node to destination node
  - IP, routing protocols e.g. OSPF, BGP
- link: data transfer between neighboring network nodes (usually directly connected through a communication link)
  - PPP, Ethernet
- physical: representing information bits in form of electromagnetic signals on the wire, fiber or in air.



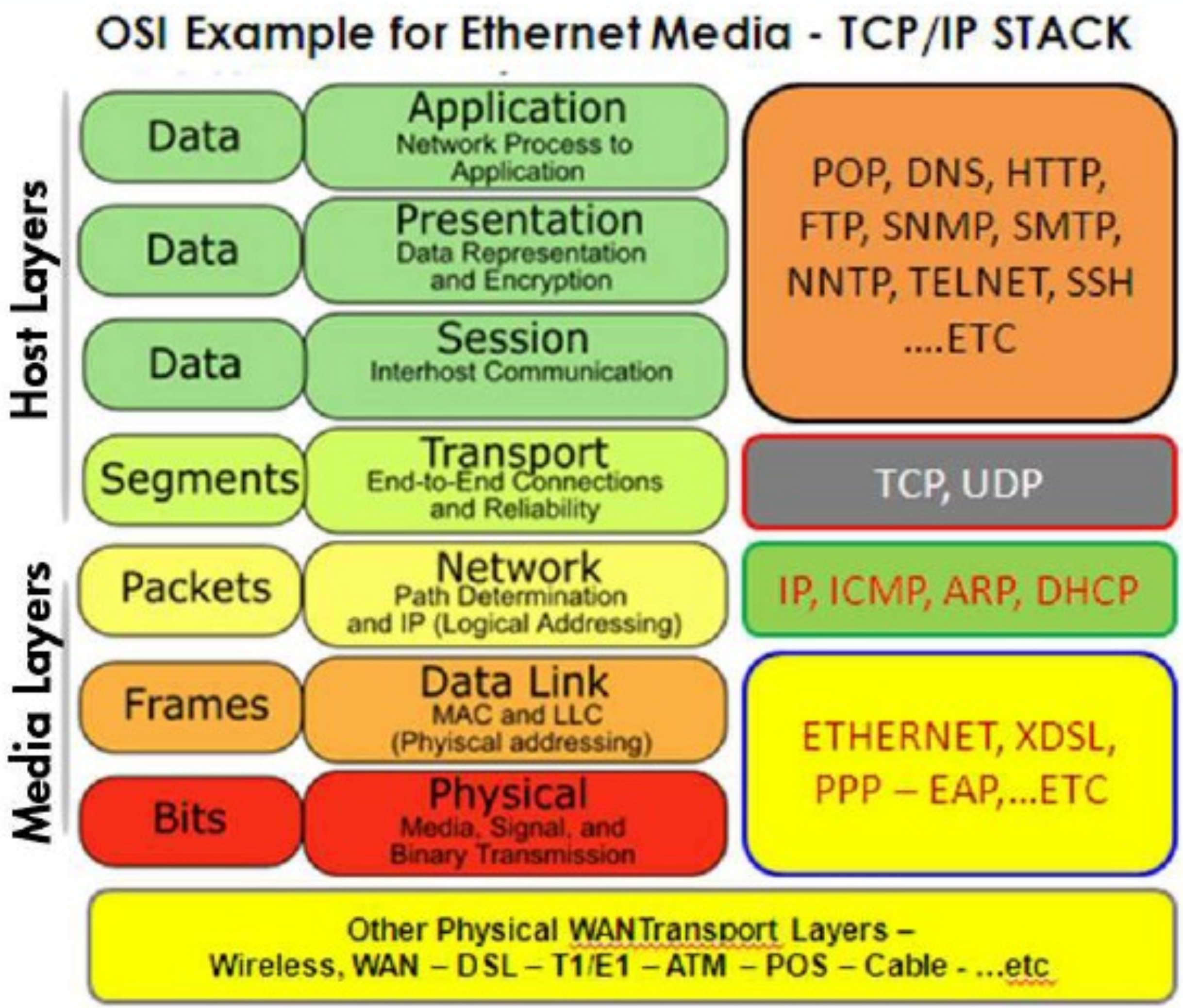
Protocol layering and data

Each layer takes data from above

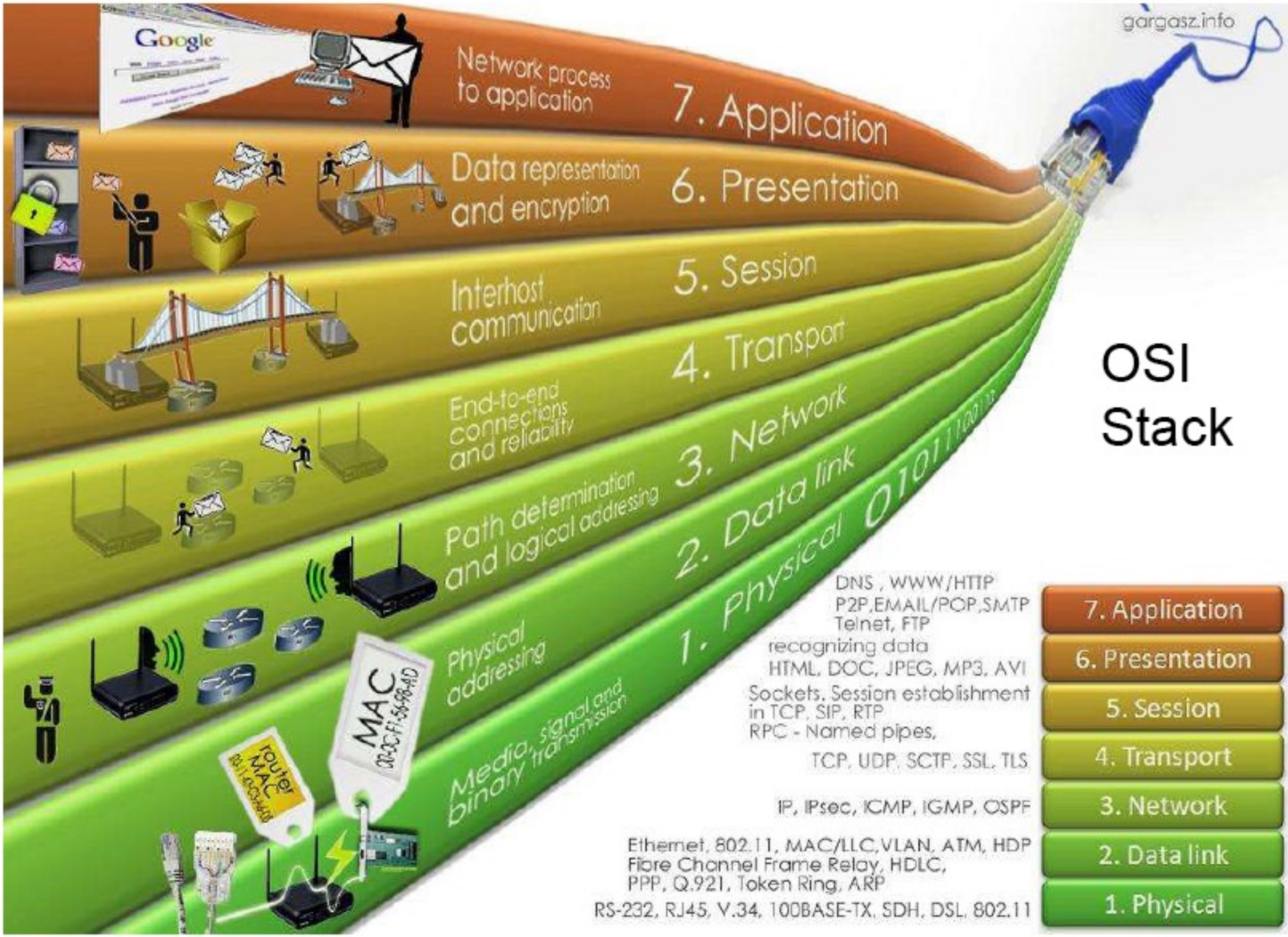
- adds header information to create new data unit
- passes new data unit to layer below



Layered Design

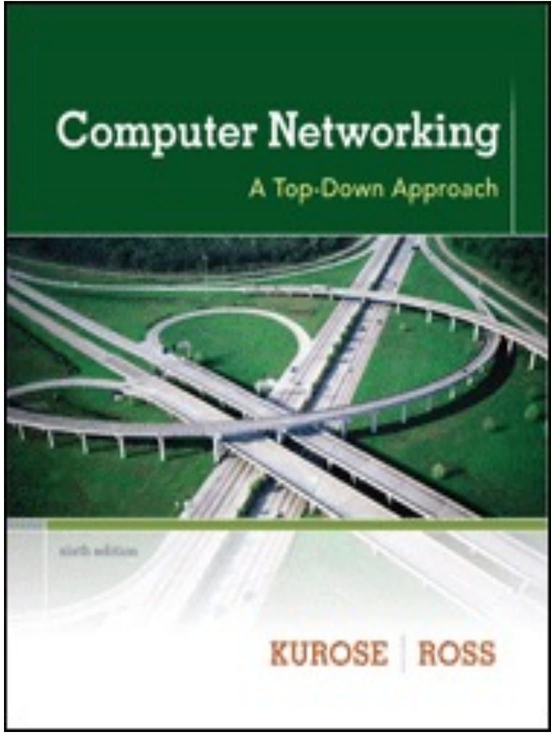


Source: http://blog.anuesystems.com/category/span-and-taps/



Source: http://walkwidnetwork.blogspot.com/2013/04/physical-layer-osi-model.html

A day in the life of a web request



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

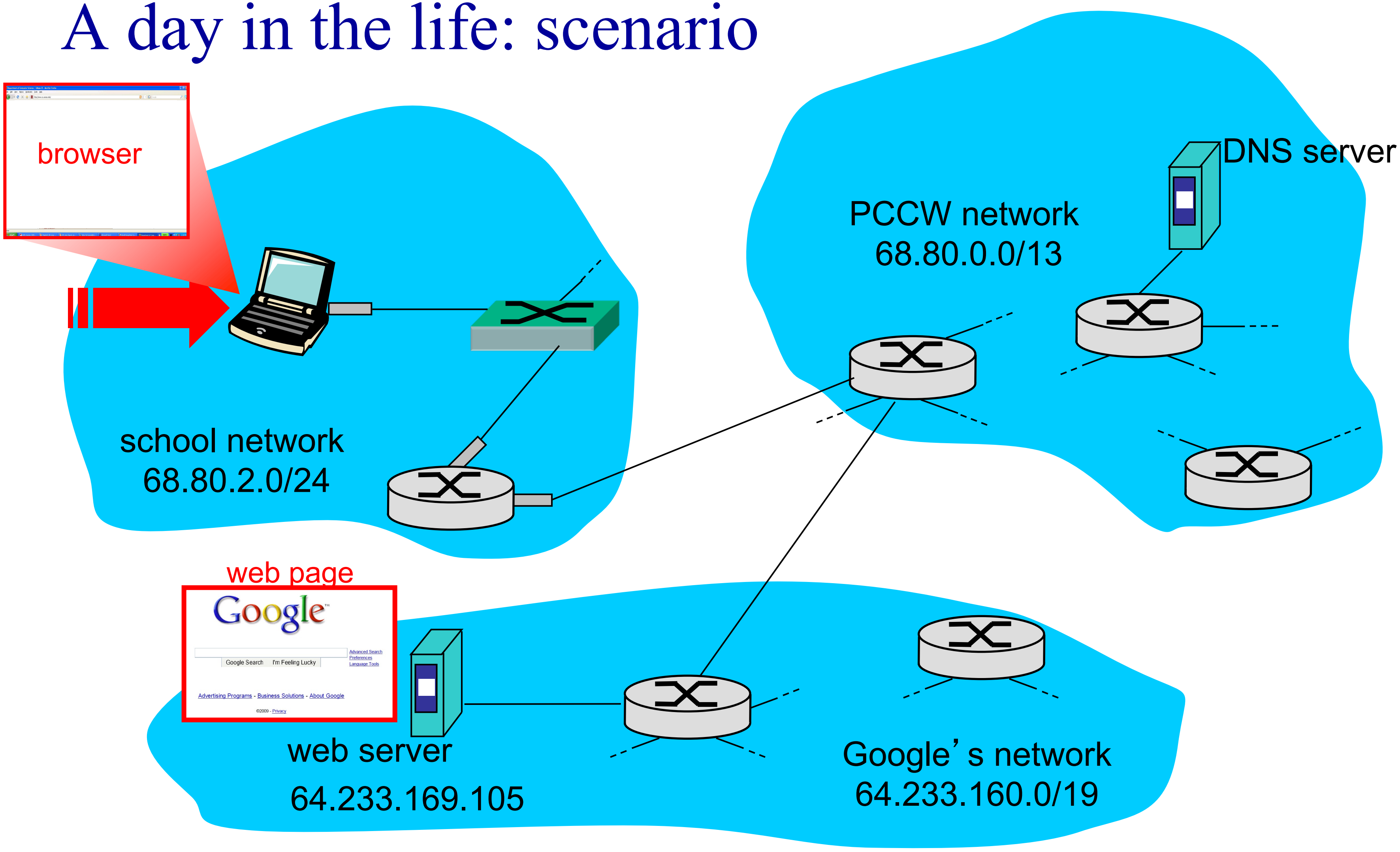
Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
6th edition.
Jim Kurose, Keith Ross
Addison-Wesley, 2012.

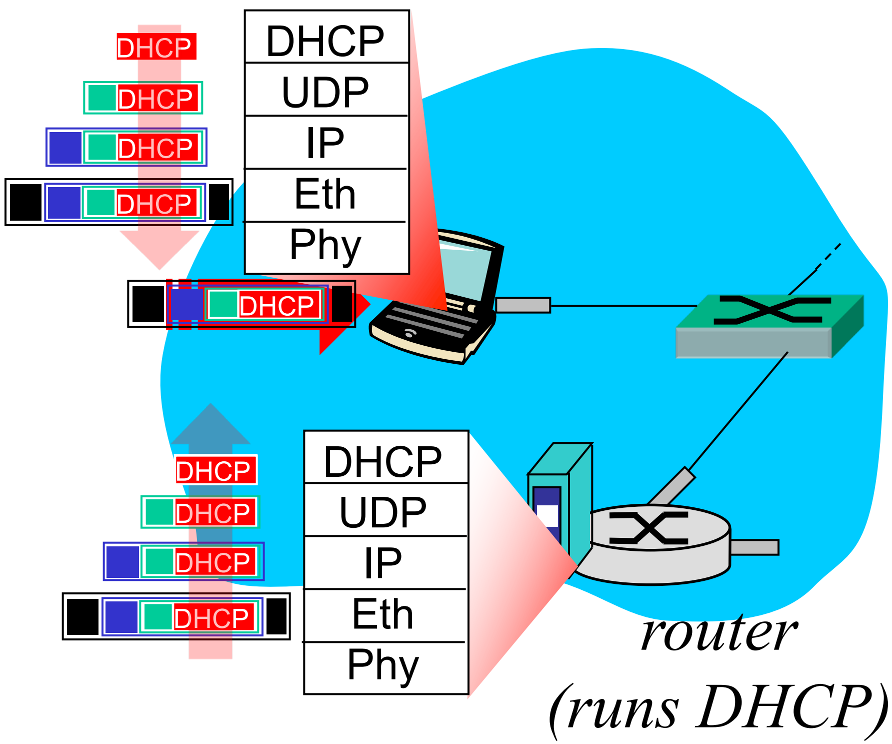
Synthesis: a day in the life of a web request

- journey down protocol stack complete!
  - application, transport, network, link
- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

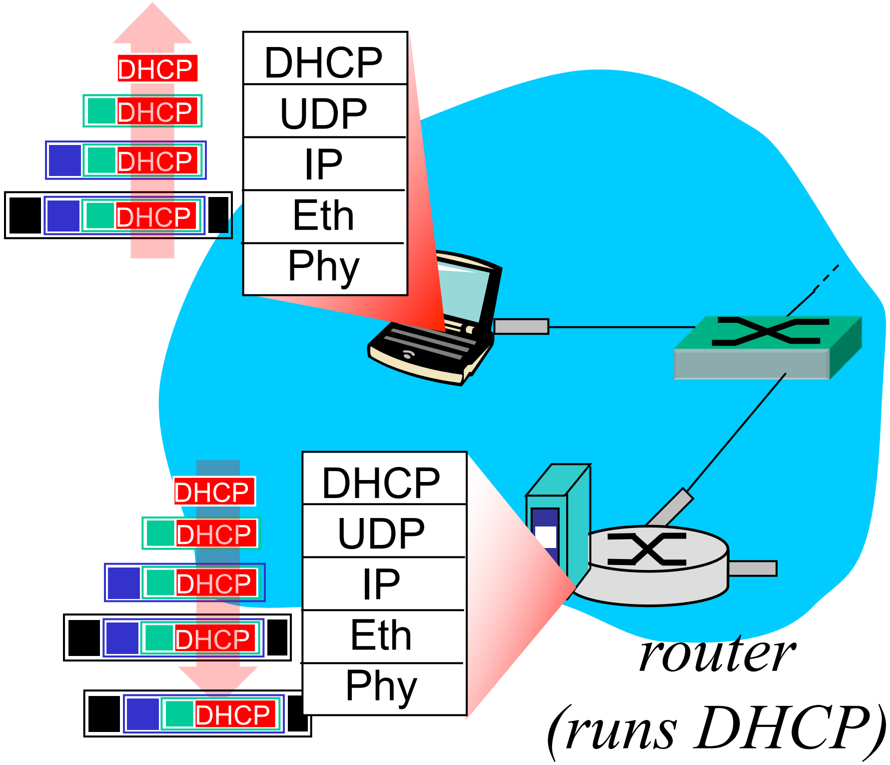


A day in the life… connecting to the Internet



- Connecting laptop needs to get its own *IP* address, addr of first-hop router, addr of *DNS* server: use *DHCP*
- DHCP request *encapsulated* in ***UDP***, encapsulated in ***IP***, encapsulated in ***802.1 Ethernet***
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received at router running ***DHCP*** server
- Ethernet *demuxed* to IP demuxed, UDP demuxed to DHCP

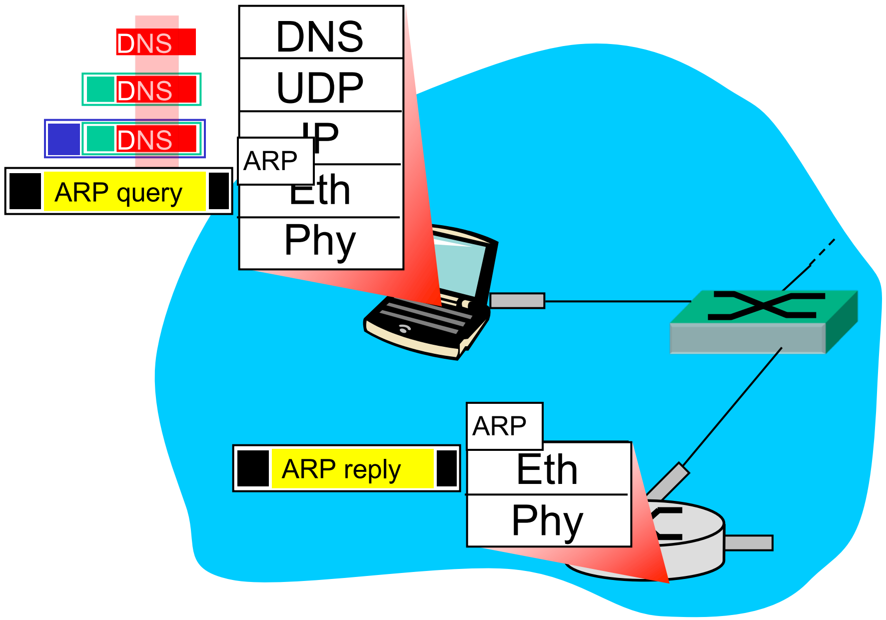
A day in the life… connecting to the Internet



- DHCP server formulates ***DHCP ACK*** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- Encapsulation at DHCP server, frame forwarded (***switch learning***) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

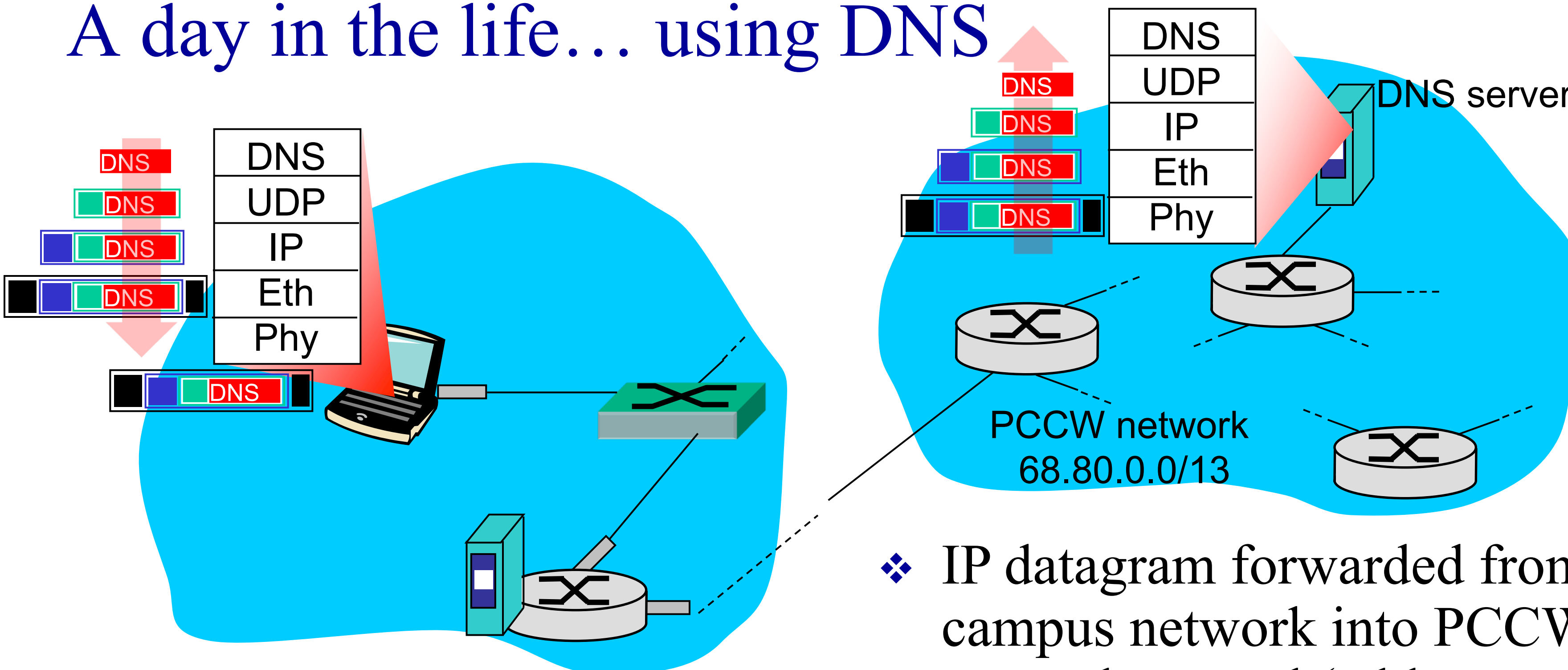
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



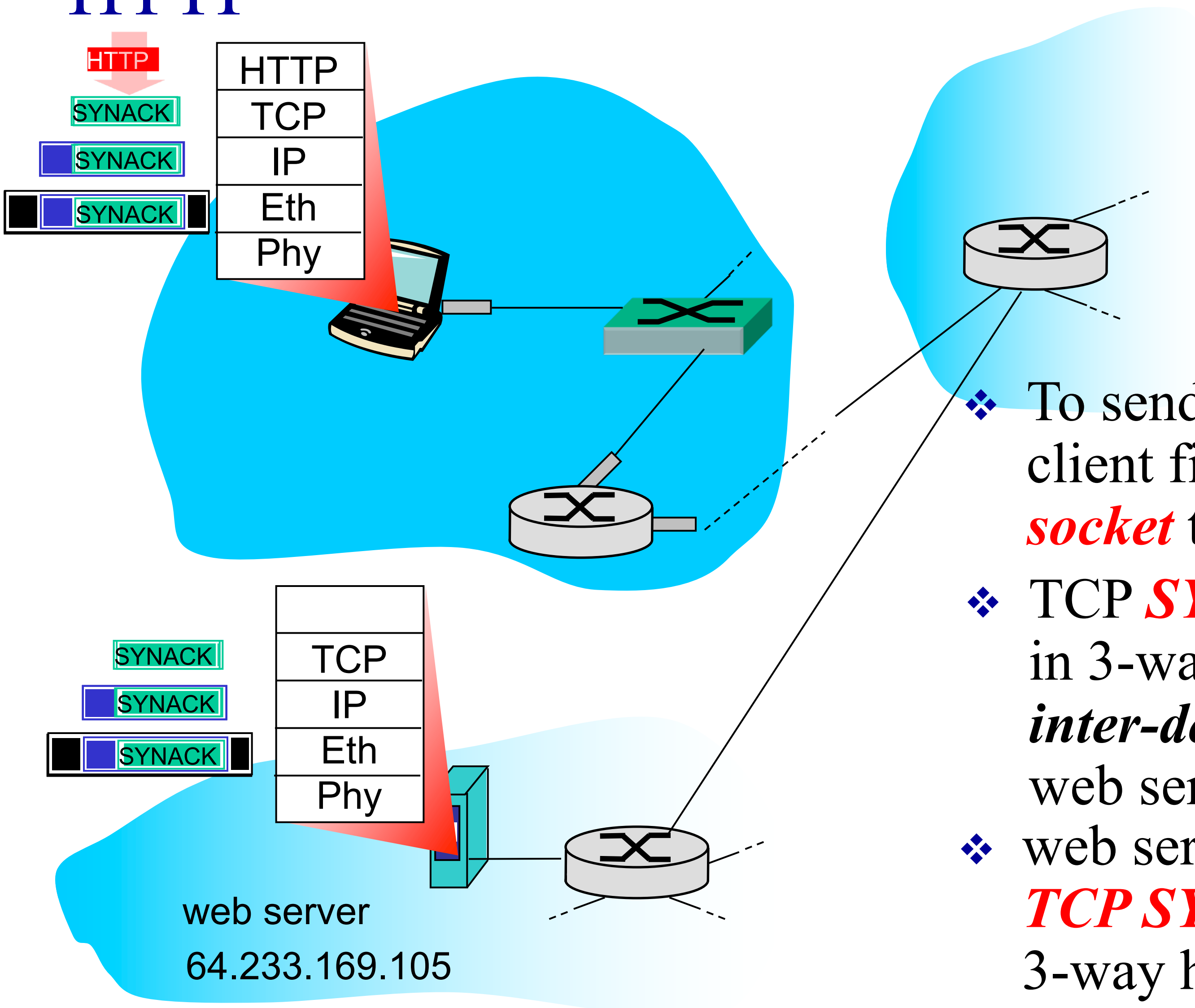
- Before sending ***HTTP*** request, need IP address of www.google.com: Use ***DNS***
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. In order to send frame to router, need MAC address of router interface: Use ***ARP***
- ***ARP query*** broadcast, received by router, which replies with ***ARP reply*** giving MAC address of router interface
- Client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS



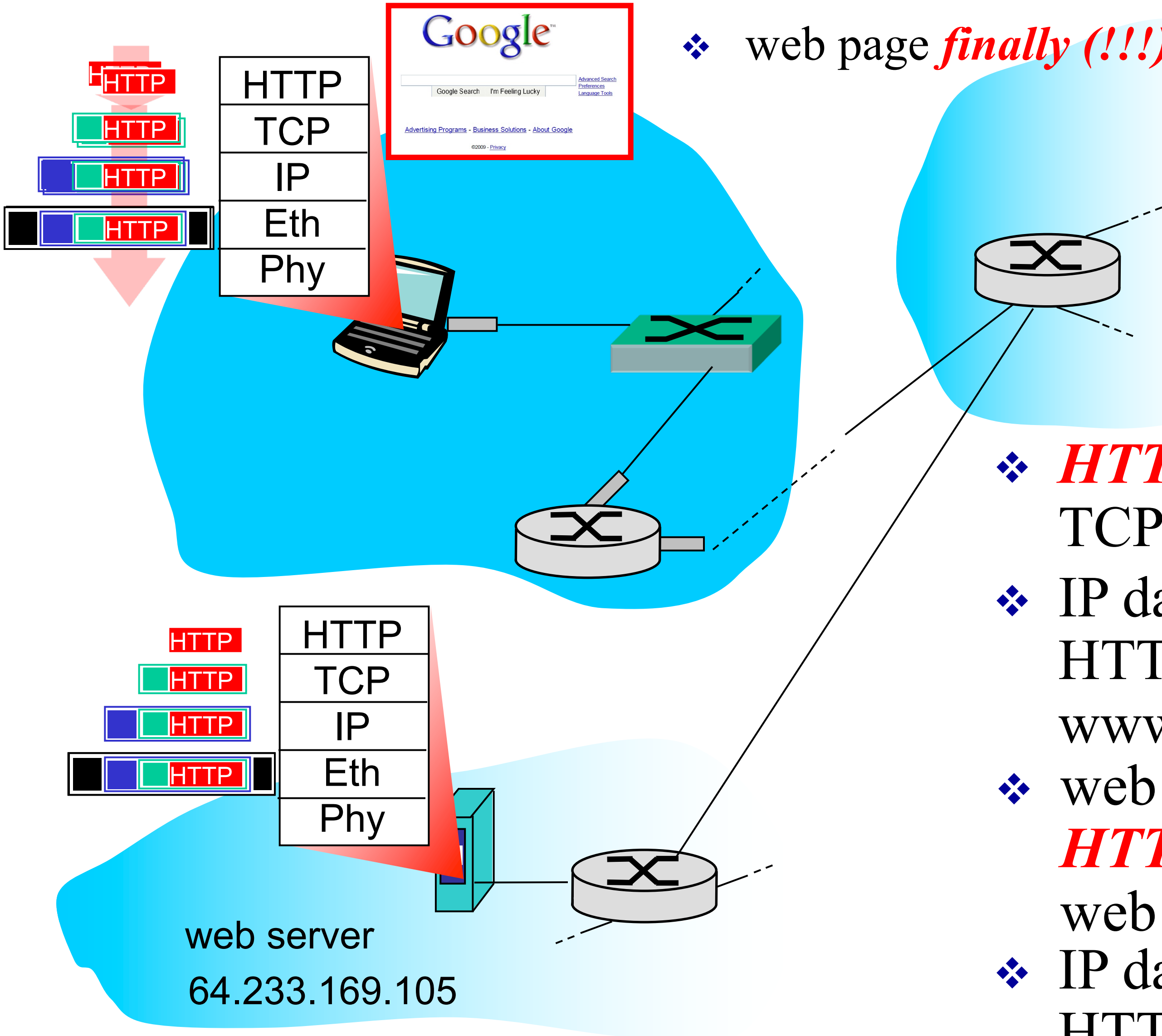
- ❖ IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- ❖ IP datagram forwarded from campus network into PCCW network, routed (tables created by ***RIP, OSPF, IS-IS*** and/or ***BGP*** routing protocols) to DNS server
- ❖ Demuxed to DNS server
- ❖ DNS server replies to client with IP address of www.google.com

A day in the life… TCP connection carrying HTTP



- To send HTTP request, client first opens ***TCP socket*** to web server
- TCP ***SYN segment*** (step 1 in 3-way handshake) ***inter-domain routed*** to web server
- web server responds with ***TCP SYNACK*** (step 2 in 3-way handshake)
- TCP ***connection established!***

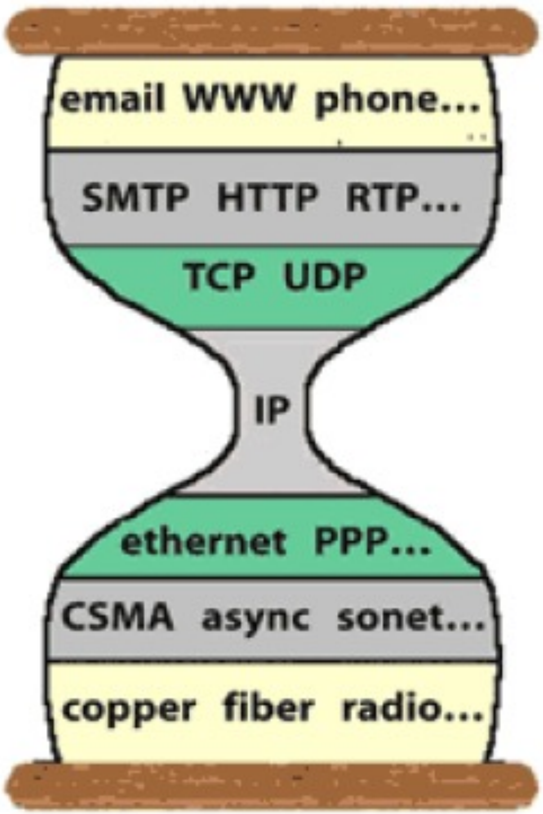
A day in the life… HTTP request/reply



- web page *finally (!!!)* displayed
- ***HTTP request*** sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with ***HTTP reply*** (containing web page)
- IP datagram containing HTTP reply routed back to client

Hourglass Architecture

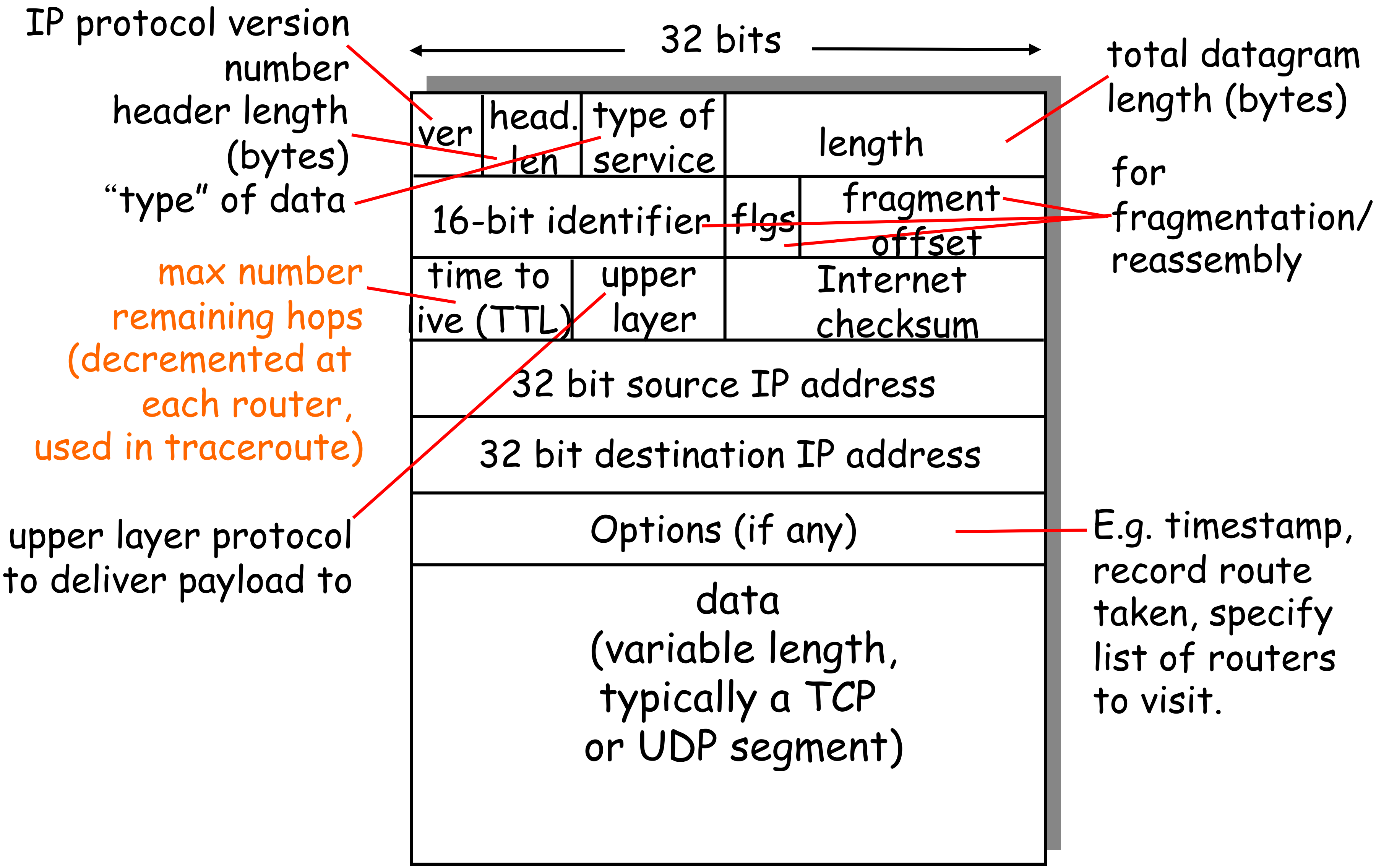
- Why internet layer?
  - Make a big network
  - Global addressing
  - Virtualized network to isolate protocols from network details
- Why only IP?
  - Maximum interoperability
  - Minimize no of service interfaces
- Why IP is so narrow?
  - Assumes least common network functionality → many networks



Steve Deering's hourglass showing "waist" of the Internet

Source: http://isoc.org/wp/ietfjournal/?p=454

Review on IP datagram format



Review on Naming and Addressing and Translation Mechanisms

- Given the Domain-name of the destination host, e.g. www.ie.cuhk.edu.hk, the end-host ask a Domain Name System (DNS) server to translate it to an IP address, e.g. 137.189.96.168
- This IP address is put into the destination IP address field of the packets sent to the destination host ; this destination IP address is used for routing table lookup along the path
- When the packet arrives at the "destination network", i.e. the LAN segment connected to the destination host, the last-hop router use the Address Request Protocol (ARP) to translate the destination IP address to the MAC address of the ethernet network interface card (NIC) on the destination host
- This MAC address is used as the destination address to deliver the Ethernet frame to the destination host

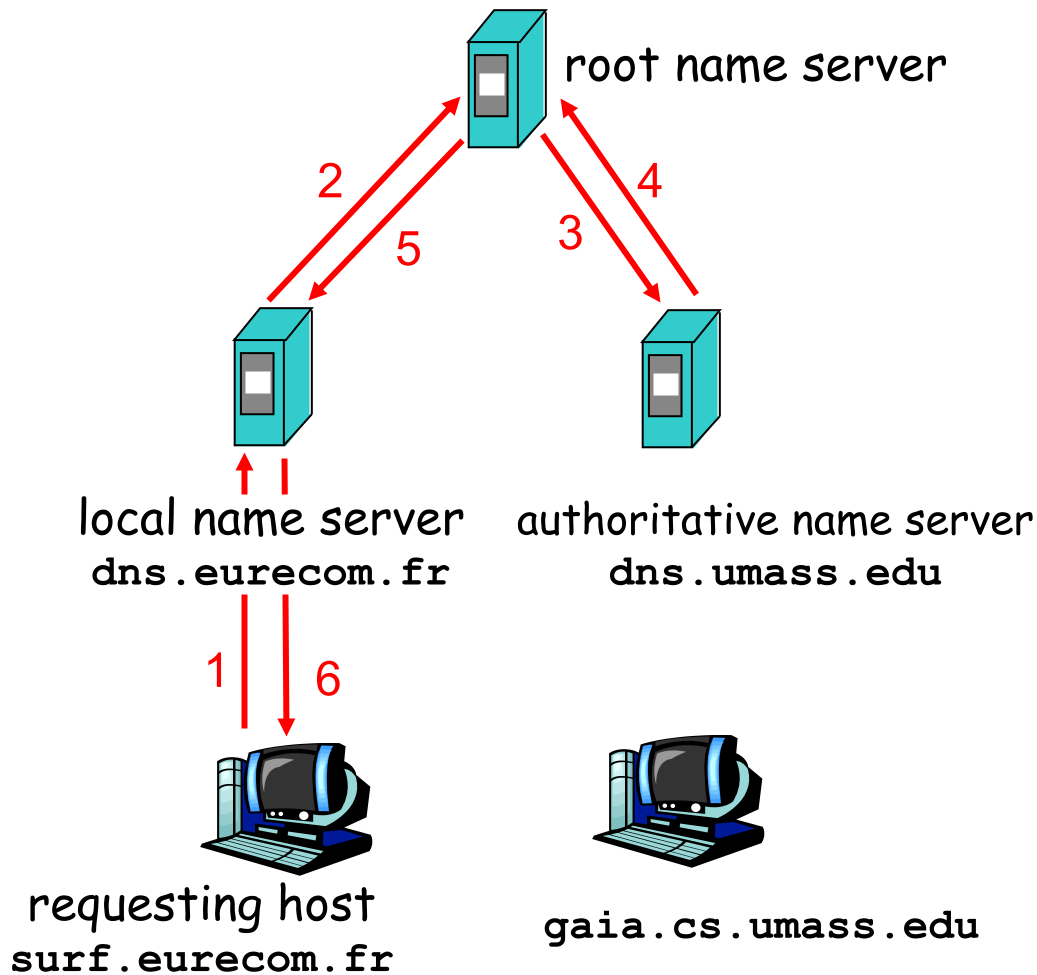
Domain Name Service (DNS) Review

- **Given a domain name,e.g. www.cuhk.edu.hk, DNS is used to translate it to the corresponding IP address e.g. 137.189.11.73**

An Example:

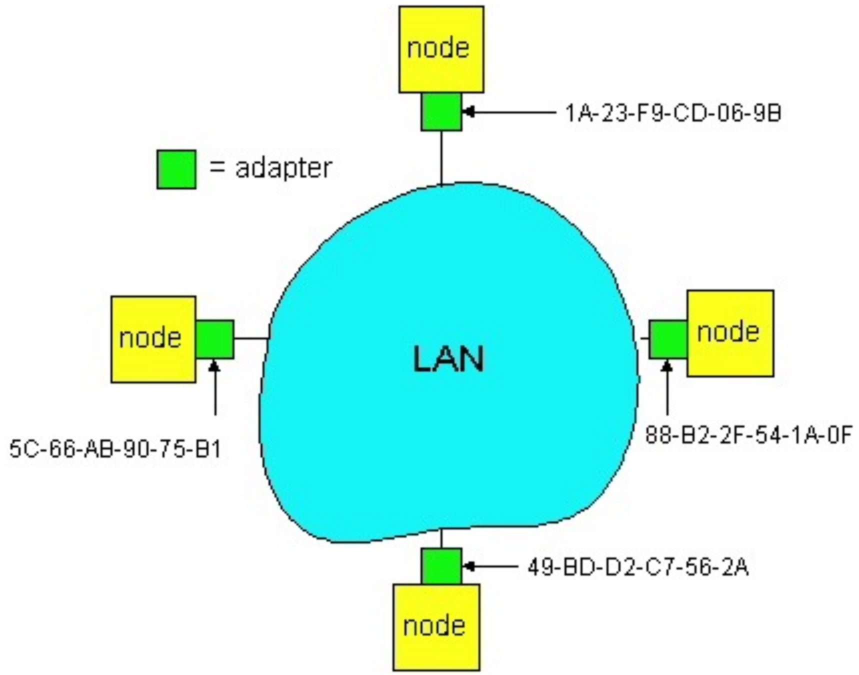
host **`surf.eurecom.fr`** wants IP address of **`gaia.cs.umass.edu`**

1. Contacts its local DNS server, **`dns.eurecom.fr`**
2. **`dns.eurecom.fr`** contacts root name server, if necessary
3. root name server contacts authoritative name server, **`dns.umass.edu`**, if necessary



LAN Addresses and ARP

- **IP address**: drives the packet to destination **network**
- **LAN** (or MAC or Physical) **address**: drives the packet to the destination node's LAN interface card (adapter card) on the **local LAN**
- **48 bit MAC address** (for most LANs); burned in the adapter ROM

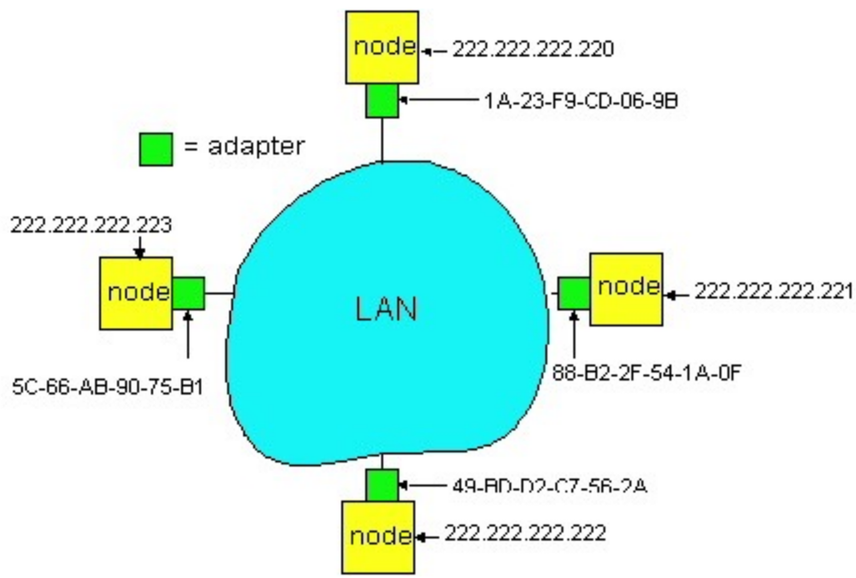


ARP: Address Resolution Protocol

- Each IP node (Host, Router) on the LAN has **ARP** module and Table (aka ARP cache)
- ARP Table: IP/MAC address mappings for **some** LAN nodes

  < IP address; MAC address; TTL>

  < .... >
- TTL (Time To Live): timer, typically 20 min



ARP (cont'd)

- Host A wants to send packet to destination IP addr XYZ on same LAN, (A knows the dest. is in the same LAN by comparing the subnet mask with the dest. IP addr.)
- Source Host first checks own ARP Table for IP addr XYZ
- If XYZ **not** in the ARP Table, ARP module **broadcasts** ARP pkt:

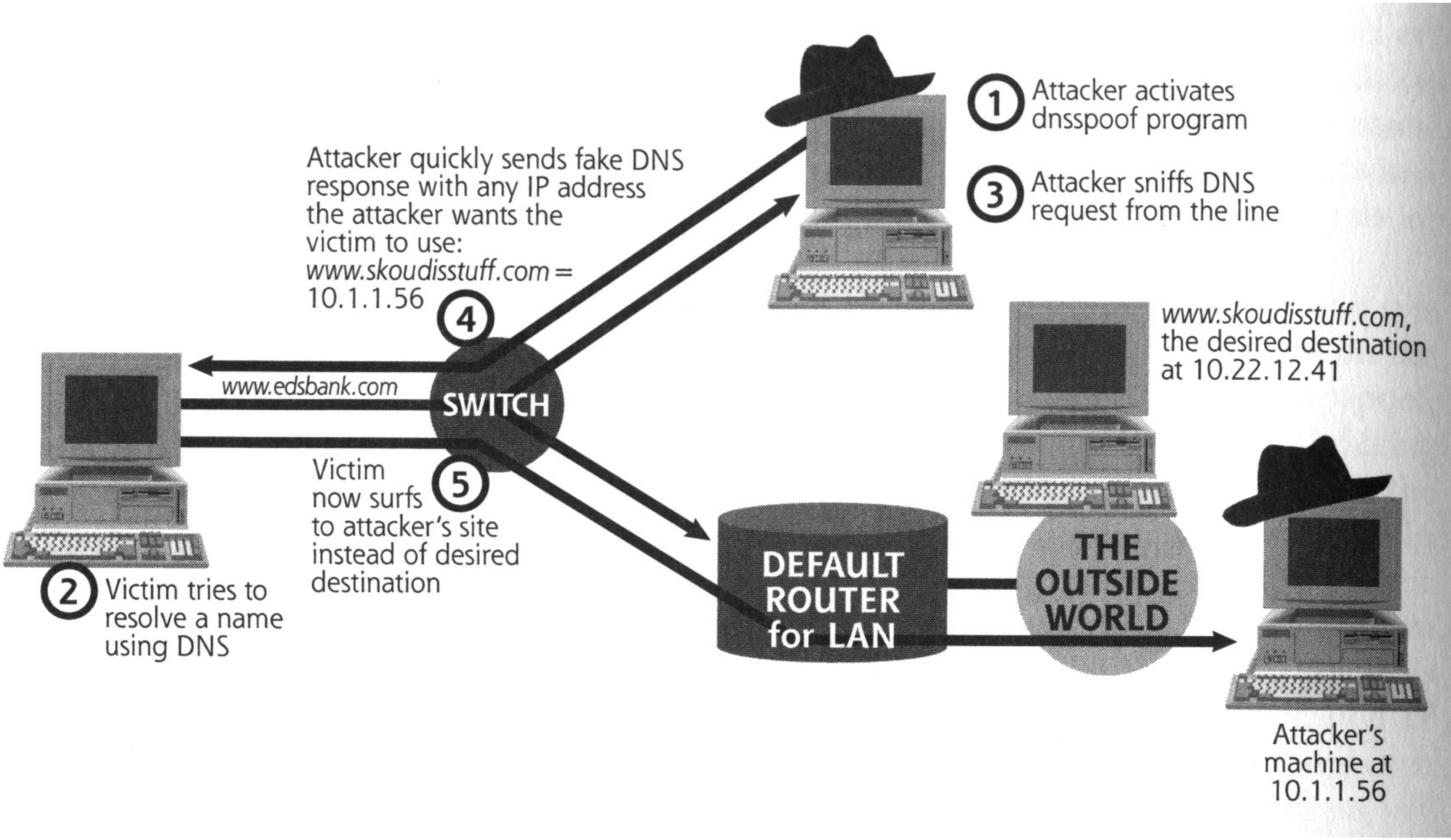
  < XYZ, MAC (?) >

- ALL nodes on the LAN accept and inspect the ARP pkt
- Node XYZ responds with **unicast** ARP pkt carrying own MAC addr:

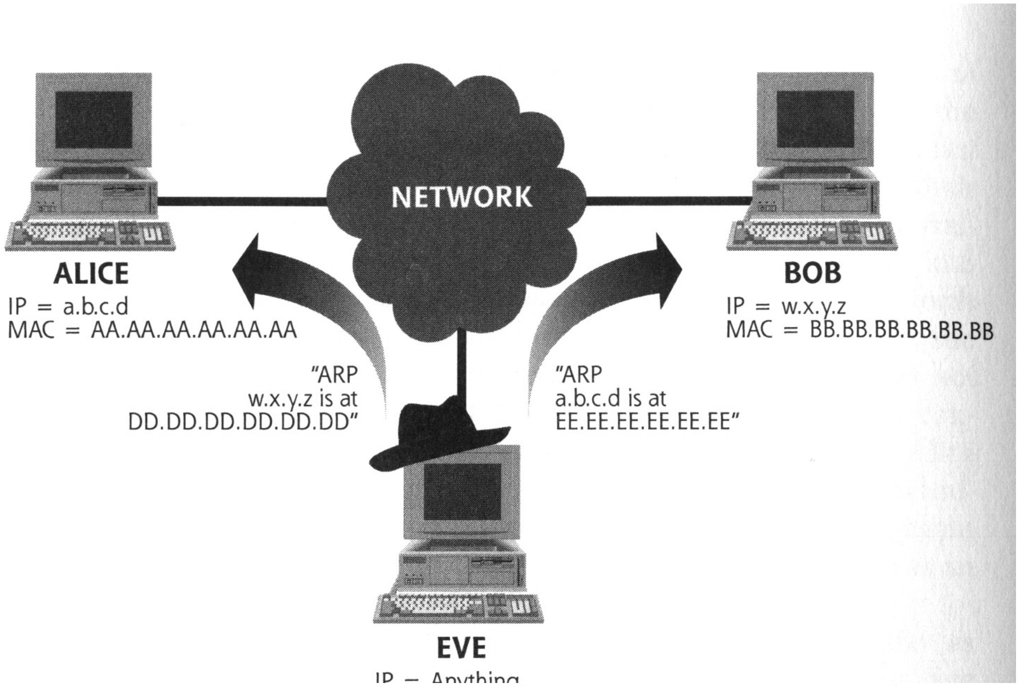
  < XYZ, MAC (XYZ) >

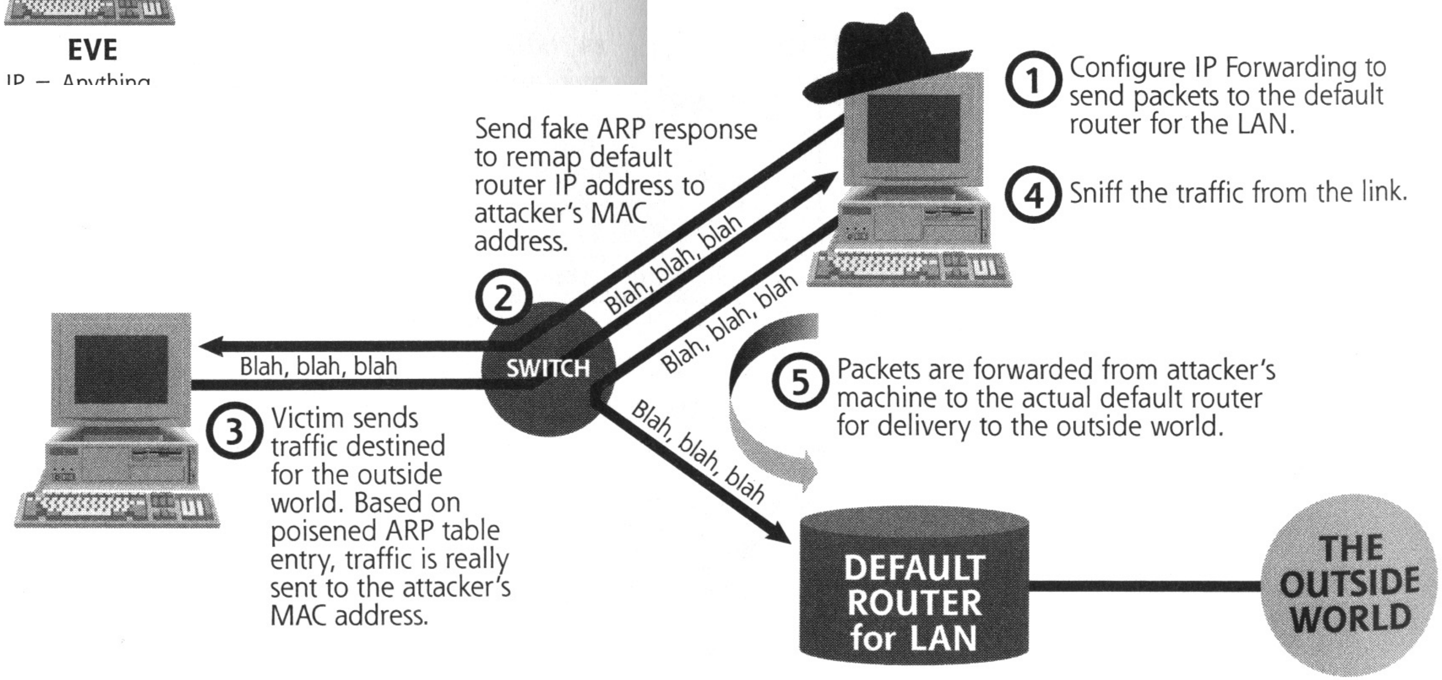
- MAC address **cached** in ARP Table

DNS Spoofing



ARP Spoofing





Specific Threats on Internet Protocol

- Spoofing: exploiting/altering mapping between

  Domain Name → IP address → MAC address

  - IP Spoofing (e.g. attacker as sender)
  - DNS Spoofing (e.g. attacker as receiver)
  - ARP Spoofing (e.g. attacker as receiver)
- Sniffing: Watching/recording contents of IP traffic passing-by, e.g. see Wireshark (used to be called Ethereal).
- Session Hijacking: take over the control of an existing session, e.g. TCP connection, from the legitimate sender/receiver
- Man-in-the-Middle attack: The attacker injects itself into the communication path between the 2 communication parties. Instead of talking directly to the intended communication partner, each communication party is fooled to send its packets to the attacker, which in turn, relays them to the legitimate receiver.
- Denial of Service (DOS) Attacks: take a victim network entities, e.g. host/routers, out of service
  - Can be used as a step to pull off a more sophisticated attack
- Attacks on Routing Protocols and Routing Infrastructure
  - DOS on Routers, Spoofing, False Route Injection, Routing Black-hole

Spoofing (Impersonating)

Impersonating the sender (or receiver) of an IP packet

- IP Spoofing: an attacker impersonate A to send an IP packet by simply setting the source IP address of the packet to A's IP address, so that:
  - The attacker can gain access to a network/host, if A is trusted by the network/host ; OR
  - Unexpected (reply) packets will be sent to A

Counter-measures:

- Require ALL edge ISP to perform ingress filtering (incomplete still)
- Use IPSec for authentication

- DNS Spoofing: an attacker redirects the packets sent by A, intended for B to a different destination Eve by sending A an incorrect DNS reply which maps B's domain-name to Eve's IP address (instead of B's)

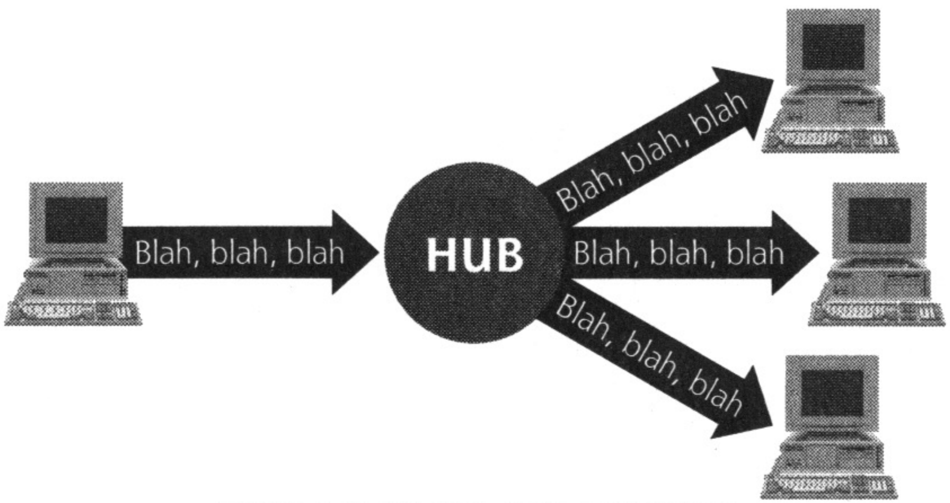
Counter-measures:

- Deploy Secure DNS (DNSsec)
- Use update version of DNS software (BIND)

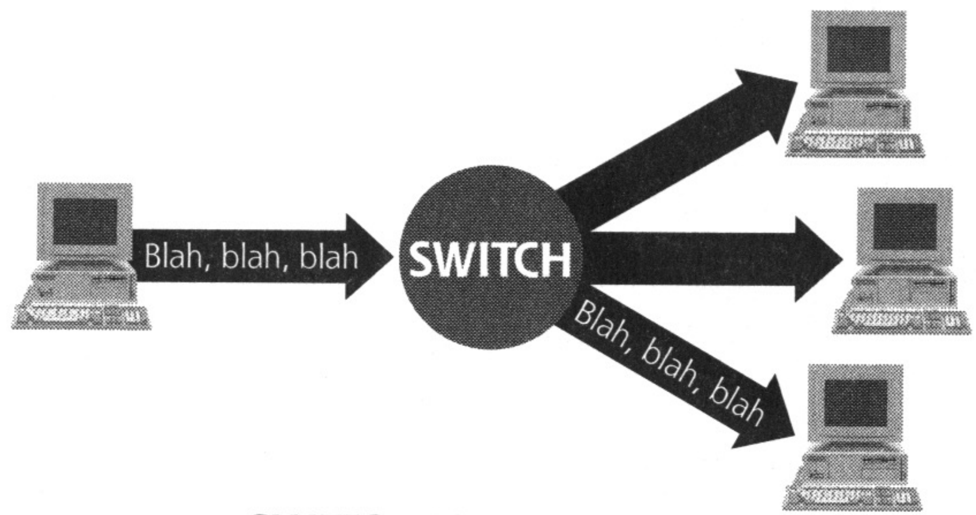
- ARP Spoofing: an attacker redirects the packets sent by A, intended for B to a different destination Eve by sending A an incorrect ARP reply which maps B's IP address to Eve's MAC address (instead of B's)

Counter-measure: Hardwire IP-to-MAC address mapping

Hub Vs. Switch (Bridge) review



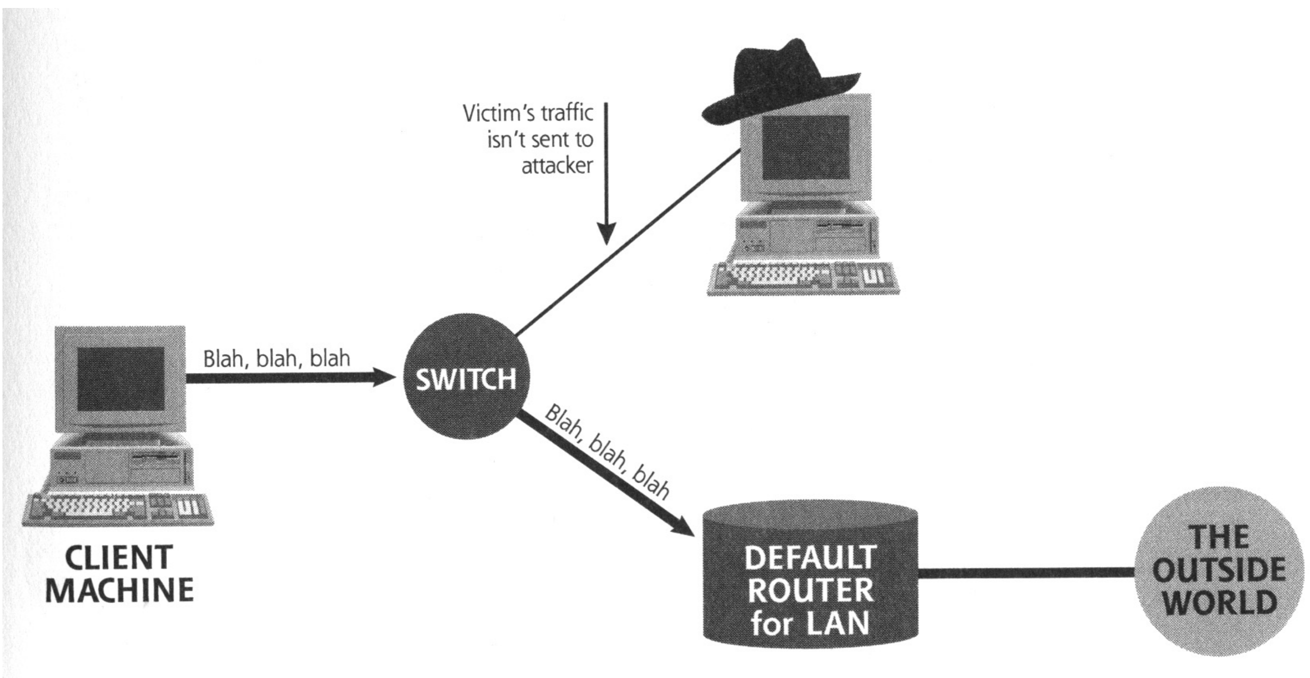
BROADCAST ETHERNET



SWITCHED ETHERNET

Sniffing

- Turn on the Promiscuous mode of an (Ethernet) Network Interface Card (NIC) to read ALL packets passing by regardless of their destination address
- Since standard IP traffic is not encrypted, the attacker can see the complete content of the packets
- Hub environment Vs. Switch environment

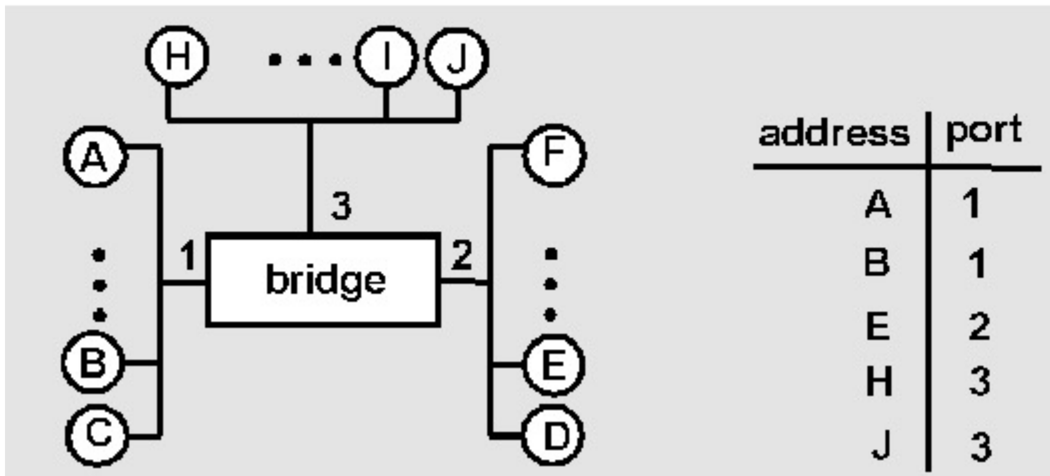


Review on the Self learning Mechanism of a Switch (Bridge)

- A bridge has a bridge table
- entry in bridge table:
  - (Node MAC Address, Bridge Interface, Time Stamp)
  - stale entries in table dropped (TTL can be 60 min)
- bridges *learn* which hosts can be reached through which interfaces
  - when frame received, bridge "learns" location of sender: incoming LAN segment
  - records sender/location pair in bridge table
- After the whereabout (i.e. associated switch port, say p1) of a MAC address, say M1, is learned, future frames destined to M1 will only be forwarded to its associated port p1, no broadcast is necessary

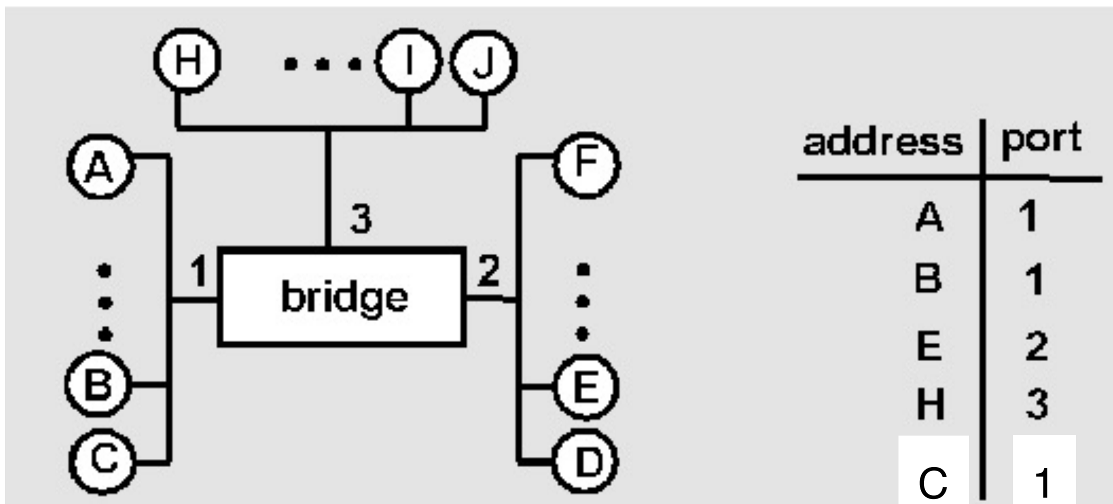
Self-learning Bridge (Switch) example

Suppose C sends frame to D and D replies back with frame to C.



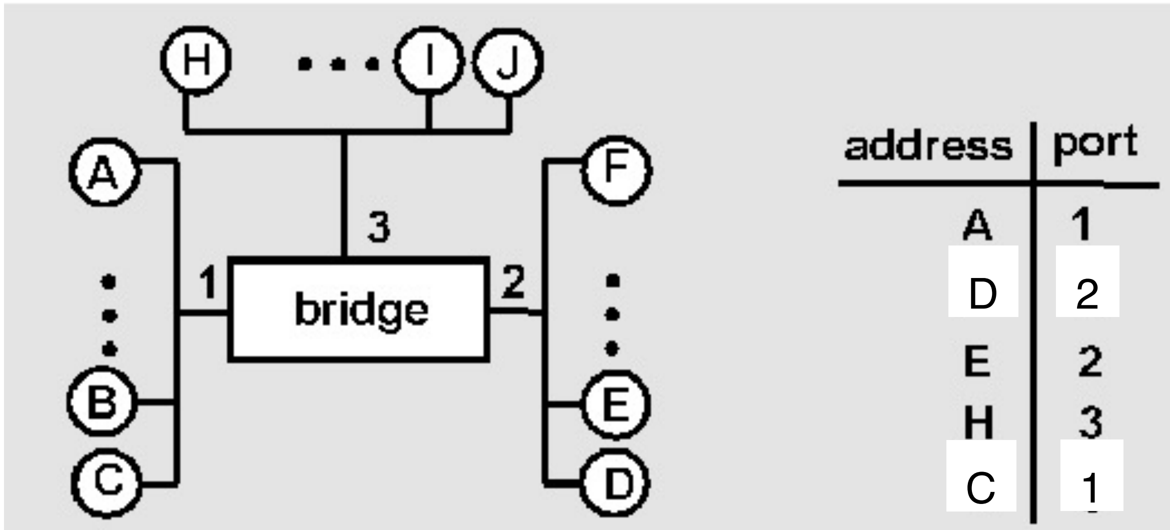
- Bridge receives frame from C
  - notes in bridge table that C is on interface 1
  - because D is not in table, bridge sends frame into interfaces 2 and 3
- frame eventually arrives at D

Self-learning Bridge (Switch) example (cont'd)



- Since the bridge just learns that C is connected to it via Interface 1, the internal bridge table is updated accordingly.

Self-learning Bridge (switch): example (cont'd)



- Later on, when D replies a frame to C, the bridge receives frame via interface 2 and it adds an entry for D in its internal bridge table.
- Since the bridge already knows C is on interface 1, so it selectively forwards frame to interface 1
- What if the learning table is full ?
  - Some old entries got replaced by newly learned ones

=> Has been exploited by Hackers to sniff thru a bridge by forcing it always broadcast

Sniffing on a Switched LAN

- The attacker fills up the Self-learning Table on the Switch with bogus entries by sending out a large number of frames with relevant source MAC address

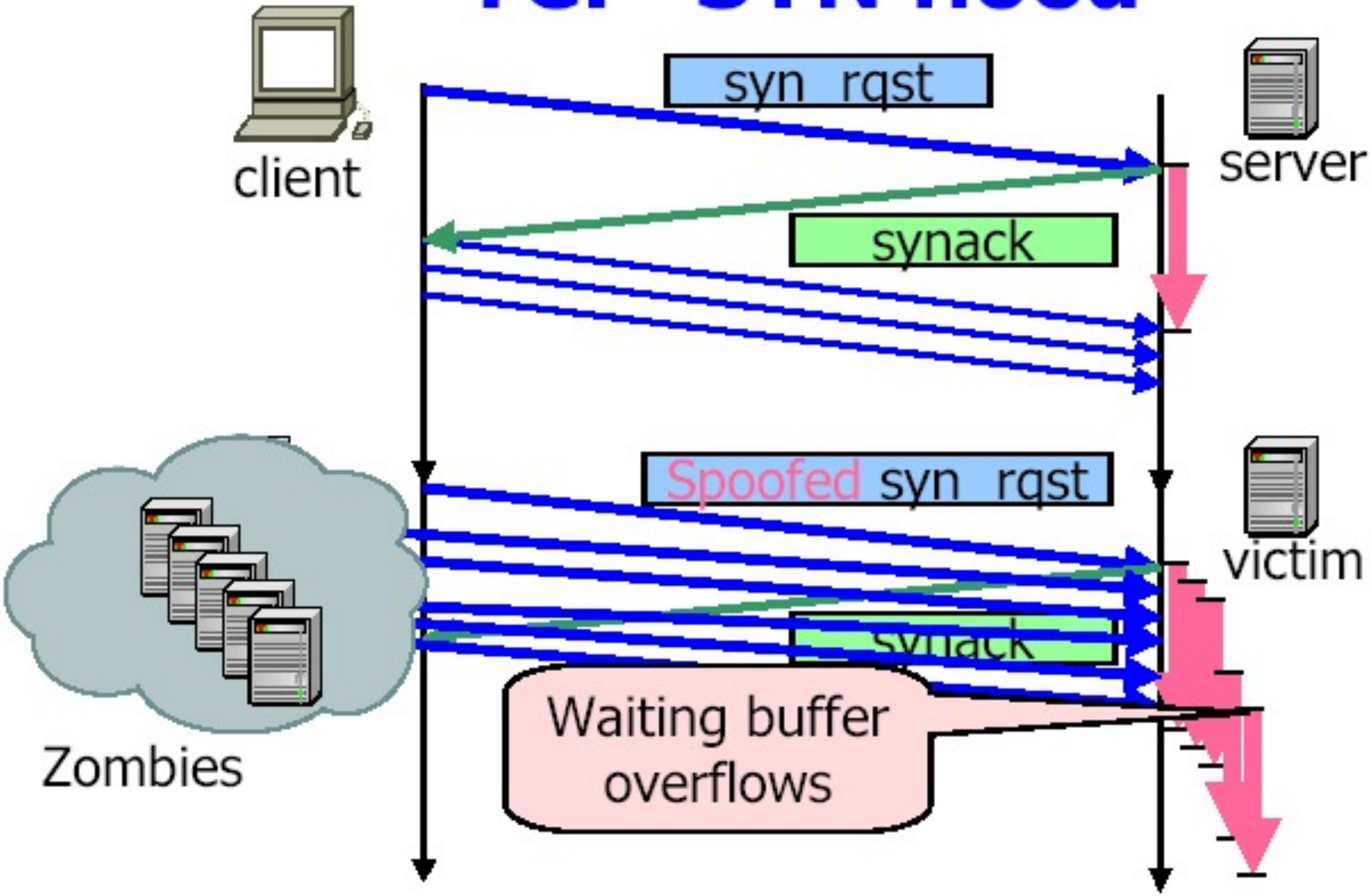
=> Entries of legitimate MAC address cannot be found on the self-learning table

=>The switch has no choice but to broadcast the frame to all switch ports

⇒ Now, the switch effectively operates like a hub and is susceptible to sniffing

e.g.Dsniff supports this feature

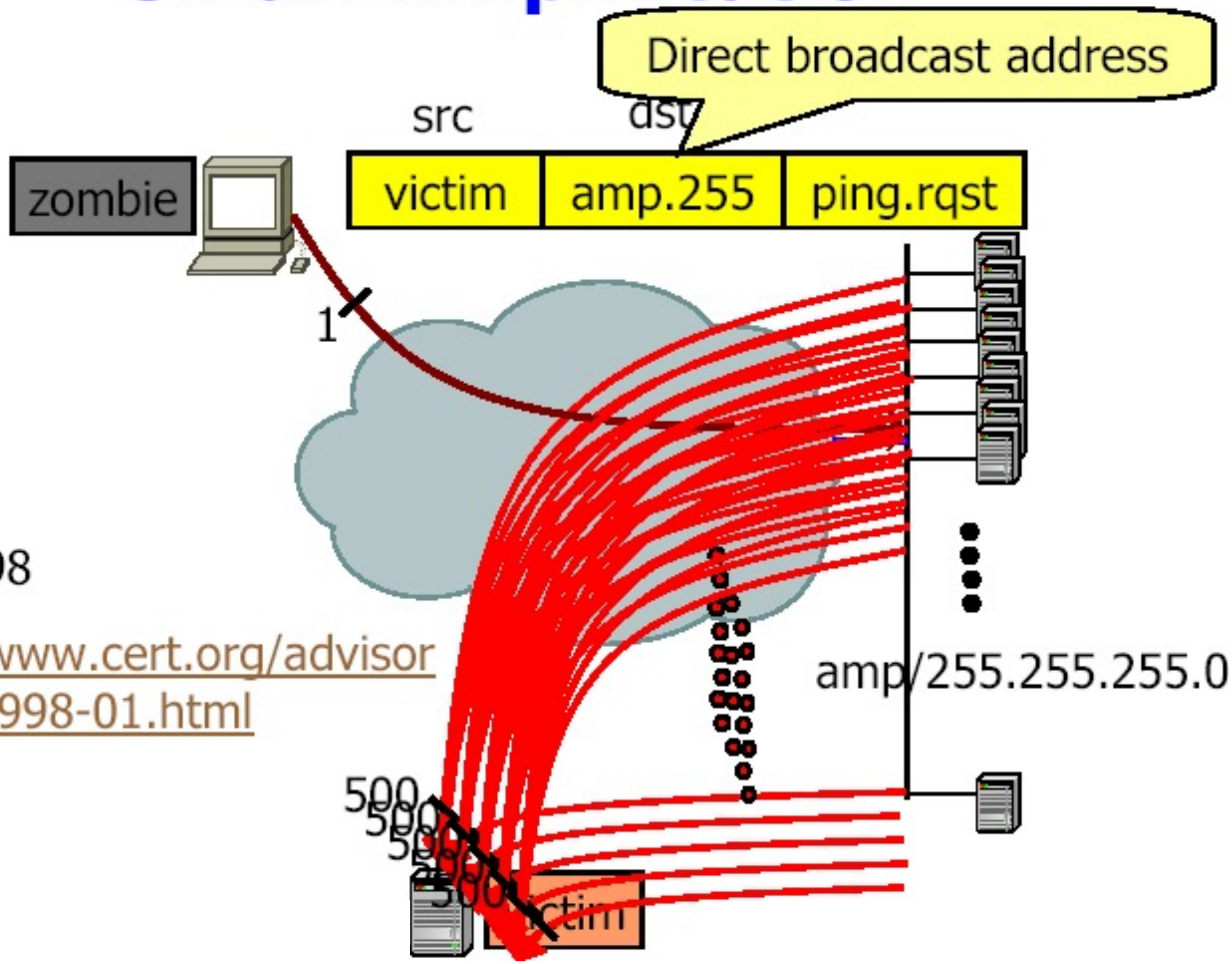
TCP SYN flood



Counter-measures:

- Tune time-out values in Server OS
- Use sync-cookies, a new extension during TCP conn. setup

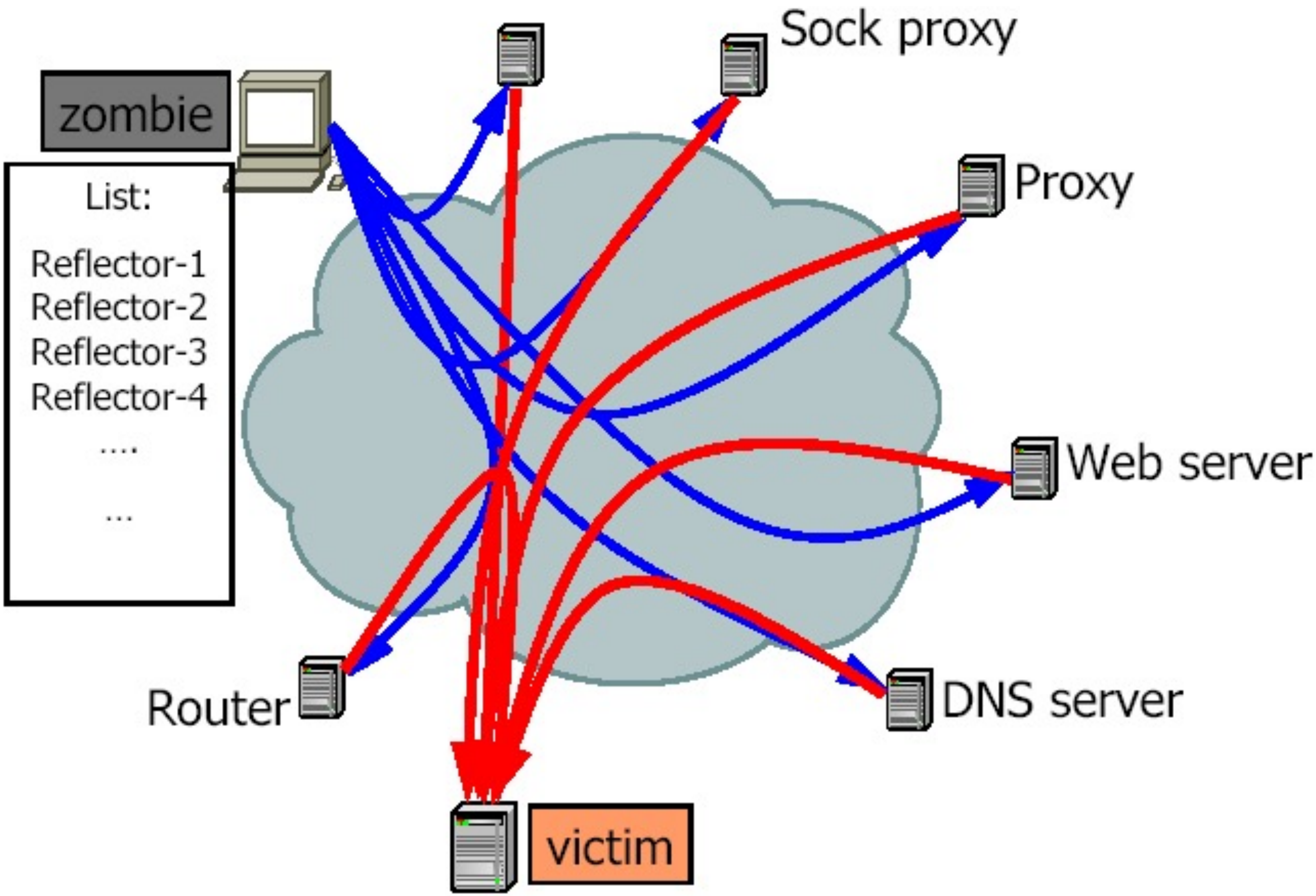
Smurf Amplification



- Jan 1998
- http://www.cert.org/advisories/CA-1998-01.html

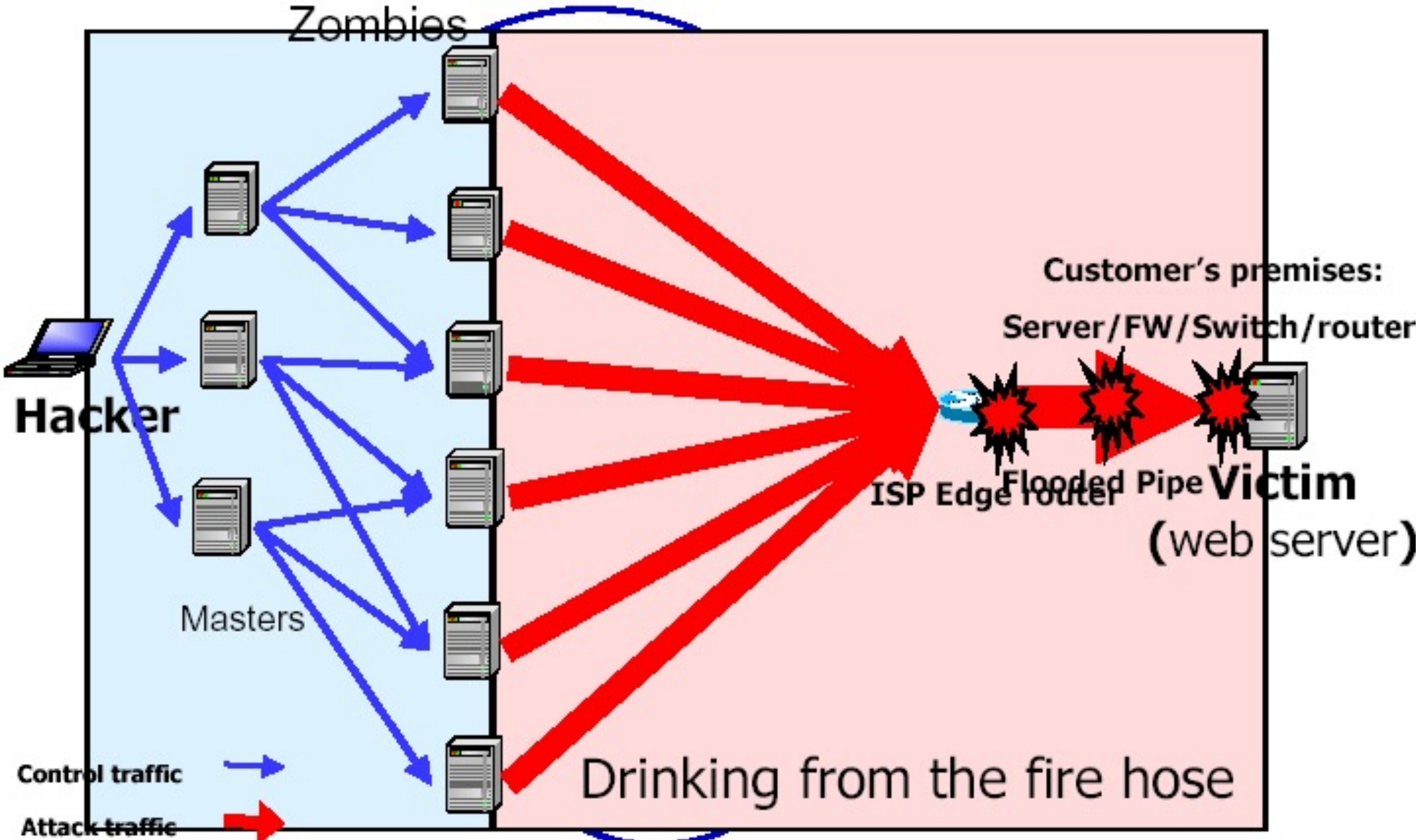
Counter-measures: Disable direct broadcast across subnets

Reflectors



Counter-measures: Use IPSec for authentication

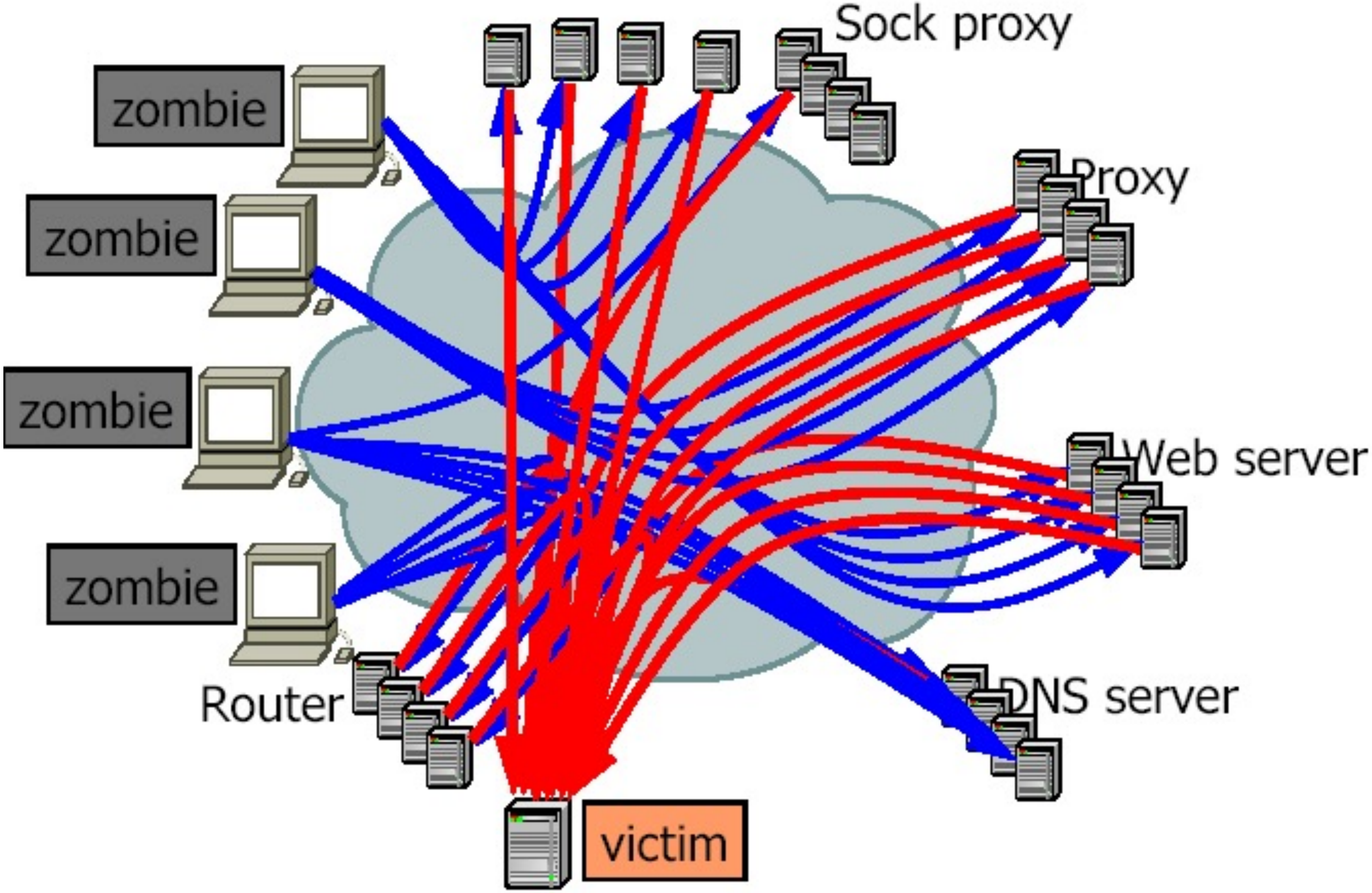
DDoS



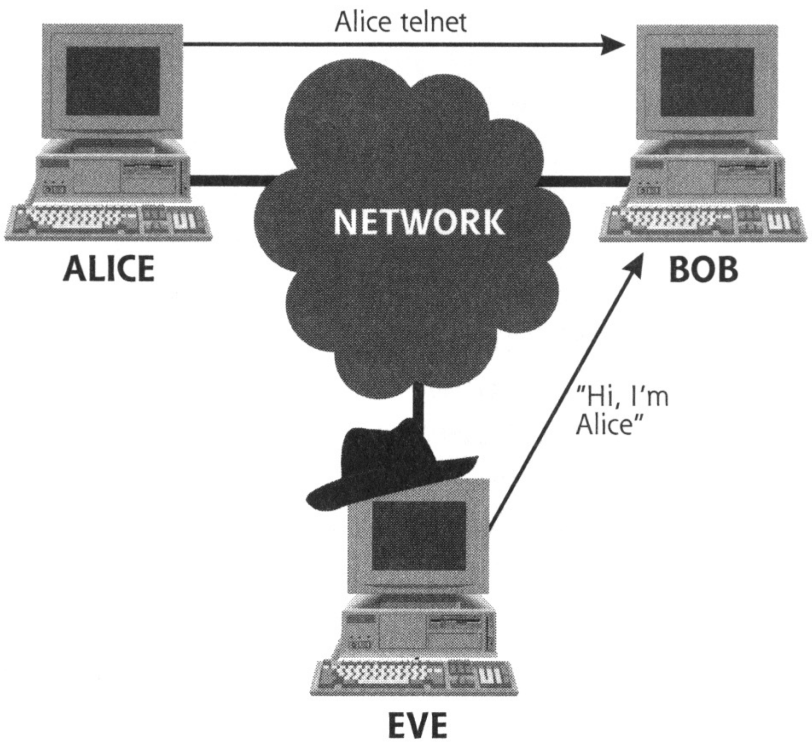
Counter-measures:

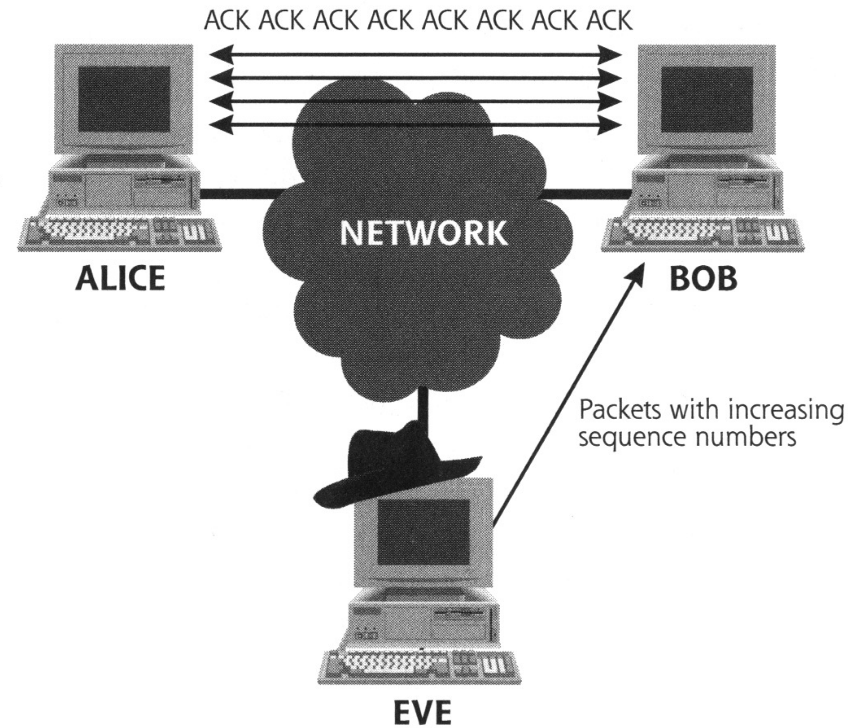
- Increase Capacity (Akamai services) ;
- Products from Start-up for overload control, e.g. : Riverhead, Mazu, Arbor etc

Reflectors


Sock proxy
zombie
Proxy
zombie
zombie
Web server
zombie
Router
DNS server
victim

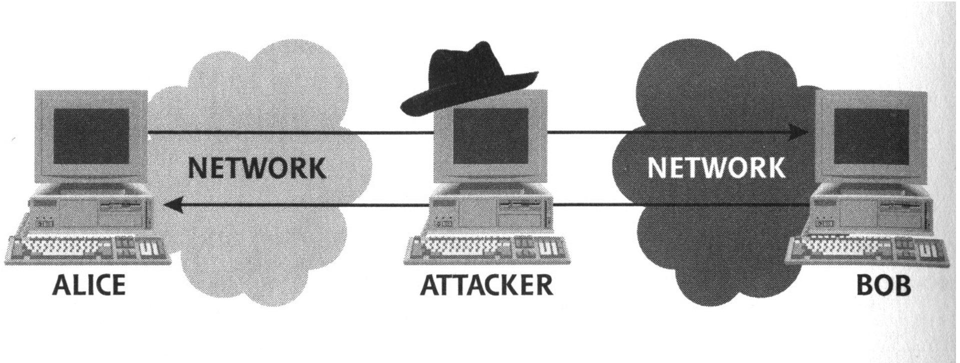
TCP Hijacking



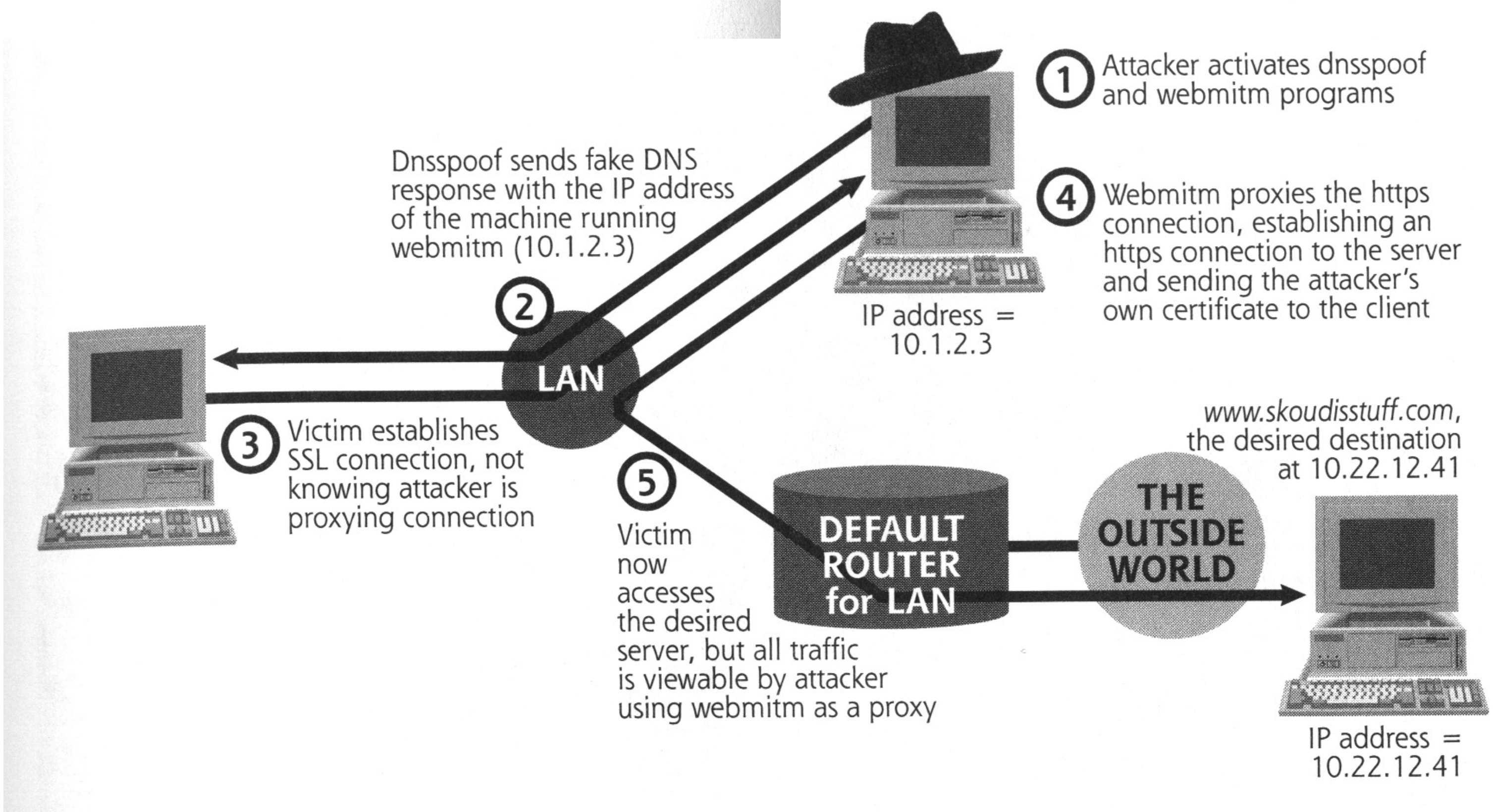


- Eve first sniffs at the Network to discover an ongoing TCP connection ; note the corresponding TCP ACK sequence numbers being used
- Eve creates an IP packet to carry its command of choice, using Alice IP's address of the packet's source IP address ; also with the right TCP ACK sequence numbers
- To avoid the ACK-storm, Eve can choose to bring Alice down by launching a DOS attack against Alice ; or Eve can perform an arp or dns spoofing on Alice and Bob to redirect their outgoing packets to a blackhole

Man-In-The-Middle Attack



- A Conceptual View



Counter-measure: Read carefully before accepting a Digital Certificate

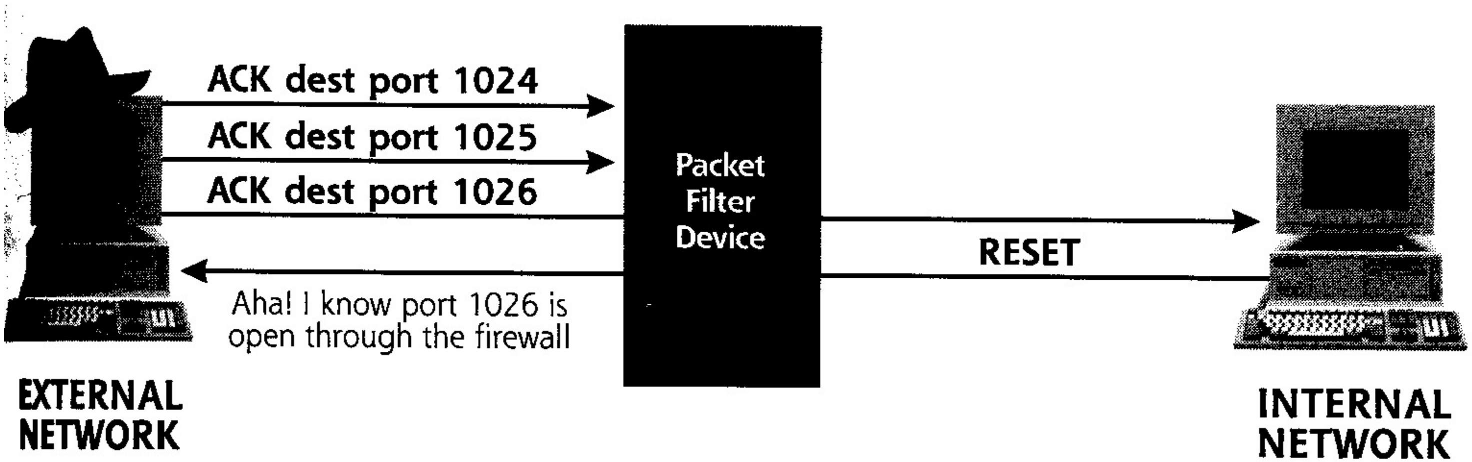
Stages of a Typical Network Attack

1. Reconnaissance
   - Social engineering
   - Information collection via Web surfing (e.g. using SamSpade or Google)
   - Port Scanning (e.g. using nmap of insecure.org)
   - OS/Device finger-printing (e.g. using nmap)
   - Vulnerability discovery (e.g. using Nessus, OpenVAS, Nikto)
   - Network Mapping (e.g. using traceroute, Kismet)

2. Gaining Access to accounts/machines
   - Social Engineering
   - Trojan horse, Worms
   - By exploiting known vulnerability lists on specific OS, device, protocols, application programs/ server programs
     - Buffer-overflow
     - Default Passwords for Admin/guest accounts,Lapse in default configurations
     - Password eavedrop (using sniffers, e.g. Wireshark)
   - Using other tools such as Metaspolit, Ettercap, Burp Suite, Backtrack, w3af, sqlmap, etc

- See http://www.insecure.org/tools.html for the popular tools

Port Scanning



Stages of a Typical Network Attack (cont'd)

4. Cover the tracks and hiddings
   - Erase log files,
   - Install hidding tools (rootkits) : plant tainted version of system utilities,e.g. ps, ls
5. Maintaining access
   - Planting Backdoor servers and trojans to allow future remote control e.g. BackOrifice, Netcat, remote-control zombies.
6. Use the compromised machine as jumping board to the next set of attacks
   - DNS/ARP Spoof enabling (e.g. Dsniff)
   - Planting tools for info collection by installing sniffers (e.g. Wireshark, Snort, Ettercap), key-stroke loggers, Cracking local password files
   - Exploit local trust
   - Use compromised machine as zombie to launch DDoS attacks