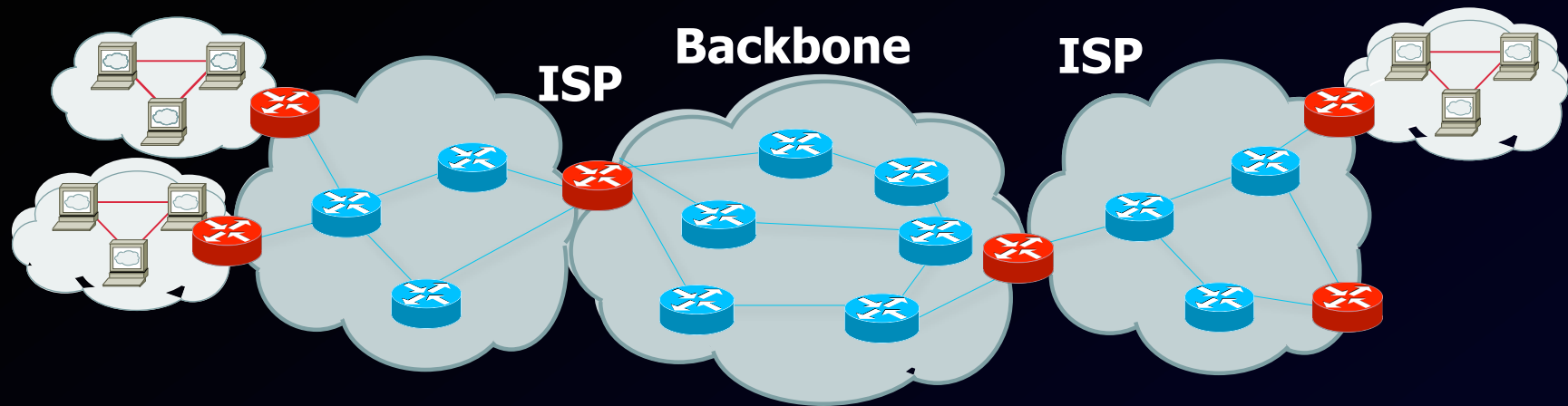


# UMSSIA

**DAY III: NET/PROFIT...**

# THE INTERNET



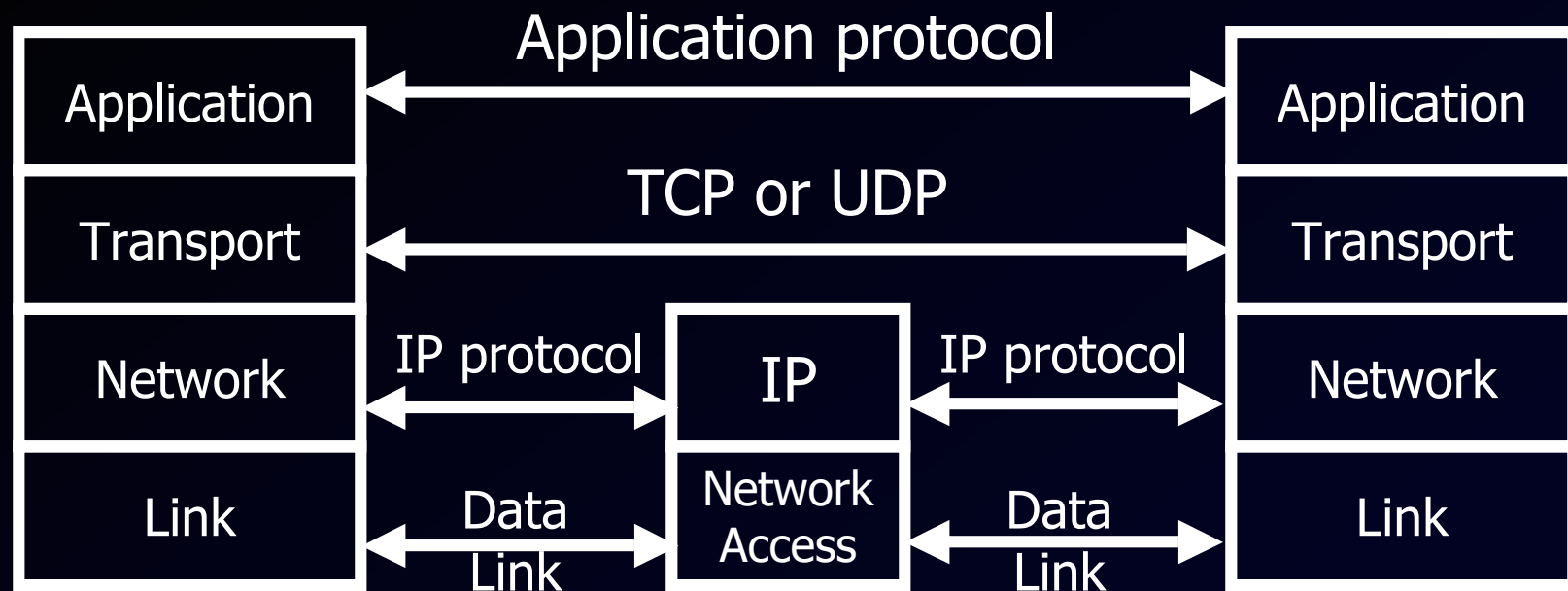
The internet consists of **end system** networks connected to other networks by ISPs or **Autonomous Systems (AS)**.

ASes run **routing** protocols (BGP, EGP, iBGP) to determine how to reach different addresses.

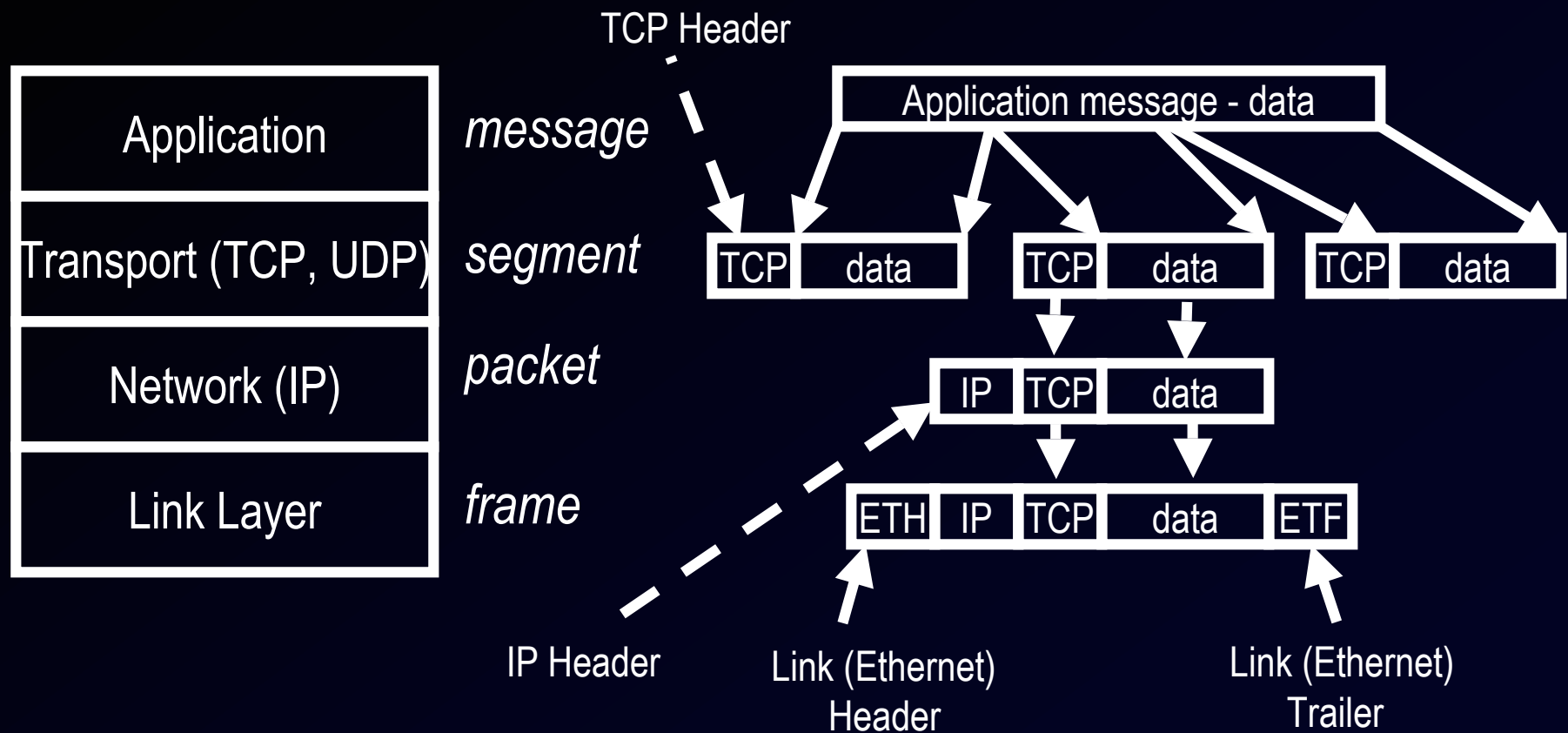
**Packets** of data are **forwarded** between hosts via TCP/IP.

**Addresses** are translated to **Domain Names** by DNS.

# IP PROTOCOL STACK



# PACKET FORMATS



# INTERNET PROTOCOL

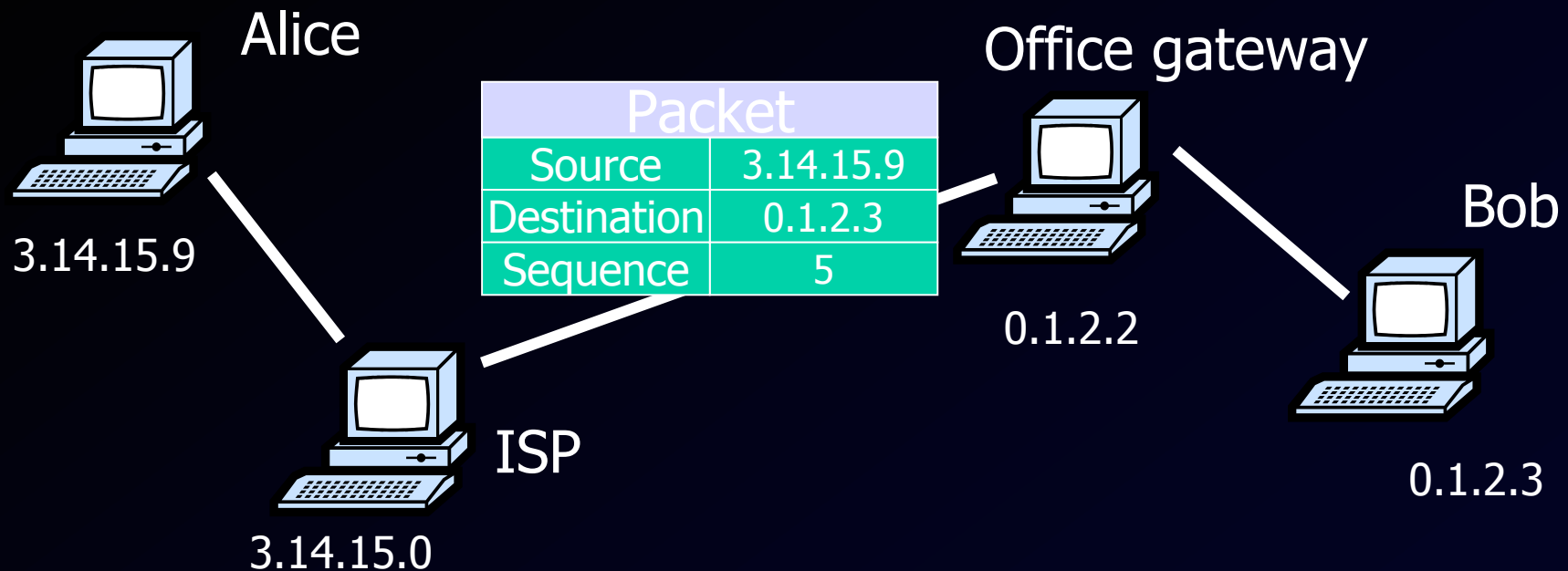
... or IP, is the “base” internetwork protocol.

IP is a **connectionless, unreliable, best effort** datagram delivery service

An IP Datagram is a **header** followed by data:

V#	HLen	TOS	Length	
Ident		Flag	Offset	
TTL	Protocol	Checksum		
Source IP Address				
Destination IP Address				
Options (opt)			Pad	
Data				

# IP FORWARDING



Datagrams are forwarded using the 4-byte **IP address** of the destination host.

Typical routes use several (**10-15**) **hops**.

# IP “FEATURES”

- **IP provides forwarding, if:**
  - The source knows the location of a router
  - Each router knows the next hop to other networks
- **IP provides packet **fragmentation** when traversing different networks. End hosts perform reassembly.**
- **IP provides error reporting, via the ICMP protocol, e.g. packet is dropped, host is unreachable, etc.**

# USER DATAGRAM PROTOCOL

UDP is a transport protocol that provides limited application (de)multiplexing via ports.

Each application sends & accept datagrams from its own port, specified by 16-bit port numbers.

The **destination** port specifies the remote app being addressed, and the **source** port provides the "return address."

UDP retains IP's statelessness and unreliability

# TRANSMISSION CONTROL PROTOCOL

TCP is a reliable transport protocol. A TCP connection is defined by (src:port, dst:port).

TCP Senders break a byte stream down into **packets** with sequence numbers.

TCP Receivers **acknowledge** receipt of packets in order.

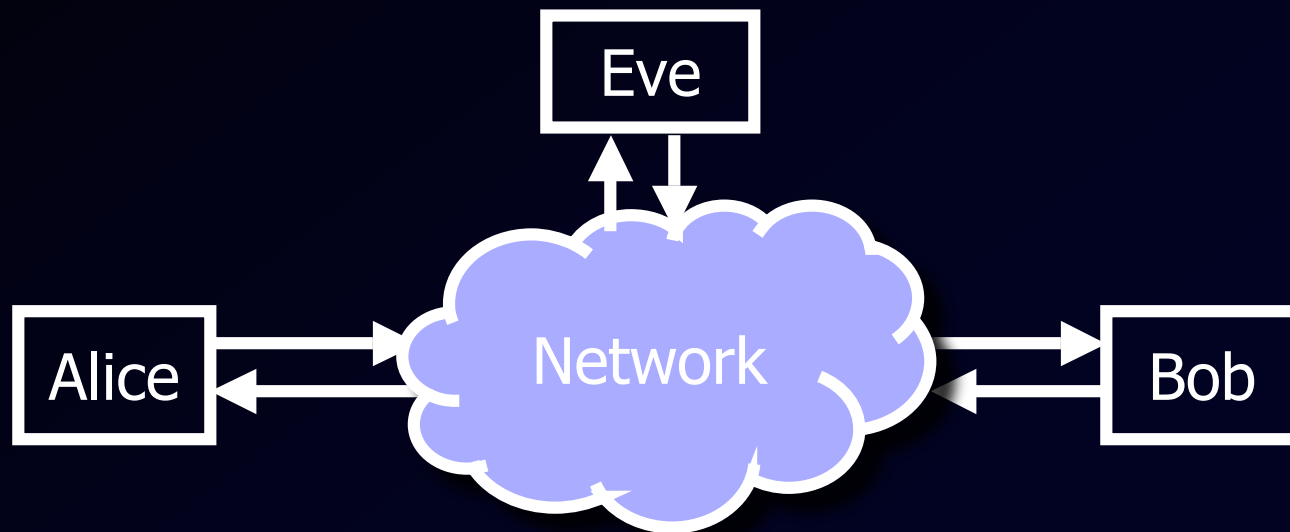
Dropped packets aren't ACKed, causing **timeouts**. The sender resends NAKed packets after a timeout.

Timeouts also signal network congestion. Senders use AIMD algorithm (voluntarily) to control their send rate.

# ATTACKS: PACKET SNIFFING

... Is when an intermediate host or one with a "shared medium" looks at packets sent to others.

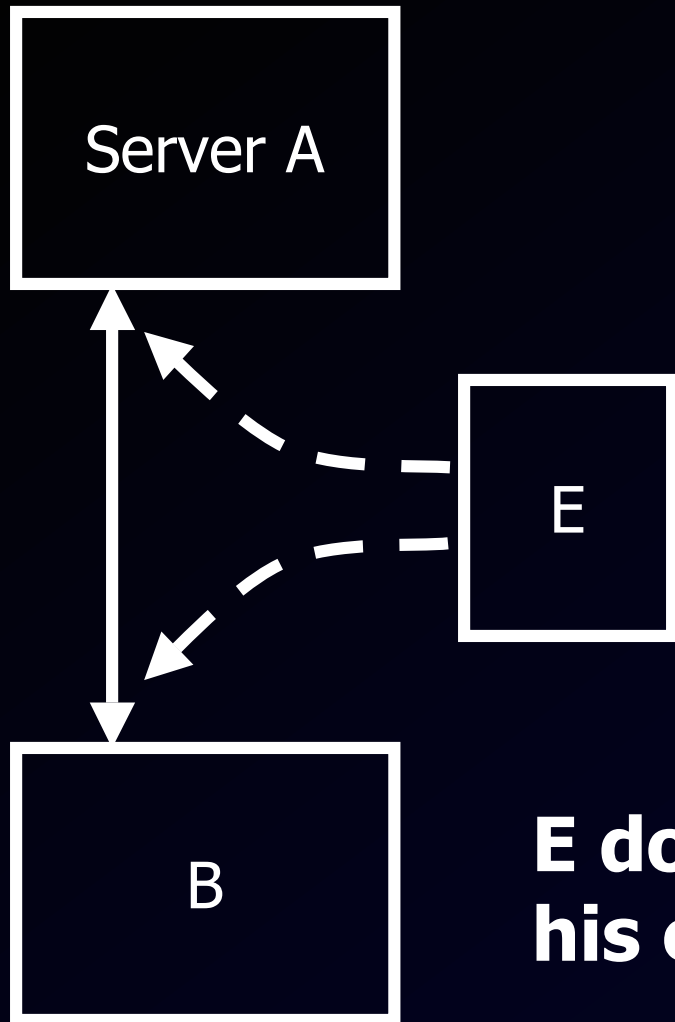
Many network applications send sensitive info (e.g. passwords, credit cards...) "in the clear."



# CONNECTION SPOOFING

- Each TCP connection has a **state**:
  - ClientIP:Port:SeqN, ServerIP:Port:SeqN,
- An attacker who can guess this state can “inject” packets into a connection.
- **Problem: It is easy to guess the state!**
  - IPs are public, port numbers are standard
  - Sequence numbers are often chosen in a predictable way
  - Attacker can usually guess correct SN with a few hundred packets.
- **(Partial) solution: unpredictable ISNs.**

# IP SPOOFING



- 1. B connects to A; both assume this connection is trustworthy**
- 2. E connects to A (as E) to get TCP ISN**
- 3. E initiates DoS on B.**
- 4. E sends packets to A with forged headers from B.**
- 5. A accepts E's packets as trusted.**

**E doesn't get to see the result of his commands (eg `"rm -rf /*"`)**

# ROUTING VULNERABILITIES

- The goal of a **routing attack** is to direct some traffic through a compromised host or hosts.  
... As a means to do some other attack e.g.:
- Read responses from TCP spoofing
- Read unencrypted connections, sniff passwords...
- Selectively modify packets, inject/replay commands, etc...
- Drop or delay packets on some connection...

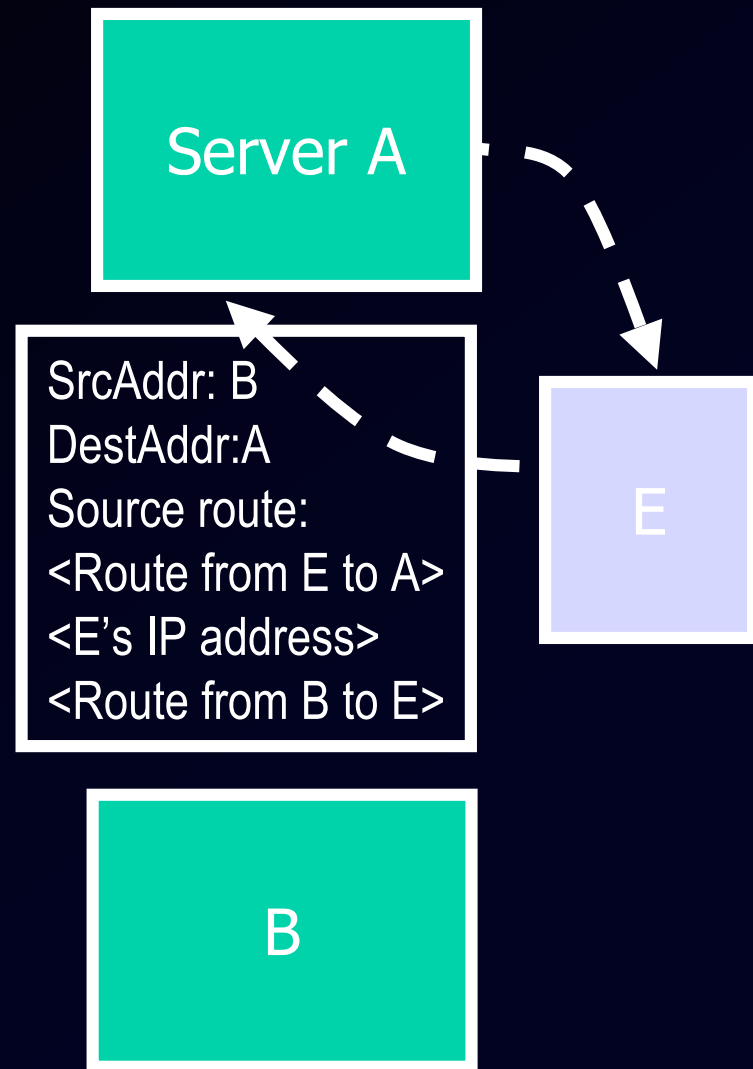
# SOURCE ROUTING ATTACK

In **source routing**, the source node chooses a route to the destination, and includes it in each packet

The destination host uses the reverse of the source route to return traffic.

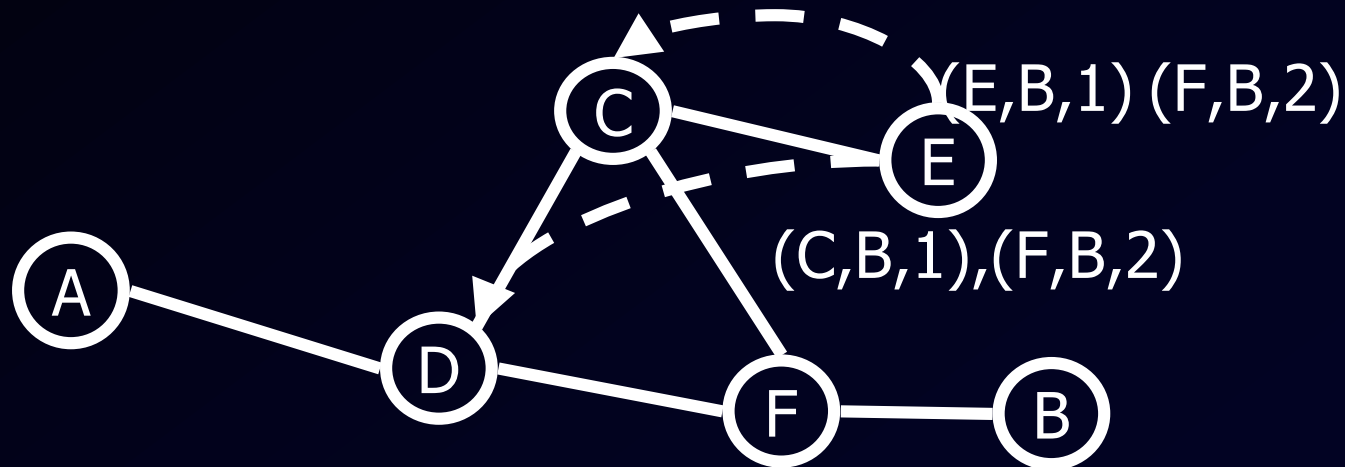
Source routing allows an attacker to route traffic through machines controlled by attacker, send and receive from different addresses

Defense: disable source routing

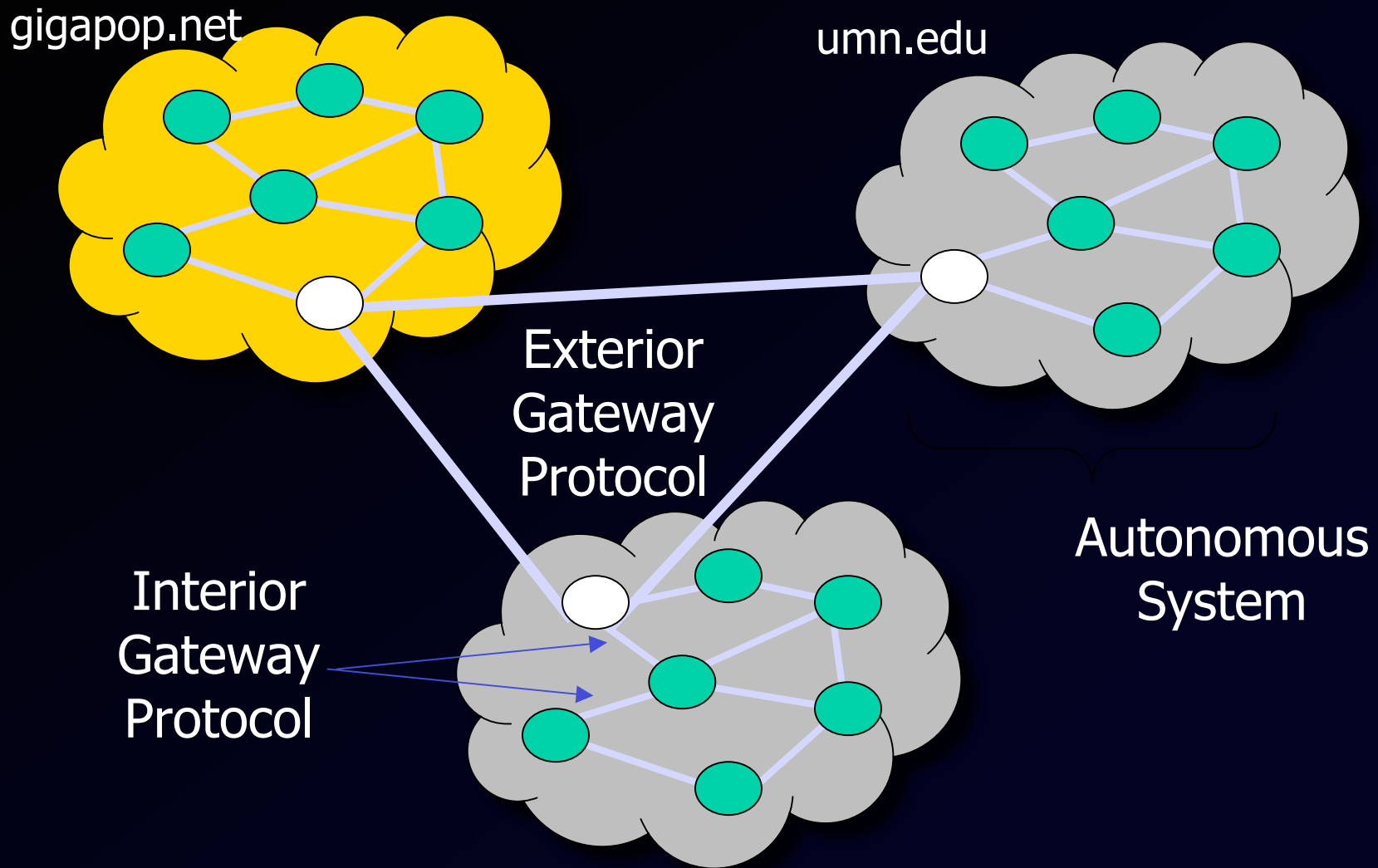


# ROUTING INFORMATION PROTOCOL

- ... is a **Distance Vector** protocol: each gateway sends a list of (network, hop-count) pairs to his neighbors
- **Attack:** send bogus routing information to target and/or gateways along the route
- **Resend to host w/ source routing**



# INTERDOMAIN ROUTING



# BGP OVERVIEW

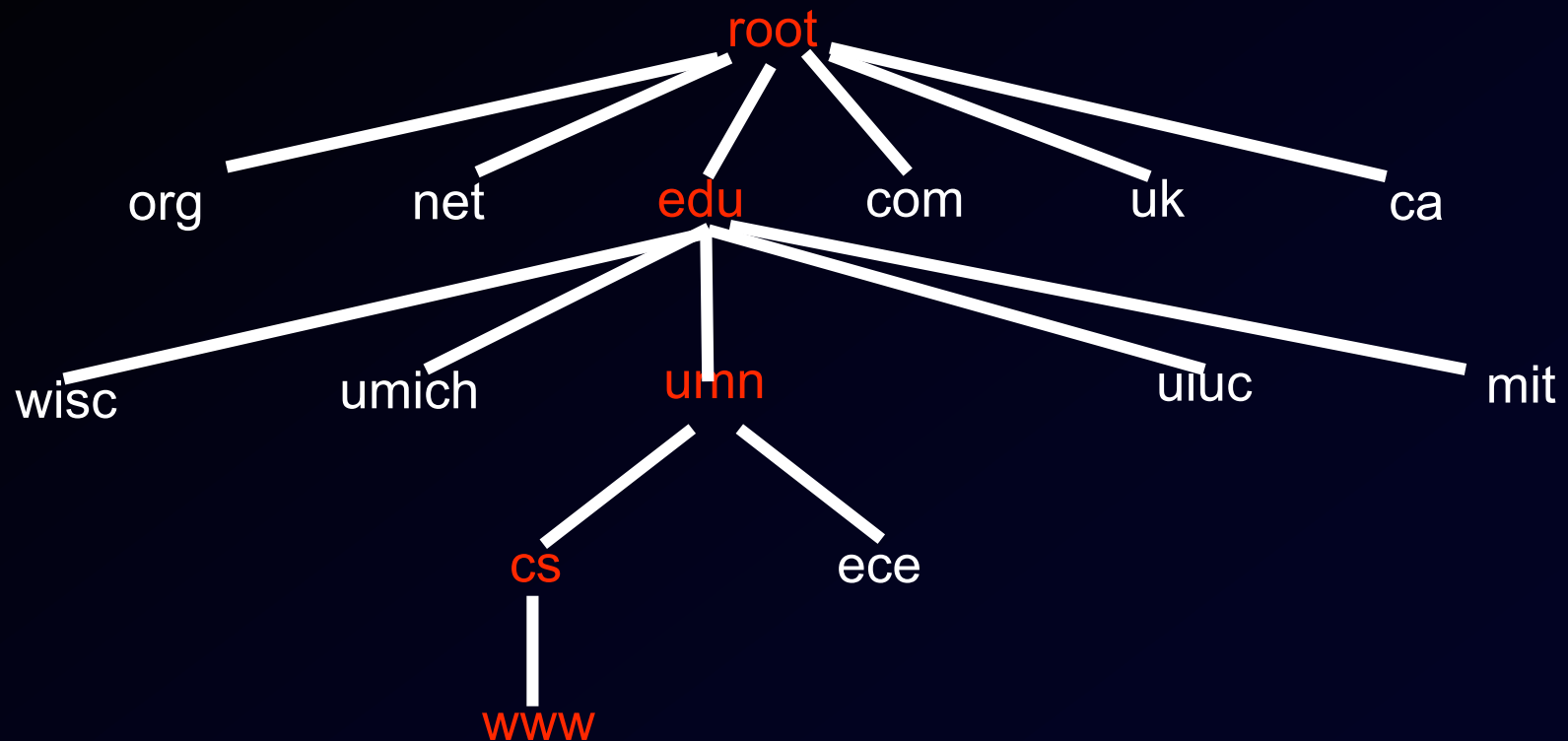
- BGP uses **iterative path announcement** for route discovery:
  - Paths grow from destination to source
  - Packets flow in the reverse direction
- Announcements **may** be the shortest path
- Nodes are **allowed** to use other policies
- An AS is not obligated to use the announced path when forwarding a packet.

# BGP ISSUES

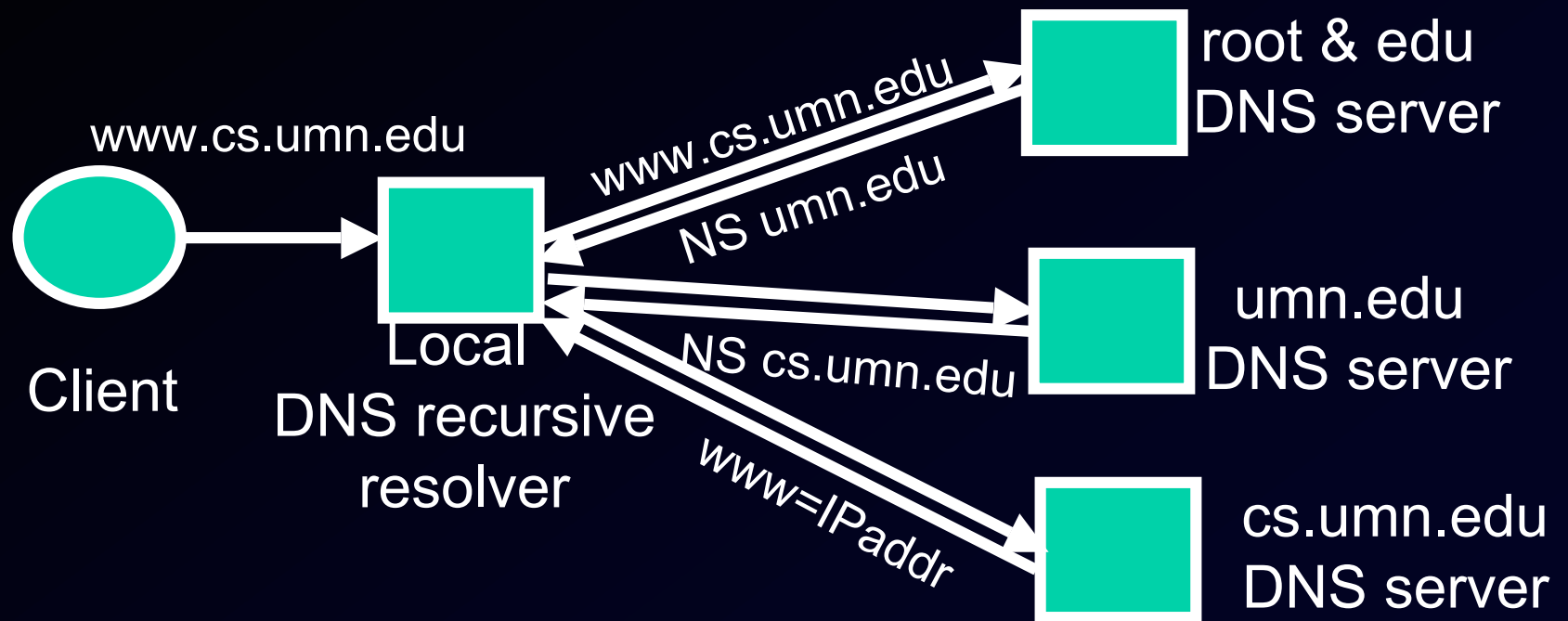
- **BGP packets are un-authenticated: anybody can claim to speak for any AS.**
- **BGP has a history of **Black Hole** incidents due to misconfiguration**
- **One of the most important services is one of the least-secured.**
- **BGP gives ASes incentives for dishonesty: ISPs pay for some routes, others (e.g. lower quality) are free or costly to competitors.**
- **Frequent issue: asymmetric paths due to **hot potato** routing.**

# DNS

... Maps between flat IP addresses and a semantic, hierarchical address space



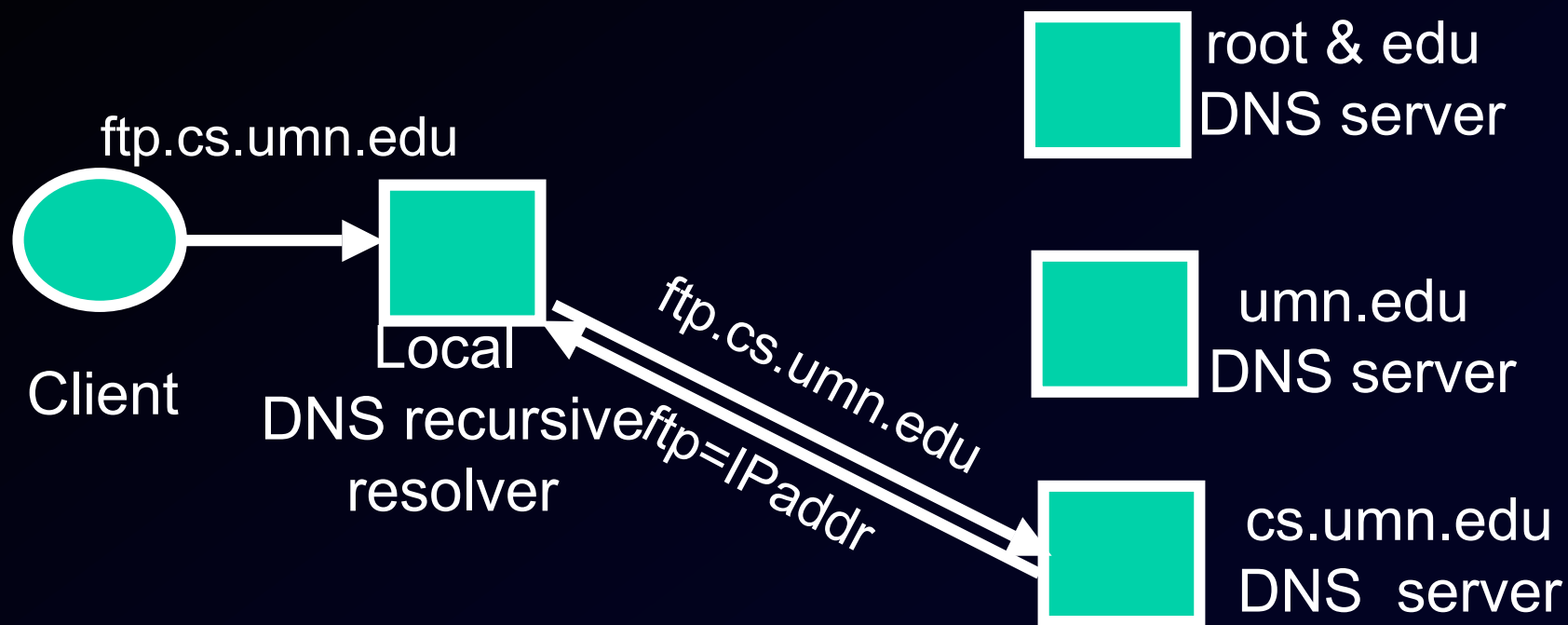
# DNS LOOKUPS



# DNS CACHING

- **DNS responses are usually **cached****
  - Quick response for repeated translations
  - Recursive results are reused by other queries
- **Negative queries** are also cached
  - We don't have to repeat past mistakes
- **Cached data periodically times out**
  - Lifetime of cache data is controlled by the owner of the data: cache TTL is passed with every record

# REPEAT DNS LOOKUP



# INHERENT DNS VULNERABILITIES

- **Users and hosts typically trust the host-address mapping provided by DNS**
- **Interception** of requests or **compromise** of DNS servers can result in bogus responses (Filipino DNS)
- **“Zone transfers” can provide useful list of target hosts**
- **Partial Solution – authenticated requests and responses limit us to trusting the “required” parties**

# RSH ATTACK

- **RSH and Rlogin's trust relationships use symbolic addresses, e.g. /etc/hosts.equiv contains friend.my.domain**
- **IP requests come with numeric source addresses. How do we determine permissions?**
  - **Use reverse DNS to find symbolic name**
  - **Decide access based on /etc/hosts.equiv**
- **Attack: spoof reverse DNS to make the host trust the attacker**

# REVERSE DNS

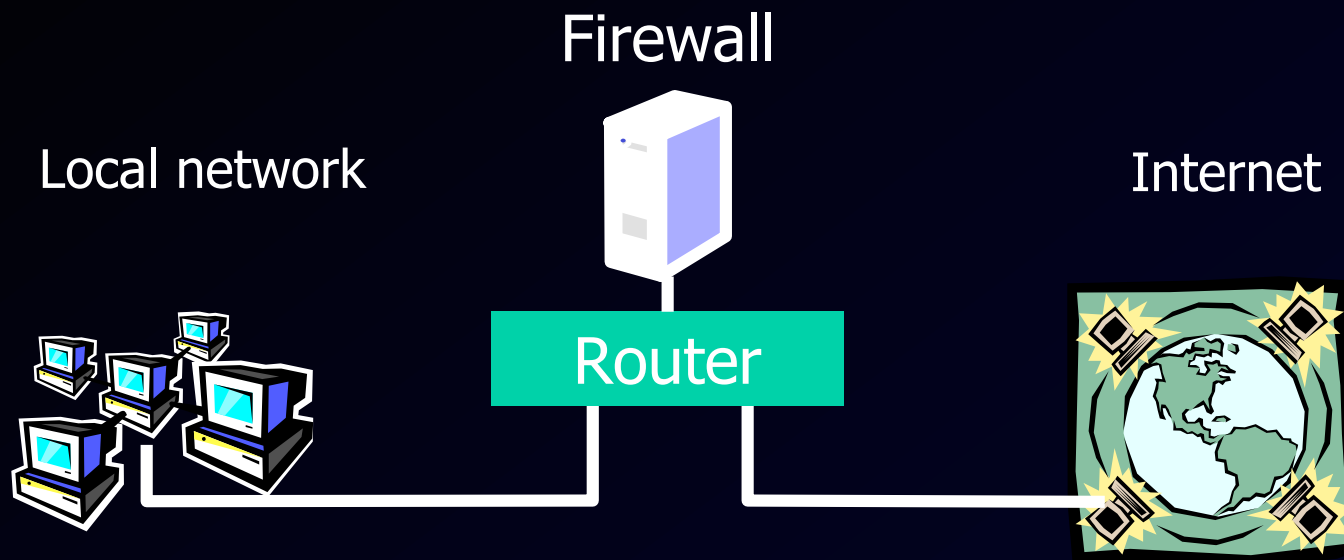
- **Given a numeric IP address, reverse DNS finds the associated domain name.**
- **E.g. to find 222.33.44.5,**
  - **Query 5.44.33.222.in-addr.arpa**
  - **Get list of symbolic addresses, e.g.,**
    - 1 IN PTR server.small.com
    - 2 IN PTR boss.small.com
    - 3 IN PTR ws1.small.com
    - 4 IN PTR ws2.small.com
- **One problem: Reverse DNS is controlled by the IP address holder.**

# RSH ATTACK II

- 1. Gain control of reverse DNS service for evil.org**
  - 2. Select target machine in good.net**
  - 3. Find trust relationships**
    - SNMP, finger can find active sessions, etc.
    - Example: target trusts host1.good.net
  - 4. Connect**
    - Attempt rlogin from coyote.evil.org
    - Target contacts reverse DNS server with IP addr
    - Use modified reverse DNS to say "addr belongs to host1.good.net"
    - Target allows rlogin
- "Cache poisoning" wrecks the obvious fix**

# FIREWALLS

A **firewall** is a router that sits between a local network and the internet. All packets from the LAN are routed through the firewall.



To exploit a hole in my software you need to talk to it.

With diverse software, firewalls provide **defense in depth**.

# PACKET FILTERING

- **Basic packet filtering uses only “transport-layer” or header information, e.g.**
  - Source and destination host and port
  - Protocol, flags, packet length, message type
- **Examples:**
  - HTTP uses port 80: allow internal connections to external port 80.
  - SMTP uses port 25: Block connections from external hosts to port 25, except on “gateway” mail host.

**A**

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

**B**

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

**C**

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

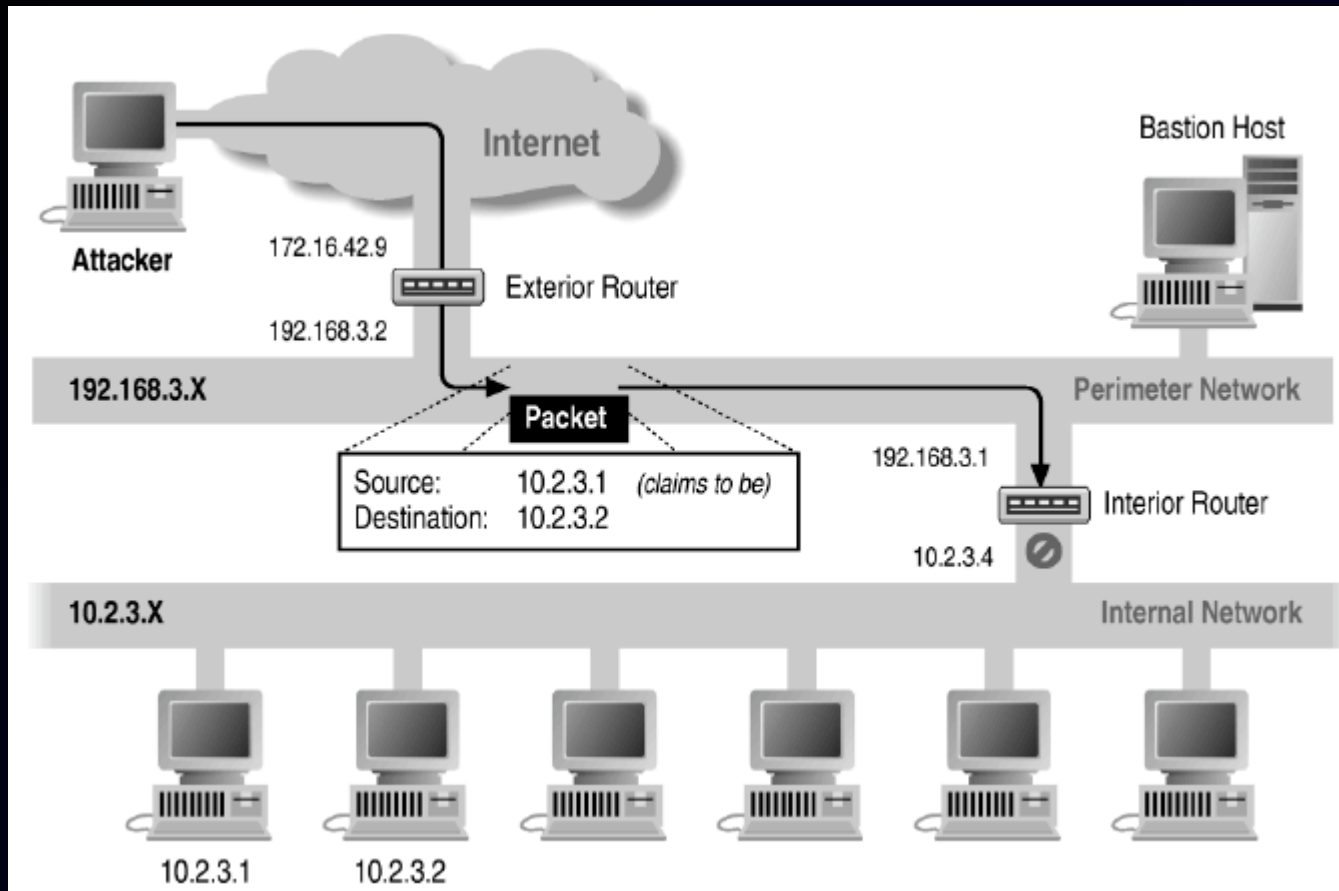
**D**

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

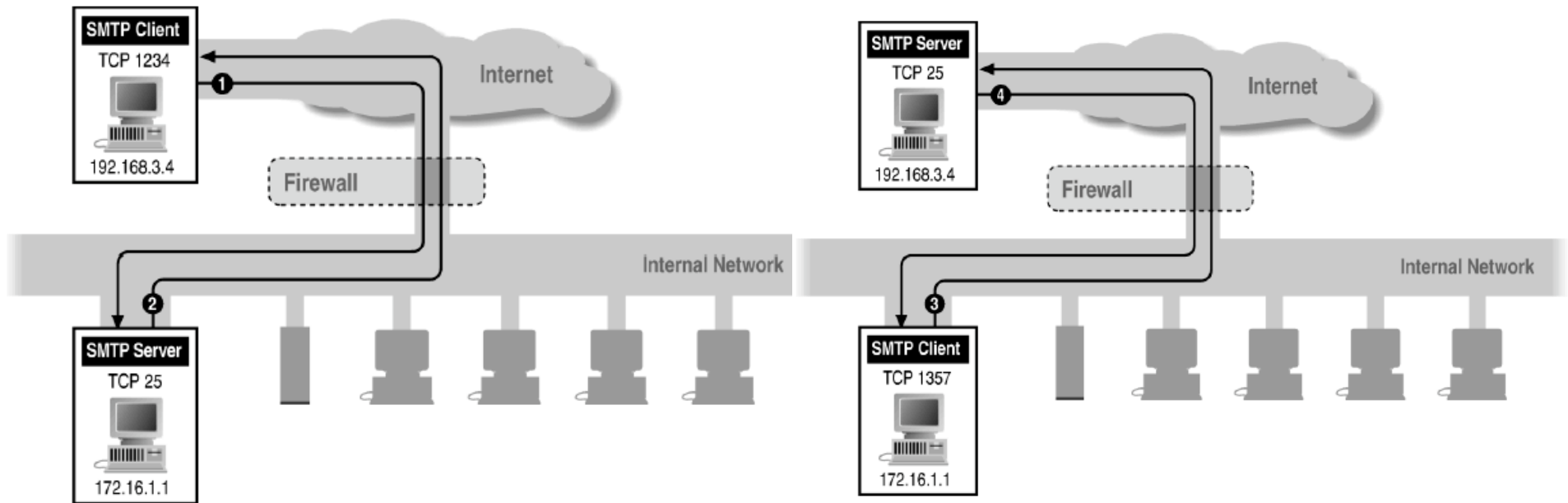
**E**

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

# ADDRESS FORGERY



# EXAMPLE: SMTP

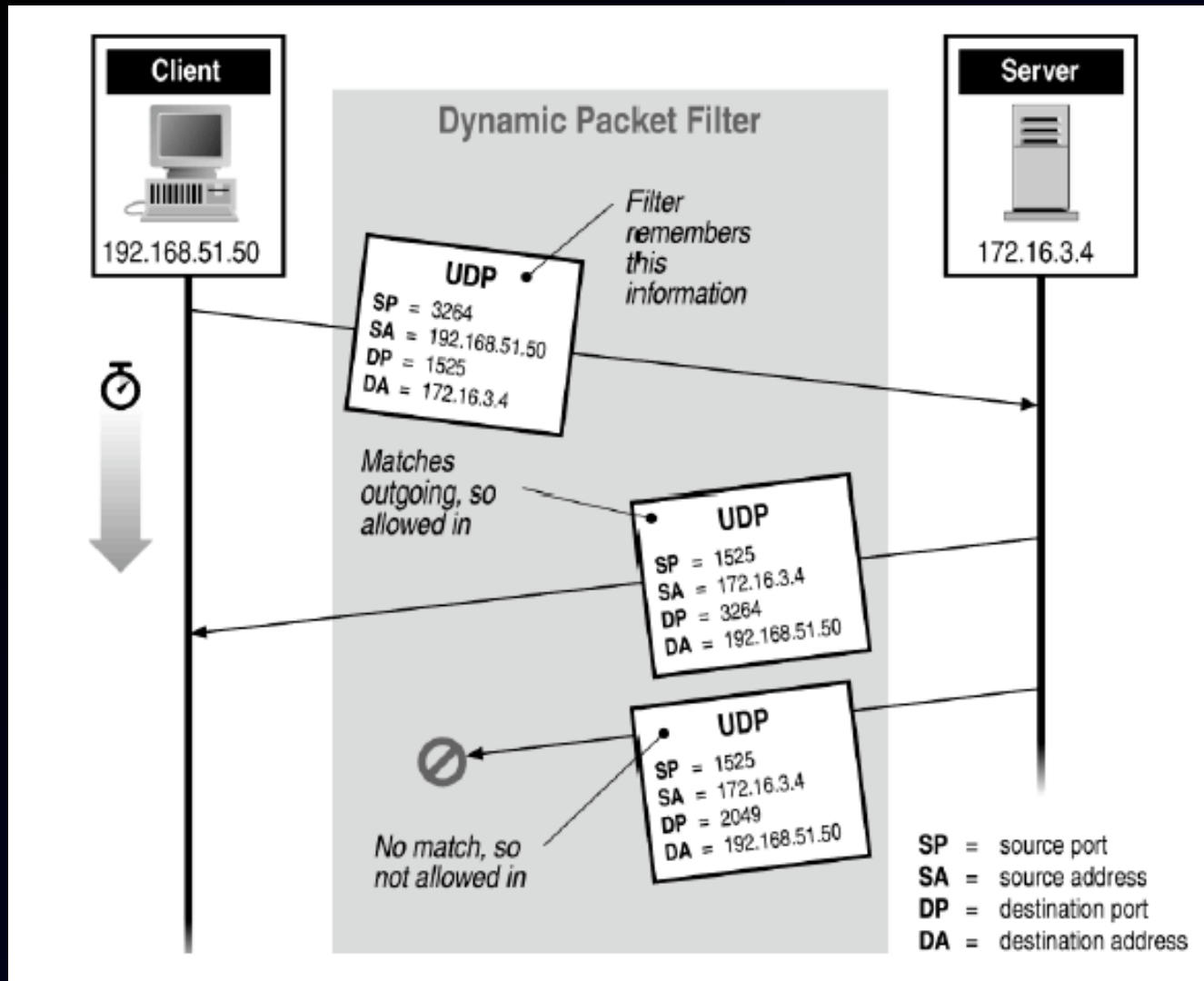


**Block external requests to internal server based on port number**  
**Internal requests to external servers will use known port out ... ?**

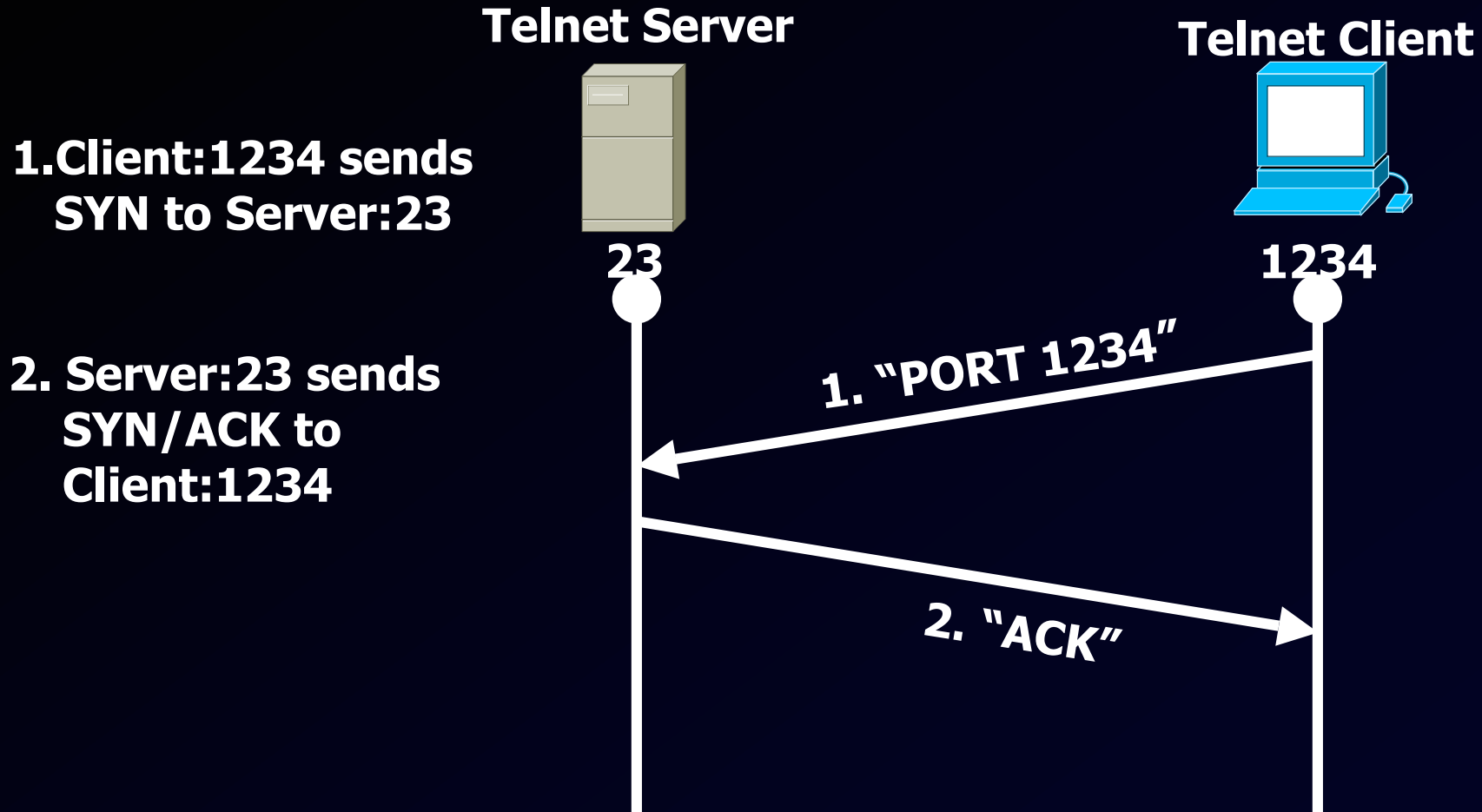
# PORT NUMBERS

- TCP connections typically have a "Server port"  $< 1024$ , or a "client port"  $> 1024$ .
- "Server" ports  $< 1024$  are standardized, e.g.
  - 20,21 for FTP
  - 23 for Telnet
  - 25 for SMTP
  - 80 for HTTP
  - 22 for SSH
  - (see /etc/services)
- Typically ports  $> 1024$  are used "as needed"
  - Ports  $> 1024$  must be "open" for clients to make connections!
- A **Stateful packet filter** watches outgoing requests so that it knows what incoming traffic to allow.

# STATEFUL PACKET FILTERING



# EXAMPLE: TELNET



Firewall sets "allow" rule for Server:23 to Client:1234 until closed

# EXAMPLE: FTP

FTP Server

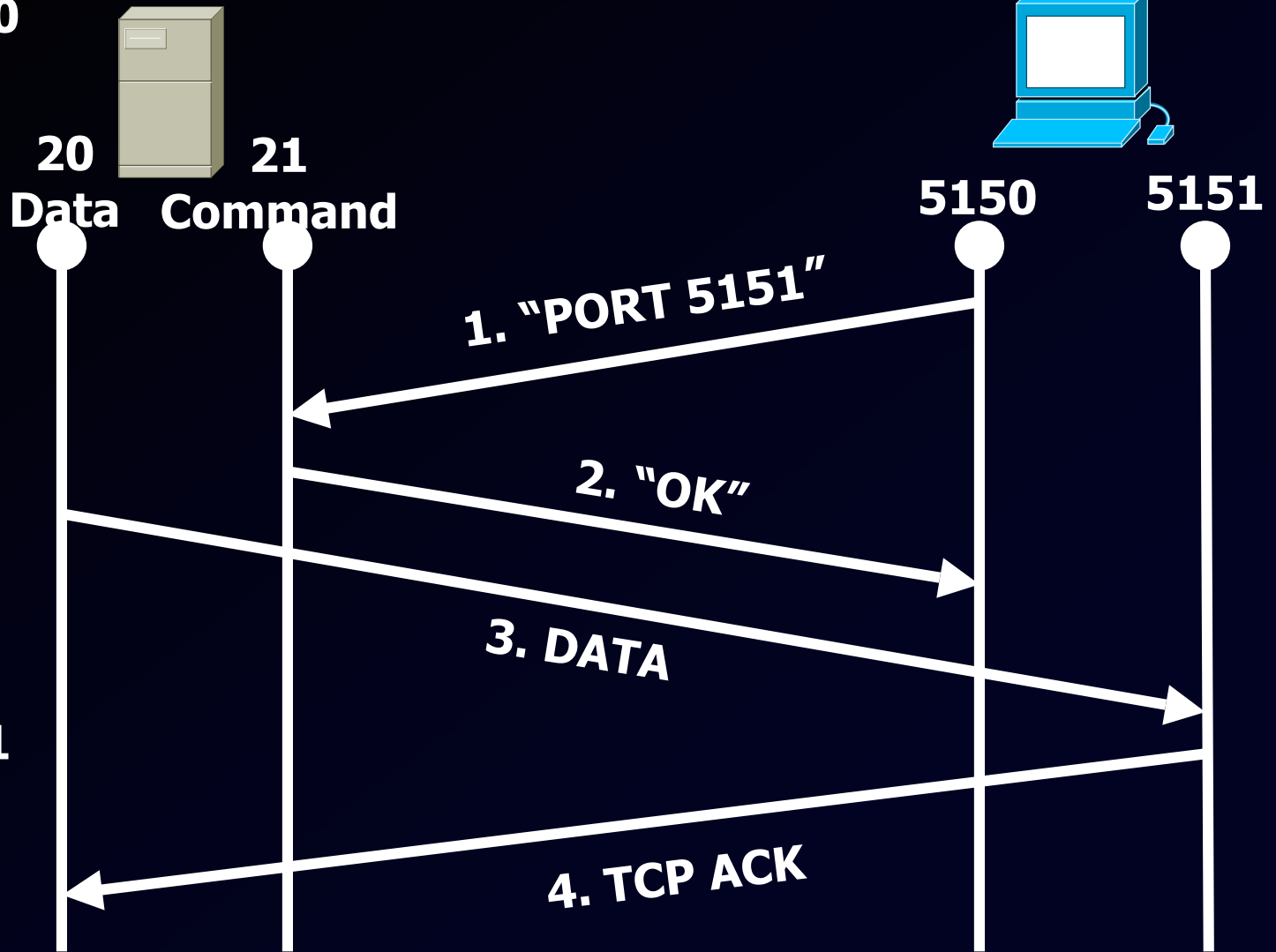
FTP Client

1. Client:5150 connects to server:21; sends 5151

2. Server:21 sends ACK to client:5150

3. Server:20 connects to client:5151

4. Client:5151 sends ACK to server:20



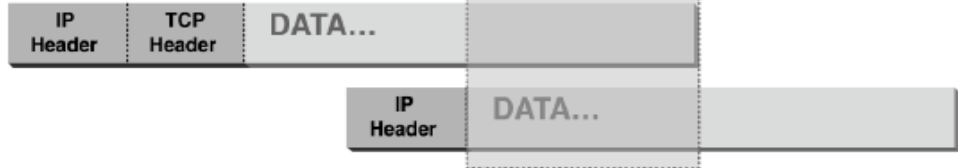
# FRAGMENTATION



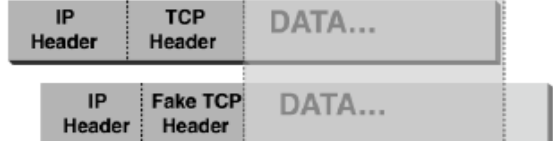
Normal



Overlapping data



Overlapping headers



# PROXIES

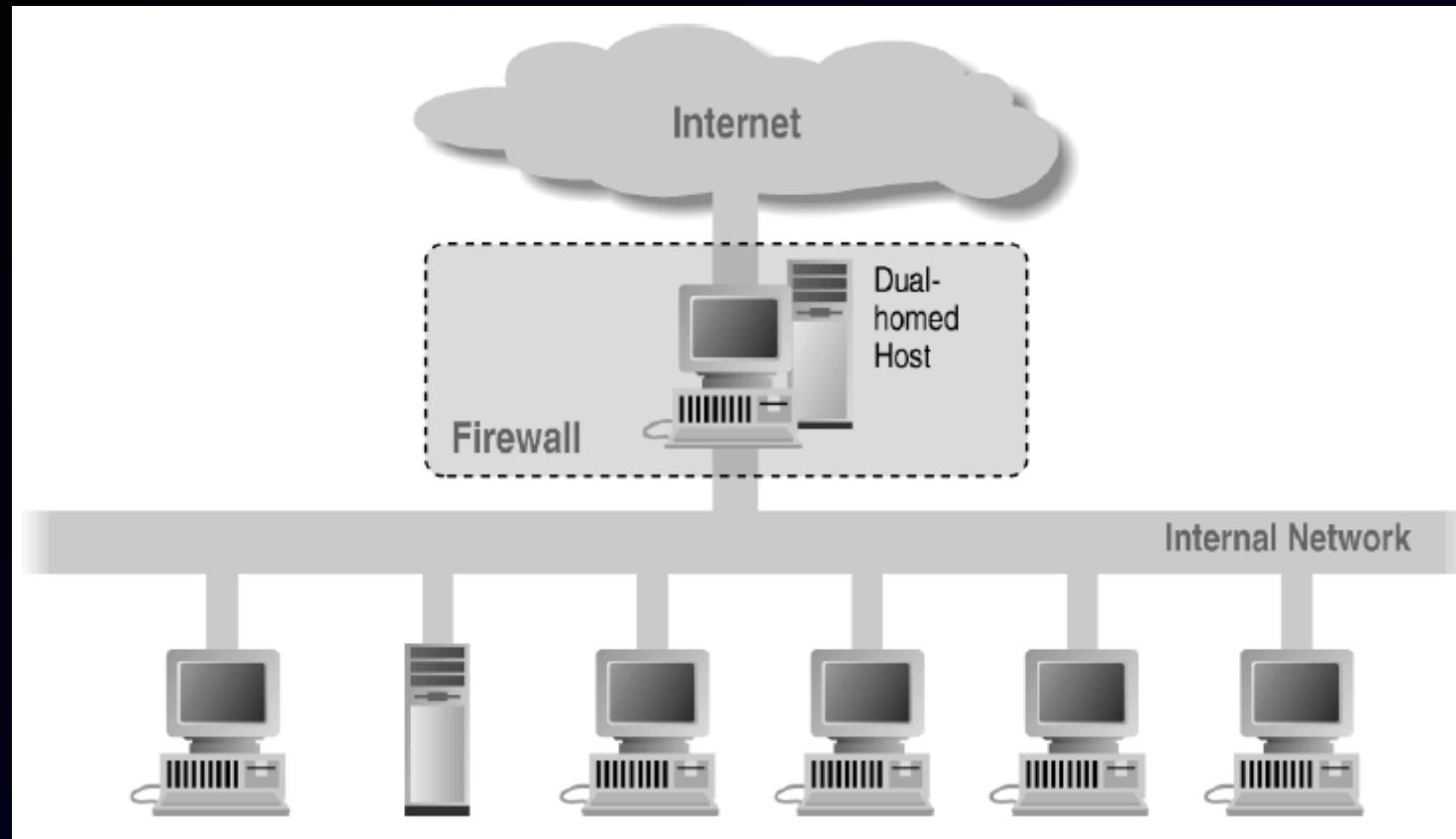
A **proxy** is a host that relays connections to external machines. Incoming connections go to the proxy, and outgoing packets appear to come from it.

The proxy can enforce policies for specific protocols, e.g. virus scanning for SMTP or “NetNanny” list for HTTP.

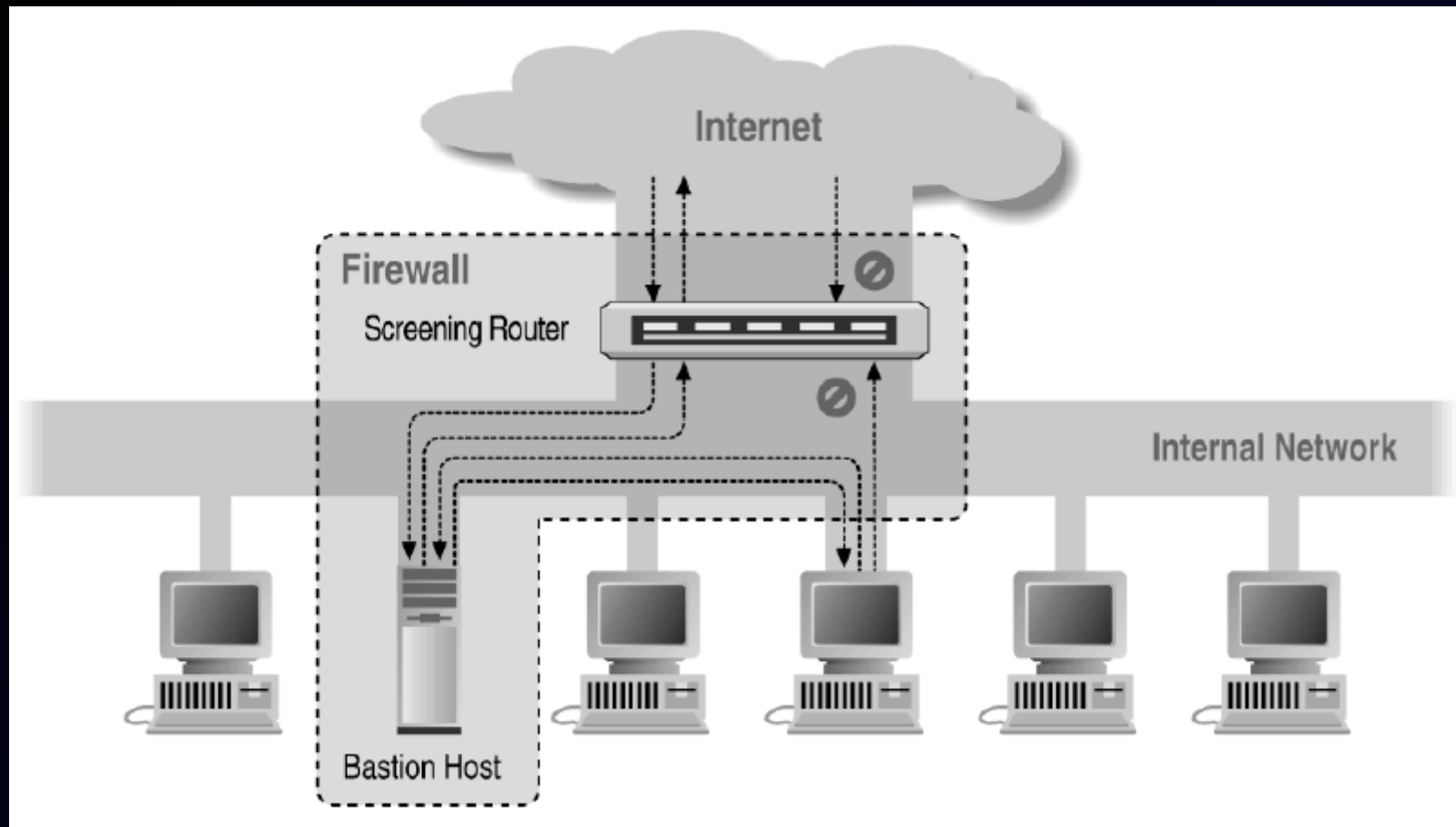
A proxy host (sometimes called a “bastion”) must be carefully secured – where should it be located?

There are two basic types of proxy programs: circuit-level and application-level.

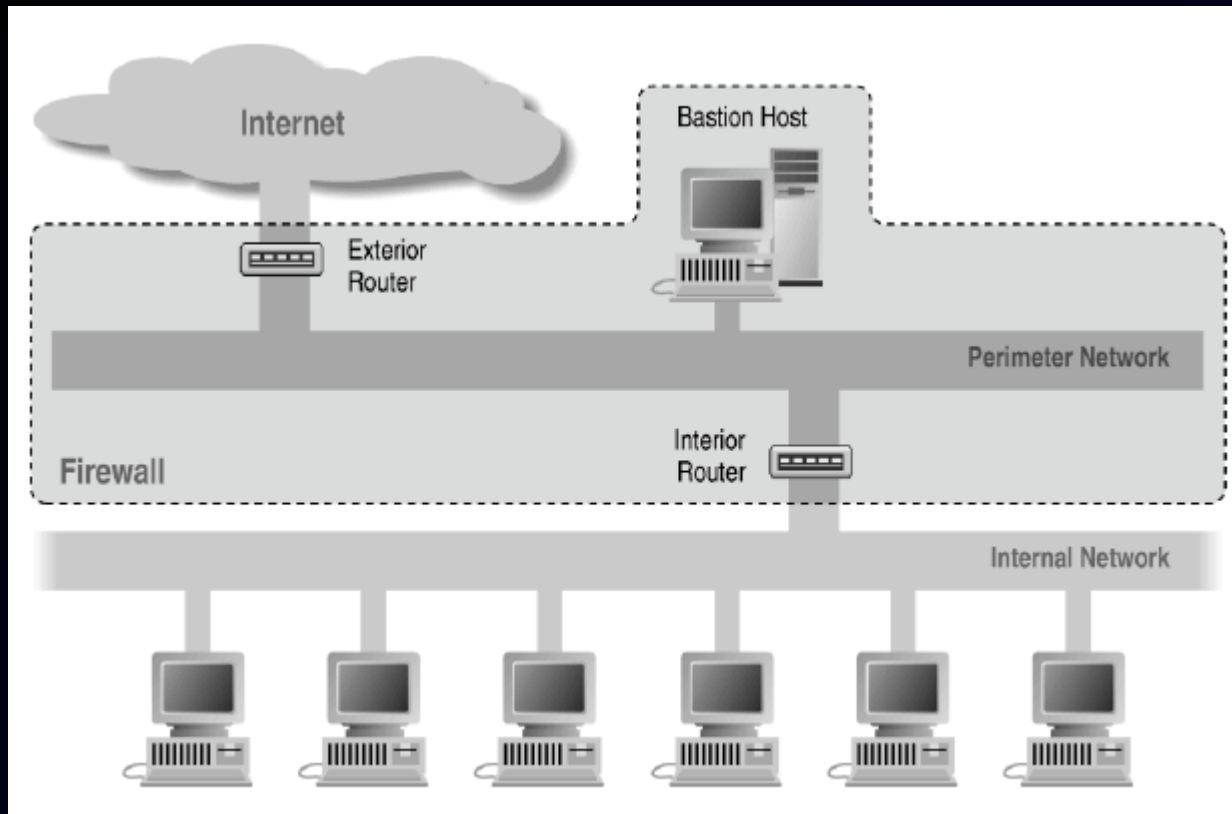
# DUAL-HOMED PROXY



# SCREENED HOST PROXY



# SCREENED SUBNET PROXY



# CIRCUIT-LEVEL GATEWAYS

- ... Relay the contents of two TCP connections: Internal to gateway and gateway to External.
- ... Can simplify dynamic packet filtering, e.g.  
**IP Fragmentation:**
  - streams are reassembled at the proxy**Dynamic filtering:**
  - allow packets from server ports to any port on the proxy
- ... Typically log connections, ports, and number of bytes, but nothing else.
- Popular examples include:
  - SOCKS: a “Drop-in replacement” for the TCP API; just replace system calls with the equivalent SOCKS calls.
  - WinSOCK: a “nearly generic” proxy for microsoft

# **APPLICATION-LEVEL PROXIES**

**... are proxies dedicated to a single application protocol, like HTTP or FTP.**

**... Understand applications, so they can enforce policies, e.g.**

- Easy: validate email headers, filter URLs, etc...**
- Harder: inspect for malware...**

**... are natural for some applications, e.g. SMTP (E-Mail), NNTP (Net news), DNS (Domain Name System), NTP (Network Time Protocol)**

**... can also improve efficiency via caching.**

# PERSONAL FIREWALLS

- A **software firewall** is an application or OS module that filters packets at the host level. This can afford some protection against internal hosts
- Software firewalls can offer flexibility, e.g.:
  - Filter based on calling process: Firefox or Opera, but not IE
  - Pop-up dialog on connection attempts...
- Not effective after host compromise!
- Some popular examples: ipchains, ZoneAlarm, Kerio, Windows Firewall

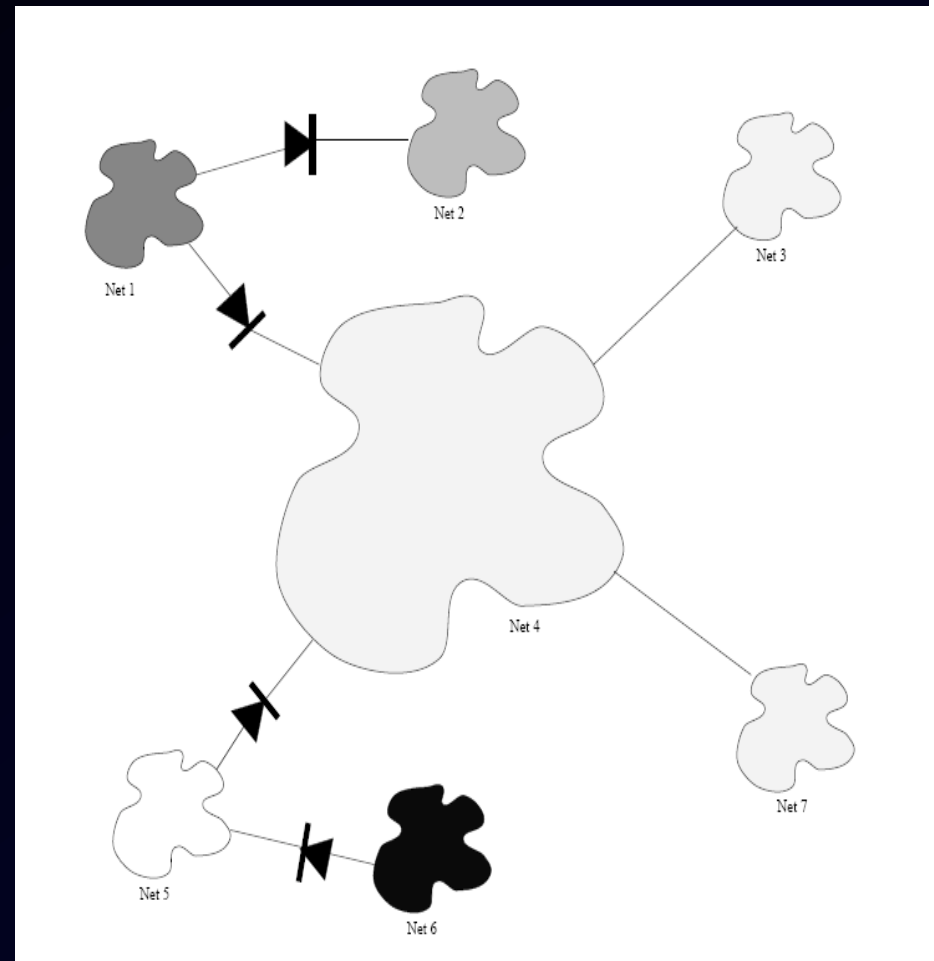
# NETWORK ARCHITECTURE

**Within an organization, users need access to different machines:**

- **Separate domains: compartmentalization**

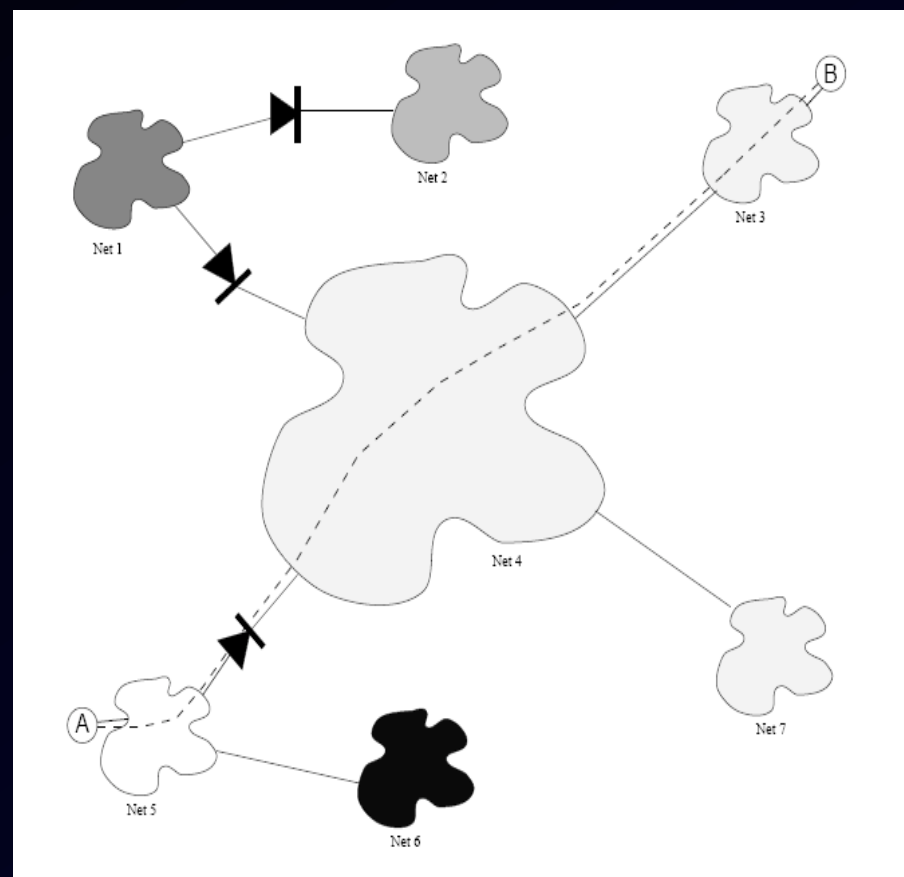
**Some domains may need mutual access**

- **E.g., support personnel**
- **Authentication becomes important. (why?)**



# TUNNELS

- **IP can run over any “link layer:”**
  - Ethernet, 802.11, ATM, Token Ring, FDDI...
  - **BUT Also: IP, TCP, HTTP, DNS...**
  - As with IPsec, this is called “encapsulation”
- A “Tunnel” is a connection that uses encapsulation over some network links.
- Tunnels can “burrow under” a firewall.



# TUNNELS BAD...

- **Tunneling over other protocols avoids firewall policy, a la Skype.**
- **Minimally this allows an external host to access internal hosts**
- **In the worst case, an external host might advertise routes to internal hosts over the tunnel!**

# ... AND TUNNELS GOOD

- If tunnel endpoints are trustworthy, they can extend the internal network, creating a **Virtual Private Network (VPN)**
- This allows building a “private” network by tunneling over public links, so remote networks can:
  - Share server resources
  - Simplify videoconferencing
  - Use address-based authentication
  - Etc...
- Typically VPNs use encryption and MACs to authenticate tunnels.
- The most common protocols for this are IPsec, SSH, and PPTP.

# **FIREWALL PROBLEMS**

- **Performance**
- **Limitations**
  - **Don't solve the real problems**
  - **Cannot prevent Denial of Service**
  - **Cannot prevent application-level attacks**
  - **Do not prevent insider attacks**
- **Administration**
  - **Many commercial firewalls permit very complex configurations**
  - **Implementations vary (ruleset ordering, interface filtering, etc...)**