CMPE 209 Network Security

Chapter 9. Firewalls

Dr. Younghee Park

SAN JOSÉ STATE UNIVERSITY

Introduction

- Internet connectivity is essential
  - However it creates a threat

- The Need For Firewalls
  - Effective means of protecting LANs
  - Inserted between the premises network and the Internet to establish a controlled link
    - Can be a single computer system or a set of two or more systems working together

Design Goals for Firewall

- Used as a perimeter defense
  - Single choke point to impose security and auditing
  - Insulates the internal systems from external networks

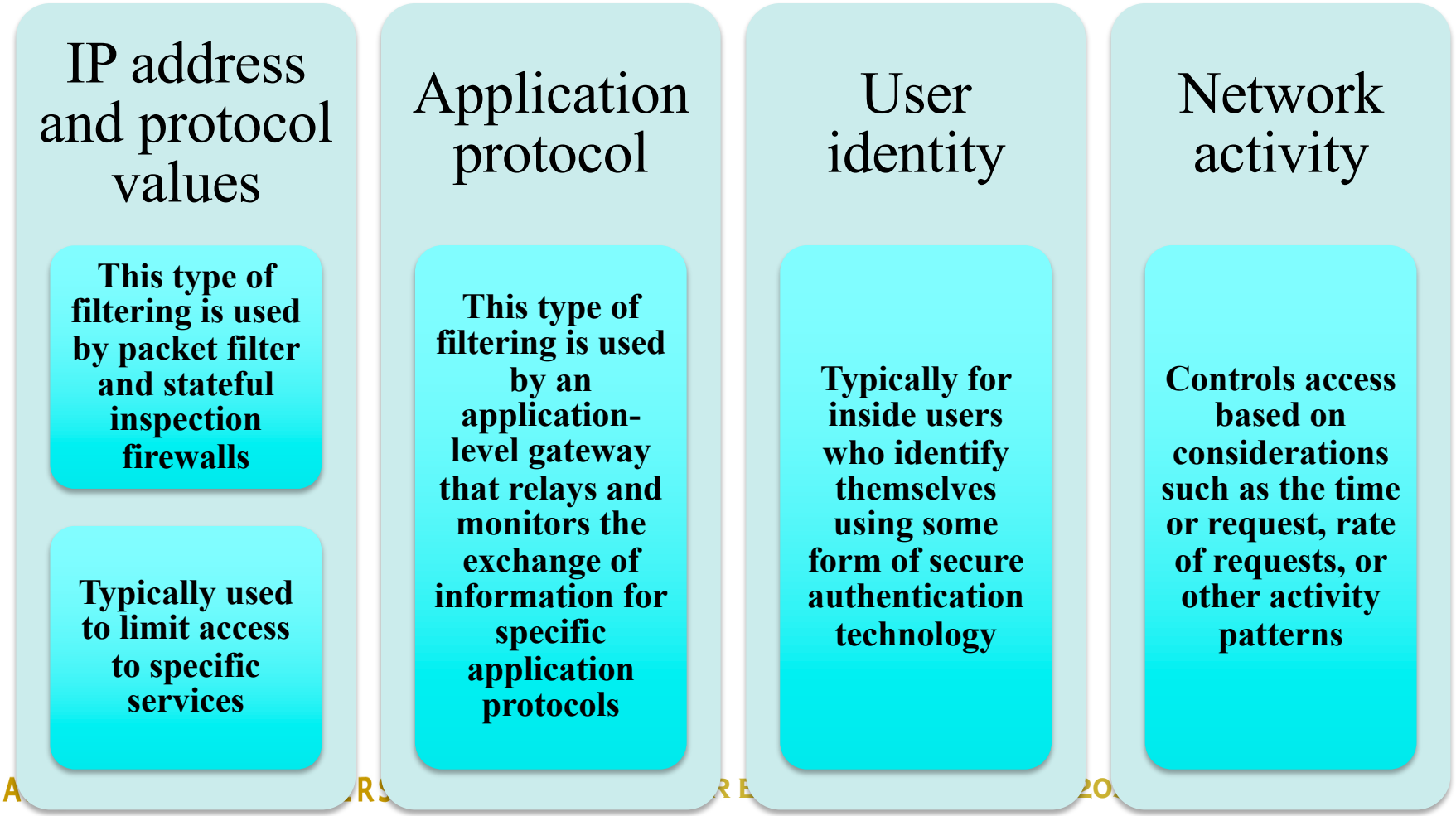
- All traffic from inside to outside, and vice versa, must pass through the firewall
- Only authorized traffic as defined by the local security policy will be allowed to pass

Firewall Access Policy

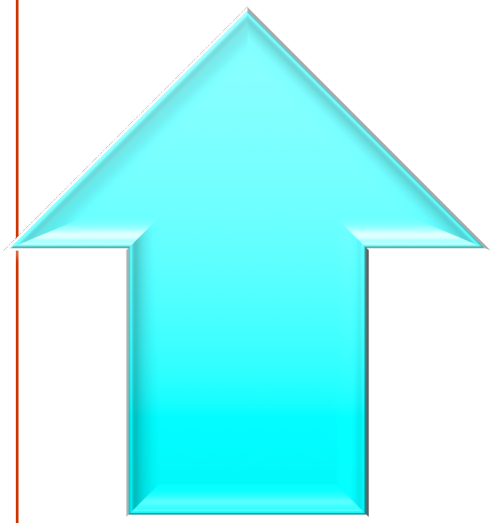
- A critical component in the implementation of a firewall is specifying a suitable access policy
    - This lists the types of traffic authorized to pass through the firewall
    - Includes address ranges, protocols, applications and content types
- This policy should be developed from organization's information security risk assessment and policy
- Should be developed from a broad specification of which traffic types the organization needs to support
    - Then refined to detail the filter elements which can then be implemented within an appropriate firewall topology

Firewall Filter Characteristics

- Characteristics that a firewall access policy could use to filter traffic include:



Firewall Capabilities And Limits



Capabilities:

- **Defines a single choke point**
- **Provides a location for monitoring security events**
- **Convenient platform for several Internet functions that are not security related**
- **Can serve as the platform for IPSec**

Limitations:

- **Cannot protect against attacks bypassing firewall**
- **May not protect fully against internal threats**
- **Improperly secured wireless LAN can be accessed from outside the organization**
- **Laptop, PDA, or portable storage device may be infected outside the corporate network then used internally**

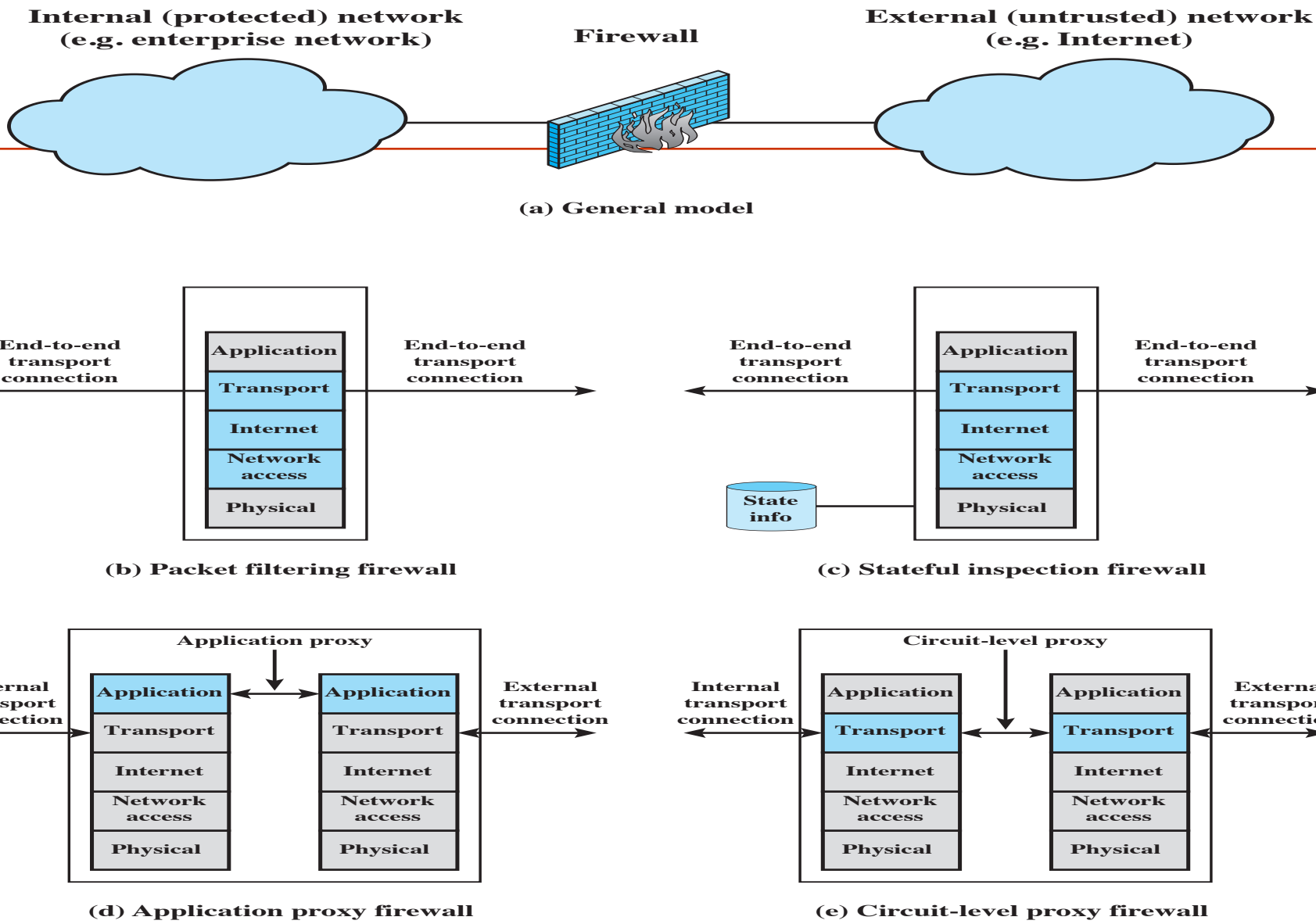
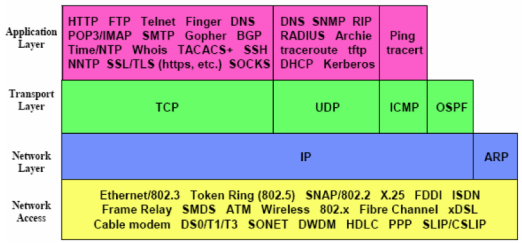


Figure 9.1 Types of Firewalls



Filtering

- Packet filtering
  - Access Control Lists
  - Decisions made on a per-packet basis
  - No state information saved
  - Typical configuration
    - Ports > 1024 left open
    - If dynamic protocols are in use, *entire ranges of ports must be allowed* for the protocol to work.
- Session filtering (Stateful Inspection Firewall)
  - Dynamic Packet Filtering
  - Stateful Inspection
  - Context based access control for connections

Packet Filtering Firewall

- Applies rules to each incoming and outgoing IP packet
  - Typically a list of rules based on matches in the IP or TCP header
  - Forwards or discards the packet based on rules match

Filtering rules are based on information contained in a network packet

- **Source IP address**
- **Destination IP address**
- **Source and destination transport-level address**
- **IP protocol field**
- **Firewall Interface**

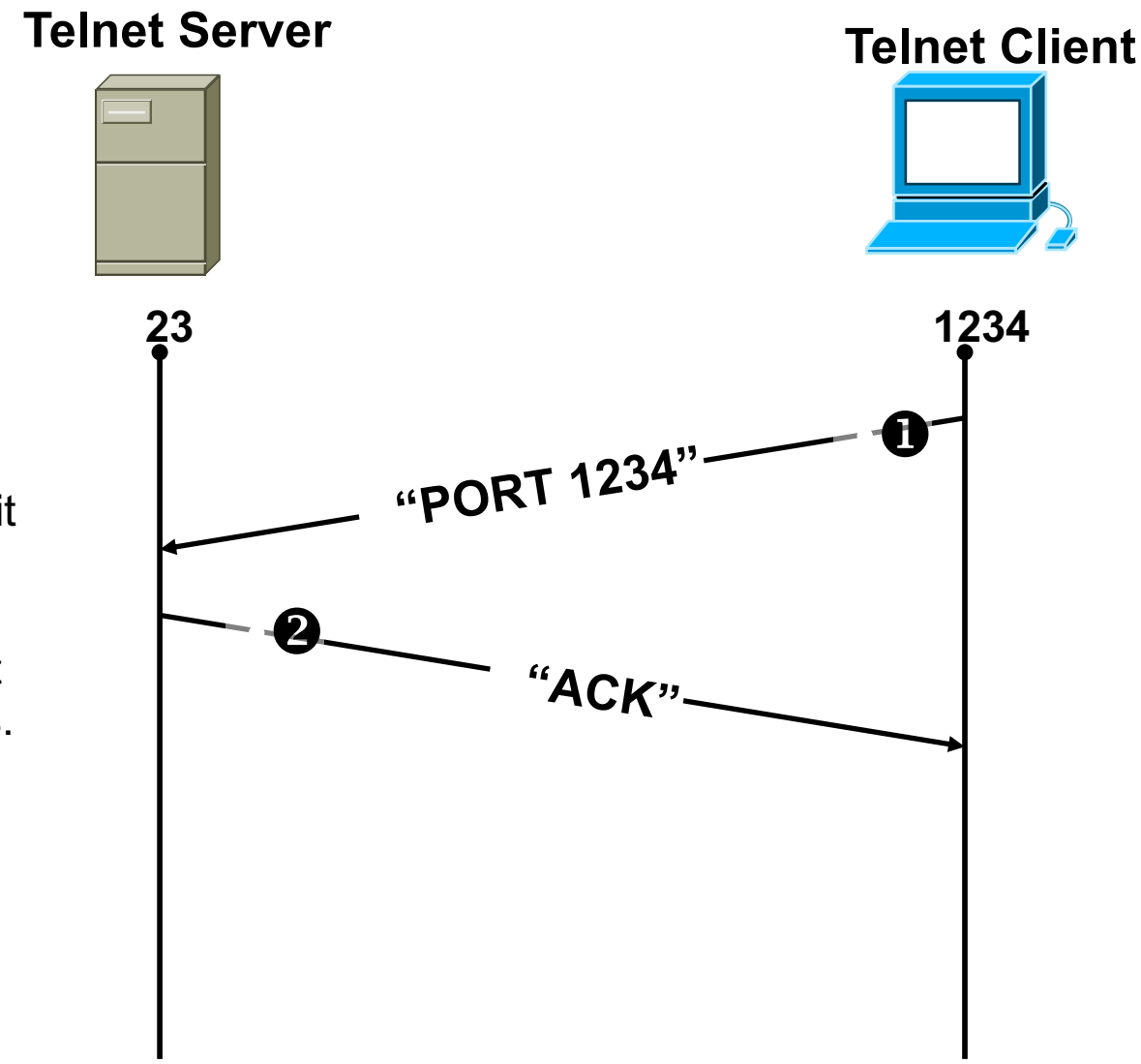
- Two default policies:
  - Discard - prohibit unless expressly permitted
    - More conservative, controlled, visible to users
  - Forward - permit unless expressly prohibited
    - Easier to manage and use but less secure

Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Table 9.1 is a simplified example of a rule set for SMTP traffic.

Telnet



❶ Client opens channel to server; tells server its port number. The ACK bit is not set while establishing the connection but will be set on the remaining packets.

❷ Server acknowleges.

Example: Telnet

Format:

access-list *<rule number> <permit|deny> <protocol> <SOURCE host with IP address| any|IP address and mask> [<gt|eq port number>] <DEST host with IP address| any|IP address and mask> [<gt|eq port number>]*

The following allows user to telnet from an IP address (172.168.10.11) to any destination, but not vice-versa:

```
access-list 100 permit tcp host 172.168.10.11 gt 1023 any eq 23
  ! Allows packets out to remote Telnet servers
access-list 101 permit tcp any eq 23 host 172.168.10.11 established
  ! Allows returning packets to come back in.  It verifies that the ACK bit is set

interface Ethernet 0
 access-list 100 out   ! Apply the first rule to outbound traffic
 access-list 101 in    ! Apply the second rule to inbound traffic
```

- ***Note*: anything not explicitly permitted in an access-list is denied.**

Pros and Cons of Packet Filter

- Advantages
  - Simplicity
  - Typically transparent to users and are very fast
- Weaknesses
  - Cannot prevent attacks that employ application specific vulnerabilities or functions
  - Limited logging functionality
  - Do not support advanced user authentication
  - Vulnerable to attacks on TCP/IP protocol bugs

    (e.g. *network layer address spoofing)*
  - Improper configuration can lead to breaches

Some Attacks on packet filtering

- IP address spoofing : The intruder transmits packets from the outside with a source IP address field containing an address of an internal host. The attacker hopes that the use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from specific trusted internal hosts are accepted. The countermeasure is to discard packets with an inside source address if the packet arrives on an external interface. In fact, this countermeasure is often implemented at the router external to the firewall.

Some Attacks on packet filtering

- Source routing attacks: The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. A countermeasure is to discard all packets that use this option.

Some Attacks on packet filtering

- **Tiny fragment attacks** : The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. This attack is designed to circumvent filtering rules that depend on TCP header information. Typically, a packet filter will make a filtering decision on the first fragment of a packet. All subsequent fragments of that packet are filtered out solely on the basis that they are part of the packet whose first fragment was rejected. **The attacker hopes that the filtering firewall examines only the first fragment and that the remaining fragments are passed through.** A tiny fragment attack can be defeated by enforcing a rule that the first fragment of a packet must contain a predefined minimum amount of the transport header. If the first fragment is rejected, the filter can remember the packet and discard all subsequent fragments.

Stateful Inspection Firewall

Tightens rules for TCP traffic by creating a directory of outbound TCP connections

- **There is an entry for each currently established connection**
- **Packet filter allows incoming traffic to high numbered ports only for those packets that fit the profile of one of the entries in this directory**

Reviews packet information but also records information about TCP connections

- **Keeps track of TCP sequence numbers to prevent attacks that depend on the sequence number**
- **Inspects data for protocols like FTP, IM and SIPS commands**

Table 9.2 Example Stateful Firewall Connection State Table

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

Application-Level Gateway

- Also called an application proxy
- Acts as a relay of application-level traffic
  - User contacts gateway using a TCP/IP application
  - User is authenticated
  - Gateway contacts application on remote host and relays TCP segments between server and user
- Must have proxy code for each application
  - May restrict application features supported
- Tend to be more secure than packet filters
- Disadvantage is the additional processing overhead on each connection

Application Gateways

- Understands specific applications
  - Limited proxies available
  - Proxy 'impersonates' both sides of connection
- Resource intensive
  - process per connection
- Must write a new proxy application to support new protocols - Not trivial!
- Example
  - HTTP proxies may cache web pages
  - *Block all unless specifically allowed*

Circuit-Level Gateway

- A circuit-level gateway is a type of firewall.
- Circuit level gateways work at the session layer of the OSI model, or as a "shim-layer" between the application layer and the transport layer of the TCP/IP stack. They monitor TCP handshaking between packets to determine whether a requested session is legitimate.

Circuit-Level Gateway

Circuit level proxy

- Sets up two TCP connections, one between itself and a TCP user on an inner host and one on an outside host
- Relays TCP segments from one connection to the other without examining contents
- Security function consists of determining which connections will be allowed

Typically used when inside users are trusted

- May use application-level gateway inbound and circuit-level gateway outbound
- Lower overheads

SOCKS

- **Socket Secure (SOCKS) is an Internet protocol that routes network packets between a client and server through a proxy server.**
- **OpenSSH** allows dynamic creation of tunnels, specified via a subset of the SOCKS protocol, supporting the CONNECT command.
- **PuTTY** is a Win32 SSH client that supports local creation of SOCKS (dynamic) tunnels through remote SSH servers.
- **Tor** is a system intended to enable online anonymity. Tor offers a SOCKS server interface to its clients.

Comparison

	Security	Performance	Service Support
Packet Filter	3	1	No dynamic w/o holes
Session Filter	2	2	Dependent on vendor for dynamic support
Circuit GW	2	3	
App. GW	1	4	Typically < 20

Lower is better for security & performance

Comparison (Cont'd)

	Modify Client Applications?
Packet Filter	No
Session Filter	No
Circuit GW	Typical, SOCKS-ify client applications
App. GW	Unless transparent, client application must be proxy-aware & configured

Comparison (Cont'd)

	ICMP	Fragmentation
Packet Filter	Yes	No
Session Filter	Yes	Maybe
Circuit GW	(SOCKS v5)	Yes
App. GW	No	Yes

Linux Firewall: iptables

- Building blocks of Linux firewall framework
- iptables is **a user-space application program** that allows a system administrator **to configure the tables** provided by the Linux kernel firewall (implemented as different Netfilter modules) and the chains and rules it stores. Different kernel modules and programs are currently used for different protocols; **iptables** applies to IPv4, **ip6tables** to IPv6, **arptables** to ARP, and **ebtables** to Ethernet frames.
- https://en.wikipedia.org/wiki/Iptables

IPTables

- A packet selection system
  - Direct descendent of ipchains
- Used for
  - Packet filtering
  - Session filtering
  - Network Address Translation (NAT)
    - Masquerading, port forwarding, transparent proxying

User Space Tool: iptables

- iptables
  - Command to configure and communicate with the kernel modules
- iptables for packet filtering
  - Three chains
    - INPUT
    - OUTPUT
    - FORWARD

Iptables for Packet Filtering

- You need three things to configure a firewall rule
  - Which chain?
  - What packet pattern?
  - What action to apply?
- Example
  - Drop all packets from 200.200.200.1
  - iptables -A INPUT -s 200.200.200.1 -j DROP
  - Use "man iptables" on Linux to get more information.

Iptables for Session Filtering

- Similar to Packet Filtering
- Add rules for connection tracking
  - modules state and conntrack
    - state: match packet state
    - conntrack: recognize if a given packet belongs to an existing connection, and update connection table according to its decisions
  - mainly at pre-routing and local-out w/in netfilter

Iptables for Session Filtering (Cont'd)

- Fedora Core 1: /etc/sysconfig/iptables

```
# Firewall configuration written by redhat-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j
   ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -p tcp --dport SSH -j
   ACCEPT
```

- Reference: https://help.ubuntu.com/community/IptablesHowTo

Iptables: extensible infrastructure

- Another extension: time
- adds CONFIG_IP_NF_MATCH_TIME, which supplies a time match module. This match allows you to filter based on the packet arrival time (arrival time at the machine which the netfilter is running on) or departure time (for locally generated packets).
- Supported options (all required for use):

```
--timestart HH:MM
--timestop HH:MM
--days Tue,Mon...
  Days of the week to match separated by
 a coma, no space
  (one of Sun,Mon,Tue,Wed,Thu,Fri,Sat)
```

Iptables: extensible infrastructure (Cont'd)

- Examples:
- Disable http access between 0400 and 0630 on Saturday and Sunday morning for routine maintenance on web server:

```
iptables -A INPUT -p tcp --dports 80 -m
time --timestart 04:00 --timestop 06:30
--Days Sat,Sun -j REJECT
```

- Disable outbound http access between 0800 and 1830 Monday through Friday to prevent employee misuse during business hours:

```
iptables -A OUTPUT -p tcp --dports 80 -m
time --timestart 08:00 --timestop 18:30 --
Days Mon,Tue,Wed,Thu,Fri -j REJECT
```

Firewall Topologies

Host-resident firewall	• Includes personal firewall software and firewall software on servers
Screening router	• Single router between internal and external networks with stateless or full packet filtering
Single bastion inline	• Single firewall device between an internal and external router
Single bastion T	• Has a third network interface on bastion to a DMZ where externally visible servers are placed
Double bastion inline	• DMZ is sandwiched between bastion firewalls
Double bastion T	• DMZ is on a separate network interface on the bastion firewall
Distributed firewall configuration	• Used by large businesses and government organizations

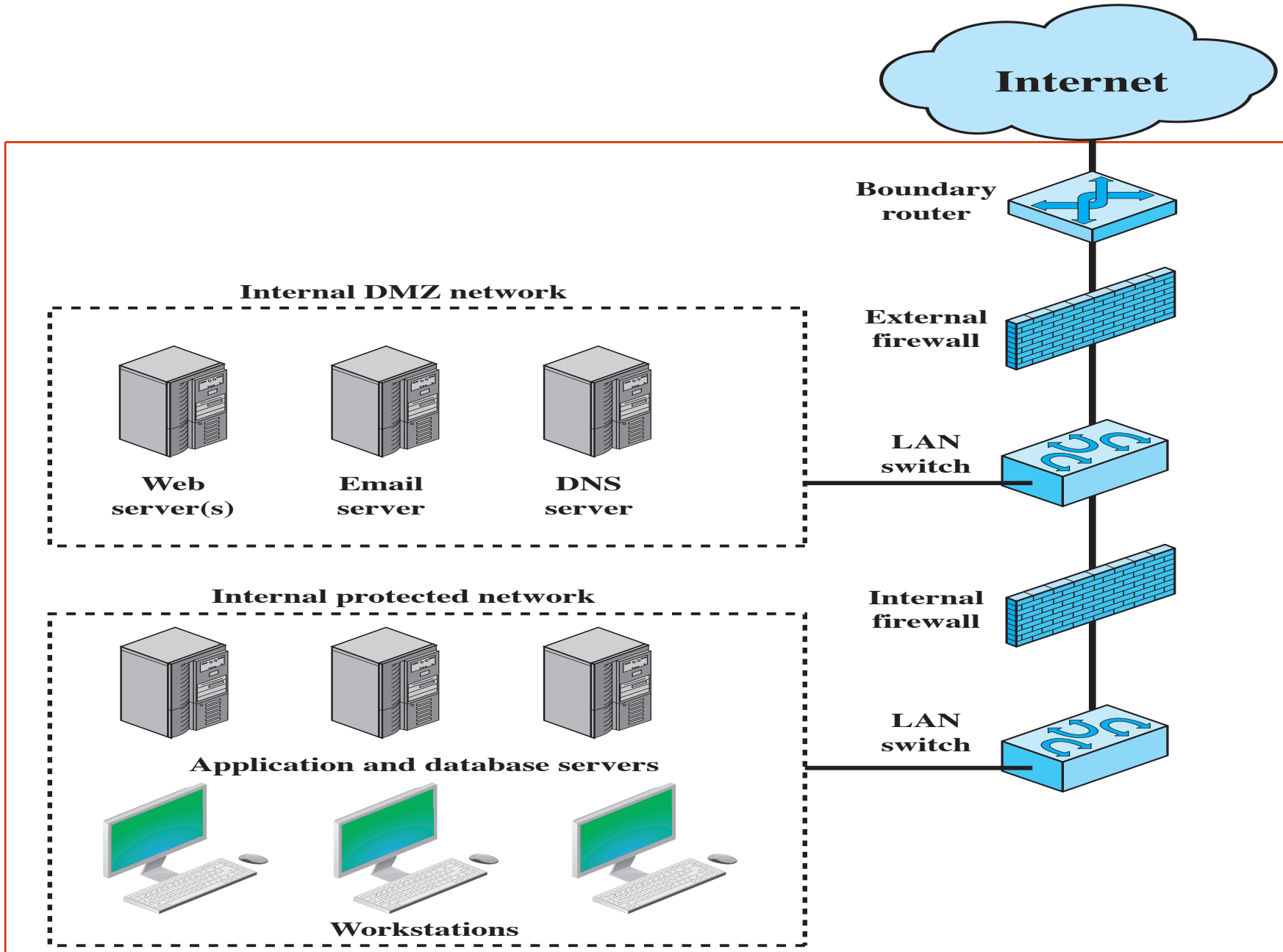


Figure 9.2 Example Firewall Configuration

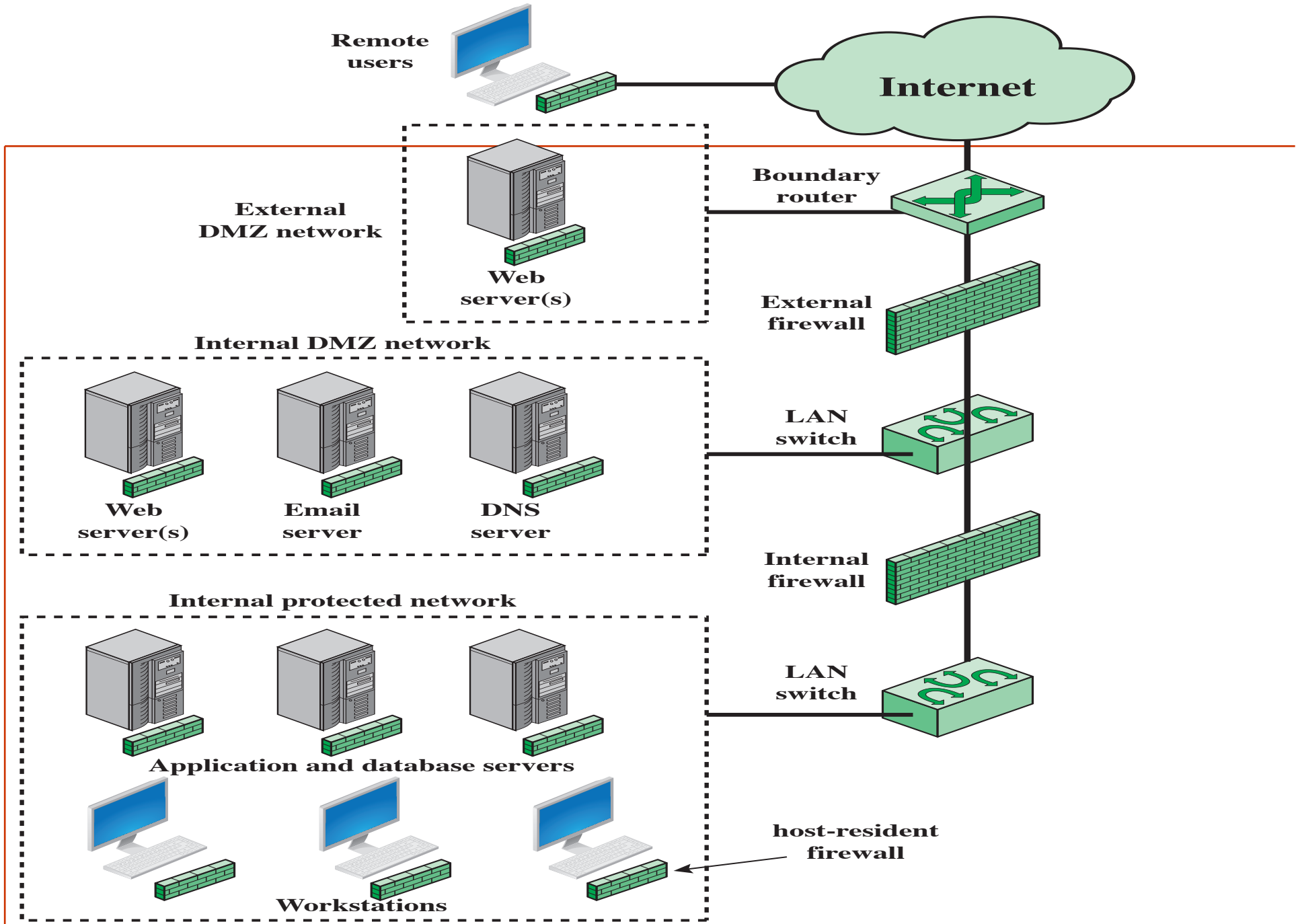


Figure 9.4 Example Distributed Firewall Configuration