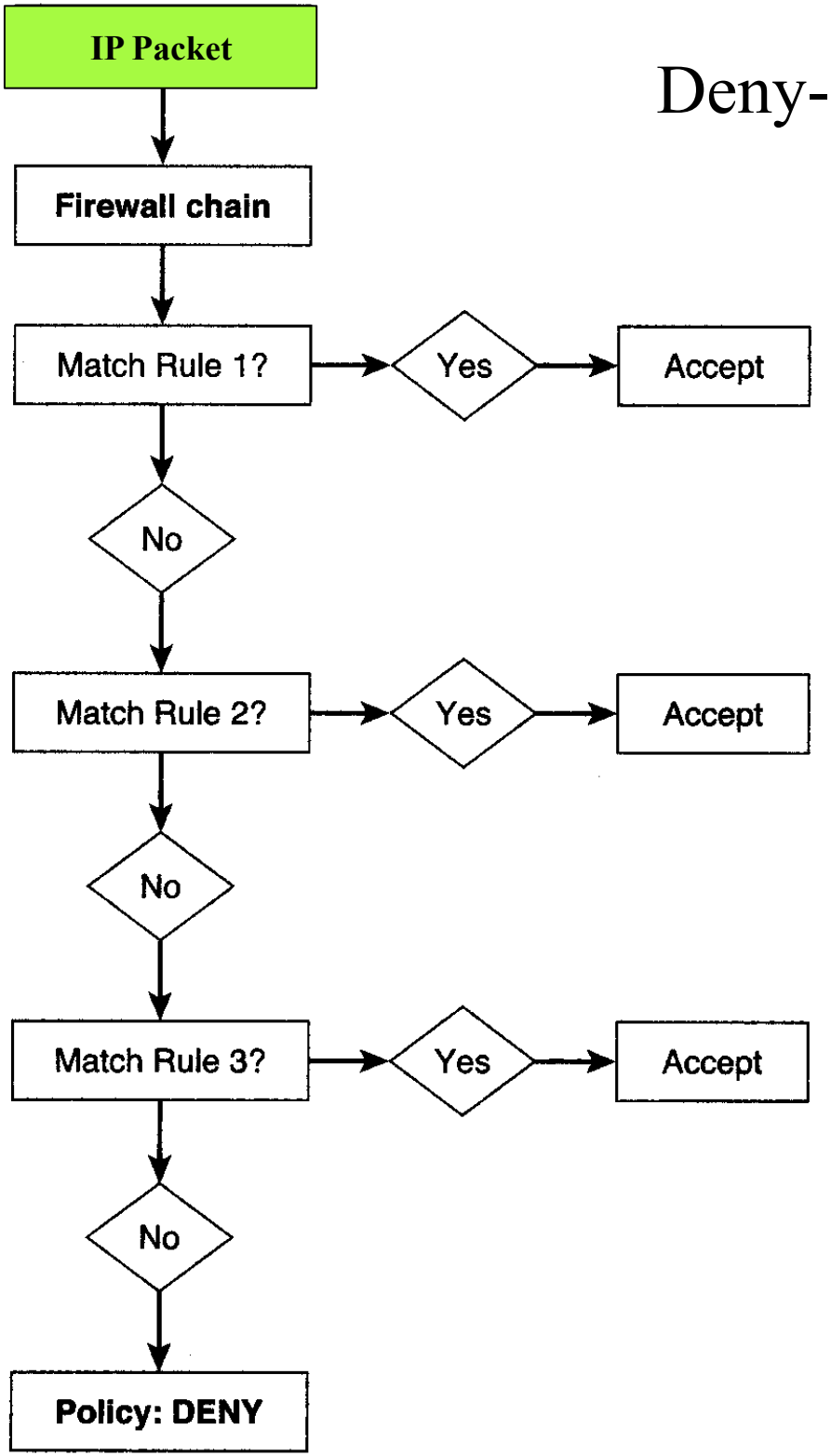
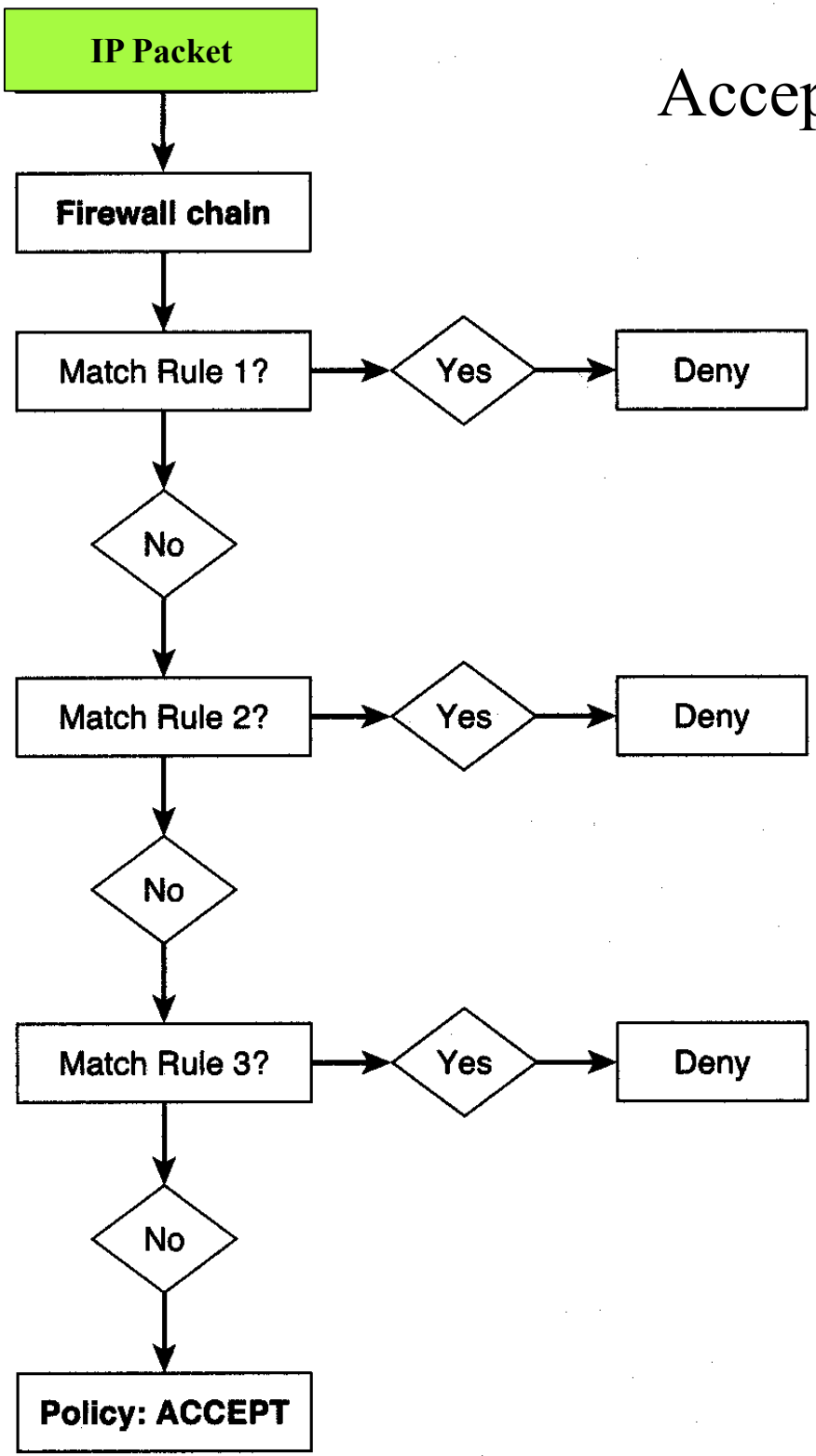
Deny-everything-by-default-policy



Accept-everything-by-default-policy

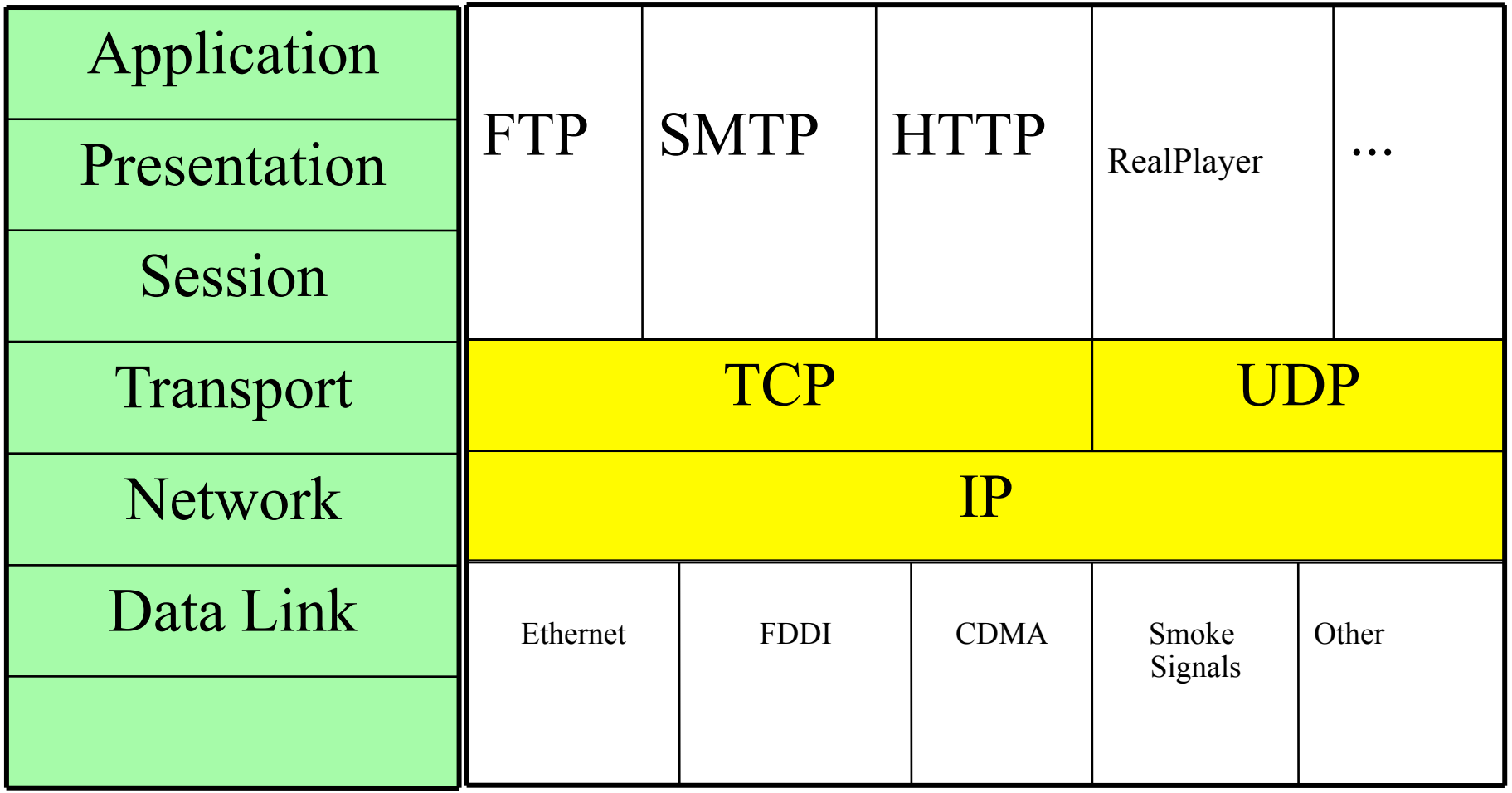


iptables syntax

```
iptables -I INPUT -i eth0 -p tcp -s
 192.168.56.1 --sport 1024:65535 -d
 192.168.56.2 --dport 22 -j ACCEPT
```

```
iptables -I OUTPUT -o eth1 -p tcp !
 --syn -s 192.168.56.2 --sport 22 -d
 192.168.56.1 --dport 1024:65535 -j
 ACCEPT
```

Packet Filter Layers



Command

- The compulsory command section of the command above is the most important part of the iptables command.
- It tells the iptables command what to do, for example, to insert a rule, to add a rule to the end of the chain, or to delete a rule.

Command

- The following are the most often-used commands:
  - -A or --append: This command appends a rule to the end of a chain. Example:
  - `$ iptables -A INPUT -s 205.168.0.1 -j ACCEPT`
  - This example command appends a rule at the end of the INPUT chain that specifies packets coming from source address 205.168.0.1 to be ACCEPT

Commands

- -D or --delete: This command deletes a rule from the chain, either by specifying the rule to match with -D or by specifying the rule's position number in the chain. The following examples shows both ways.
- Examples:
- `$ iptables -D INPUT --dport 80 -j DROP`
- `$ iptables -D OUTPUT 3`
- The first command deletes a rule from the INPUT chain that specifies packets destined for port 80 to be DROPped. The second command simply deletes rule number 3 from the OUTPUT chain.

- -P or --policy: This command sets a default target, i.e. policy, for a chain. All packets that don't match any rule in the chain will then be forced to use the policy of the chain.
- Example:
- `$ iptables -P INPUT DROP`
- This command specifies the default target of the INPUT chain to be DROP. That means all the packets not matching any rule in the INPUT chain will be dropped.

Command

- # -N or --new-chain: This creates a new chain with the name specified in the command.
- Example:
- `$ iptables -N allowed-chain`

Command

- # -F or --flush:
  - This command deletes all rules inside a chain if a chain name is specified or all rules in all chains if no chain name is specified. Used for quick cleanup.
- Examples:
- `$ iptables -F FORWARD`
- `$ iptables -F`

Command

- # -L or --list: Lists all rules in the specified chain.
- Example:
- `$ iptables -L allowed-chain`
- `$ iptables -L`

Match

- The optional match section of the iptables command specifies the characteristics that a packet should have to match the rule, such as source and destination address, protocol, etc. T
- he matches are divided in two major categories:
  - generic matches and
  - protocol-specific matches.

Match

- -p or --protocol: This generic protocol match is used to check for certain protocols.
  - Examples of protocols are TCP, UDP, ICMP, comma-delimited list of any combination of these three protocols and ALL (for all protocols).
  - ALL is the default match. This option can be inverted by using the ! sign.
- Examples:
- `$ iptables -A INPUT -p TCP, UDP`
- `$ iptables -A INPUT -p ! ICMP`
  - Both commands perform the same task -- they specify that all TCP and UDP packets will match this rule. By specifying ! ICMP, we mean to allow all other protocols (TCP and UDP, in this case) except ICMP.

Match

- -s or --source: This source match is used to match packets based on their source IP address.
  - This match also allows IP address range matching and it can be inverted using the ! sign.
  - The default source match matches all IP addresses.
  - `$ iptables -A OUTPUT -s 192.168.1.1`
  - `$ iptables -A OUTPUT -s 192.168.0.0/24`
  - `$ iptables -A OUTPUT -s ! 203.16.1.89`
  - The second command specifies that this rule matches all packets coming from IP addresses ranging 192.168.0.0 to 192.168.0.24.
  - The third command specifies that this rule will match any packets not from source address 203.16.1.89.

Match

- # -d or --destination:
  - This destination match is used to match packets based on their destination IP address.
  - This match also allows IP address range matching and it can be inverted using the ! sign.
- `$ iptables -A INPUT -d 192.168.1.1`
- `$ iptables -A INPUT -d 192.168.0.0/24`
- `$ iptables -A OUTPUT -d ! 203.16.1.89`

Block	Range		CIDR Notation	Default Subnet Mask	Number of hosts
24 bit block in class A	10.0.0.0	10.255.255.255	10.0.0.0/8	255.0.0.0	16,777,216
20 bit block in class B	172.16.0.0	172.31.255.255	172.16.0.0/12	255.240.0.0	1,048,576
16 bit block in class C	192.168.0.0	192.168.255.255	192.168.0.0/16	255.255.0.0	65,536

Targets

- We already know that targets are the actions specified by rules to be performed on packets that match those rules.
- There are many target options available along with allowances for user-defined targets.
- The following are often-used targets, their examples and explanations:
  - Accept
  - Reject
  - Drop

Target

- # ACCEPT:
  - When a packet is perfectly matched with a rule that has an ACCEPT target, it is accepted (allowed to go wherever it is destined to) and it will stop traversing the chain (though that packet may traverse through another chain in another table and may be dropped there).
  - This target is specified as -j ACCEPT.
- # DROP:
  - A packet that matches a rule perfectly that has a DROP target will be blocked and no further processing will be done on it.
  - This target is specified as -j DROP.

Target

- # REJECT:
  - This target works the same way as the DROP target, except that it is better than DROP.
  - Unlike DROP, REJECT doesn't leave dead sockets around on the server and client.
  - Also, REJECT sends back an error message to the sender of the packet.
  - This target is specified as -j REJECT.
  - `$ iptables -A FORWARD -p TCP --dport 22 -j REJECT`

Target

- # RETURN:
  - The RETURN target set in a rule makes the packet matching that rule stop traversing through the chain containing the rule.
  - If the chain is a main chain like INPUT, the packet will be handled using the default policy of that chain.
  - It is specified as -jump RETURN. Example:
  - `$ iptables -A FORWARD -d 203.16.1.89 -jump RETURN`

Stateful firewalls

- The biggest advantage of netfilter/iptables is that it can configure stateful firewalls
  - A stateful firewall is capable of assigning and remembering the state of connections made for sending or receiving packets.
  - Firewall gets this information from the connection tracking state of the packets.
  - This information about states is used by a firewall when it is making new packet filtering decisions to increase its efficiency and speed.
  - There are 4 valid states, namely ESTABLISHED, INVALID, NEW and RELATED.

States

- The state ESTABLISHED indicates that the packet is part of an already established connection that has been used to both send and receive packets and is fully valid.
- INVALID state indicates that the packet is not associated with any known stream or connection and it may contain faulty data or headers.
- The state NEW means that the packet has or will start a new connection or that it is associated with a connection that has not been used to both send and receive packets.
- Finally, RELATED means that the packet is starting a new connection and it is associated with an already established connection

- Rate-limited connection and logging capability.
- Now you can limit both connection attempts, as in SYN-flooding Denial of Service (DOS) attacks, and also prevent your logs being flooded, as happened in the Jolt2 fragment-driven DOS attack against Checkpoint's Firewall-1.

TCP Extensions

- --tcp-flags
  - Followed by an optional `!', then two strings of flags, allows you to filter on specific TCP flags.
  - The first string of flags is the mask: a list of flags you want to examine. The second string of flags tells which one(s) should be set. For example,
  - `# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DENY`
  - This indicates that all flags should be examined (`ALL' is synonymous with `SYN,ACK,FIN,RST,URG,PSH'), but only SYN and ACK should be set.
  - There is also an argument `NONE' meaning no flags.

TCP Extensions

- --syn
  - Optionally preceded by a `!', this is shorthand for `--tcp-flags SYN,RST,ACK SYN'.
- --source-port
  - Followed by an optional `!', then either a single TCP port, or a range of ports.
  - Ports can be port names, as listed in /etc/services, or numeric.
  - Ranges are either two port names separated by a `-', or (to specify greater than or equal to a given port) a port with a `-' appended, or (to specify less than or equal to a given port), a port preceded by a `-'.

Deny a specific host:

```
iptables -I INPUT -s XXX.XXX.XXX.XXX -j DROP
```

```
# Allow loopback access. This rule must come before the rules denying port access!!
iptables -A INPUT -i lo -p all -j ACCEPT
#This rule is essential if you want your own computer to be able to access itself
#through the loopback interface
iptables -A OUTPUT -o lo -p all -j ACCEPT

iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 2049 -j DROP       - Block NFS
iptables -A INPUT -p udp -s 0/0 -d 0/0 --dport 2049 -j DROP       - Block NFS
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 6000:6009 -j DROP  - Block X-Windows
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 7100 -j DROP    -Block X-Windows font server
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 515 -j DROP        - Block printer port
iptables -A INPUT -p udp -s 0/0 -d 0/0 --dport 515 -j DROP        - Block printer port
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 111 -j DROP        - Block Sun rpc/NFS
iptables -A INPUT -p udp -s 0/0 -d 0/0 --dport 111 -j DROP        - Block Sun rpc/NFS
```

```
iptables -F
iptables -A INPUT -i lo -p all -j ACCEPT
#Allow self access by loopback interface
iptables -A OUTPUT -o lo -p all -j ACCEPT

iptables -A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEP
# Accept established connections
iptables -A INPUT -p tcp --tcp-option ! 2 -j REJECT --reject-with tcp-reset
iptables -A INPUT -p tcp -i eth0 --dport 21 -j ACCEPT      - Open ftp port
iptables -A INPUT -p udp -i eth0 --dport 21 -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 22 -j ACCEPT    - Open secure shell por
iptables -A INPUT -p udp -i eth0 --dport 22 -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 80 -j ACCEPT     - Open HTTP port
iptables -A INPUT -p udp -i eth0 --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --syn -s 192.168.10.0/24 --dport 139 -j ACCEPT
#Accept local network Samba connection
iptables -A INPUT -p tcp --syn -s trancas --destination-port 139 -j ACCEPT
iptables -P INPUT DROP
```

IP Headers

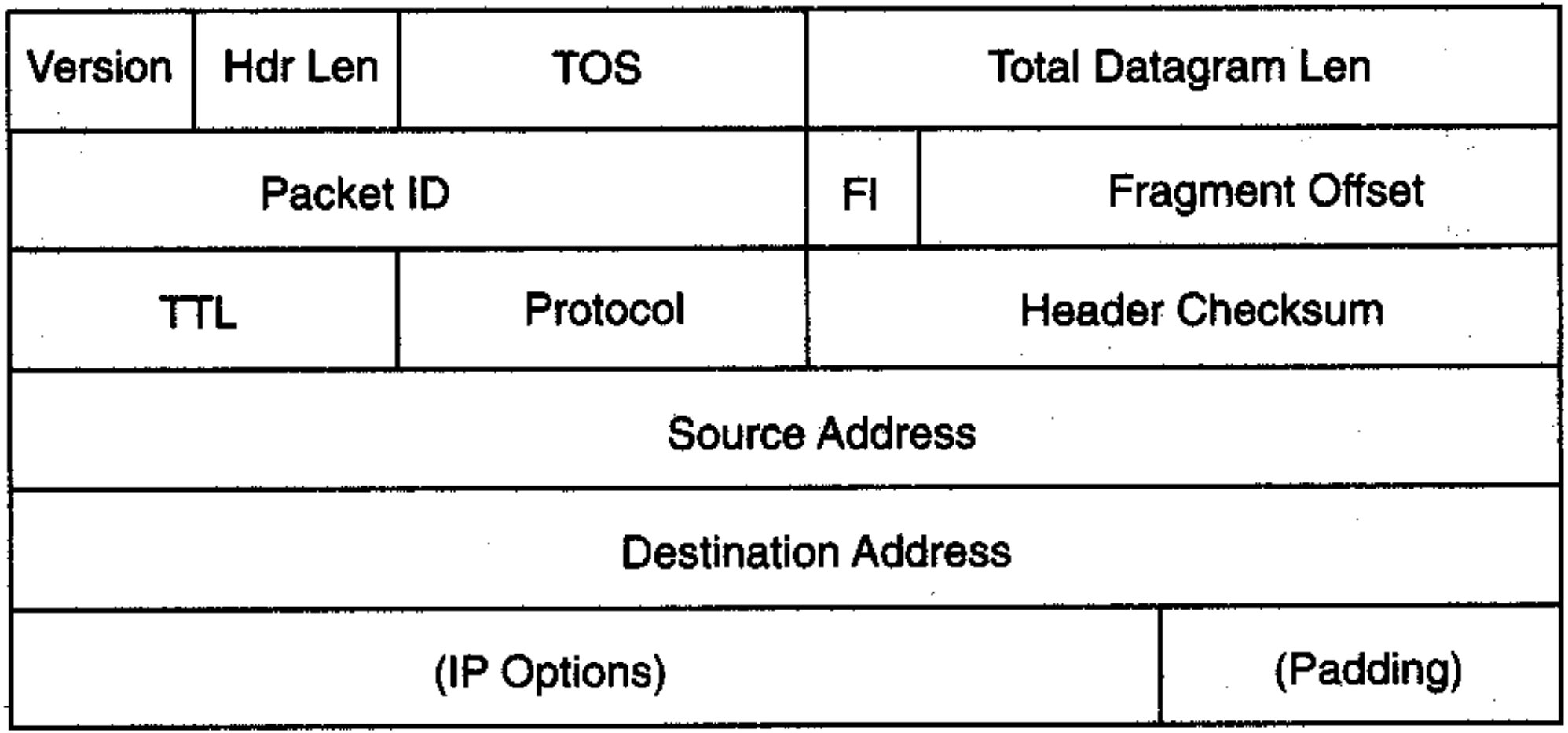
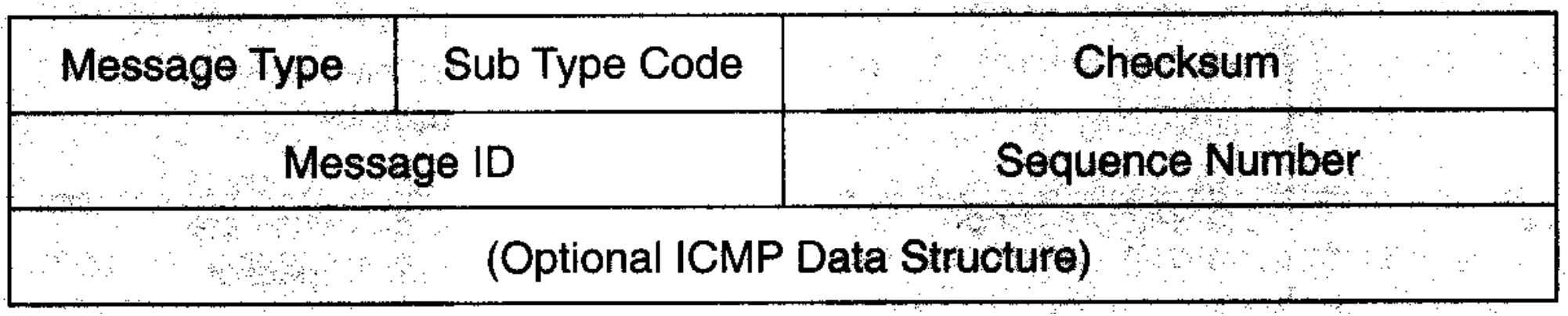
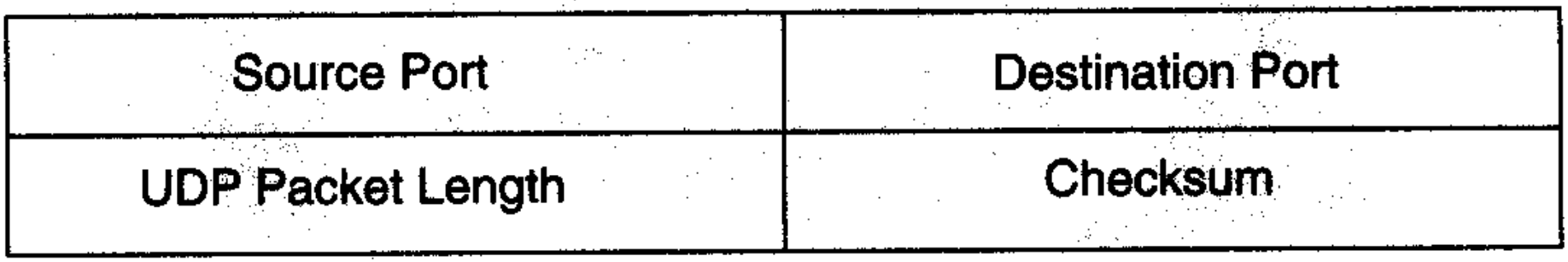


Figure 1.5 IP Header.

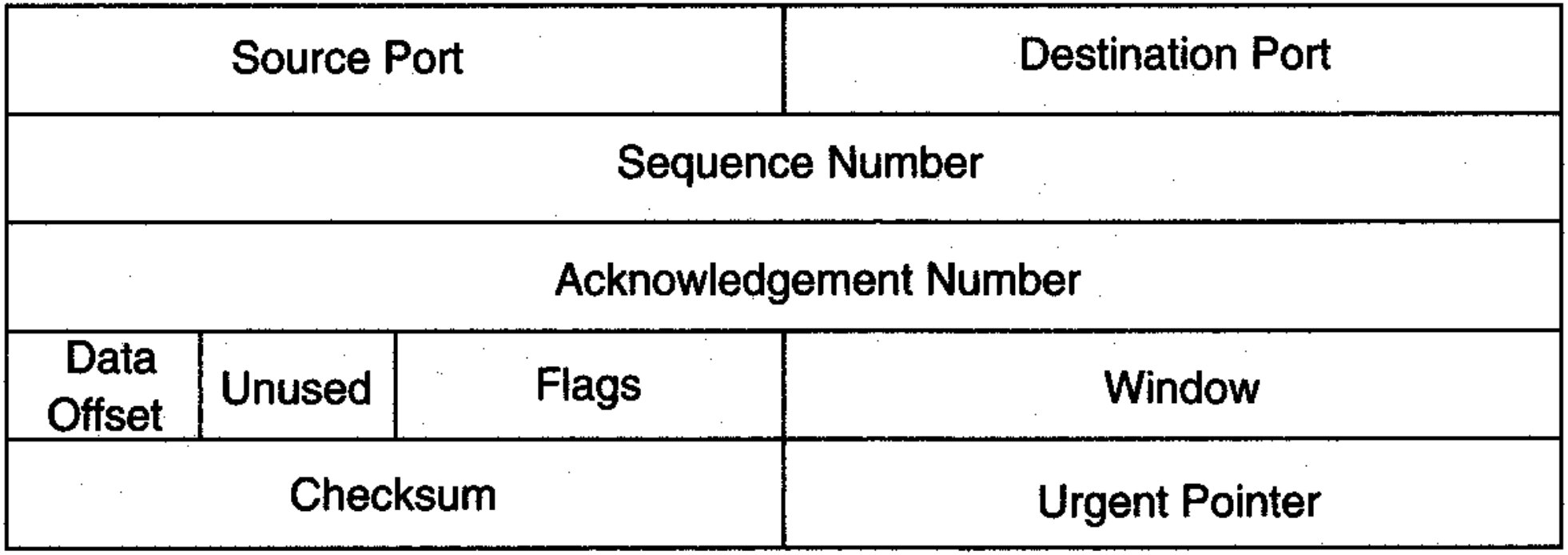
ICMP Header


Message Type
Sub Type Code
Checksum
Message ID
Sequence Number
(Optional ICMP Data Structure)

UDP Header



TCP Header


Source Port
Destination Port
Sequence Number
Acknowledgement Number
Data Offset
Unused
Flags
Window
Checksum
Urgent Pointer

Command	Description
`-N ¦ --new-chain <chain>`	Creates a user-defined chain.
`-F ¦ --flush [<chain>]`	Flushes the chain, or all chains if none is specified.
`-X ¦ --delete-chain [<chain>]`	Deletes the user-defined chain, or all chains if none is specified.
`-P ¦ --policy <chain><policy>`	Defines the default policy for one of the built-in chains, `INPUT`, `OUTPUT`, or `FORWARD`. The policy is either `ACCEPT` or `DROP`.
`-L ¦ --list [<chain>]`	Lists the rules in the chain, or all chains if none is specified.
`-Z ¦ --zero`	Resets the packet and byte counters associated with each chain.
`-h ¦ <some command> -h`	Lists the `iptables` commands and options, or if preceeded by an `iptables` command, lists the syntax and options for that command.

Note: The ¦ in the table means "or."

Command	Description
`-A ¦ --append <chain>`	Appends a rule to the end of the chain.
`-I ¦ --insert <chain>`	Inserts a rule at the beginning of the chain.
`-D ¦ --delete <chain> <rule number>`	Deletes the rule at position rule number within the chain.

Command	Description
`-i ¦ --in-interface [!] [<interface>]`	For incoming packets on either the `INPUT` or `FORWARD` chains, or their user-defined subchains, specifies the interface name that the rule applies to. If no interface is specified, all interfaces are implied.
`-o ¦ --out-interface [!] [<interface>]`	For outgoing packets on either the `OUTPUT` or the `FORWARD` chains, or their user-defined subchains, specifies the interface name that the rule applies to. If no interface is specified, all interfaces are implied.
`-p ¦ --protocol [!] [<protocol>]`	Specifies the IP protocol that the rule applies to. The built-in protocols are `tcp`, `udp`, `icmp`, and `all`. The protocol value can be either the name or the numeric value, as listed in `/etc/protocols`.

TCP Filter Matches

-s ¦ --source ¦ --src [!] <address>[/<mask>]	Specifies the host or network source address in the IP header.
-d ¦ --destination ¦ --dst [!] <address>[/<mask>]	Specifies the host or network destination address in the IP header.
-j ¦ --jump <target>	Specifies the target disposition for the packet, if it matches the rule. The default targets are ACCEPT or DROP, or a user-defined chain.

TCP Filter Matches

-p tcp	Description
--source-port ¦ --sport [!] <port>[:<port>]	This command specifies the source ports.
--destination-port ¦ --dport [!] <port>[:<port>]	This command specifies the destination ports.
--tcp-flags [!] <mask>[,<mask>] <set>[,<set>]	This command tests the bits in the mask list, out of which the following bits must be set in order to match.
[!] --syn	The SYN flag must be set as an initial connection request.
--tcp-option [!] <number>	The only legal tcp option is the maximum packet size that the sending host is willing to accept.

UDP MATCHES

-p udp	Description
--source-port ¦ --sport [!] <port>[:<port>]	Specifies the source ports.
--destination-port ¦ --dport [!] <port>[:<port>]	Specifies the destination ports.

- `echo-reply` (0)
- `destination-unreachable` (3)
    - network-unreachable
    - host-unreachable
    - protocol-unreachable
    - port-unreachable
    - fragmentation-needed
    - network-unknown
    - host-unknown
    - network-prohibited
    - host-prohibited
- `source-quench` (4)
- `redirect` (5)
- `echo-request` (8)

ICMP Matches

Match	Description
`--icmp-type [!] <type>`	Specifies the ICMP type name or number. The ICMP type is used in place of a source port.

StandAlone Firewall

```
# Remove any existing rules from all chains
iptables --flush
iptables -t nat --flush
iptables -t mangle --flush

# Unlimited traffic on the loopback interface
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Set the default policy to drop
iptables --policy INPUT   DROP
iptables --policy OUTPUT  DROP
iptables --policy FORWARD DROP
```

```
# All of the bits are cleared
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

# SYN and FIN are both set
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

# SYN and RST are both set
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP

# FIN and RST are both set
iptables -A INPUT -p tcp --tcp-flags FIN,RST FIN,RST -j DROP

# FIN is the only bit set, without the expected accompanying ACK
iptables -A INPUT -p tcp --tcp-flags ACK,FIN FIN -j DROP

# PSH is the only bit set, without the expected accompanying ACK
iptables -A INPUT -p tcp --tcp-flags ACK,PSH PSH -j DROP

# URG is the only bit set, without the expected accompanying ACK
iptables -A INPUT -p tcp --tcp-flags ACK,URG URG -j DROP
```

Stateful Packet Filtering

```
# Using Connection State to By-pass Rule Checking

if [ "$CONNECTION_TRACKING" = "1" ]; then
    iptables -A INPUT  -m state --state ESTABLISHED,RELATED -j
ACCEPT
    iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

Source Address Spoofing and Other Bad Addresses

```
# Refuse spoofed packets pretending to be from
# the external interface's IP address
iptables -A INPUT  -i $INTERNET -s $IPADDR -j DROP

# Refuse packets claiming to be from a Class A private network
iptables -A INPUT  -i $INTERNET -s $CLASS_A -j DROP

# Refuse packets claiming to be from a Class B private network
iptables -A INPUT  -i $INTERNET -s $CLASS_B -j DROP

# Refuse packets claiming to be from a Class C private network
iptables -A INPUT  -i $INTERNET -s $CLASS_C -j DROP

# Refuse packets claiming to be from the loopback interface
iptables -A INPUT  -i $INTERNET -s $LOOPBACK -j DROP
```

```
NAMESERVER="isp.name.server.1"       # address of a remote name server
POP_SERVER="isp.pop.server"          # address of a remote pop server

NEWS_SERVER="isp.news.server"        # address of a remote news server
TIME_SERVER="some.timne.server"      # address of a remote time server
DHCP_SERVER="isp.dhcp.server"        # address of your ISP dhcp server

LOOPBACK="127.0.0.0/8"               # reserved loopback address range
CLASS_A="10.0.0.0/8"                 # class A private networks
CLASS_B="172.16.0.0/12"              # class B private networks
CLASS_C="192.168.0.0/16"             # class C private networks
CLASS_D_MULTICAST="224.0.0.0/4"      # class D multicast addresses
CLASS_E_RESERVED_NET="240.0.0.0/5"   # class E reserved addresses
BROADCAST_SRC="0.0.0.0"              # broadcast source address
BROADCAST_DEST="255.255.255.255"     # broadcast destination address
```

```
CONNECTION_TRACKING="1"
ACCEPT_AUTH="0"
SSH_SERVER="0"
FTP_SERVER="0"
WEB_SERVER="0"
SSL_SERVER="0"
DHCP_CLIENT="1“
INTERNET="eth0"                    # Internet-connected interface
LOOPBACK_INTERFACE="lo“    # however your system names it
IPADDR=”132.170.148.222"     # your IP address
SUBNET_BASE="132.170.148.0"
# ISP network segment base address
SUBNET_BROADCAST="132.170.148.255"
# network segment broadcast address
MY_ISP="144.144.144.144"
# ISP server & NOC address range
```