NYU TANDON SCHOOL OF ENGINEERING

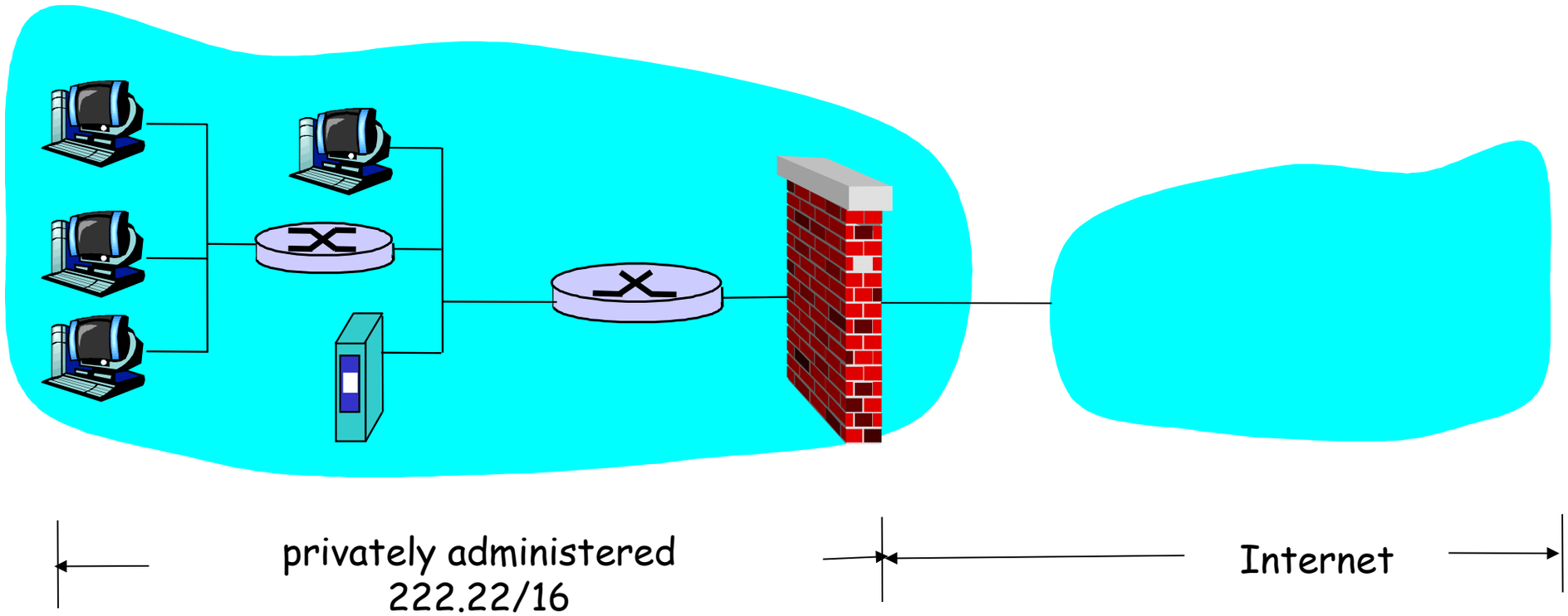
Firewalls

CS6823 – Network Security

Phillip Mak
pmak@nyu.edu

Firewall

- **Firewalls enforces security between networks of different security policies**
- Data firewall isolates an organization's internal net from an untrusted network such as the Internet, allowing some packets to pass and blocking others.



Firewall Goals:

- All traffic from outside to inside and vice-versa passes through the firewall
- Only authorized traffic, as defined by local security policy, will be allowed to pass
- The firewall itself is immune to penetration

Firewalls Terms

- **Packet Filters (Traditional)** – fast, stateless firewalls
- **Stateful Firewalls** – FWs that maintain connection info
- **Proxies (Application Gateway)** – a server used as an intermediary on an network, such as web, email, FTP
- **Bastion Host** – host intended to withstand attacks
- **DMZ (Demilitarized Zone or Perimeter Network)** – The area of the network where outward facing services are exposed
- **Access Control Lists** – Another term for packet filter firewall, commonly used when referring to non-firewall devices

Packet Filters (Stateless)

- Analyzes each datagram going through it; makes drop decision based on:
  - **Source IP address**
  - **Destination IP address**
  - **Source port**
  - **Destination port**
  - **TCP flag bits**
    - SYN bit set: datagram for connection initiation
    - ACK bit set: part of established connection
  - **TCP, UDP or ICMP**
    - Firewalls often configured to block all UDP
  - **Direction**
    - Is the datagram leaving or entering the internal network
  - **Router Interface**
    - Decisions can be different for different interfaces

Stateless Firewalls (Traditional or Packet Filtering)

- Advantages
  - One screening router can protect entire network
  - Can be efficient if filtering rules are kept simple
  - Widely available. Almost any router even Linux boxes

- Disadvantages
  - Cannot enforce some policies. For example, permit certain users
  - Rules can get complicated and difficult to test
  - Example: Attacker can send a malformed packet attack by sending ACK=1 segments, with source port 80

Filtering Rules - Examples

Policy	Firewall Setting
No outside Web access	Drop all outgoing packet to any IP address on port 80
External connections to public Web server only.	Drop all incoming TCP SYN packets to any IP except 222.22.44.203 port 80
Prevent IPTV from eating up the available bandwidth	Drop all incoming UDP packets except DNS and router broadcasts
Prevent your network from being used for a SMURF DoS attack	Drop all ICMP packets going to a broadcast address (eg 222.22.255.255)
Prevent your network from being tracerouted	Drop all outgoing ICMP

Cortesty: Wikipedia

Access control lists – Apply rules from top to bottom

Action	Source Address	Destination Address	Protocol	Source Port	Destination Port	Flag Bit
Allow	222.22/16	Outside of 222.22/16	TCP	>1023	80	Any
Allow	Outside of 222.22/16	222.22/16	TCP	80	>1023	ACK
Allow	222.22/16	Outside of 222.22/16	UDP	>1023	53	--
Allow	Outside of 222.22/16	222.22/16	UDP	53	>1023	__
Reject	All	All	All	All	All	All

Stateful Filters: Example

1) Packet arrives from outside: SA=37.96.87.123, SP=80, DA=222.22.1.7, DP=12699, SYN=0, ACK=1
2) Check filter table ➔ check stateful table
3) Connection is listed in connection table ➔ let packet through

Action	Source Address	Destination Address	Protocol	Source Port	Destination Port	Flag Bit	Check Connection
Allow	222.22/16	Outside of 222.22/16	TCP	>1023	80	Any	
Allow	Outside of 222.22/16	222.22/16	TCP	80	>1023	ACK	X
Allow	222.22/16	Outside of 222.22/16	UDP	>1023	53	--	
Allow	Outside of 222.22/16	222.22/16	UDP	53	>1023	--	X
Reject	All	All	All	All	All	All	

Stateful Firewalls

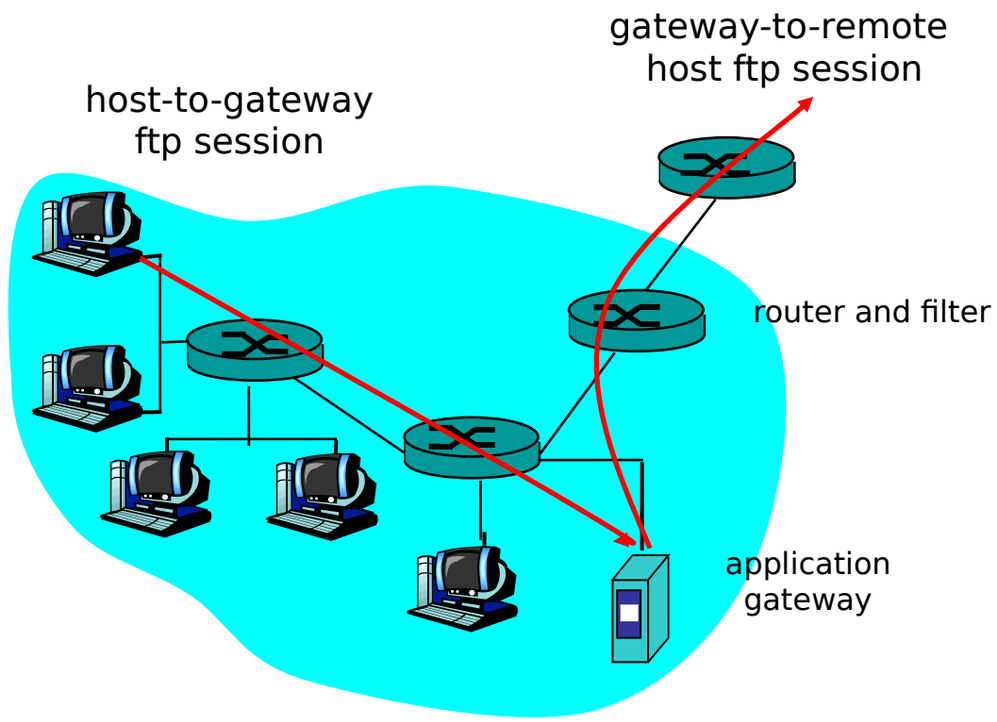
- Stateful firewalls keep track of connection states
- Each TCP, UDP, ping connection initiated through firewall is logged
- Timeout entries which see no activity for, say, 60 seconds
- If rule table indicates that stateful table must be checked: check to see if there is already a connection in stateful table

Protocol	State	source address	dest address	source port	dest port
tcp	SYN Sent	222.22.1.7	37.96.87.123	12699	80
tcp	Established	222.22.93.2	199.1.205.23	37654	80
udp	--	222.22.65.143	203.77.240.43	48712	53

Table: Stateful firewall table of open connections (Example)

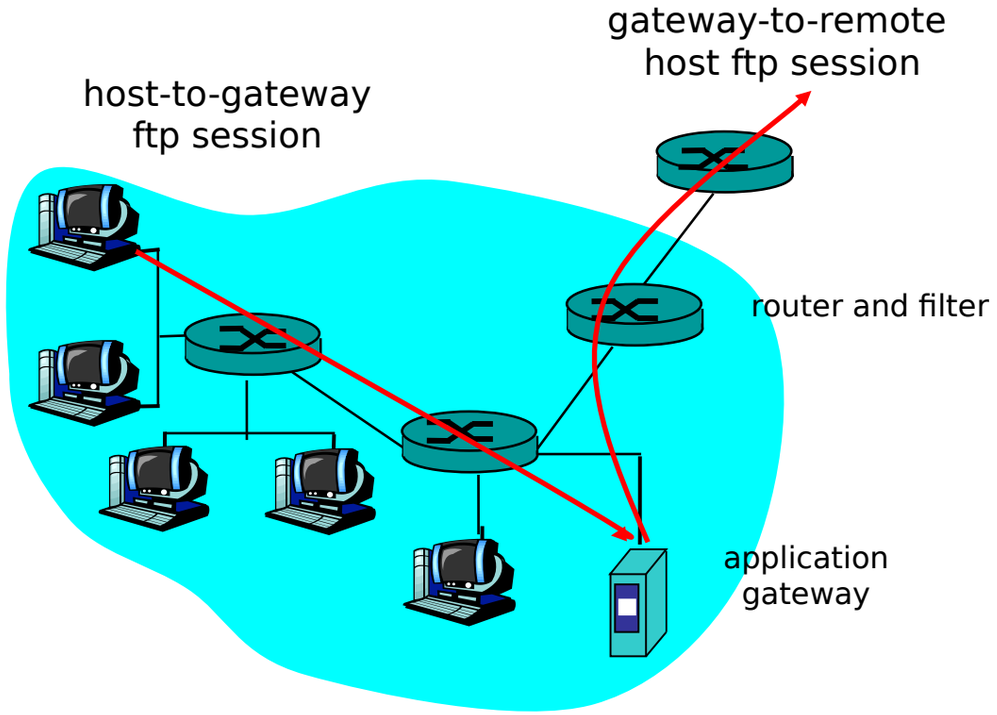
Application Gateways (Proxies) & Packet Filter

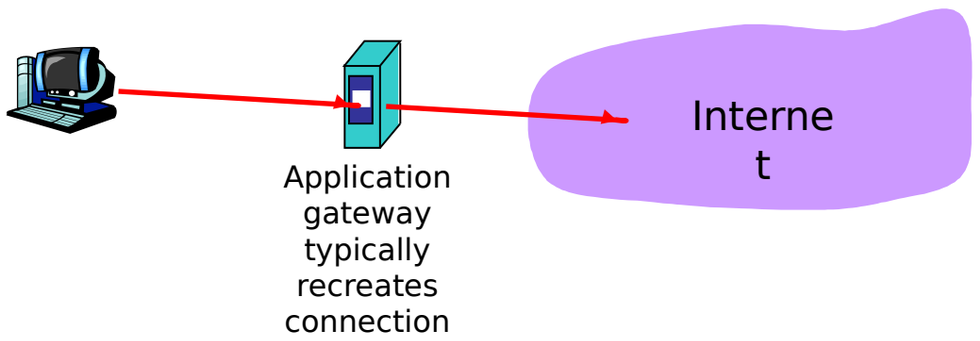
- Filters packets on application data as well as on IP/TCP/UDP fields
- Typically re-establish connections from the gateway as a form of filtering
- Example: allow select internal users to ftp outside.
  1. Require all FTP users to go through gateway
  2. For authorized users, gateway sets up ftp connection to dest host. Gateway relays data between 2 connections
  3. Router filter blocks all ftp connections not originating from gateway.



Application Gateways (Proxies)

- Gateway sits between user on inside and server on outside. Instead of talking directly, user and server talk through proxy.
- Allows more fine grained and sophisticated control than packet filtering. For example, ftp server may not allow files greater than a set size.
- A mail server is an example of an application gateway
- Can’t deposit mail in recipient’s mail server without passing through sender’s mail server



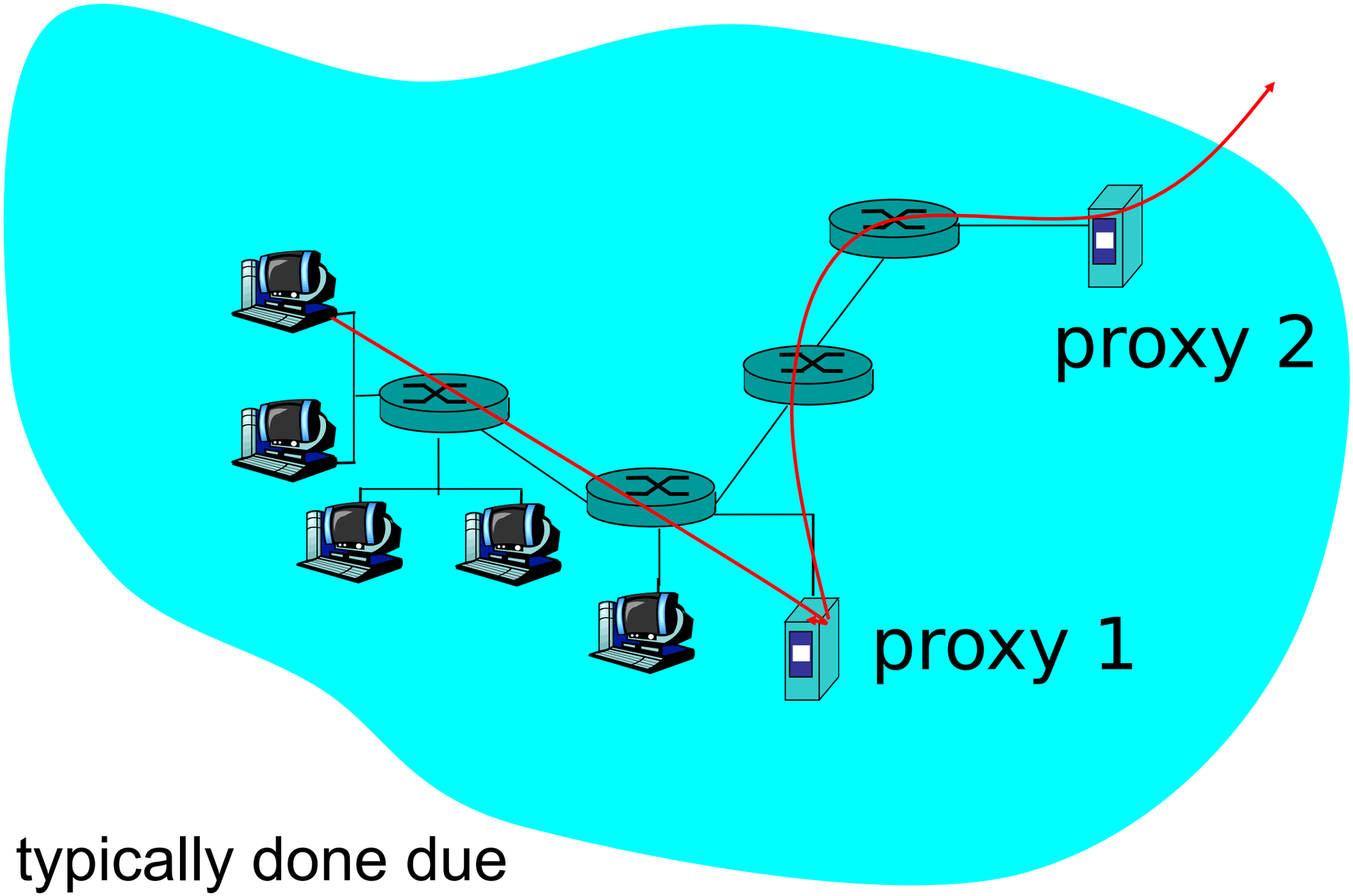


Advantages and Disadvantages of Proxy Gateways

- Advantages
  - Proxy can log all connections, activity in connections
  - Proxy can provide caching
  - Proxy can do intelligent filtering based on content
  - Proxy can perform user-level authentication
  - Supports Authentication

- Disadvantages
  - Not all services have proxied versions
  - May need different proxy server for each service
  - Requires modification of client
  - Performance
  - Cannot see encrypted traffic
    - Unless a root certificate is installed on the host

Chaining Proxies



- Not typically done due to performance issues

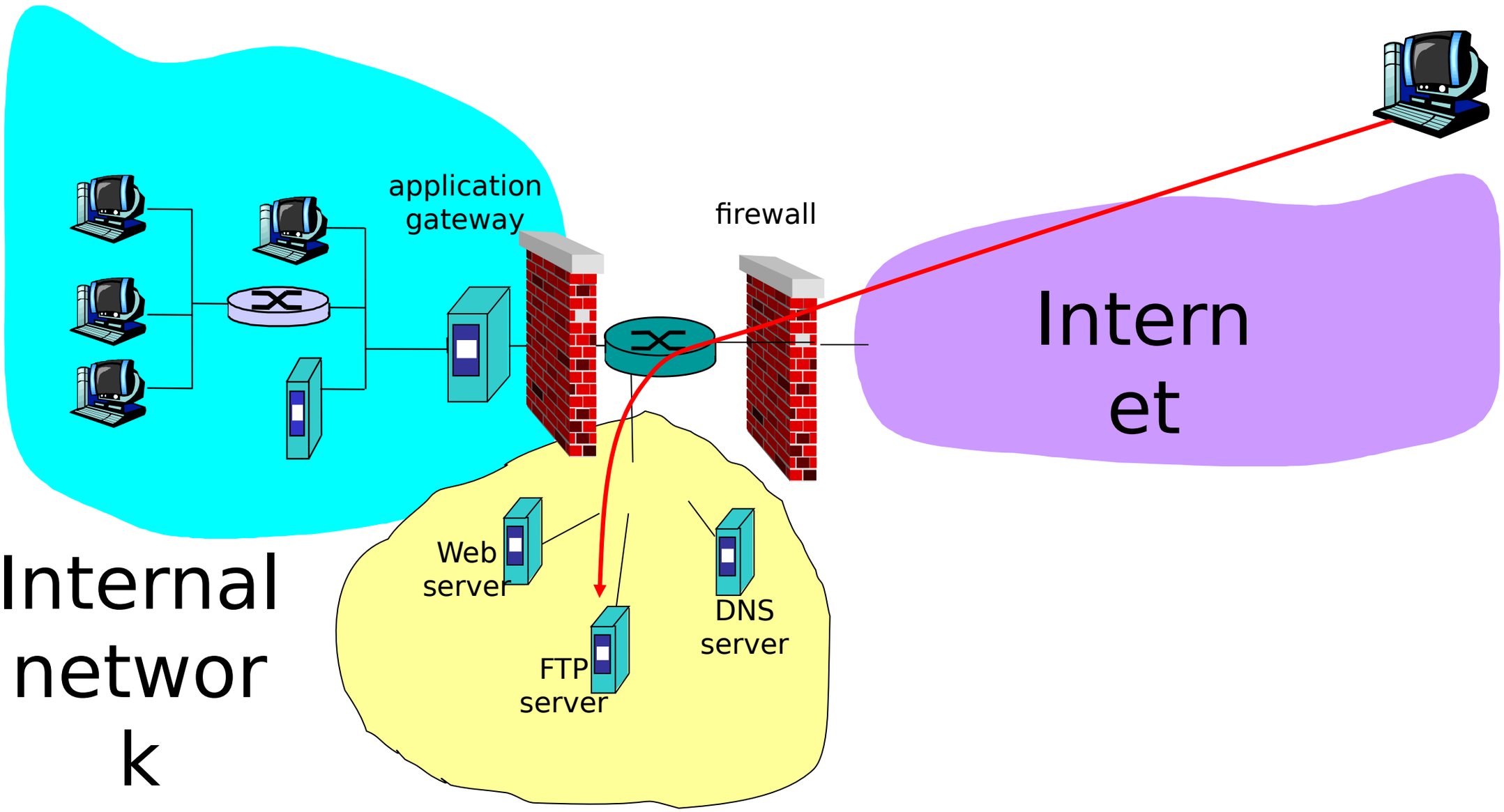
SOCKS Proxy Protocol

- Generic proxy protocol
  - Don’t have to redo all of the code when proxifying an application.

- Can be used by HTTP, FTP, telnet, SSL,…
  - Independent of application layer protocol

- Includes authentication, restricting which users/apps/IP addresses can pass through firewall.

DeMilitarized Zone (DMZ)



Linux Firewall Implementation – Netfilter/IPTables

Linux IPtables/Netfilter

- In Linux kernel 2.4/2.6 we typically use the new netfilter package with iptables commands to setup the firewall for
  - Packet filtering
  - Network Address and Port Translation (NAT|NAPT)
  - Packet mangling.

- http://www.netfilter.org/ is main site for the package.

- Tutorial and HOW-TO manual is available there.

http://www.netfilter.org/documentation/index.html#documentation-howto

Netfilter and IPtables

- netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.
- iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target/jump).
  - Tables; commands; classifiers; actions
- netfilter, ip_tables, connection tracking (ip_conntrack, nf_conntrack) and the NAT subsystem together build the major parts of the firewall framework.

NYU TANDON SCHOOL OF ENGINEERING

What can I do with netfilter/iptables?

- Build internet firewalls based on stateless and stateful packet filtering
- Use NAT and masquerading for sharing internet access if you don't have enough public IP addresses. (SNAT service; outgoing traffic/internal initiated)
- Use NAT to implement transparent proxies. Here it means clients does not know how and where the request is served. (DNAT service; incoming traffic/external requests)
- Aid the tc (traffic control) and iproute2 (utility for controlling TCP/UDP networking and traffic control) systems to build sophisticated QoS and policy-based routing
- Do further packet manipulation (mangling) like altering bits of the IP header
  - Type of Service (TOS; 2nd Byte in IP header for QoS RFC791)
  - Differential Service Control Point (DSCP upper 6bits of TOS field; RFC2474)
  - Explicit Congestion Notification (ECN bit 6 and 7 of TOS field; RFC3168)

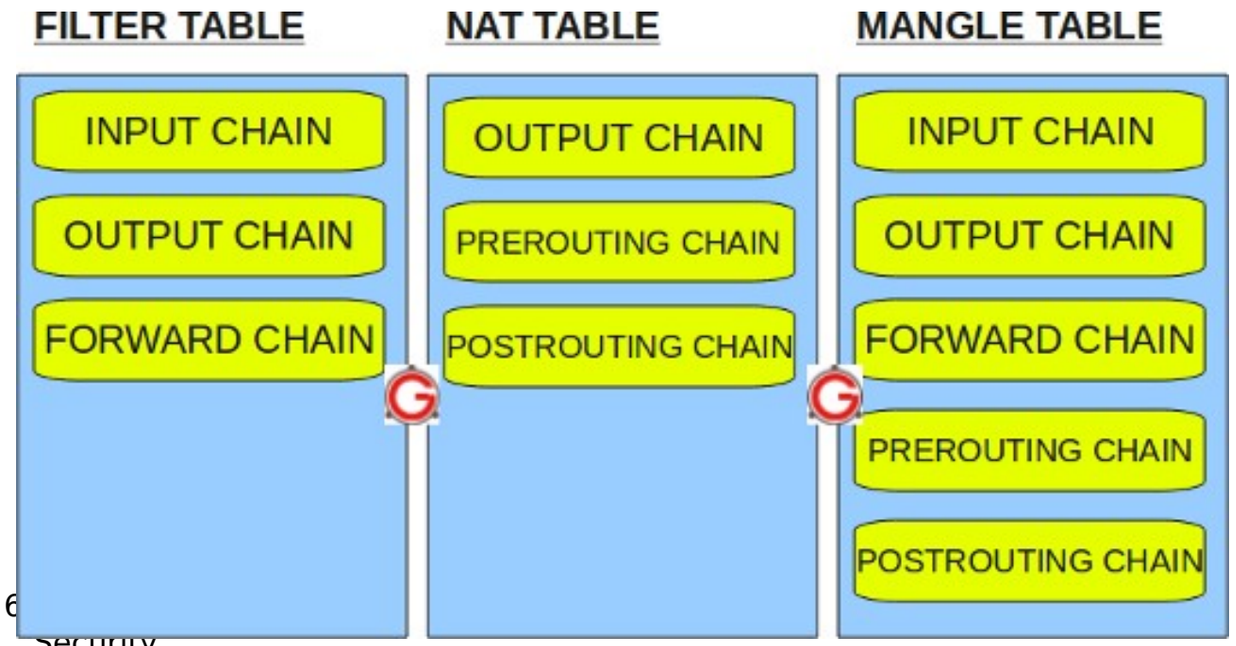
IPtables Table types

- FILTER:
  - What we have been talking about so far!
  - 3 chain types: INPUT, OUTPUT, and FORWARD

- NAT:
  - Hide internal network hosts from outside world. Outside world only sees the gateway's external IP address, and no other internal IP addresses
  - PREROUTING, POSTROUTING, and others

- MANGLE
  - Used for packet alterations

- RAW
  - Used for removing packets from connection tracking



Tables, Chains & Rules

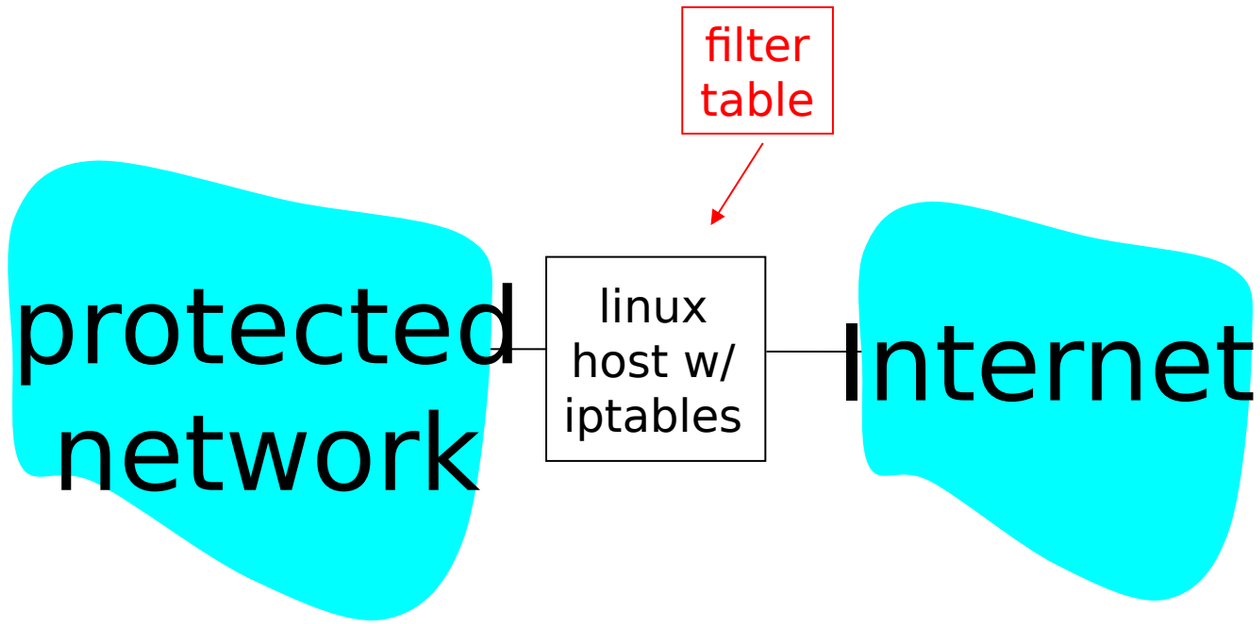
- Four types of tables: FILTER, NAT, MANGLE, and RAW

- A table consists of chains.
  - For example, a filter table can have an INPUT chain, OUTPUT chain, and a FORWARD chain.

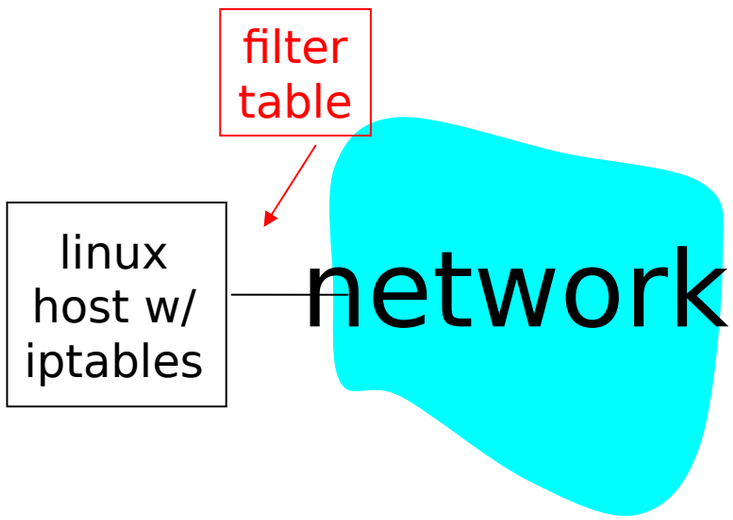
- A chain consists of a set of rules.

Network or Host Firewall?

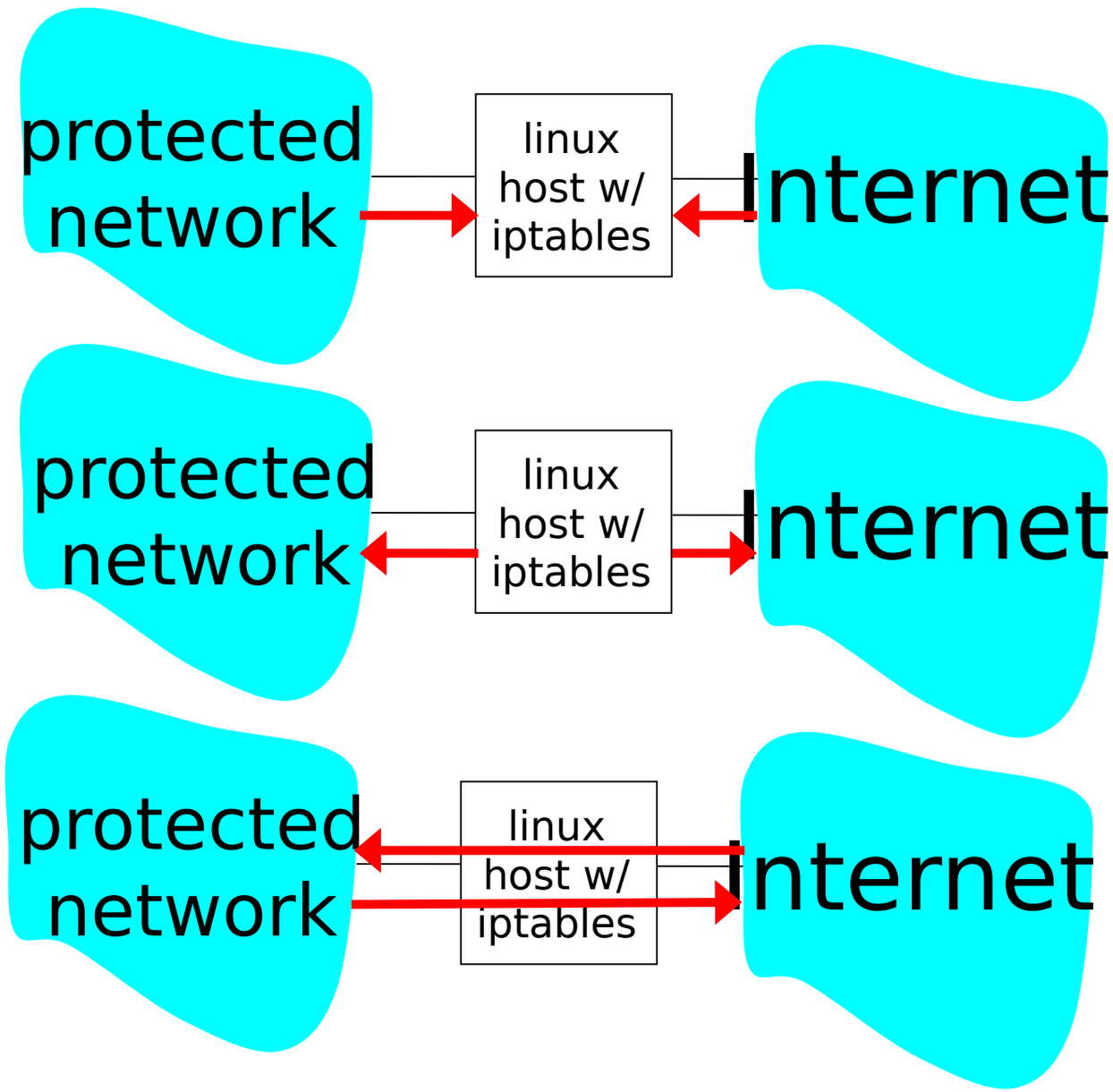
- Network Firewall: linux host with 2 Interfaces



- Host Firewall: linux host with 1 Interface



Chain Types for Host Firewall



INPUT chain: applies for all packets destined to firewall

OUTPUT chain: applies for all packets originating from firewall

FORWARD chain: applies for all packets passing through the firewall

iptables: Examples

```
iptables –t FILTER –A INPUT –i eth0 –j REJECT
```

- Sets a rule
  - Rejects all packets that enter from the interface eth0 (except for those accepted by previous rules)

- `-A INPUT, --append`: add rule to end of INPUT chain
- `-i, --in-interface`: interface via which a packet was received
- `-j REJECT`: if this packet matches the rule, jump to the REJECT rule (which simply denies the packet)

iptables: Examples

```
iptables -A INPUT -i eth0 -s 232.16.4.0/24 -j ACCEPT
```

- Sets a rule
  - Accepts packets that enter from interface eth0 and have a source address in 232.16.4.0/24
- Kernel applies the rules in order
  - The first rule that matches the packet determines the action for that packet
- `-A INPUT, --append`: add rule to end of INPUT chain
- `-i, --in-interface`: interface via which a packet was received
- `-s, --source`: source address/mask
- `-j ACCEPT`: if this packet matches the rule, ACCEPT it

iptables: More Examples

`iptables –L (iptables -L -v)`

- Lists current rules

```
Chain ACCEPT_ALL (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  br0    !eth3   0.0.0.0/0            0.0.0.0/0           state NEW
    0     0 ACCEPT     all  --  ipsec+ !eth3   0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     icmp --  *      eth3    0.0.0.0/0            0.0.0.0/0           icmp type 8

Chain BADTCP (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 PSCAN      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x3F/0x29
    0     0 PSCAN      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x3F/0x00
    4   208 PSCAN      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x3F/0x01
    2    80 PSCAN      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x06/0x06
    1    44 PSCAN      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:0x03/0x03
  937  620K NEWNOTSYN  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp flags:!0x17/0x02 st
```

`iptables –F (-t table)`

- Flush all rules in the filter table

```
iptables –I INPUT 1 –p tcp --tcp-flags SYN –s 232.16.4.0/24 –d 0/0:22 –j
  ACCEPT
```

- Default table is filter (so it's unspecified)
- -I INPUT 1: insert INPUT rules at top
- Accept TCP SYN to from 232.16.4.0.24 to port 22 (ssh)

Iptables: Default policy

- Iptable rules are read from top to bottom, so order of the rules matter greatly
- It's a good idea to set a default policy in case the packet does not match any of the rules

```
-iptables -P INPUT DROP
-iptables -P OUTPUT DROP
-iptables -P FORWARD DROP
```

iptables Options

```
-p protocol type (tcp, udp, icmp)
-s source -d dest IP address & port number
-i in, -o out interface name (lo, ppp0, eth0)
-j target (ACCEPT, REJECT, DROP)
--sport source port, --dport dest port
--icmp-type
```

Connection tracking for stateful firewall

```
-m conntrack --ctstate RELATED,ESTABLISHED
```

ctstate options:

- NEW: Allow new connections
- ESTABLISHED: Allow established sessions to receive traffic
- RELATED: Allow related established connections, e.g., FTP 20/21

```
-m state --state RELATED,ESTABLISHED
```

- Older method, but still fine

iptables Connection Tracking (Stateful Firewall)

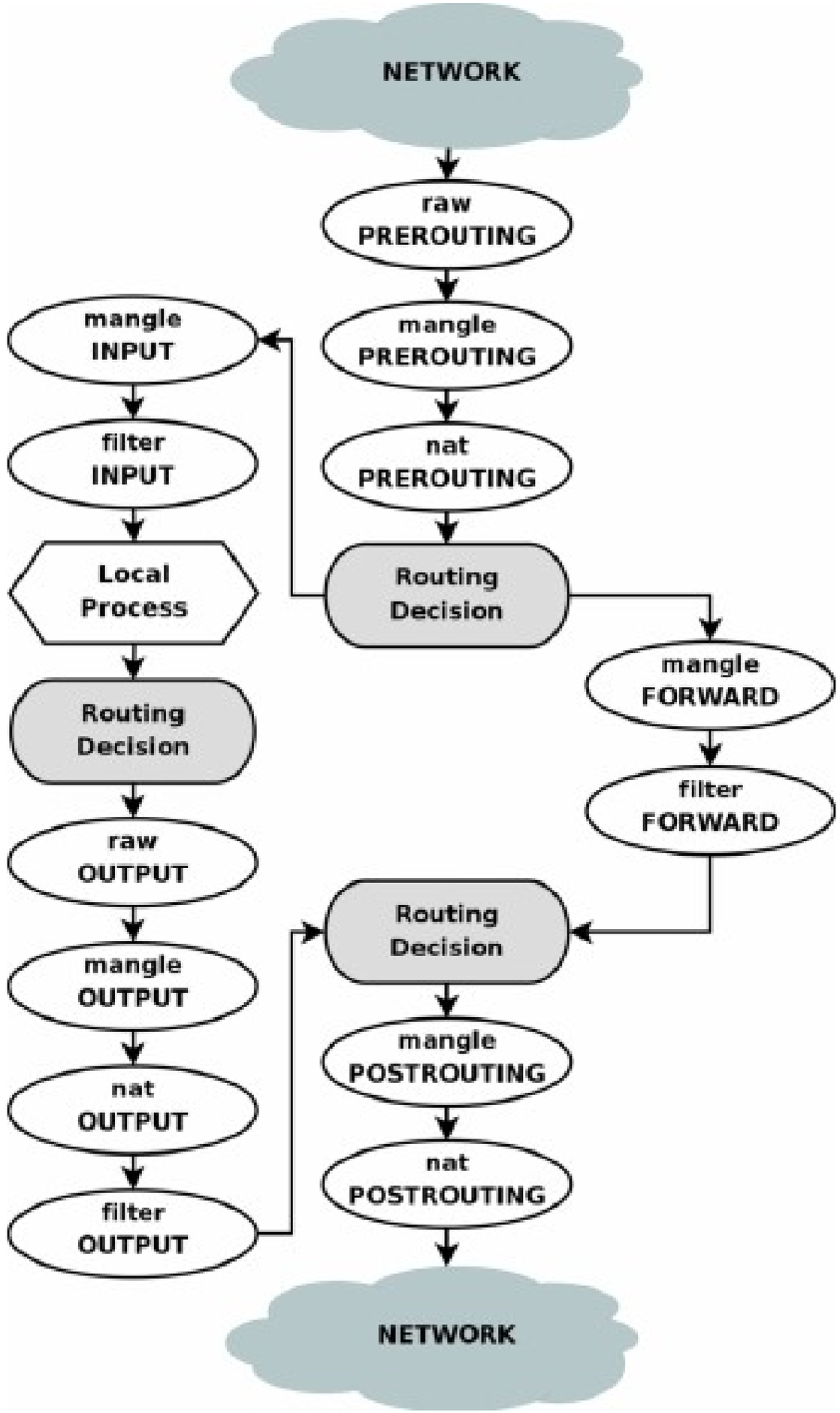
```
-A FORWARD -s 192.168.1.0/24 -i eth0 -o eth1 -m
conntrack --ctstate NEW -j ACCEPT
-A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j
ACCEPT
```

-A FORWARD: append to the FORWARD chain
-s: source address
-i eth0: coming in interface eth0
-o eth1: outgoing to interface eth1
-m conntrack --ctstate:
NEW,ESTABLISHED,RELATED

The first rule ensures that only the 192.168.1.0/24 network can start new connections

NYU TANDON SCHOOL OF ENGINEERING

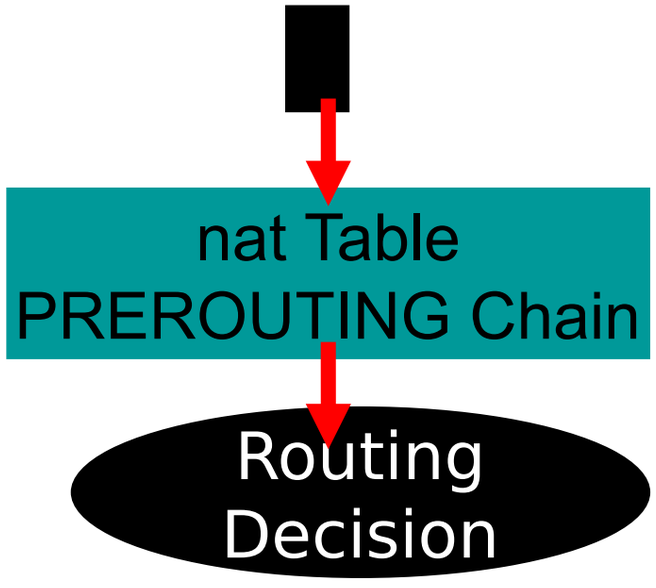
Packet Traversal Through IPTables



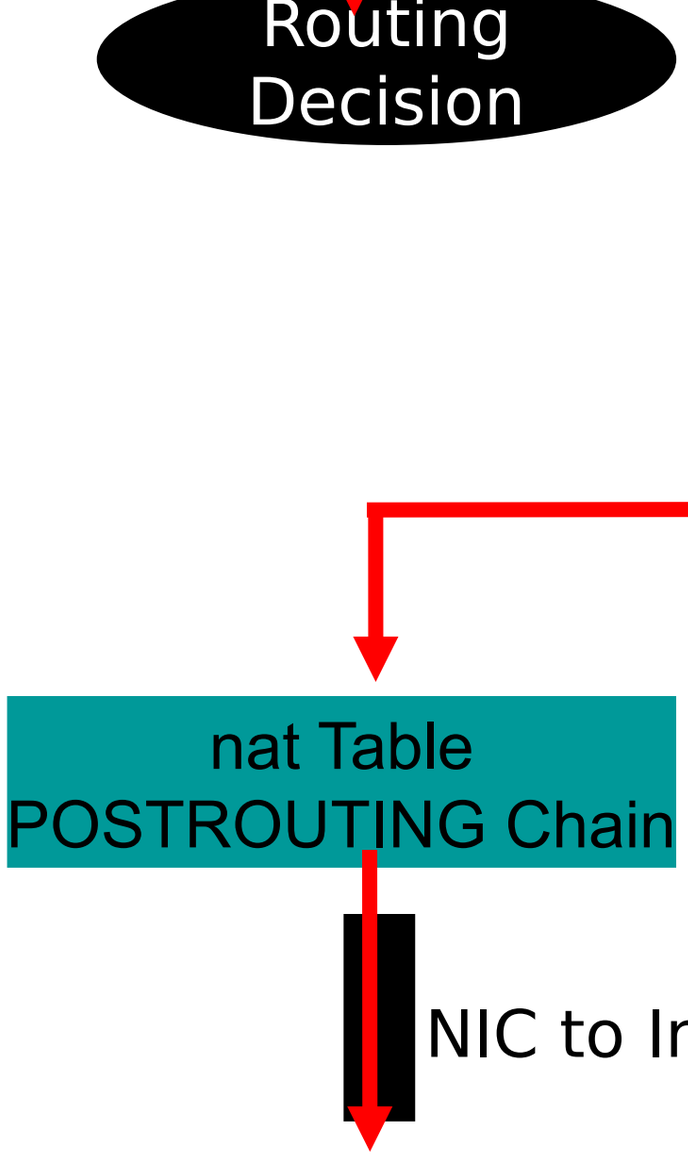
IPtables: Traversing of tables and chain http://www.iptables.info/en/structure-of-iptables.html

NYU TANDON SCHOOL OF ENGINEERING

Incoming Packet Journey through Linux Firewall




```
iptables -t nat -A PREROUTING -p TCP
        -i eth0 -d 128.168.60.12 --dport 80
        -j DNAT --to-destination 192.168.10.2
```



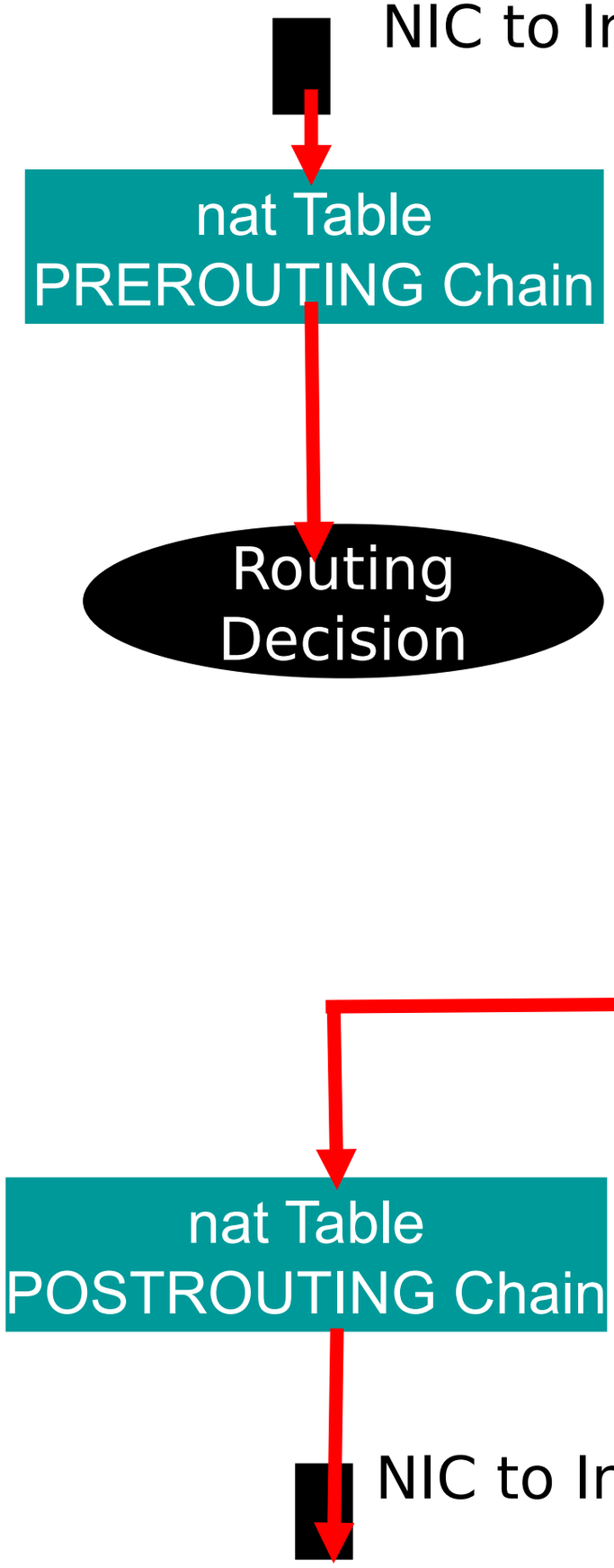

```
iptables -A FORWARD -p ALL
     -s 128.199.66.1 -j REJECT
iptables -A FORWARD -p ALL -s 128.200.0.2
     -j LOG --log-prefix "bad guy:"
iptables -A FORWARD -p ALL -s 128.200.0.2
     -j DROP
```

Note: IPTables continues processing after –j LOG

DNAT and IPtables command

- DNAT: Destination Network Address Translation.
- Deal with packets from Internet to our Internet exposed servers.
- It translates the destination (external) IP addresses to the corresponding internal IP address of DMZ servers.
- ```
  iptables -t nat -A PREROUTING -p TCP
          -i eth0 -d 128.168.60.12 --dport 80
          -j DNAT --to-destination 192.168.10.2
  ```
- -t specify the type of tables
  -A Append to a specific chain
  -p specify the protocol
  -i specify the incoming interface
  -d specify the matched destination IP address in packet
  -j specify the “target” or operation to be performed.
  --to-destination substitute the destination IP address.
  ```

Forwarded Packet Journey through Linux Firewall (2)




```
iptables -A FORWARD
     -s 192.168.10.10 -j DROP
```

- Certain system in Intranet not allowed out
- What happens to the return packets?

```
iptables -t nat -A POSTROUTING -o eth0
     -j MASQUERADE
```

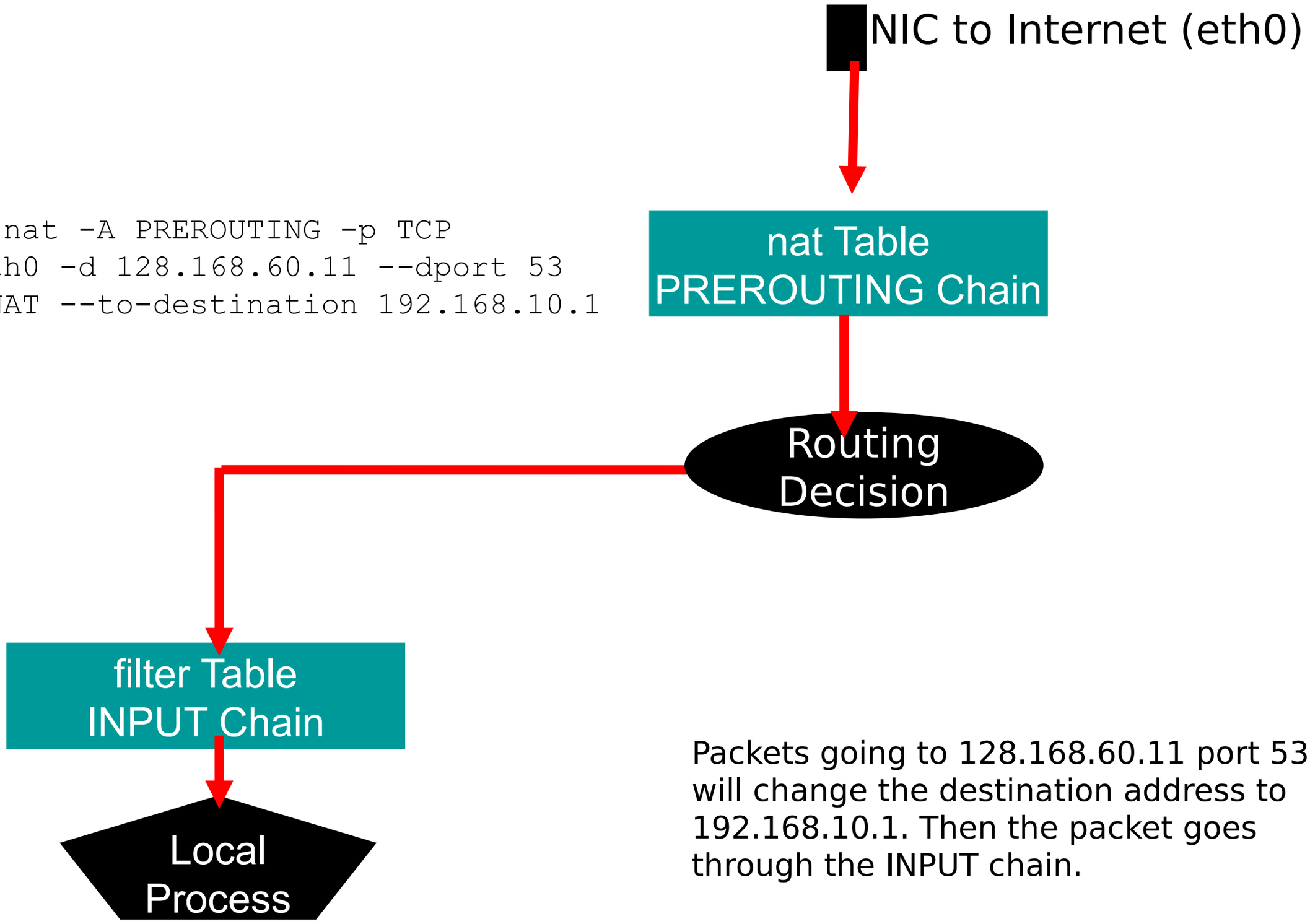
Alter the IP address to look like interface eth0

SNAT vs. MASQUERADE

- SNAT which translates only the IP addresses, the port number is preserved unchanged.
- However, it requires that you have the equal number of outgoing IP addresses as IP address in your intranet that are carrying in the source address field of the outgoing packets.
- Since it does not have to search for the available port or available IP address, SNAT is faster than MASQUERADE.
- For smaller organization which only have a few static IP addresses, MASQUERADE is the typically method.

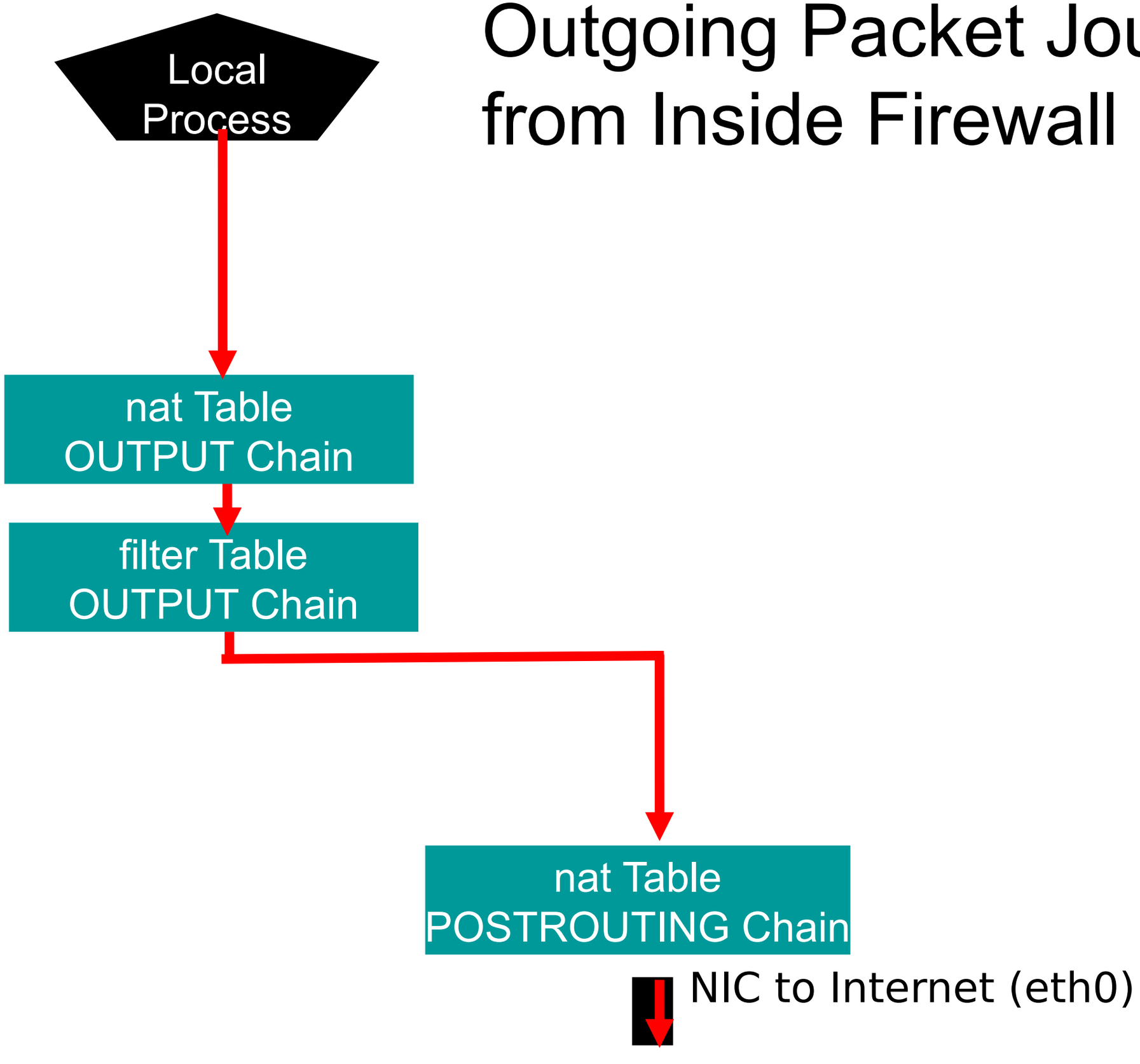
Incoming Packet to Service in Firewall

```
iptables -t nat -A PREROUTING -p TCP
        -i eth0 -d 128.168.60.11 --dport 53
        -j DNAT --to-destination 192.168.10.1
```



Packets going to 128.168.60.11 port 53 will change the destination address to 192.168.10.1. Then the packet goes through the INPUT chain.

Outgoing Packet Journey from Inside Firewall



IPTables Tables and Targets

- IPTables has 4 built in tables
  - Raw
  - Mangle
  - NAT
  - FILTER
- IPTables has numerous Targets (-j), interesting ones are:
  - ACCEPT/REJECT/DROP
  - LOG
  - TOS
  - TTL
  - MARK
  - SECMARK
  - CONNSECMARK

FILTER Table

- Primarily used for filtering packets. Packets are matched according to the pattern and then filtered
- Actions can be performed based on header and content
- The FILTER table is not the only place that filtering can be performed, but it's the best place due to design and convention
- Almost all targets are usable in this table (e.g., DROP , LOG , ACCEPT or REJECT)
- The FILTER table includes the built in chains: INPUT, OUTPUT, and FORWARD

```
iptables -A INPUT -i eth1 -s 192.168.0.0/24 -j DROP
```

Add a rule to drop packets coming from eth1 and 192.168.0.0/24 destined to the host. Table is FILTER (default, so it's not shown), chain is INPUT, and target is DROP.

RAW Table

- Mainly used to mark a packet to not be tracked by the connection tracking system; to make IPTables treat certain packets in a stateless manner
- This is done by using the NOTRACK target on the packet. If a connection is hit with the NOTRACK target, then `conntrack` will simply not track the connection.
- The RAW table only has the PREROUTING and OUTPUT chains
- This is the only place that packets can be dealt with before connection tracking

Silly example:

```
iptables -t raw -A PREROUTING -p tcp --dport 21 -j NOTRACK
```

Don’t track FTP connections. Note, that the return connection is still being tracked

MANGLE Table

- This table should mainly be used for mangling packets. In other words, you may freely use the mangle targets within this table, to change TOS (Type Of Service) fields, etc.
- You are strongly advised not to use this table for any filtering; nor will any DNAT, SNAT or Masquerading work in this table.
- Valid Targets
  - TOS
  - TTL
  - MARK
  - SECMARK
  - CONNSECMARK

TOS Target

- Changes the Type of Service field in the IPv4 header.
- Generally this field is ignored
- Don’t set this in other words for packets going to the Internet unless you want to make routing decisions on it, with iproute2.

```
iptables -t mangle -A PREROUTING -p TCP --dport 22 -j
TOS –set-tos 0x10
```

Set the TOS value to HEX 10

TTL Target

- Changes the Time To Live field of the packet.
- We could tell packets to only have a specific TTL and so on. One good reason for this could be that we don’t want to give ourself away to nosy Internet Service Providers. Some Internet Service Providers do not like users running multiple computers on one single connection, and there are some Internet Service Providers known to look for a single host generating different TTL values, and take this as one of many signs of multiple computers connected to a single connection.

```
iptables -t mangle -A POSTROUTING -o eth0 -j TTL --ttl-
set 64
```

- Set all outgoing packets on eth0 to TTL 64

MARK Target

- The MARK target is used to set special mark values to the packet. These marks could then be recognized by the iproute2 programs to do different routing on the packet depending on what mark they have, or if they don't have any. We could also do bandwidth limiting and Class Based Queuing based on these marks.

```
iptables -A PREROUTING -i eth0 -t mangle -p tcp --dport
25 -j MARK --set-mark 1
```

- Market SMTP packets with "1". Then, the rules can be added to the routing tables to route packets marked with 1 to a specific connection

SECMARK and CONNSECMARK Target

- SECMARK and CONNSECMARK are used increase the security of the system by bringing the firewall functions to specific processes.
- For example, only the ftpd service should be able to use ports 21 and 22.

- The SECMARK target can be used to set security context marks on single packets for usage in SELinux and other security systems that are able to handle these marks. This is then used for very fine grained security on what subsystems of the system can touch what packets.
- CONNSECMARK is used for the entire session

NAT Table

- This table should only be used for NAT (Network Address Translation) on different packets. In other words, it should only be used to translate the packet's source field or destination field. Note that, as we have said before, only the first packet in a stream will hit this table. After this, the rest of the packets will automatically have the same action taken on them as the first packet. The actual targets that do these kind of things
- Valid Targets
  - DNAT
  - SNAT
  - MASQUERADE
  - REDIRECT

DNAT Target

- The DNAT target is mainly used in cases where you have a public IP and want to redirect accesses to the firewall to some other host (on a DMZ for example). In other words, we change the destination address of the packet and reroute it to the host.

```
iptables -t nat -A PREROUTING -d 205.254.211.17
    -j DNAT --to-destination 192.168.100.17
```

Changes any packet with destination IP matching 205.254.211.17 to 192.168.100.17.

- Note that something must be done with the return packet, if any (See SNAT)

SNAT Target

- SNAT is mainly used for changing the source address of packets. For the most part you'll hide your local networks or DMZ, etc. A very good example would be that of a firewall of which we know outside IP address, but need to substitute our local network's IP numbers with that of our firewall. With this target the firewall will automatically SNAT and De-SNAT the packets, hence making it possible to make connections from the LAN to the Internet.
- Example, if your network uses 192.168.0.0/netmask for example, the packets would never get back from the Internet, because IANA has regulated these networks (among others) as private and only for use in isolated LANs.

```
iptables -t nat -A POSTROUTING -s 192.168.100.17 -j SNAT
    --to-destination 205.254.211.17
```

- For any packets with destination IP 192.168.100.17, change the destination IP to 205.254.211.17

MASQUERADE Target

- The MASQUERADE target is used in exactly the same way as SNAT, but the MASQUERADE target takes a little bit more overhead to compute. The reason for this, is that each time that the MASQUERADE target gets hit by a packet, it automatically checks for the IP address to use, instead of doing as the SNAT target does - just using the single configured IP address.
- The MASQUERADE target makes it possible to work properly with Dynamic DHCP IP addresses that your ISP might provide for your PPP , PPPoE or SLIP connections to the Internet.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

MASQUERADE changes the source address to match the interface

REDIRECT Target

- The REDIRECT target is used to redirect packets and streams to the machine itself. This means that we could for example REDIRECT all packets destined for the HTTP ports to an HTTP proxy like squid, on our own host. Locally generated packets are mapped to the 127.0.0.1 address. In other words, this rewrites the destination address to our own host for packets that are forwarded, or something alike. The REDIRECT target is extremely good to use when we want, for example, transparent proxying, where the LAN hosts do not know about the proxy at all.
- REDIRECT target is only valid within the PREROUTING and OUTPUT chains of the nat table

```
iptables -t nat -A PREROUTING -p tcp --dport 80
   -j REDIRECT --to-ports 8080
```

- Redirect packets to the local host. E.g., FORWARD packets are redirected to the local host and port number changed to 8080