Security of Embedded Linux systems

Tips and tricks how to securely configure a Linux firewall using Netfilter

Carsten Emde
Open Source Automation Development Lab (OSADL) eG

Agenda – Theory

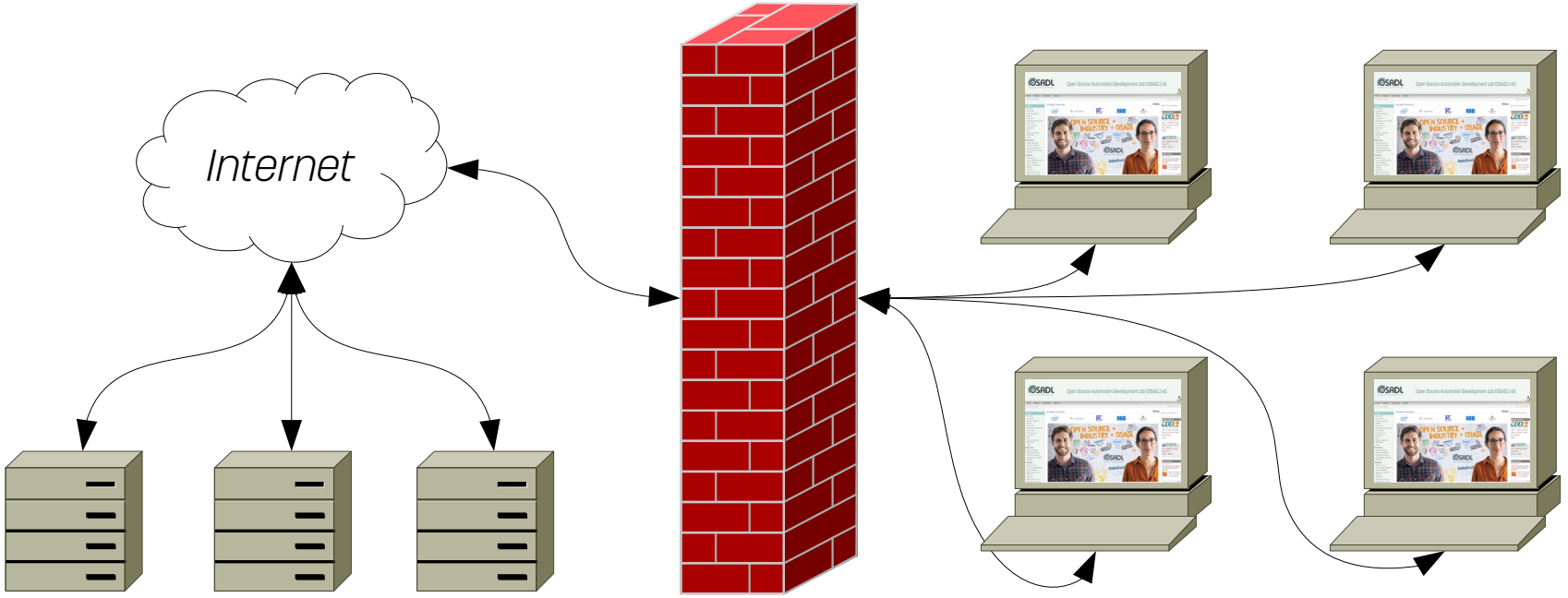
- What is a firewall?
- How Netfilter works
- Important components of Netfilter
  - Static filter
  - Network Address Translation (NAT)
  - Dynamic filter

Agenda – Practice

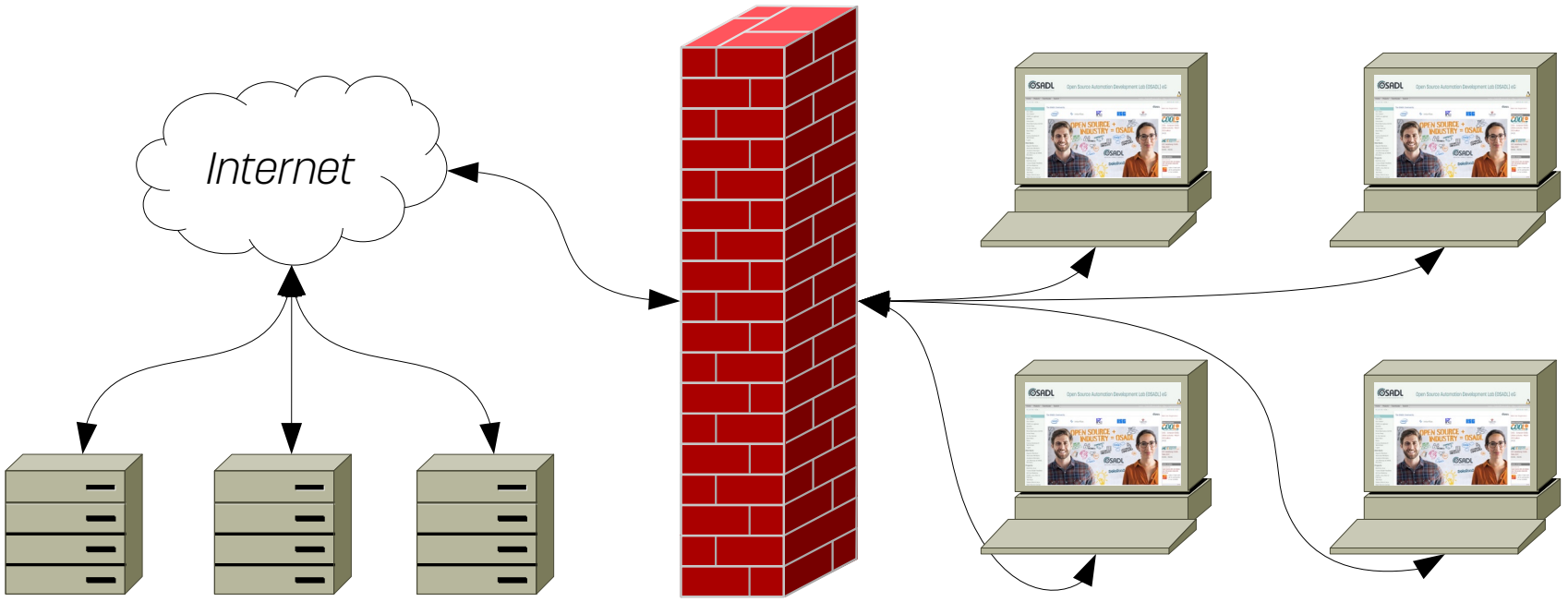
Standard firewall settings of embedded systems

1. Allow only a particular host to connect (static filter)
2. Allow several systems behind a firewall to access the Internet from a single source IP address (NAT)
3. Allow access to several systems behind a firewall through a single destination IP address (NAT)
4. Allow only peers with an IP address within selected address ranges to connect – e.g. from a particular country (dynamic filter)

What is a network firewall?

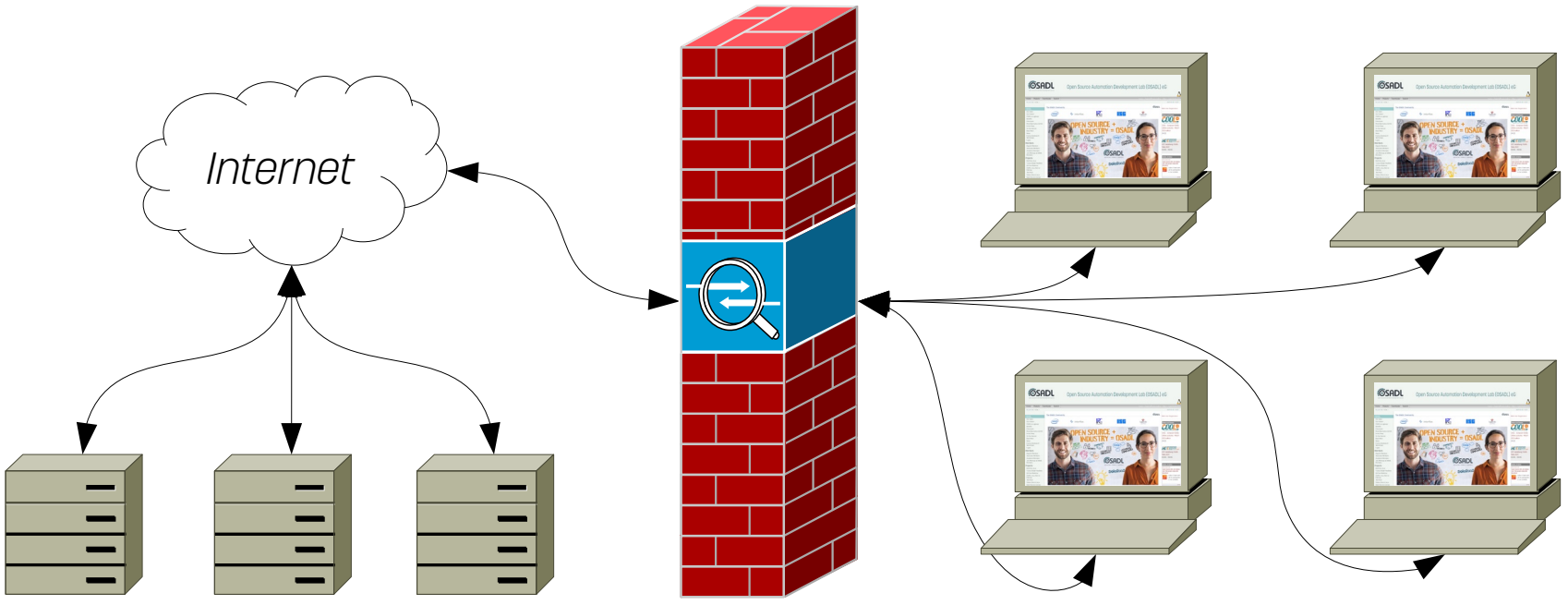


What is a network firewall?



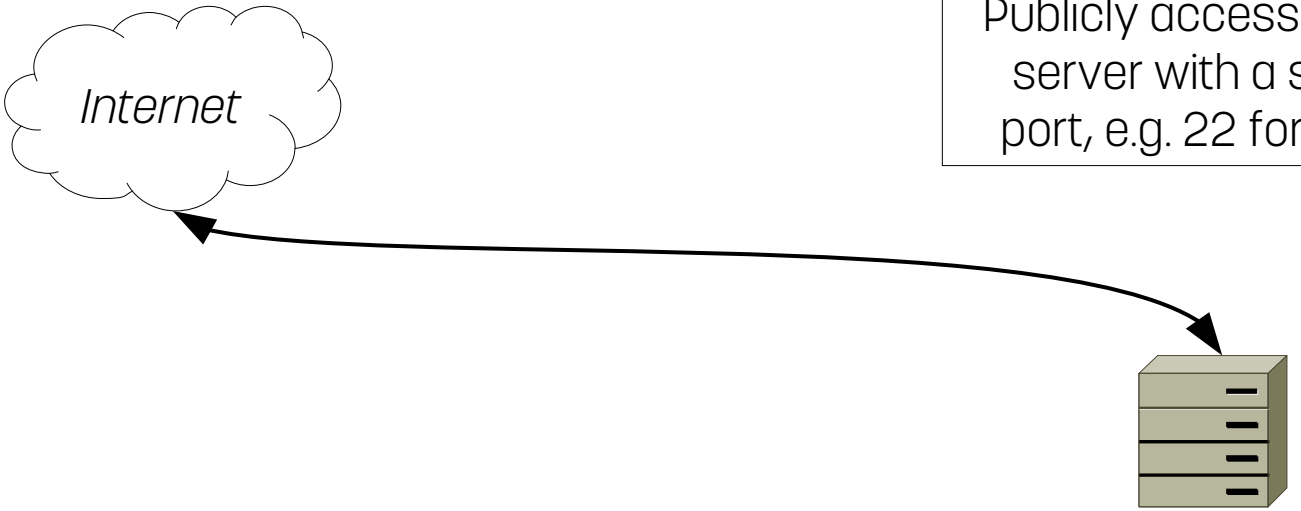
A network firewall is not a fire wall that must be tightly closed to be functional

What is a network firewall?



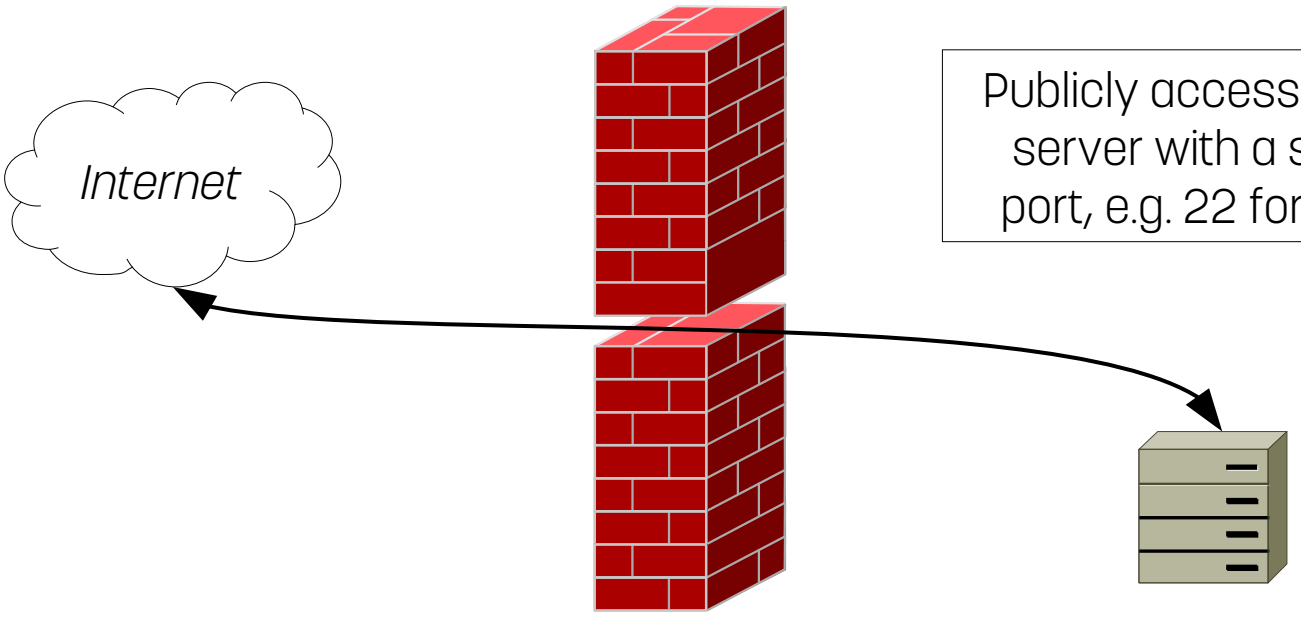
A network firewall is only functional if it has at least one hole

Is a network firewall always needed?



Publicly accessible Internet server with a single open port, e.g. 22 for ssh server

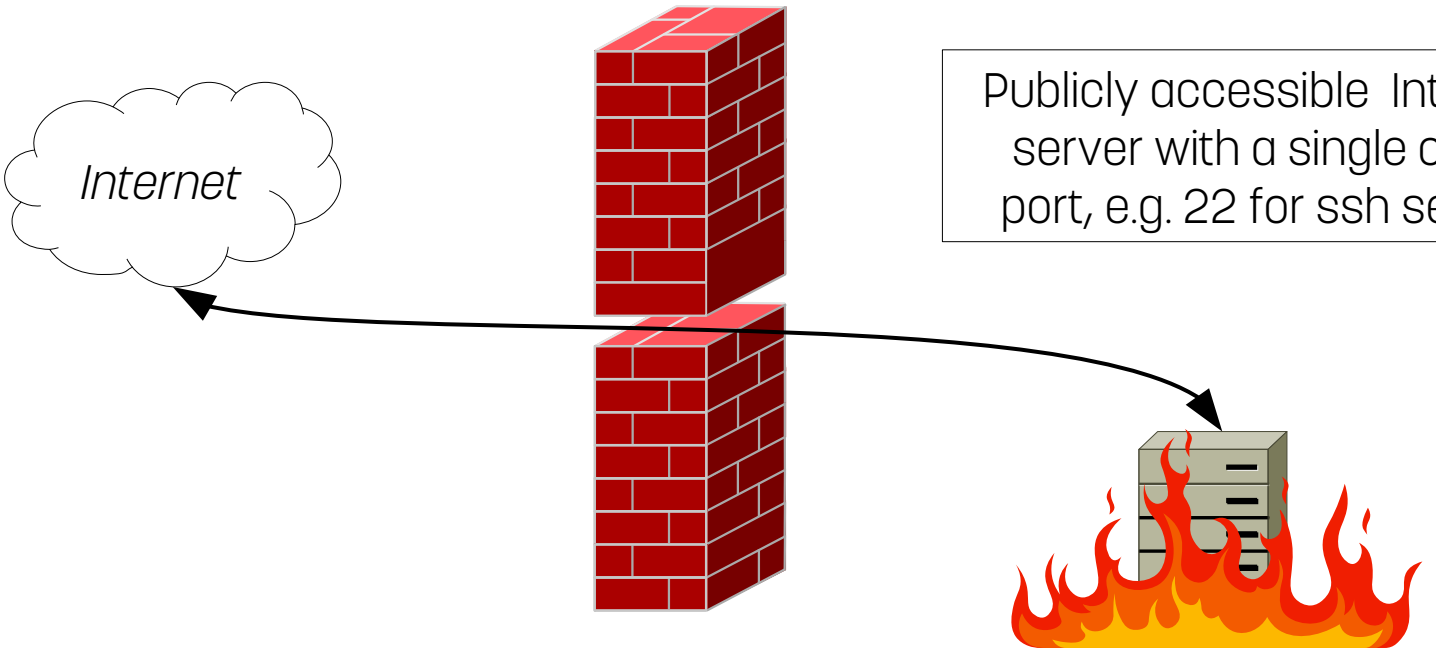
Is a network firewall always needed?



Publicly accessible Internet server with a single open port, e.g. 22 for ssh server

A standard network firewall normally does not protect an active service

Is a network firewall always needed?



Publicly accessible Internet server with a single open port, e.g. 22 for ssh server

Instead of a firewall the server software must be maintained for security

A network firewall is not a single tool

The term *firewall* is not officially defined and also not related to a particular functionality. A network firewall may consist of many different tools often used simultaneously.

- **Netfilter (iptables)**
- Proxy server
  - Authentication
  - Traffic logging
- Packet inspector
  - Malware scanner

iptables or *nftables*?

- The established Linux firewall utility *iptables* is currently being replaced by *nftables*.
- Some distributions, notably Debian 10, are now offering *nftables*, but one can switch back and forth between *iptables-nft* and *iptables-legacy* using so-called *update-alternatives*.
- In addition, tools to migrate *iptables* scripts to *nftables* scripts are available.

1. A network firewall may act as a filter

Only allow particular

- protocols
- source addresses
- destination addresses
- destination ports
- etc.

to enter (*INPUT* chain) or quit (*OUTPUT* chain) the firewall, or be routed (*FORWARD* chain) through the firewall. The related table is called *filter* which is the default if no table is given.

2. A network firewall may rewrite addresses

Network address translation (*NAT*) can rewrite

- source address (*SNAT*)
- destination address (*DNAT*)
- destination port number
- etc.

when routing packages through the firewall. The related table is called *nat*, and the most frequently used chains are *PREROUTING* and *POSTROUTING*.

The two principles of programming a firewall

- Default: **all closed**
- Allow only acceptable connections

- Default: **all open**
- Explicitly open acceptable connections
- Close all other connections

The two principles of programming a firewall

- Default: **all closed**
- Allow only acceptable connections

Advantage:
The filter is continuously operative even while the service restarts

- Default: **all open**
- Explicitly open acceptable connections
- Close all other connections

Disadvantage:
The filter function is (very) shortly interrupted while the firewall is restarting

The two principles of programming a firewall

- Default: **all closed**
- Allow only acceptable connections

Disadvantage:
Misconfiguration can mean that you cut the branch you are sitting on.

- Default: **all open**
- Explicitly open acceptable connections
- Close all other connections

Advantage:
The system remains accessible even if the service does not restart because of a misconfiguration

How do we instruct *iptables* what to do?

- Every *filter* or *nat* line of an *iptables* script needs a so-called jump target (option `-j`)
- To allow a particular condition, the *ACCEPT* jump target is specified.
- To disallow a particular condition, the *REJECT* or *DROP* jump target is specified.

The two methods of disallowing a connection

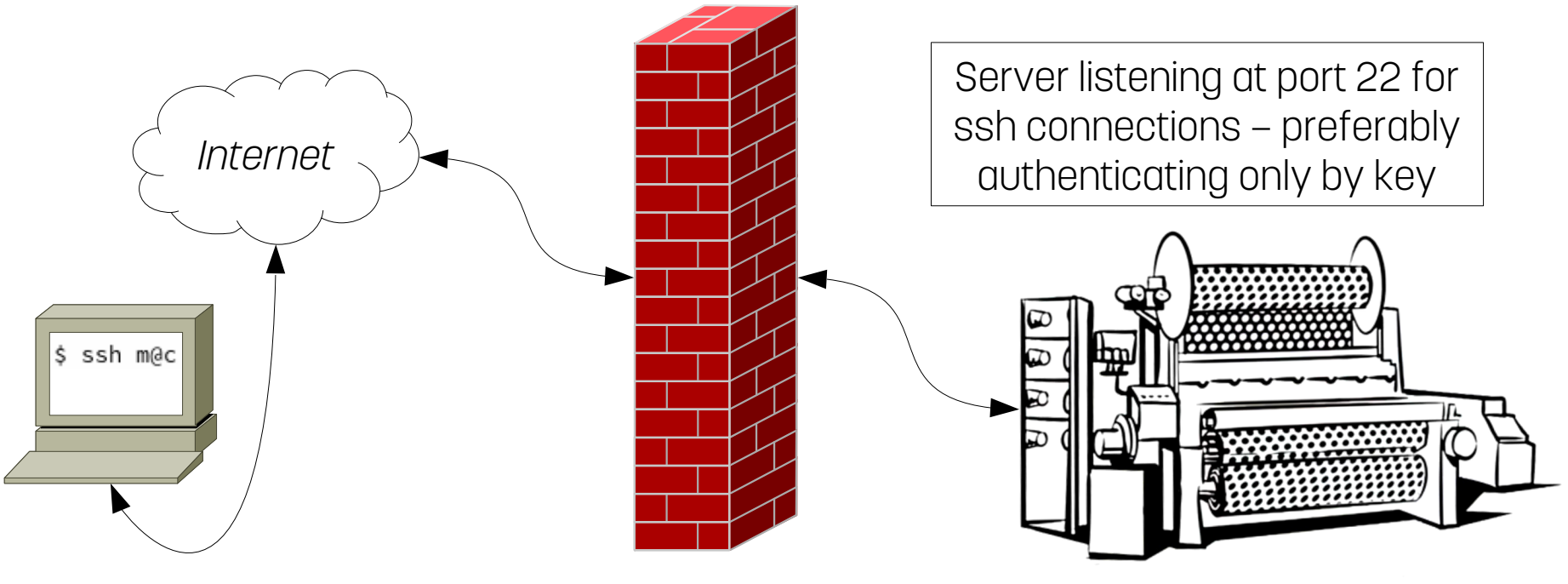
- *REJECT*
  Generate an answer packet to inform the requesting peer that the door remains closed. This is the polite way to disallow a connection and is used for friends. It helps keep the Internet responsive and usable.

- *DROP*
  Do not send an answer, but keep the callers waiting until they run into network timeout. It is as rude as attackers deserve no better. It does not help us this time, but it helps others, since it slows down the frequency of attacks.

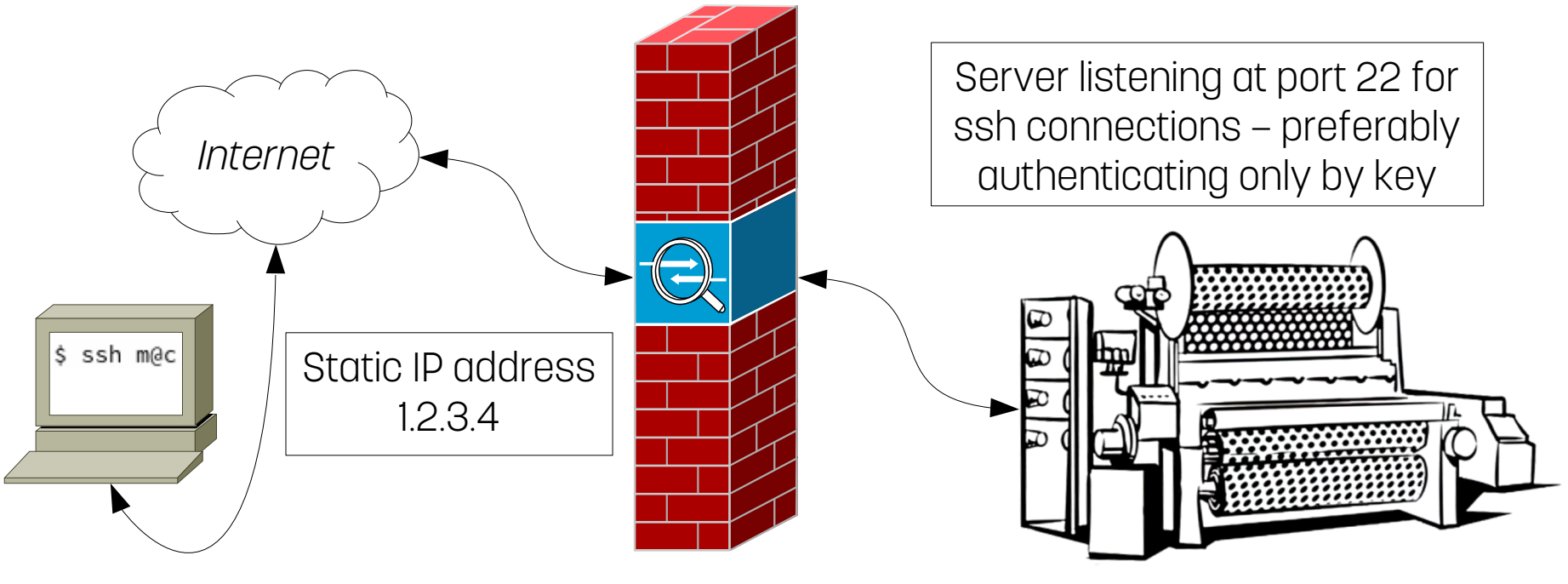
Standard firewall settings of embedded systems

1. Allow only a particular host to connect
2. Allow several systems behind a firewall to access the Internet from a single source IP address
3. Allow access to several systems behind a firewall by using a single destination IP address
4. Allow only a selection of IP addresses – e.g. from a particular country

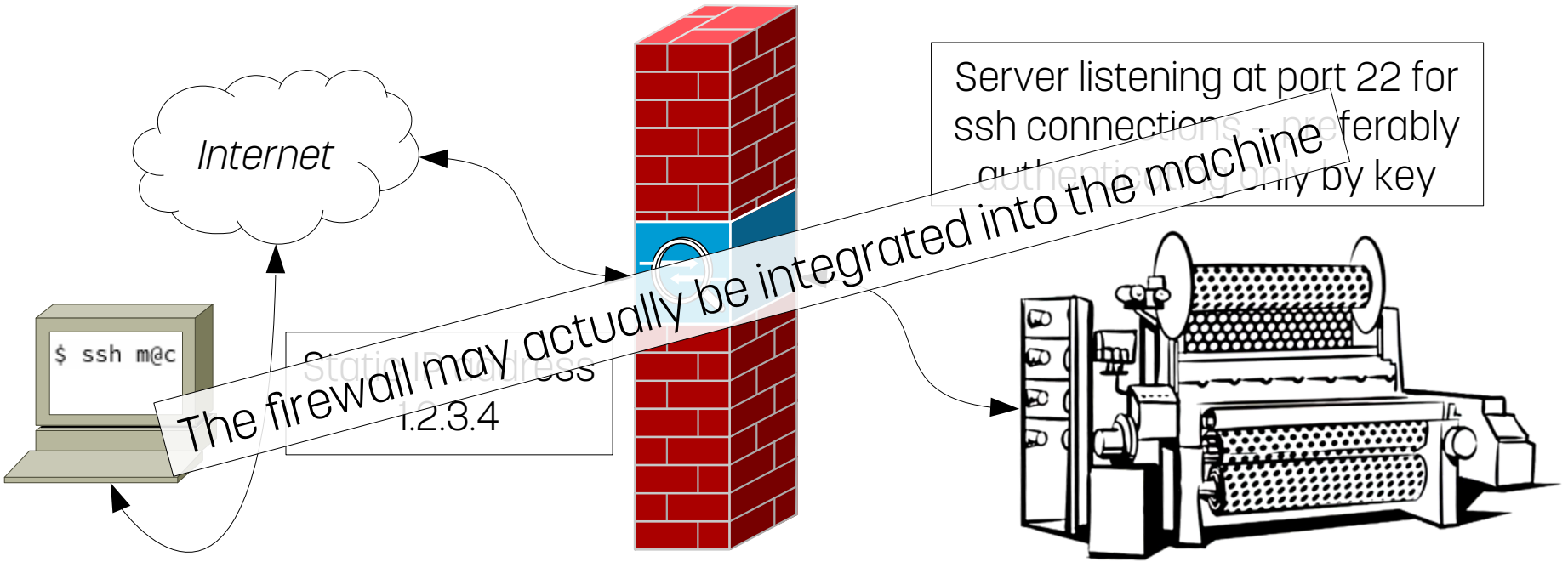
1. Allow only a particular host to connect



1. Allow only a particular host to connect



1. Allow only a particular host to connect



1. Allow only a particular host to connect

The *iptables* script commands may look like:

```
*filter
:INPUT ACCEPT [0:0]

-A INPUT -m state --state NEW -m tcp -p tcp -s 1.2.3.4 --dport 22 -j ACCEPT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -j DROP
```

Please note that the ordering of the command lines is important:

- Option `-A` appends (which would make no sense after the above DROP jump)
- Alternatively, the option `-I` is available to insert a new line by default at the top or, if specified, as a particular rule number.

1. Allow only a particular host to connect

Unfortunately, the solution so far only works, if the source address is a static IP address.

However, it is increasingly difficult to obtain such a static IP address. A later example of a standard firewall setting of embedded systems will show how to cope with dynamic IP addresses.

Why is it difficult to obtain a static IP address?

The IPv4 address range is 32 bit long resulting in a little less than 4,3 billion addresses.

Question: Where are they?

Answer: In the early days of the Internet, the 32-bit address range was considered inexhaustible, and large quantities of addresses were generously allocated even to individual companies.

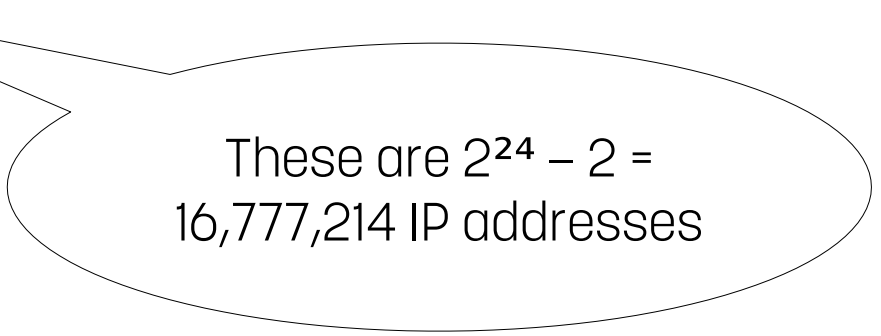
Why is it difficult to obtain a static IP address?

Some time later, dynamic IP addresses were invented and a dynamic host configuration protocol was standardized so that a company no longer needs more than a handful of IP addresses. But no one had neither the courage nor the power to take those addresses away from companies.

But again, where are they?

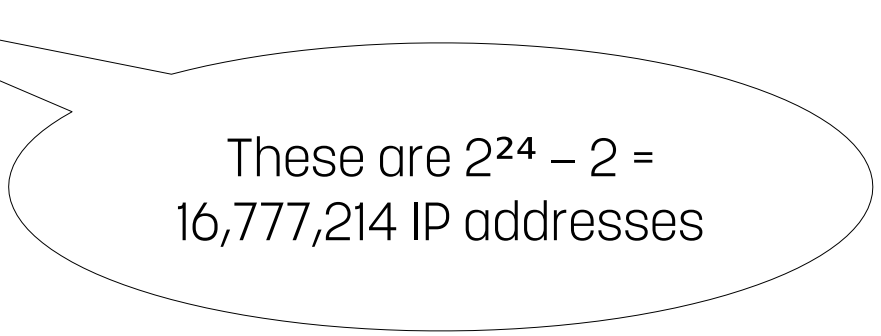
Where did the static IP addresses get lost?

```
NetRange:       19.0.0.0 - 19.255.255.255
CIDR:           19.0.0.0/8
NetName:        FINET
NetHandle:      NET-19-0-0-0-1
Parent:          ()
NetType:        Direct Assignment
OriginAS:
Organization:   Ford Motor Company (FORDMO)
RegDate:        1988-06-15
Updated:        2010-05-07
Ref:            https://rdap.arin.net/registry/ip/19.0.0.0
```



Where did the static IP addresses get lost?

```
inetnum:        53.0.0.0 - 53.255.255.255
netname:        DAIMLER-NET1
org:            ORG-DA474-RIPE
descr:          Daimler AG
country:        DE
admin-c:        BS4256-RIPE
tech-c:         BS4256-RIPE
status:         LEGACY
mnt-by:         RIPE-NCC-LEGACY-MNT
mnt-by:         DAIMLER-MNT
created:        1970-01-01T00:00:00Z
last-modified:  2021-03-01T15:09:29Z
source:         RIPE
```



Where did the static IP addresses get lost?

```
inetnum:        129.206.0.0 - 129.206.255.255
netname:        HD-NET
descr:          Ruprecht-Karls-Universitaet Heidelberg
descr:          Heidelberg, Germany
country:        DE
org:            ORG-UH5-RIPE
admin-c:        UH593-RIPE
tech-c:         UH593-RIPE
status:         LEGACY
remarks:        ************************************************
remarks:        * DEFAULT ABUSE CONTACT: abuse@uni-heidelberg.de *
remarks:        ************************************************
mnt-by:         BELWUE-MNT
created:        1970-01-01T00:00:00Z
last-modified:  2019-12-04T13:06:07Z
source:         RIPE
```

These are still $2^{16} - 2$ = 65,534 IP addresses

Where did the static IP addresses get lost?

```
inetnum:        161.42.0.0 - 161.42.255.255
netname:        KHD-NET
descr:          Universitaetsklinikum Heidelberg
descr:          Heidelberg, Germany
country:        DE
org:            ORG-UH10-RIPE
admin-c:        UH821-RIPE
tech-c:         UH821-RIPE
status:         LEGACY
remarks:        ******************************************************
remarks:        * DEFAULT ABUSE CONTACT: abuse@med.uni-heidelberg.de *
remarks:        ******************************************************
mnt-by:         BELWUE-MNT
created:        2004-01-20T10:47:33Z
last-modified:  2015-06-19T13:53:07Z
source:         RIPE
```

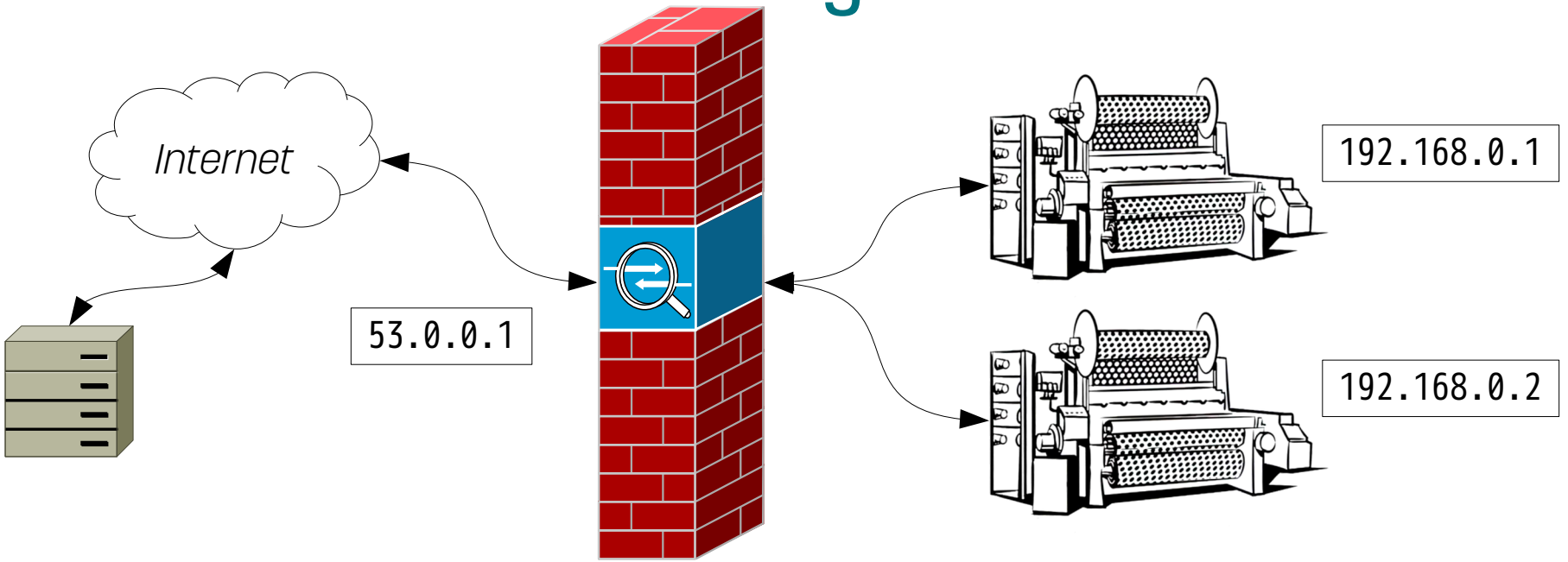
These are still $2^{16} - 2$ = 65,534 IP addresses

But another solution is ready ...

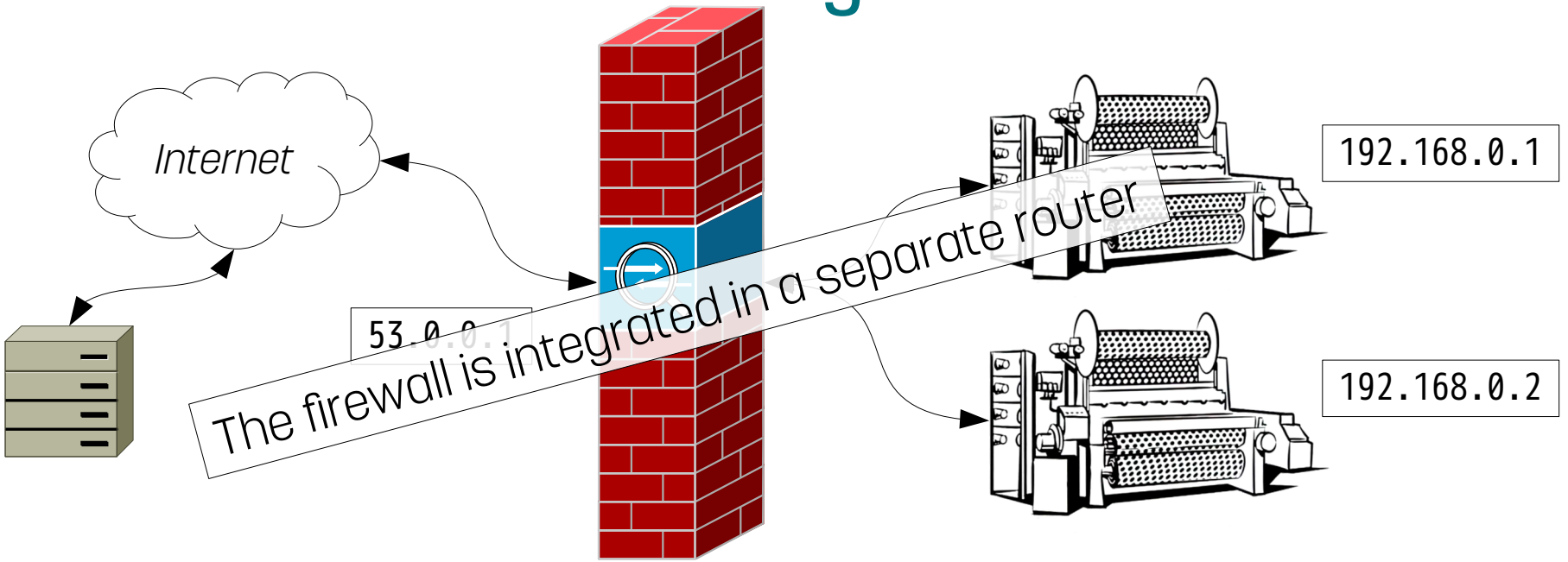
... and this solution is called IPv6, which provides such a large address space that presumably everybody and every machine on earth can be supplied with static IP addresses.

However, it may still take some time until IPv6 is fully functional everywhere.

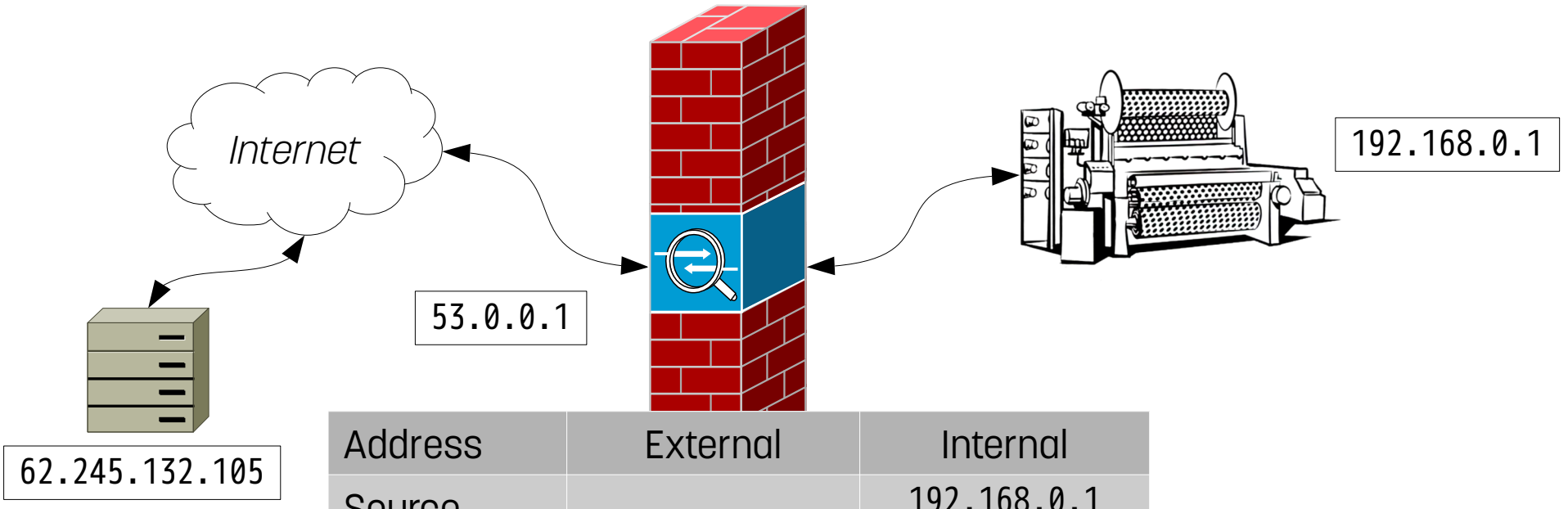
2. Allow several systems behind a firewall to access the Internet from a single source IP address



2. Allow several systems behind a firewall to access the Internet from a single source IP address

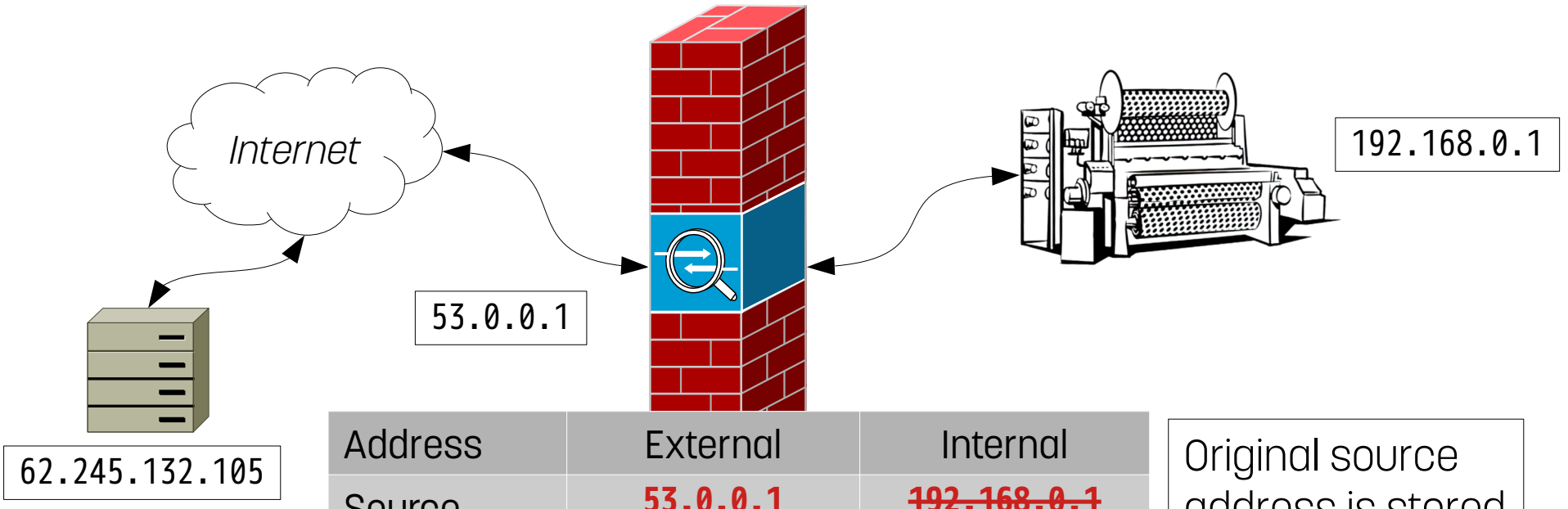


2. a) Send a request to www.osadl.org



Address	External	Internal
Source		192.168.0.1
Destination		62.245.132.105

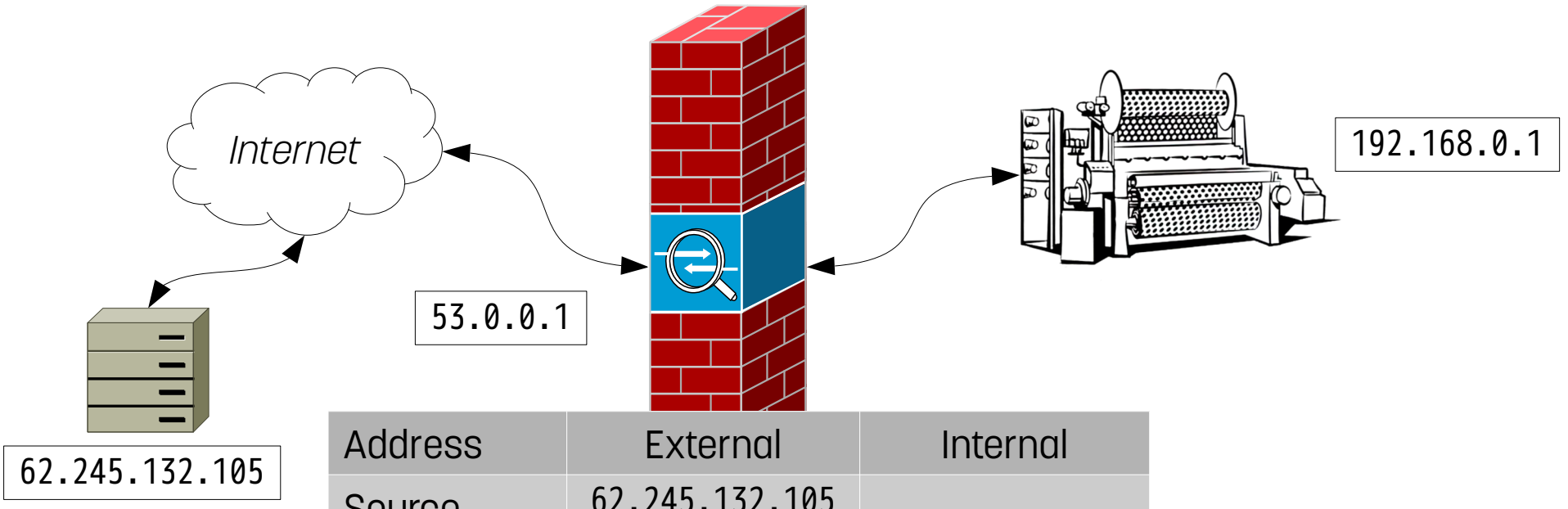
2. a) Send a request to www.osadl.org: SNAT



Address	External	Internal
Source	53.0.0.1	~~192.168.0.1~~
Destination	62.245.132.105	62.245.132.105

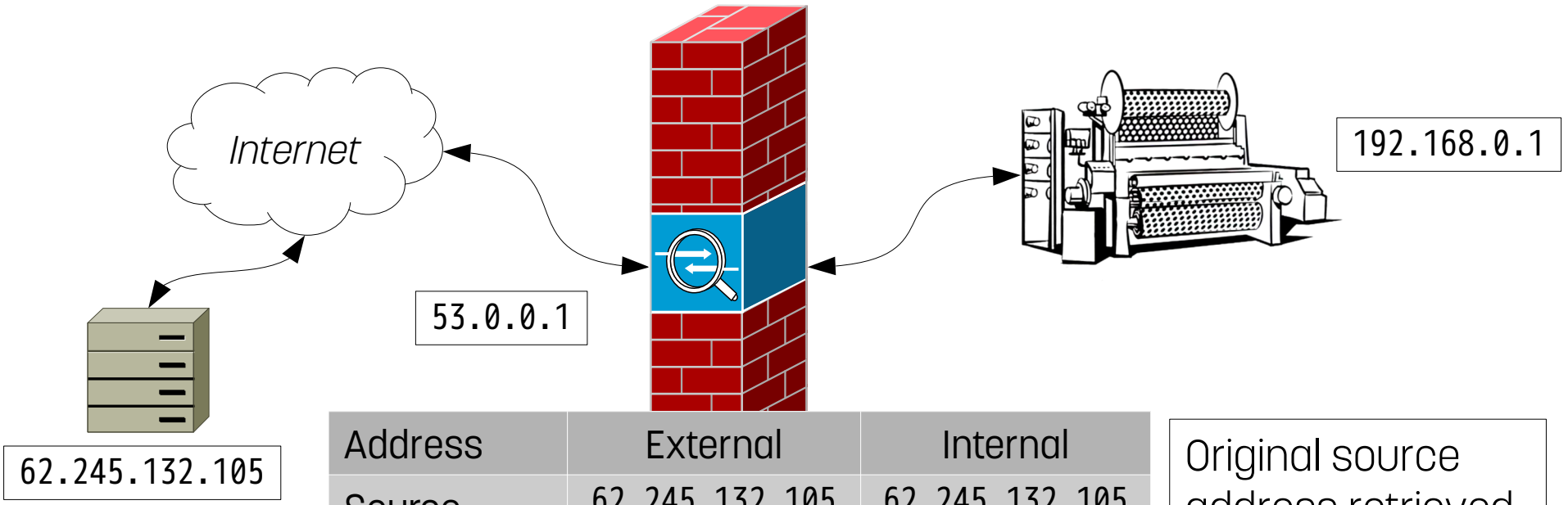
Original source address is stored in a table

2. b) Receive response from www.osadl.org: DNAT



Address	External	Internal
Source	62.245.132.105	
Destination	53.0.0.1	

2. b) Receive response from www.osadl.org: DNAT



Address	External	Internal
Source	62.245.132.105	62.245.132.105
Destination	~~53.0.0.1~~	192.168.0.1

Original source address retrieved from the table

2. Allow several systems behind a firewall to access the Internet from a single source IP address

The *iptables* script commands may look like:

```
*filter
-A FORWARD -i eth0 -s 192.168.0.0/24 -j ACCEPT
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

*nat
-A POSTROUTING -s 192.168.0.0/24 -o eth1 -j MASQUERADE
```

Please note that routing (forwarding) must additionally be enabled in the kernel:
`echo 1 >/proc/sys/net/ipv4/ip_forward`

2. Allow several systems behind a firewall to access the Internet from a single source IP address

The *iptables* script commands may look like:

```
*filter
-A FORWARD -i eth0 -s 192.168.0.0/24 -j ACCEPT
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

*nat
-A POSTROUTING -s 192.168.0.0/24 -o eth1 -j MASQUERADE
```

The combination of outgoing SNAT and incoming DNAT is called *masquerading*

Please note that routing (forwarding) must additionally be enabled in the kernel:
`echo 1 >/proc/sys/net/ipv4/ip_forward`

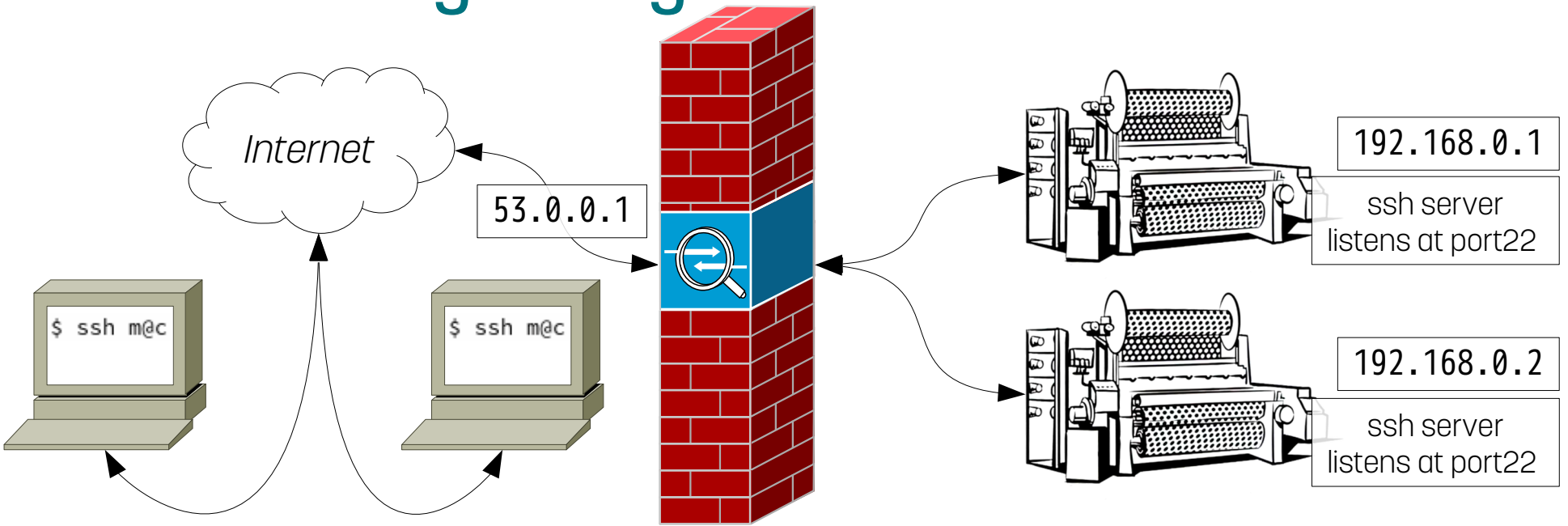
2. Allow several systems behind a firewall to access the Internet from a single source IP address

The *iptables* script commands may **better not** look like:

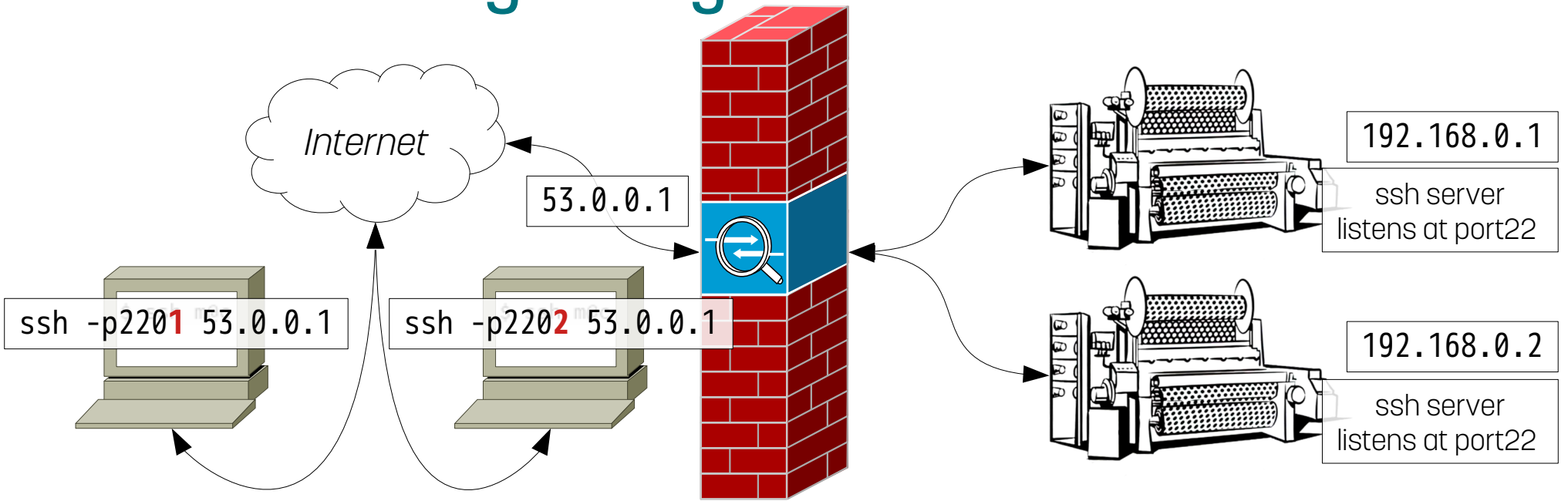
```
*filter
-A FORWARD -m tcp -p tcp -i eth0 -s 192.168.0.0/24 --dport 443 -j ACCEPT
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

*nat
-A POSTROUTING -s 192.168.0.0/24 -o eth1 -j MASQUERADE
```

Please note that routing (forwarding) must additionally be enabled in the kernel:
`echo 1 >/proc/sys/net/ipv4/ip_forward`

3. Allow access to several systems behind a firewall using a single destination IP address



3. Allow access to several systems behind a firewall using a single destination IP address



We now use DNAT plus port rewriting, and the port number decodes the destination

3. Allow access to several systems behind a firewall using a single destination IP address

The *iptables* script commands may look like:

```
*filter
-A FORWARD -m tcp -p tcp --dport 22 -o eth0 -d 192.168.0.0/24 -j ACCEPT

*nat
-A PREROUTING -m tcp -p tcp --dport 2201 -i eth1 -j DNAT --to-destination 192.168.0.1:22
-A PREROUTING -m tcp -p tcp --dport 2202 -i eth1 -j DNAT --to-destination 192.168.0.2:22
```

Please note that routing (forwarding) must additionally be enabled in the kernel:
`echo 1 >/proc/sys/net/ipv4/ip_forward`

4. Only allow peers with an IP address within selected address ranges to connect

- This is a workaround for dynamic IP addresses
- It only works, if the allowable IP range can be determined, e.g. since they belong to a particular Internet service provider or are registered in a given country.
- It uses the *ipset* utilities (`yum install ipset`, `dnf install ipset`, or `apt-get install ipset`) and connects the *ipset* setting via Linux kernel to *iptables*.

4. a) Prepare the system and fetch the data

1. Initialize kernel *ipset* list *allowssh*

```
ipset -q destroy allowssh
ipset create allowssh hash:net family inet 2> /dev/null
ipset -q flush allowssh
```

2. Generate ipset ranges (*allowssh* list)

```
curl -s https://www.nirsoft.net/countryip/de.csv >data.csv
dos2unix data.csv 2>/dev/null
grep -v ^$ data.csv >de.csv
grep Telekom de.csv | grep Deutsch | cut -d, -f1-2 >dtag.csv
```

4. b) Format of registration data is different from what *ipset* expects

1. Output format example
   ```
   2.160.0.0,2.175.255.255,1048576,28/09/10,Telekom Deutschland GmbH
   62.153.0.0,62.153.255.255,65536,04/02/00,Deutsche Telekom AG
   ```
2. Input format required by *ipset*
   `network/mask`
3. A tool is needed to convert the format, e.g. *rangemask* (can be obtained from OSADL).

4. c) Example of *rangetomask*

```
# iprangetomask 192.168.0.0 192.168.0.255
network=192.168.0.0
broadcast=192.168.0.255
mask=255.255.255.0
prefix=24
peers=254
```

4. d) Write the data to the *ipset* list

```
# for i in `cat dtag.csv`
do
  network=`echo $i | cut -d, -f1`
  broadcast=`echo $i | cut -d, -f2`
  prefixline=`iprangetomask $network $broadcast | grep prefix
  prefix=`echo $prefixline | cut -d= -f2`
  ipset -A allowssh $network/$prefix
done
```

4. e) Finally let *iptables* use the *ipset* list

```
# iptables -I INPUT -m set --match-set allowssh src \
  -p tcp --dport 22 -j ACCEPT
```

Conclusion

- Network filters and NAT techniques are indispensable tools for network security and other enhanced functionality.
- However, they cannot protect against intrusions via necessarily open ports.
- In addition to firewall installation, many more measures still need to be taken to ensure secure and safe operation of a device with network connectivity.