UNIVERSITY OF TRENTO

Network Security

AA 2020/2021

System hardening

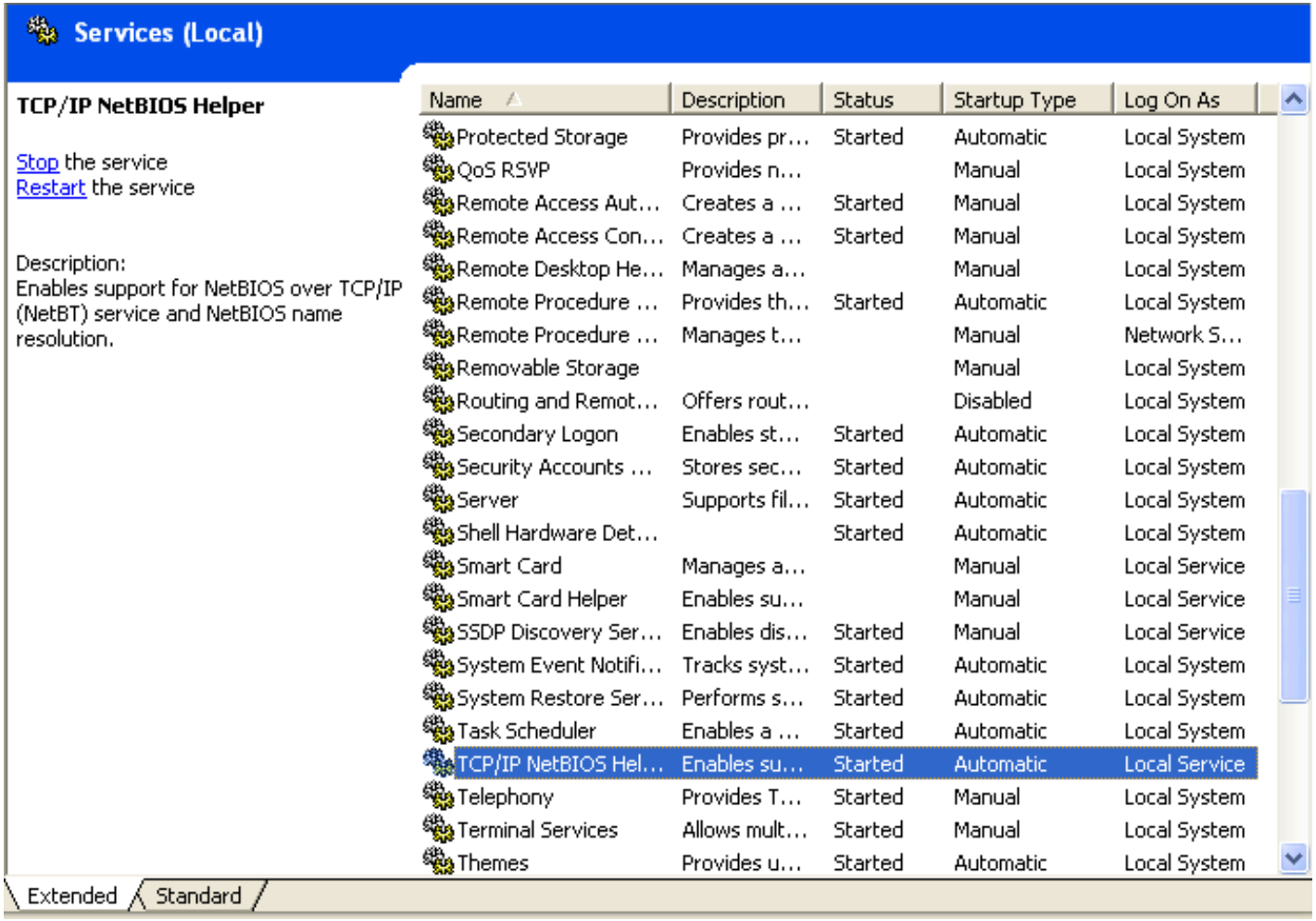
Firewalls

Default configurations

- All systems have a default configuration
  - Personal computers, servers, mainframes,..
- Fresh installation of an operating system
  - Some can be configured at installation time
    - Still limited access to full configuration settings
    - e.g. linux distr. typically allow to select packets but not all packet functionalities
- Default services
  - DHCP, RCP, NetBIOS, ..
  - SSH, VNC, ..
  - Web servers, remote interfaces
- → Default configuration satisfies vast majority of user needs

UNIVERSITY OF TRENTO

Example of default configuration



System hardening

- System hardening is the process by which a system's configuration is tuned to improve its security without compromising its functionality
  - The 100% secure system is one that is turned off
- Sys hardening process takes into account
  - System functionality → what is the role of that system?
    - Home computer
    - File server
    - Web server
    - General purpose server
  - System security → how can the security of the system be improved?
    - Minimise the attack surface of the system

Attack surfaces

- An attack surface is the set of system resources that are exposed to the attacker
  - Weak passwords
  - Software vulnerabilities
  - Misconfigurations
  - Services listening on the network
  - Inaccurate access control
  - …
- Golden rule of information security
  - "Need to know principle" → no user and no system component or process should be authorised or compiled to perform actions that are not strictly necessary for their normal operation
    - aka "If it's not there you can't brake it"

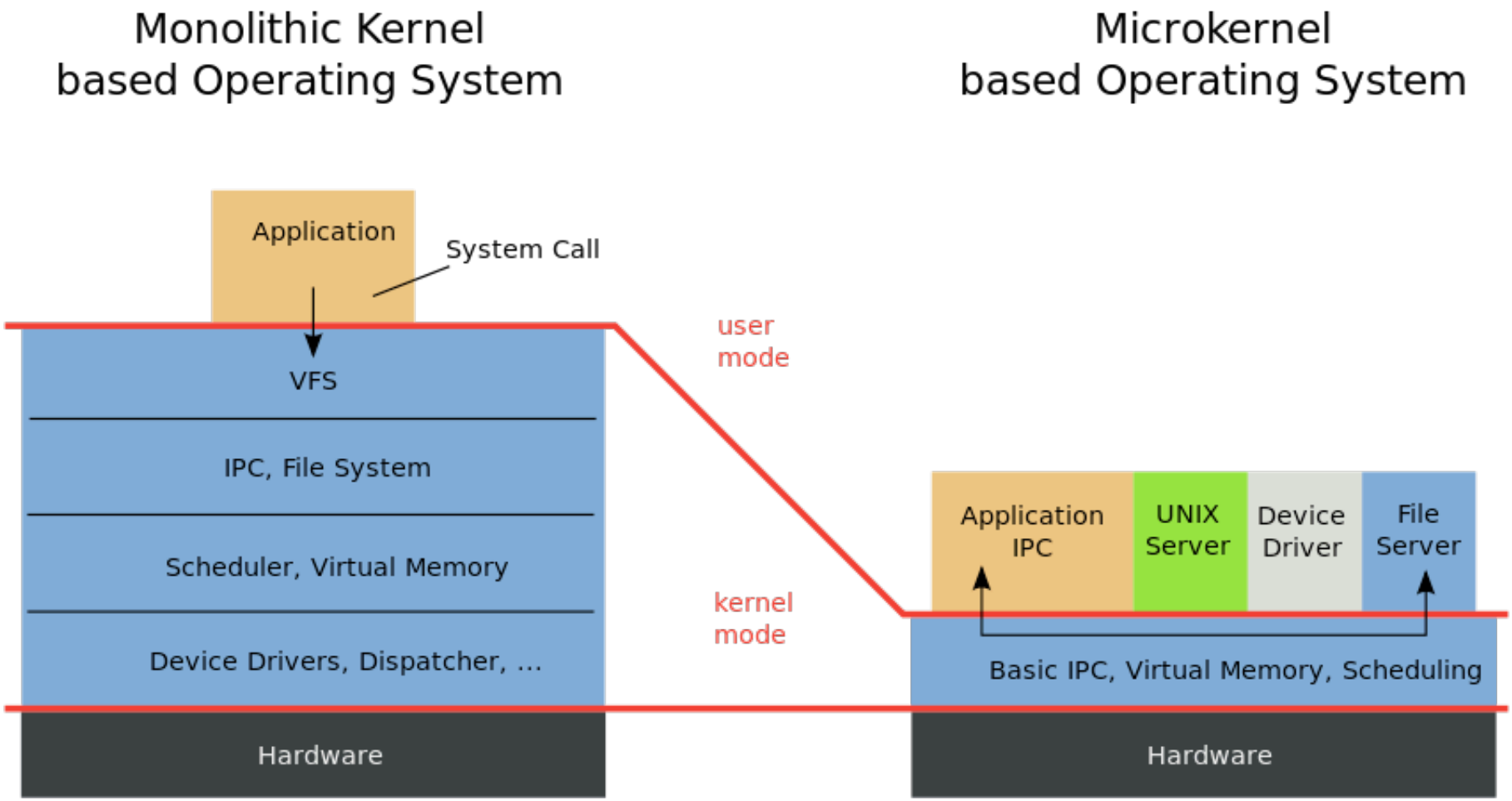
The *Need To Know* principle

- Can be applied at both system, users and processes
- **A system** should be configured such that it does not embed or enable functionalities that are not needed for normal operation

- **A user** should be authorised to only access and modify resources that are necessary for their normal operation
  - If user is NOT authorised, they will NOT be able to accomplish their tasks

UNIVERSITY OF TRENTO

OS design approaches



Minimal user privileges

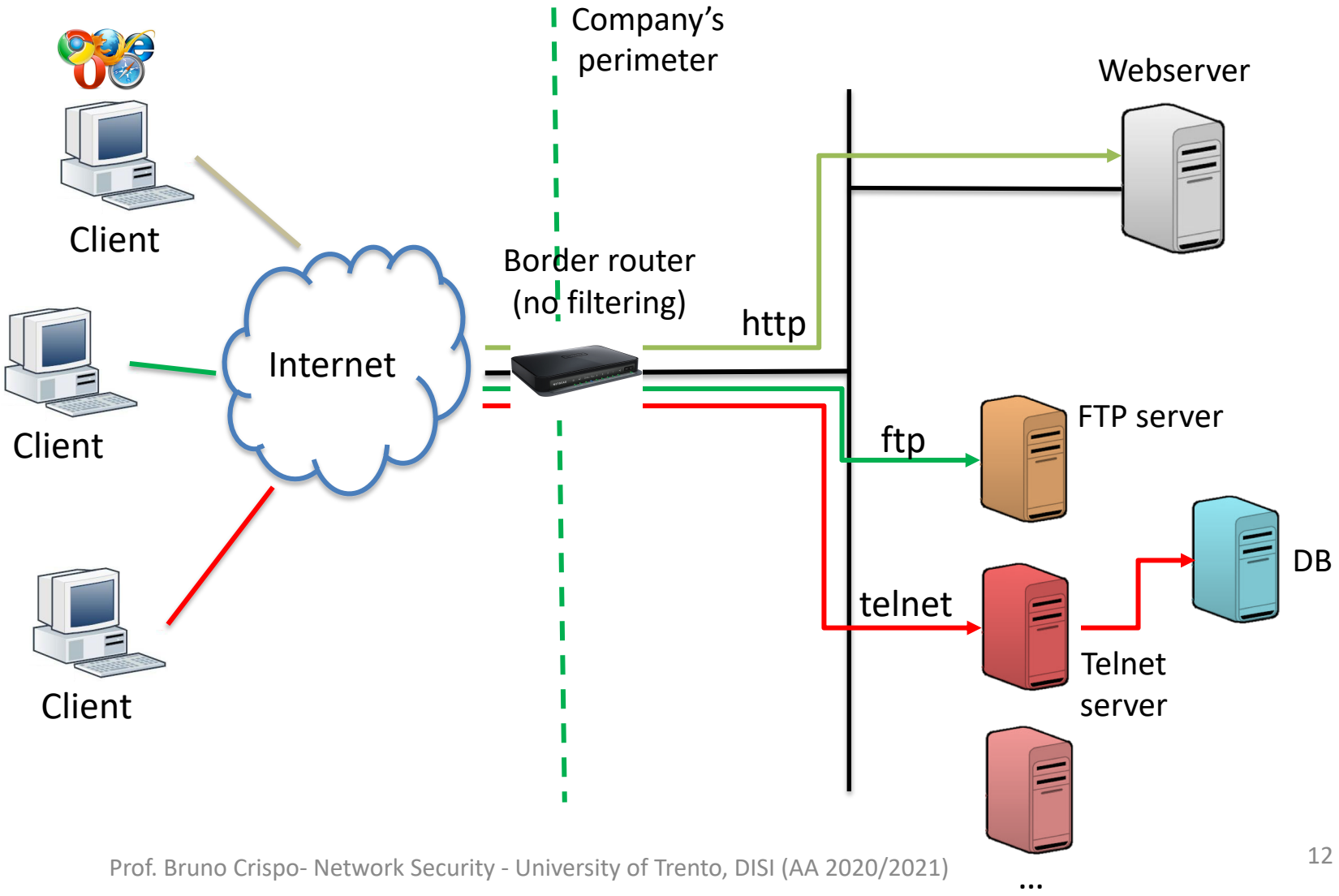
- User should not be allowed to perform more actions on the system than necessary for their operation
- Common policy requirement: restrict the behavior of a user
- To permit different users to do different things, we need a way to identify or distinguish between users
  - *Identification mechanisms to indicate identity*
  - *Authentication mechanisms to validate identity*

FIREWALLS

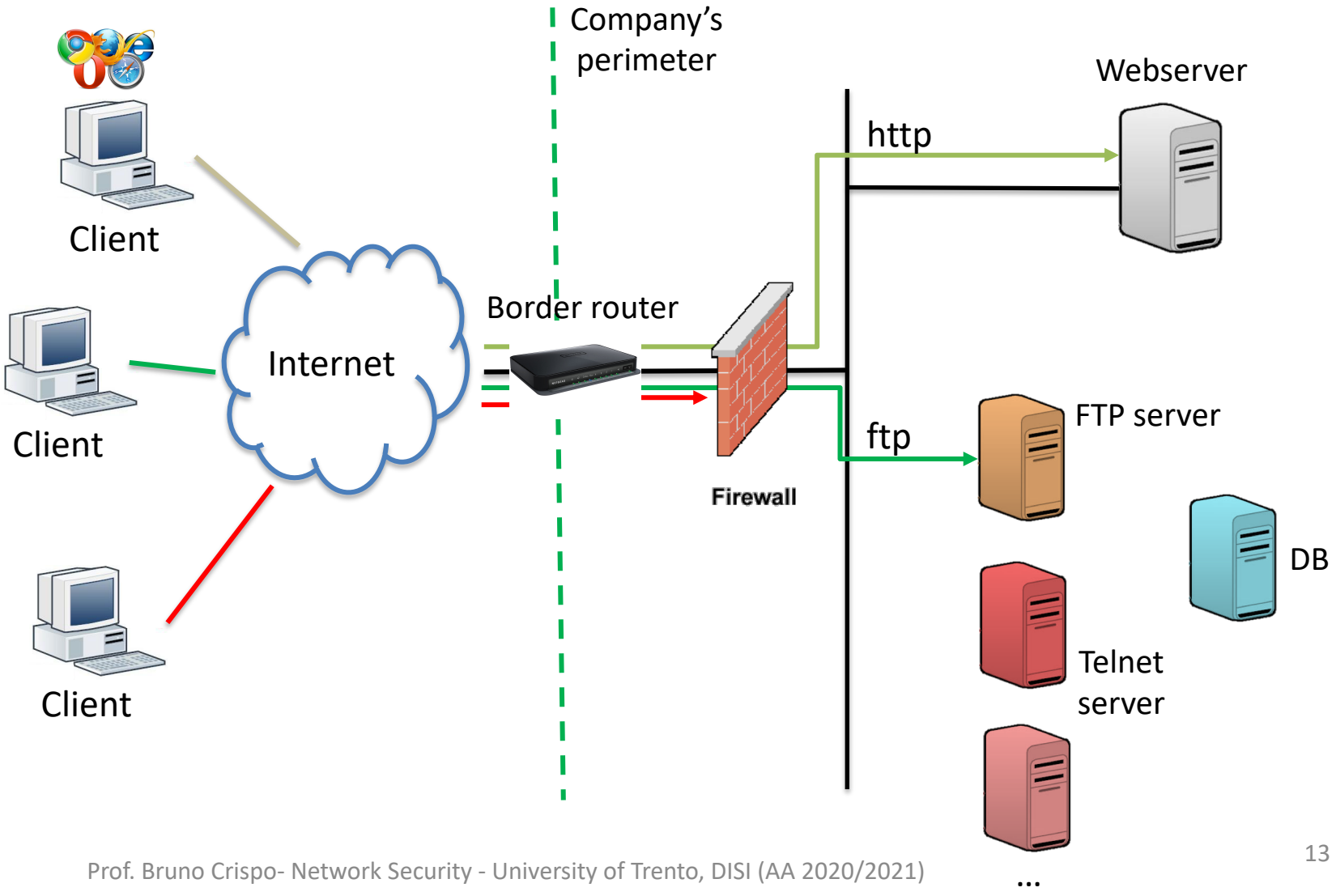
Firewalls for system minimality

- A system's minimal configuration may still have a higher attack surface than necessary
  - e.g. SSH is necessary for remote operation on server
  - However, SSH logins may only be allowed from an internal IP address
    - Additional network measures to minimise attack surface
- Firewalls are perimetral network components that filter incoming (outgoing) traffic from (to) the network
  - Embedded in network devices or as a stand-alone software

No perimetral defense

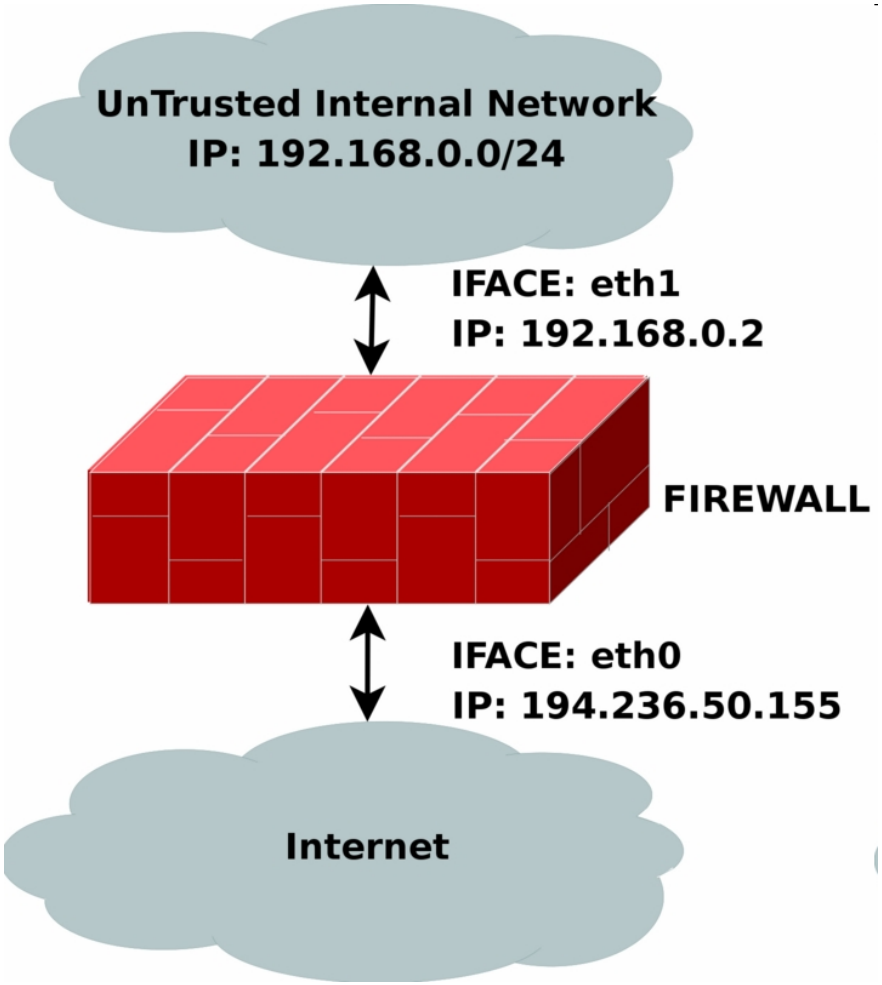


Perimetral defense



Networking with a firewall

- Internal network can be treated as untrusted
  - Do not trust outgoing traffic
  - Connections to remote servers can be regulated
    - E.g. remote storage services could be used to exfiltrate data from an organisation
- Firewalls have at least two network interfaces
  - One facing the external network
    - Or the router
    - This depends on firewall placement w.r.t border router
  - One facing internally
- More interfaces are possible if the firewall sits at the border with three or more networks



Firewall Characteristics

- Design goals
  - All traffic from inside or outside must pass through the firewall (physically blocking all access to the local network except via the firewall)
  - Only authorized traffic (defined by the local security policy) will be allowed to pass
  - The firewall itself is immune to penetration (use of a trusted system with a secure operating system)

UNIVERSITY OF TRENTO

Default Policies

Default deny:

All what is not explicitly allowed is denied

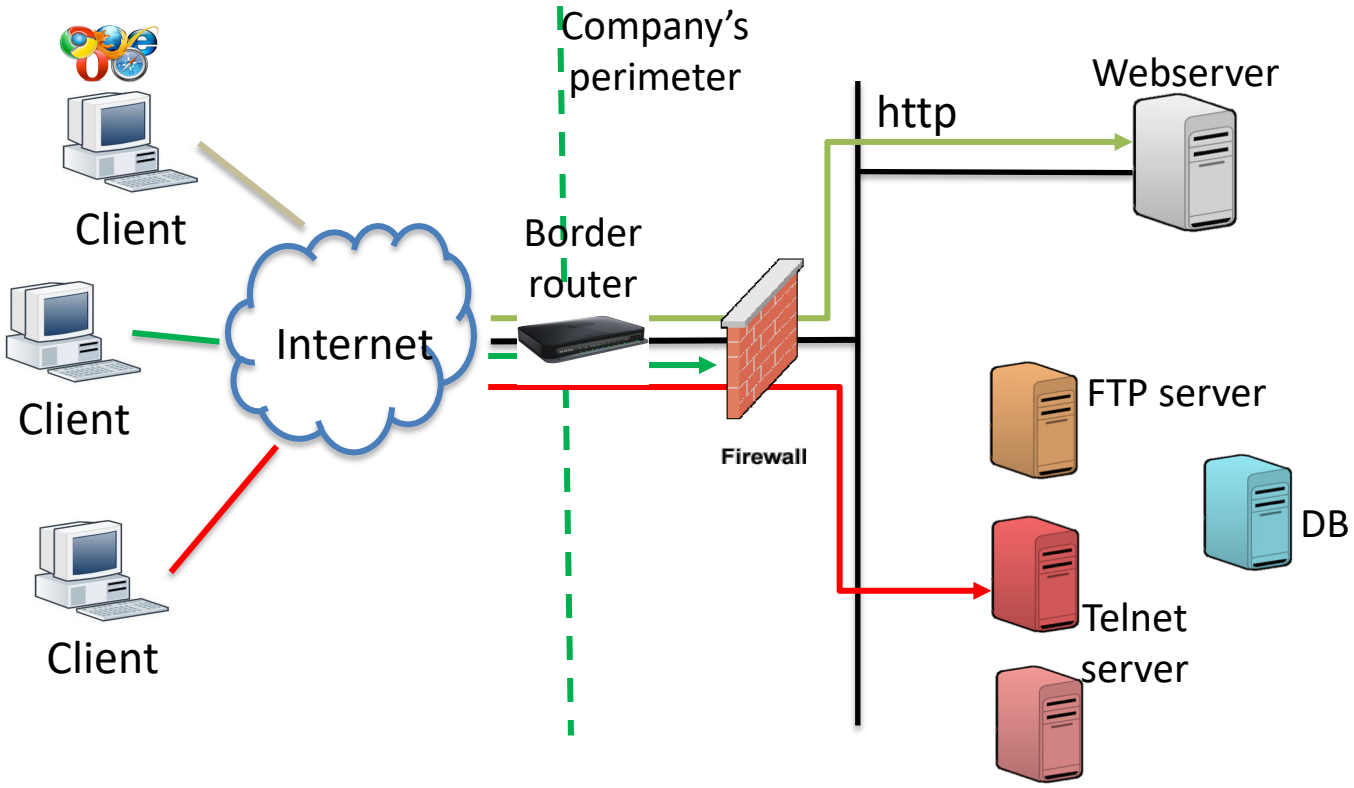
Default permit:

All what is not explicitly denied is allowed

Default Permit

- Blacklist policy → list what is blocked
- Rules to remove/reduce services are specified when a problem is discovered
- Users have more freedom on what they can do
- Suitable for small organizations or home systems

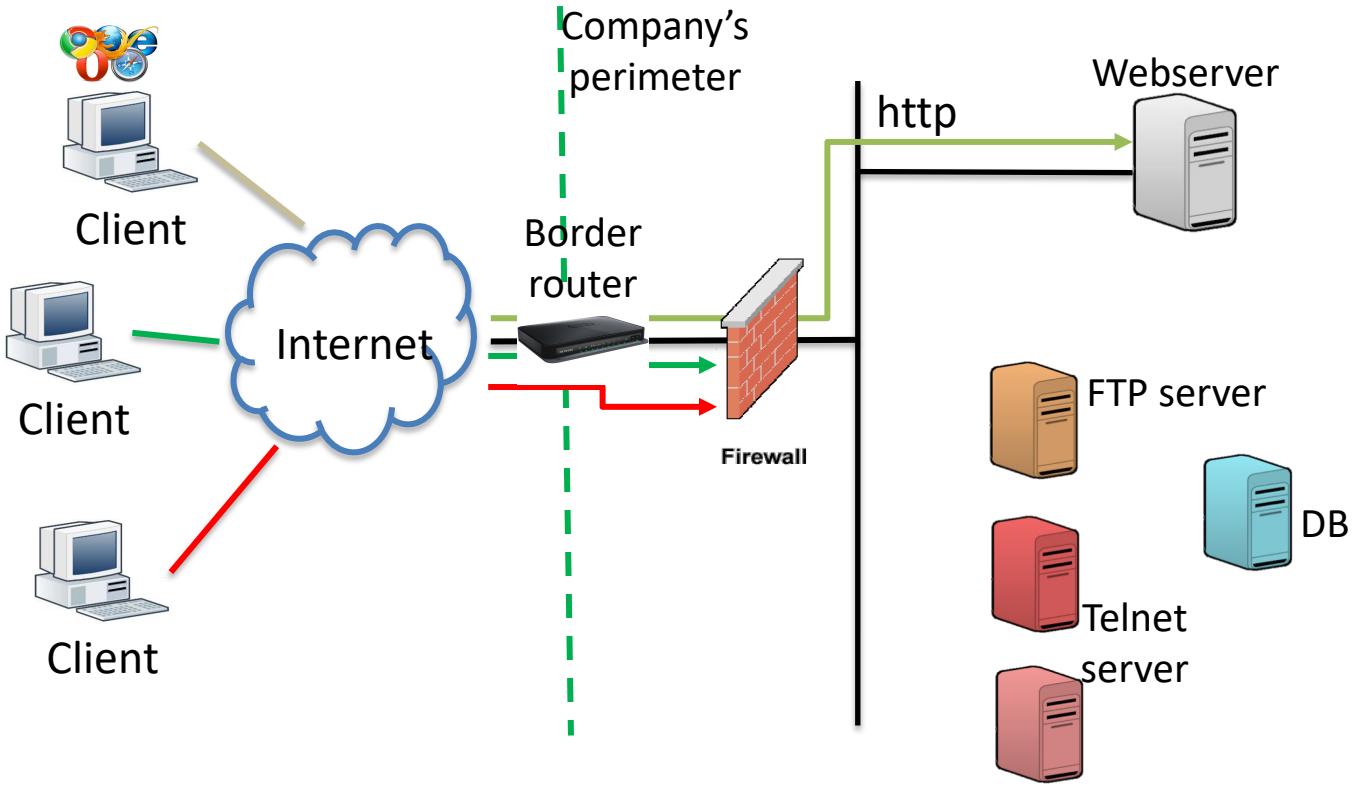
- Example permit policy
  - Deny incoming ftp traffic
  - Allow all



Default Permit

- Blacklist policy → list what is blocked
- Rules to remove/reduce services are specified when a problem is discovered
- Users have more freedom on what they can do
- Suitable for open organizations like universities or home systems

- Example permit policy
  - Deny incoming ftp traffic
  - Deny incoming telnet traffic
  - Allow all



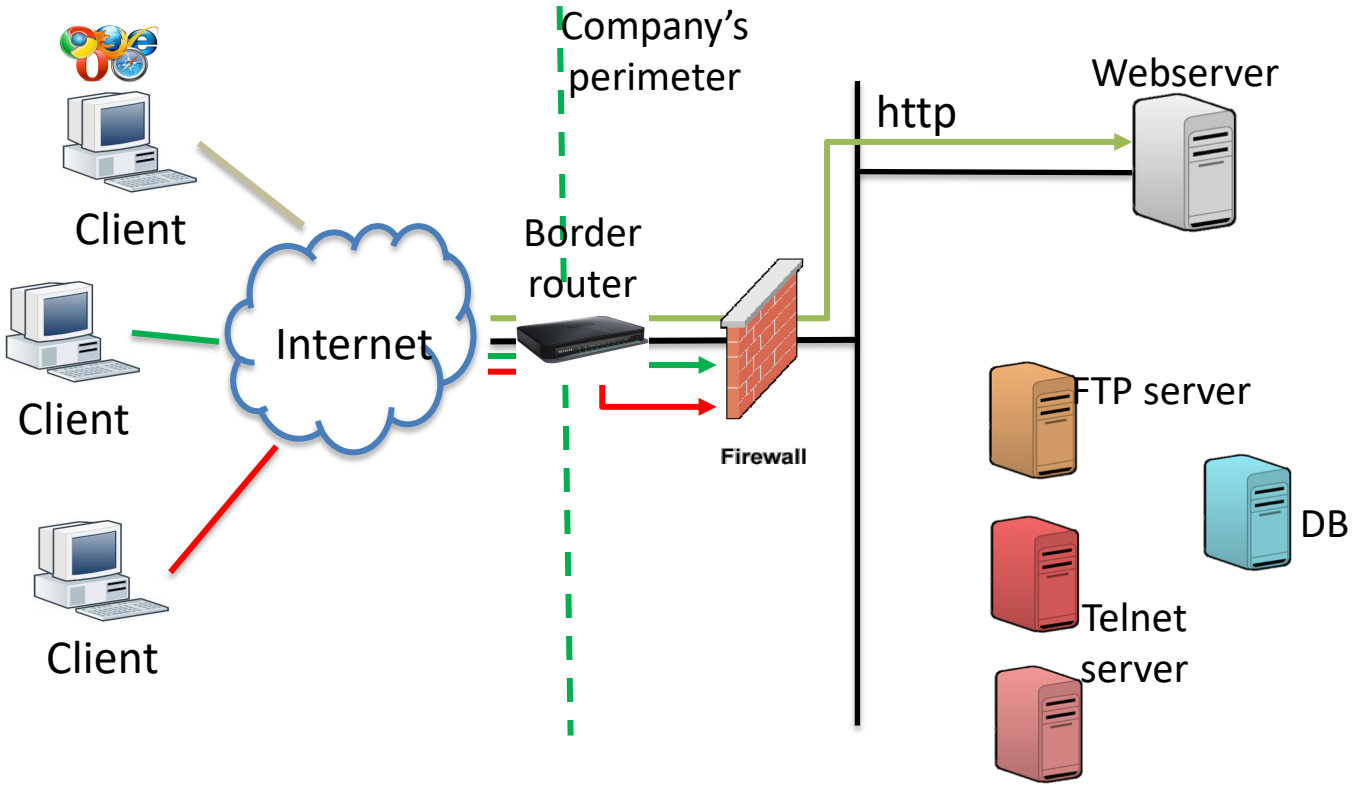
UNIVERSITY OF TRENTO

Default Deny

- Whitelist policy → list what is allowed
- Rules to allow a service are added after a careful analysis
- More visible to users (users are restricted at what they can do)
- Preferred default policy for business and governmental organizations

Order is important

- Example deny policy
  - Allow incoming http
  - Deny all



Firewall Types

- Static packet filtering
- Stateful packet filtering
- Proxies
  - Application-level gateways
  - Circuit-level gateways

Static Packet Filtering

- Applies a set of rules to each *incoming IP packet* to decide whether it should be forwarded or discarded.
- *Header information* is used for filtering (e.g., protocol number, source and destination IP, source and destination port numbers, etc.)
- *Stateless*: each IP packet is examined isolated from what has happened in the past.
- Often *implemented* by a router
- Simple and fast → low demand on resources

Access lists

- Defined by CISCO format
  - Standard ACLs

    access-list $number $action $src [wild card]
    - Number → identifies rule
    - Action → accept/deny
    - Src → source ip
    - Wild card → inverse of subnet mask → says which part of the IP should be checked for and which ignored
      - e.g. 192.168.3.1 [0.0.255.255] → "0.0.3.1" is the subnet of interest
  - Extended ACLs

    access-list $number $action $type $src [wild card] $opt $dest [wild card] [log]
    - Type → IP, tcp, udp, …
    - Opt → ports for TCP/UDP, type/code for ICMP, ..
    - Log → write in log when event is triggered
- Can assign values to variables
  - e.g. internal_net:=192.168.1.0/24

UNIVERSITY OF TRENTO

Packet Filtering

Do we actually need this?

- Yes, if default allow
- No, if default deny

Notice that this is last in the list

- First rule that matches is used

Example of (explicit) policies:

1. deny all incoming tcp connections to SSH;
2. allow outgoing TCP connections to SSH

action	src	port	dest	dport	flags	comment
allow	192.168.2.0/24	*	*	22	S	**Our outgoing traffic to remote ssh servers**
allow	*	22	192.168.2.0/24	*	SACK	**Their SYN ACK**
allow	*	22	192.168.2.0/24	*	ACK	**Rest of communication**

action	src	port	dest	dport	flags	comment
deny	*	*	192.168.2.0/24	22	S	**We do not allow remote connections to local SSH servers**

UNIVERSITY OF TRENTO

Packet Filtering

Do we actually need this?

- Yes, if default allow
- No, if default deny

Notice that this is last in the list

- First rule that matches is used

Example of (explicit) policies:

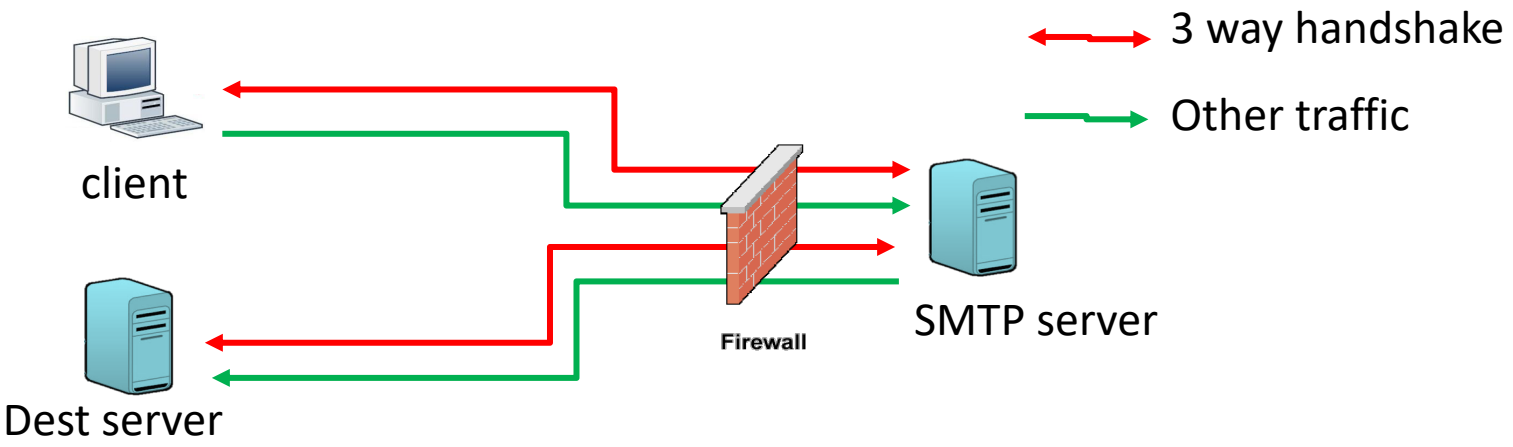
1. deny all incoming tcp connections to SSH;
2. allow outgoing TCP connections to SSH

action	src	port	dest	dport	flags	comment
allow	192.168.2.0/24	*	*	22	S or ACK	**Our outgoing traffic to remote ssh servers**
allow	*	22	192.168.2.0/24	*	SACK	**Their SYN ACK**
allow	*	22	192.168.2.0/24	*	ACK	**Rest of communication**

action	src	port	dest	dport	flags	comment
deny	*	*	192.168.2.0/24	22	S	**We do not allow remote connections to local SSH servers**

Note of caution

- Some protocols are easy to implement
  - Clear distinction between client and server
  - Other protocols are not as straightforward
- e.g. want to restrict SMTP operations
  - SMTP server acts both as a server (receives mail) and as a client (forwards mail to next server)
  - Firewall rules must match both cases



UNIVERSITY OF TRENTO

Exercise: SMTP rules

- Explicitly allow incoming SMTP traffic from 10.1.1.1 to SMTP-srv
- Allow all outgoing SMTP traffic

action	src	port	dest	dport	flags	comment
allow	10.1.1.1	*	SMTP-srv	25	S xor A	**allow everything from trusted client**
allow	SMTP-srv	25	10.1.1.1	*	S ACK	**allow server answer**
allow	SMTP-srv	25	10.1.1.1	*	ACK	**Allow rest of communication**
allow	SMTP-srv	*	*	25	S xor A	**Allow initiation of connection to remote SMTP**
allow	*	25	SMTP-srv	*	SA	
allow	*	25	SMTP-srv	*	A	
deny	*	*	*	*	*	

Packet Filtering: Pros and cons

- Pros
  - Transparent. It does not change the traffic flow or characteristics – either passes it through or doesn't
  - Simple
    - Easy to implement rules to prevent IP spoofing
      - e.g. no outgoing traffic from non-private IP address space
      - Control and log attempts to remotely connect to private services
  - Cheap
- Cons
  - It does not prevent application-specific attacks
  - Unsophisticated (protects against simple attacks)
  - Calibrating rule set may be tricky
  - Limited logging

Stateful Packet Filtering

- Called *Stateful Inspection* or *Dynamic Packet Filtering*
- Maintains a history of *previously seen packets* to make better decisions about current and future packets
  - Connection state maintained in a connection table
- Define rules to open state
- It's possible to use existent state to control future packets
  - e.g. explicit rule for TCP SYN in LISTEN state
    - "NEW" connection in IPTABLES
    - Subsequent packets can be filtered using the connection table
      - E.g. allow any packet for an ESTABLISHED connection

Pseudo-states

- Stateful firewalls allow user to define states over stateless protocols
  - e.g. UDP traffic is stateless → use <sip,sport,dip,dport> to correlate traffic
- For these protocols there is no termination sequence
  - e.g. TCP's FIN 4-way handshake
  - Typically set a time-out wherein pseudo-state is defined
- Traffic of stateless protocols depend on application, not on protocol itself
  - May be hard to manage, application-specific

Stateful firewall rule example

- Possible states (iptables with conntrack)
  - NEW → packet trying to open a not-yet existent connection
  - ESTABLISHED → incoming packet is relative to a connection already initiated
  - RELATED → packets that are stating a NEW connection but related existing one (needed by some applications – e.g. FTP)
  - INVALID → none of the above → e.g. incoming packet with ACK but not belonging to ESTABLISHED connection → can you filter this with static filtering?
- Say you want to prevent ACK scans
  - Stateful rule:

```
iptables –A INPUT –i eth0 –m state –state INVALID –
j DROP
```

Stateful firewall rule example

- Possible states (iptables with conntrack)
  - NEW → packet trying to open a not-yet existent connection
  - ESTABLISHED → incoming packet is relative to a connection already initiated
  - RELATED → packets that are stating a NEW connection but related existing one (needed by some applications – e.g. FTP)
  - INVALID → none of the above → e.g. incoming packet with ACK but not belonging to ESTABLISHED connection → can you filter this with static filtering?
- Say you want to prevent ACK scans
  - Stateful rule:

```
iptables –A INPUT –i eth0 –m state –state INVALID –j DROP
```

  - Static rule → will this be a good rule?

```
iptables –A INPUT –i eth0 -p tcp --tcp-flags ACK – j DROP
```

UNIVERSITY OF TRENTO

Another example

- Example rule: allow all incoming traffic related to an existing connection

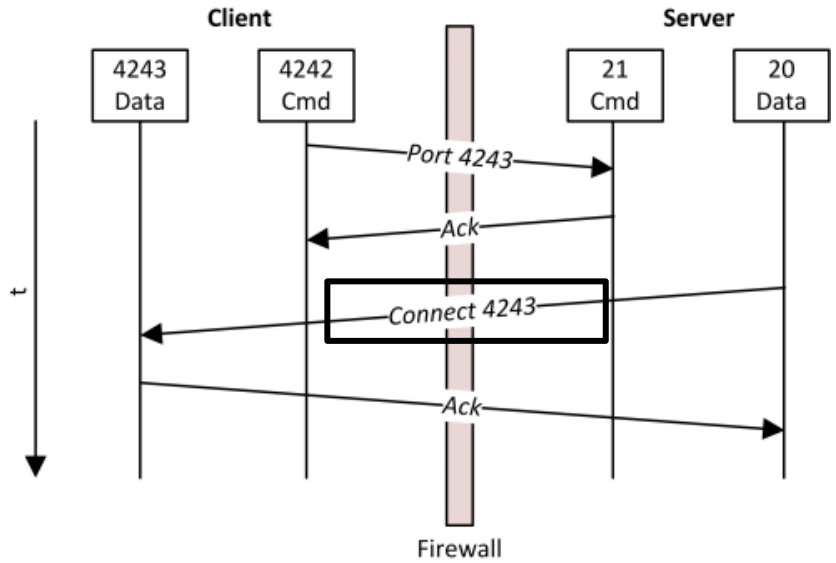
```
iptables -A INPUT -i eth0 -m state -state
   ESTABLISHED,RELATED  -j ACCEPT
```

- Mixed rules also possible:

```
iptables -A INPUT -i ! eth1 -j ACCEPT
iptables -A INPUT -m state –state
   ESTABLISHED,RELATED -j ACCEPT
iptables -P INPUT DROP
```

Application firewalls

- Statefulness consider also application layer
  - "Deep packet inspection"
  - Can keep track of some and deny others
  - e.g. FTP PORT command



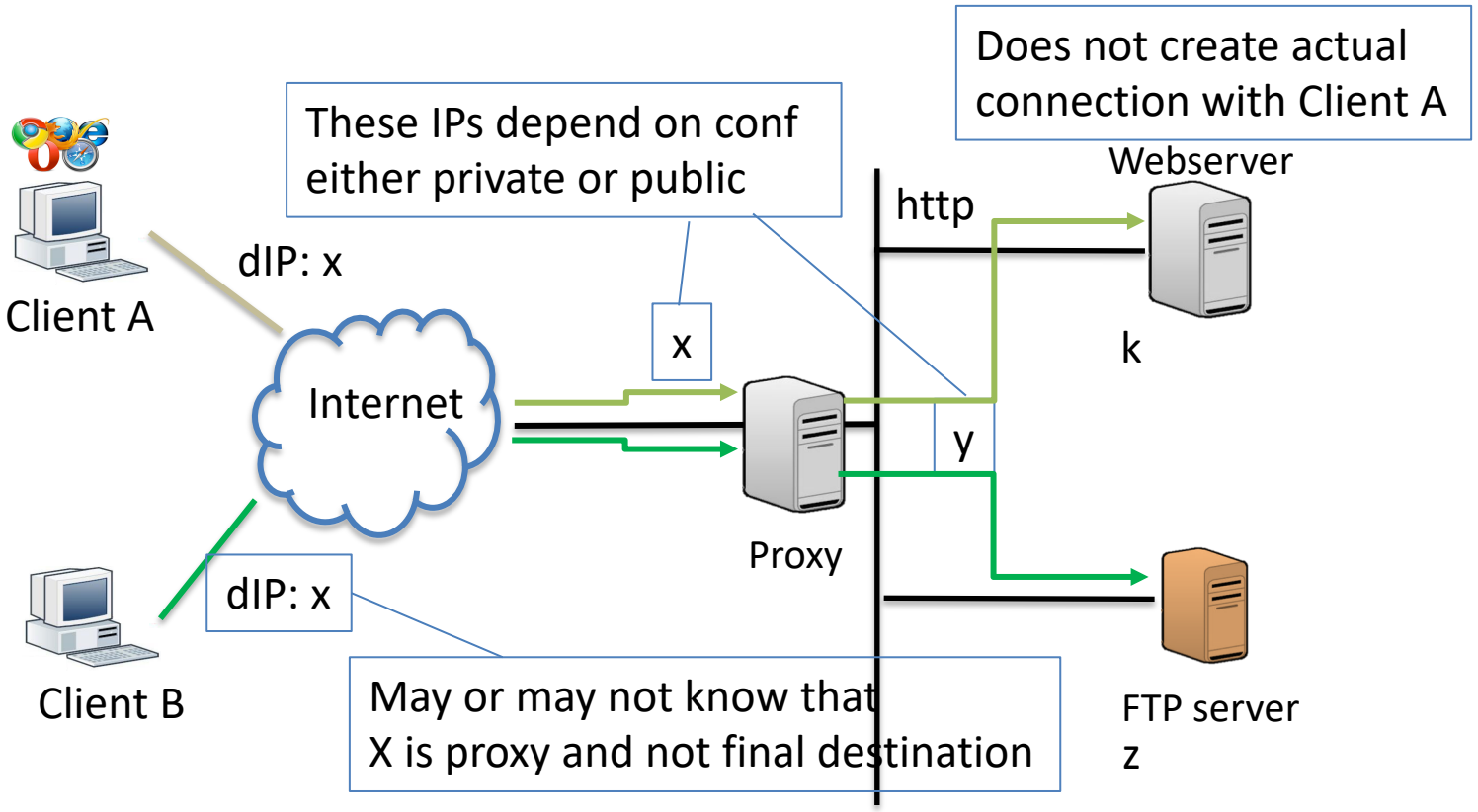
- FTP commands are passed to port 21
- In "Active mode" the server opens a connection with the client, and chooses dport
  - this happens with PORT command
- Application firewall can detect PORT command and act on packet
  - Simple stateless firewall can not manage this

Stateful and app firewalls: pros and cons

- Pros
  - Allow user to express more powerful rules
  - Policy definition is much simpler than with static packet filtering
  - Very diffused in all modern firewalls
- Cons
  - Severe impact on firewall performance
  - Deep packet inspection significantly slows down packet check
  - Application support may be very complicated
    - Typically provided as “modules”

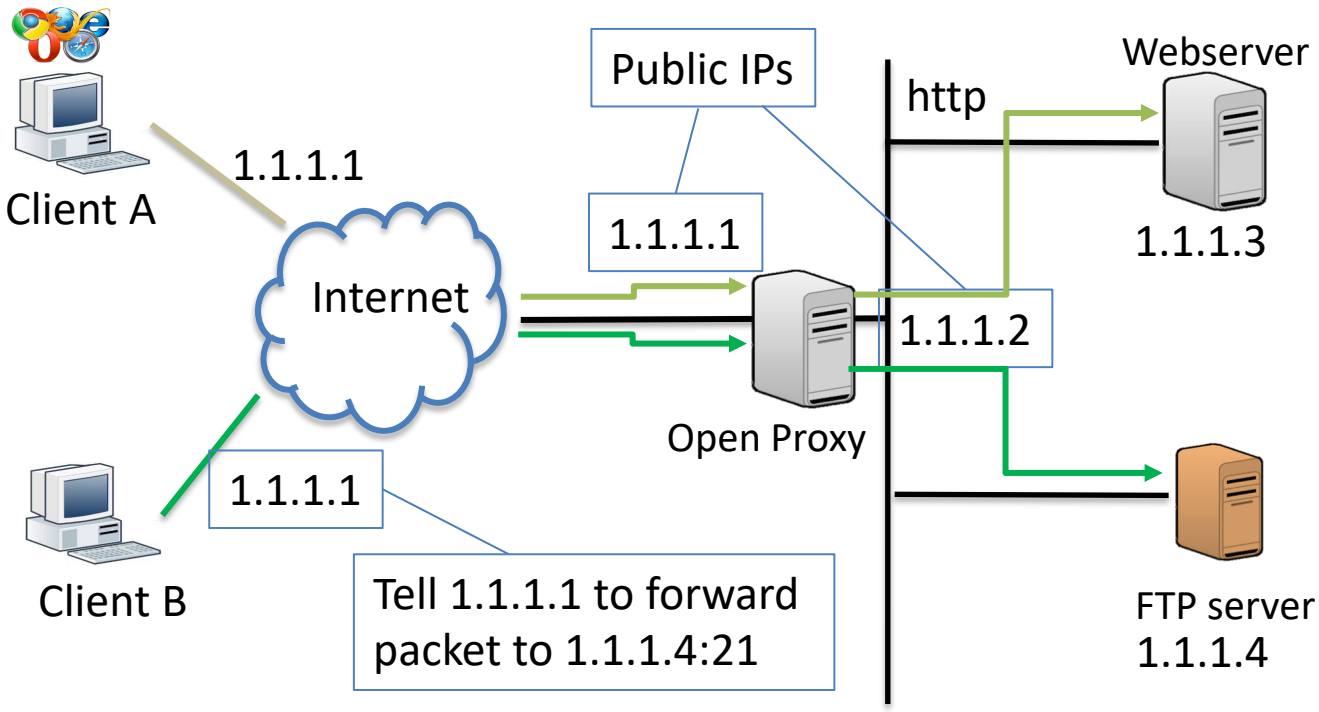
Proxy

- A network component that mediates network communications
- Untangles the otherwise direct communication between client and server
- Proxy acts both as a server (that receives remote connection) and as a client (that forwards the connection to its real destination).



Open proxy

- Proxy connects any client on the internet to any server on the internet
- Clients knows real destination of packet
- Server can not normally know by whom was the packet originated

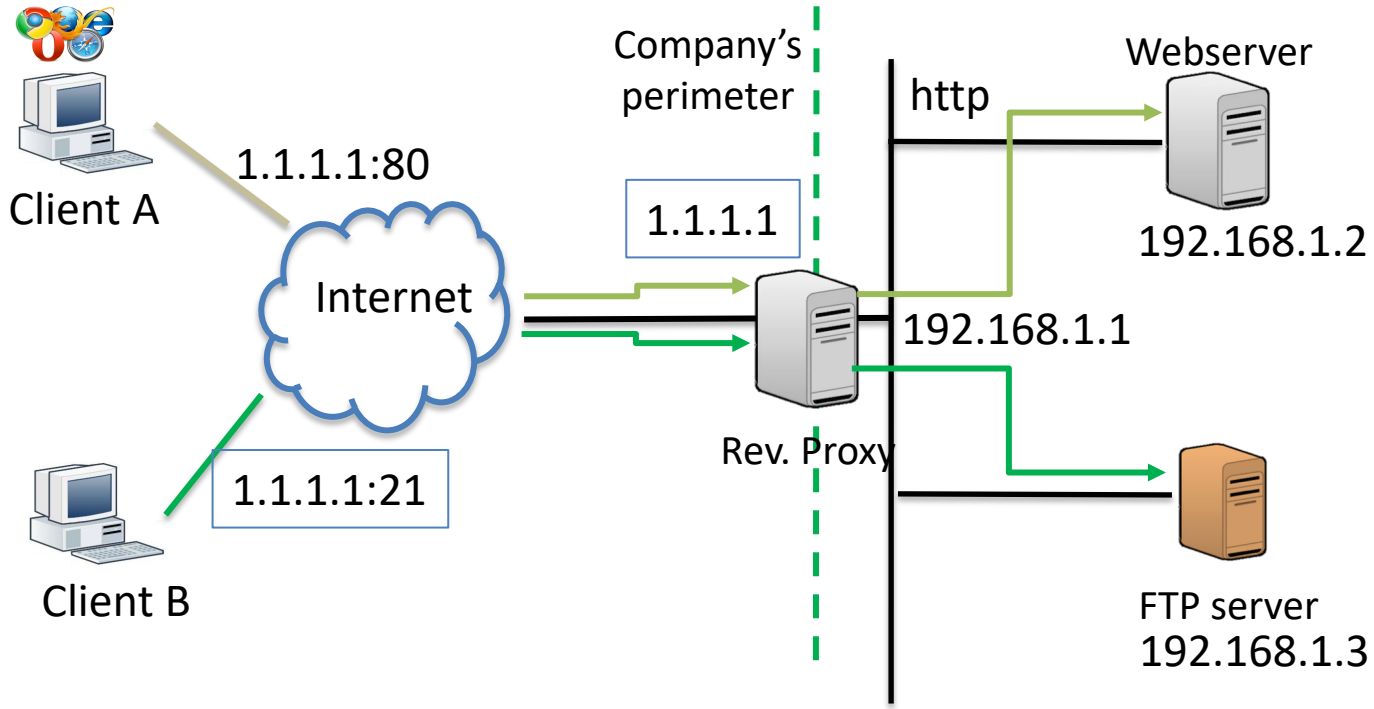


Open proxy - characteristics

- Enables the user to achieve some level of anonymity on the network
  - *Anonymous proxies*
  - Server should not be able to collect source IP
    - Some techniques exist to overcome this
    - Force the client to communicate its IP through third party services or plugins (e.g. flash)
- Trust issues → all trust is put on proxy service
  - This may or may not be sufficient depending on application
  - OK to bypass organisation's blacklist (e.g. block facebook.com)
  - Probably not trustworthy for more sensible Internet traffic
    - Confidential exchange of information
  - May be used as a malware distribution server
    - Malicious proxy embeds malware in response packet

Reverse Proxy

- Mediates connection between Internet clients and servers on an internal network it protects
- Can embed firewalling capabilities; may sit on border router.
- Client talks directly to Proxy; Proxy forward to internal servers; neither internal servers or clients know real origin/destination of packet.



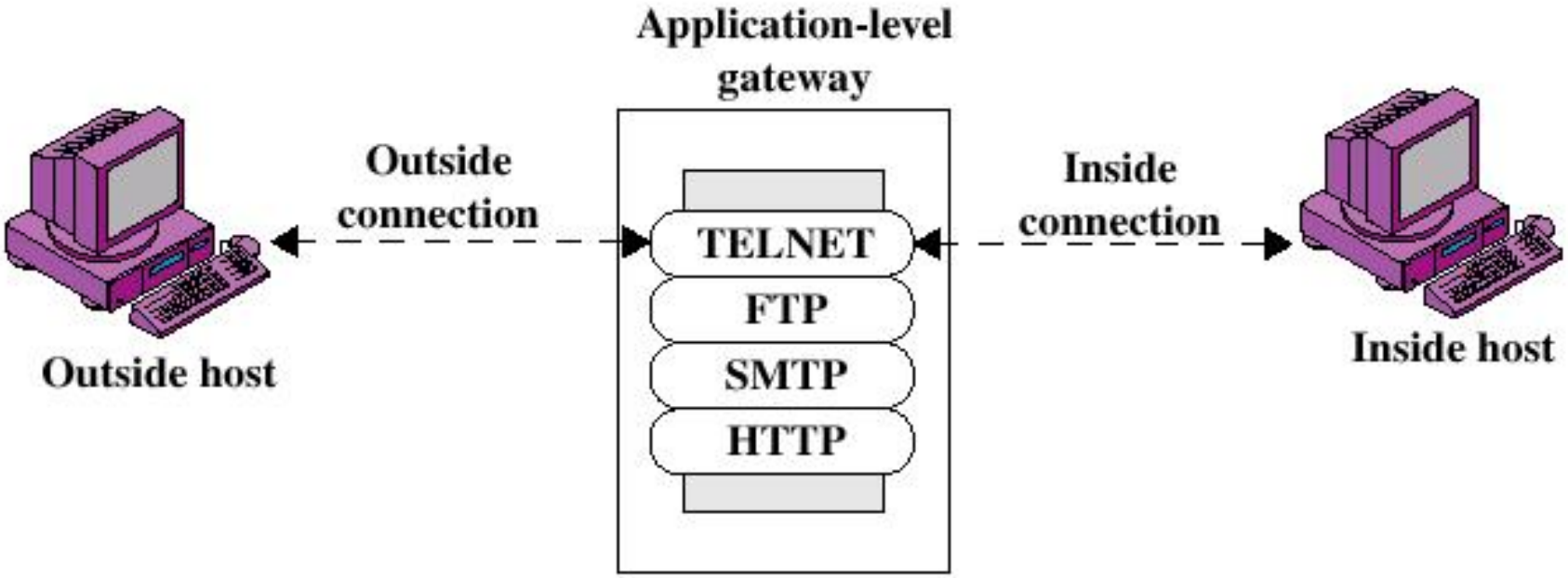
Reverse proxy - characteristics

- May hide properties of internal servers
  - IPs, non-custom service ports, versioning
    - If too aggressive may cause disservices
      - e.g. declares fake server version that breaks the protocol
- May be used for load balancing
  - Several internal replicas of a webserver
  - Proxy automatically balances the load by forwarding client's connection to most appropriate internal server
    - e.g. least busy server gets the connection
  - May be used to cache server's content → answer directly to requests for which a cache entry exists

UNIVERSITY OF TRENTO

Application Level Proxy

- Also called application proxy
- Acts as a relay of application-level traffic
- All connections are mediated by the GW

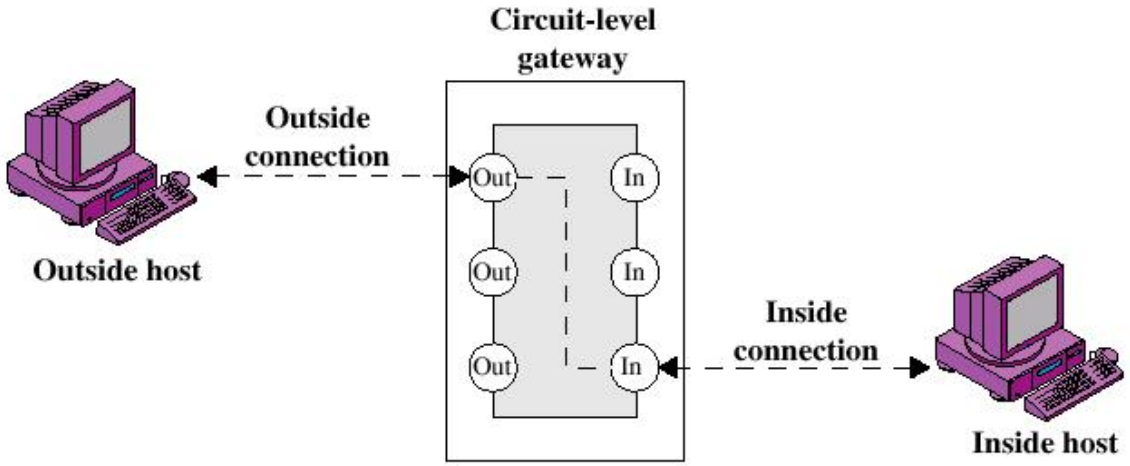


Application Gateway

- Pros: by not permitting application traffic directly to internal hosts
  - *Information hiding*: names of internal systems are not known to outside systems
  - Can limit capabilities within an application
  - *Robust authentication and logging*: application traffic can be pre-authenticated before reaching host and can be logged
  - *Cost effective*: third-party software and hardware for authentication and logging only on gateway
  - *Less-complex filtering rules for packet filtering routers*; easier stateful firewall implementations
  - More secure
- Cons
  - Keeping up with new applications
  - May need to modify application client/protocols
    - Custom implementation may be expensive

Circuit-level Gateway

- Also called circuit-level proxy
- Usual, when there is a trust to internal users
- No firewalling capabilities → simply crosses client connection to inside host
  - The gateway typically relays TCP segments from one connection to the other without examining the content
  - Operates at L4 on OSI scale



Firewall Basing

- Software module in router or LAN switch
- Bastion host. Stand-alone machine running common OS (Unix, Windows)
- Host-based firewall
- Personal firewall

UNIVERSITY OF TRENTO

Bastion Host

- A system identified by the firewall administrator as a critical strong point in the network´s security
- The bastion host serves as a platform for an application-level or circuit-level gateway
- **Characteristics:**
  - Executes on a secure version of the OS (hardened system)
  - Only essential services
  - May require additional user authentication before accessing proxy services; each proxy service may require also its own
  - Each proxy maintains detailed audit information
  - Each proxy is independent
  - Each proxy runs as a non-privileged separate user

Firewall/Bastion Administration

- Access to management console
  - By dedicated clients using encryption
  - Via SSH and https
  - Possibly using also user authentication
- Strategies of disaster recovery
  - Switches capable of Balancing/failover
- Logging
  - Use of a remote syslog server
    - Centralization of all logs
- Security incidents
  - They have different severity levels
  - The policy determines which ones are significant
    - Keep logs for legal analysis about the attacks
    - Synchronization with a time server → important to know which came first

Host-based Firewall

- Software module used to secure an individual host
- Available in many operating systems
- Common location for such firewalls is a server
- **Advantages**
  - Filtering rules can be tailored to the host environment (specific rules for the servers)
  - Protection is provided independent of topology. Thus both internal and external attacks must pass through the firewall
  - In conjunction with stand-alone firewalls, the host-based firewall provides an additional layer of protection

Personal Firewall

- Personal firewall controls the traffic between a personal computer or workstation on one side and the Internet or enterprise network on the other side
- Used in home environment and on corporate intranets
- Typically, software module on the personal computer
- Easy to configure
- Used to:
  - deny unauthorized remote access
  - detect and block worms and other malware

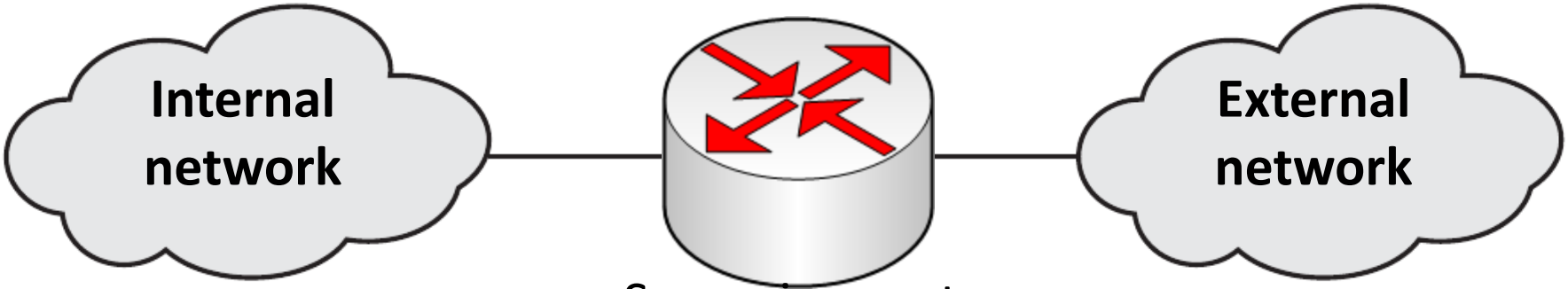
Firewall Topologies

- Host-resident firewall
- Screening router: packet filtering
- Single bastion inline
- Single bastion T, with DMZ
- Double bastion T

UNIVERSITY OF TRENTO

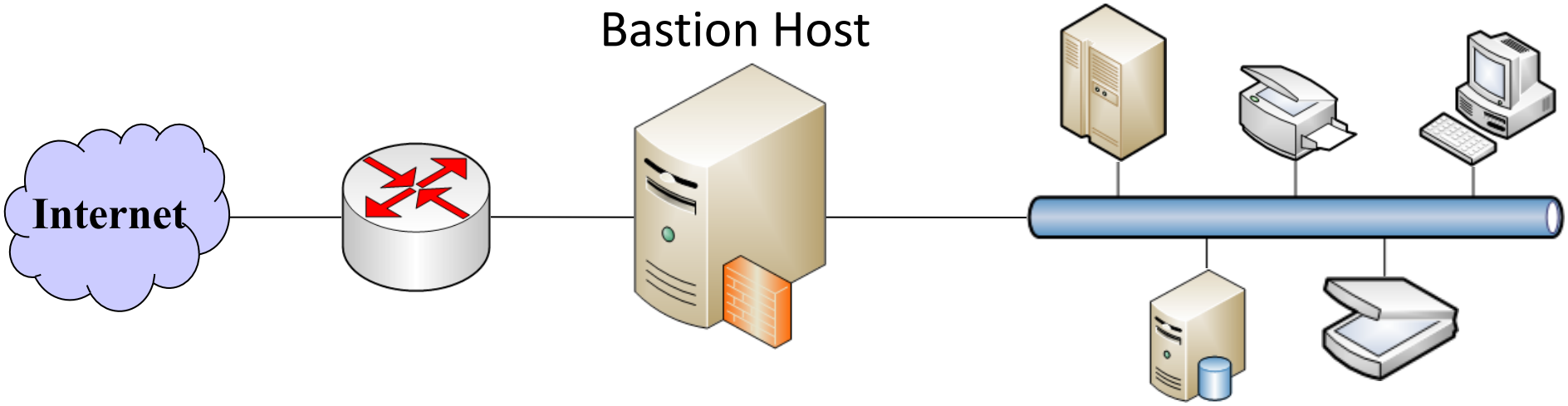
Firewall Topologies

- Host-resident firewall
  - personal firewall software and firewall software on servers
- Screening router
  - single router between internal and external networks with stateless or full packet filtering
  - typical for small office/home office (SOHO) applications



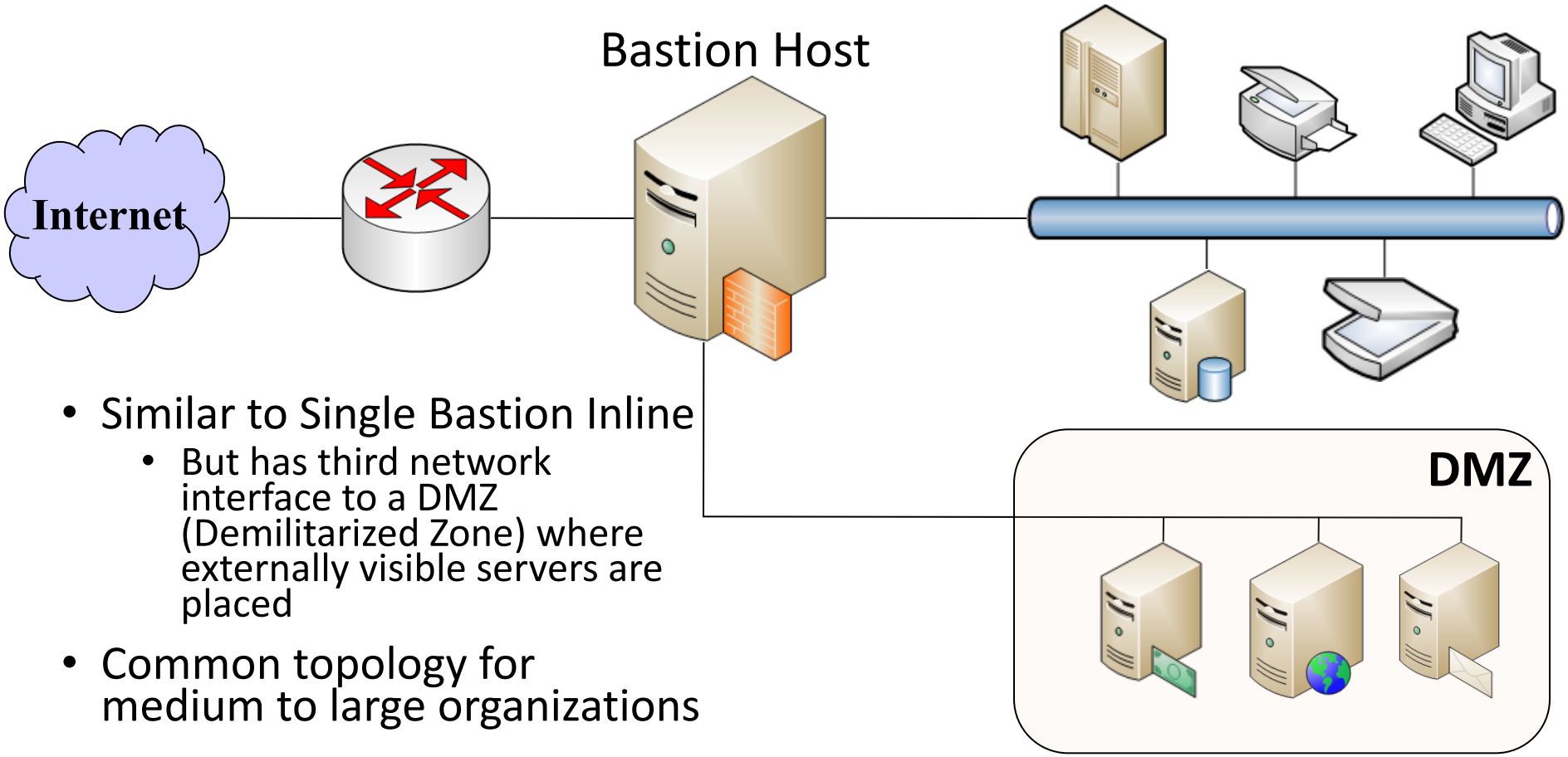
UNIVERSITY OF TRENTO

Single Bastion Inline



- Configuration for the packet-filtering router:
  - Only packets from and to the bastion host are allowed to pass through the router
- The bastion host performs authentication and proxy functions
- This configuration implements both packet-level and application-level filtering (allowing for flexibility in defining security policy)
  - An intruder must generally penetrate two separate systems

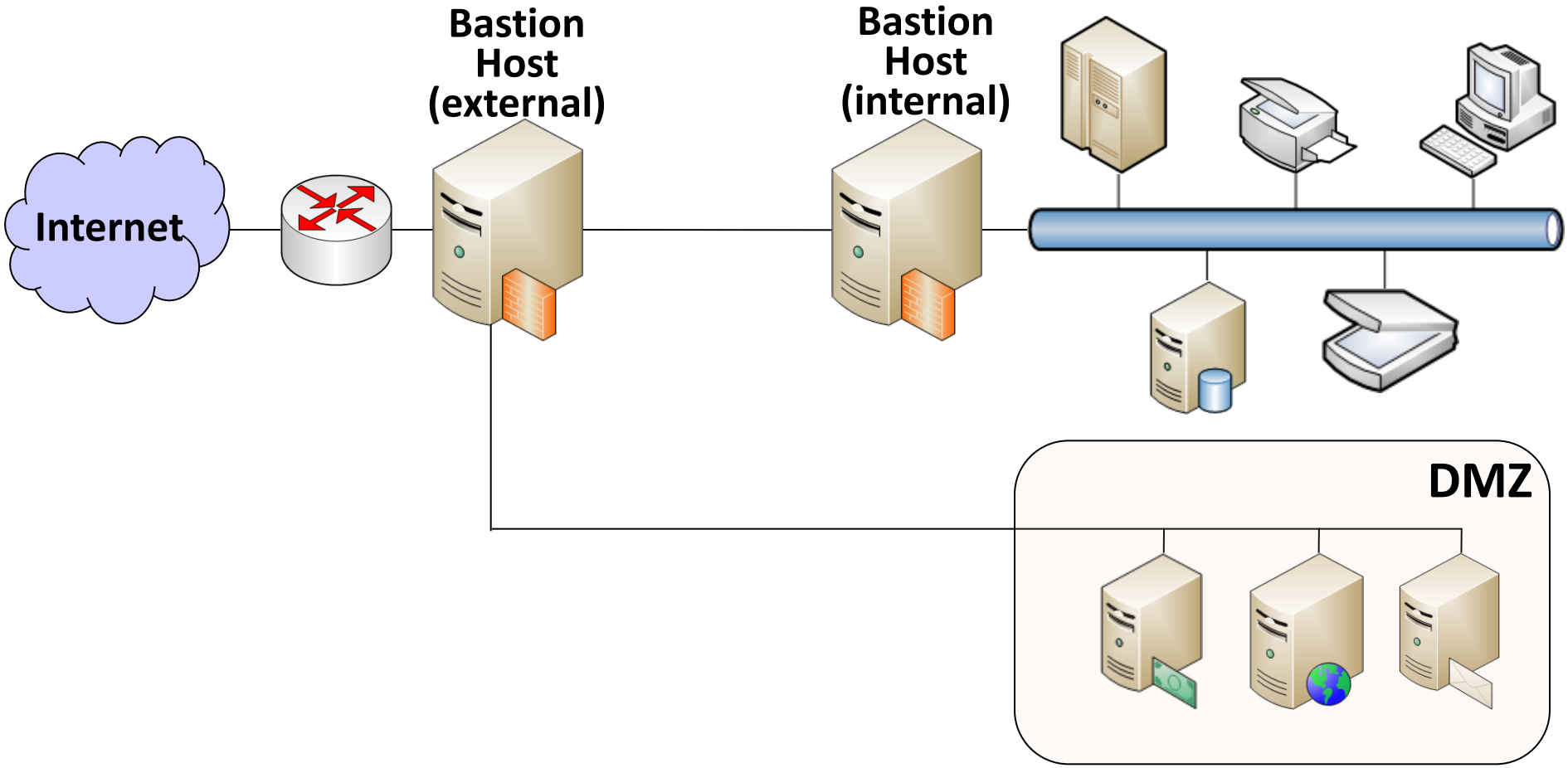
Single Bastion T



- Similar to Single Bastion Inline
  - But has third network interface to a DMZ (Demilitarized Zone) where externally visible servers are placed
- Common topology for medium to large organizations

UNIVERSITY OF TRENTO

Double Bastion T

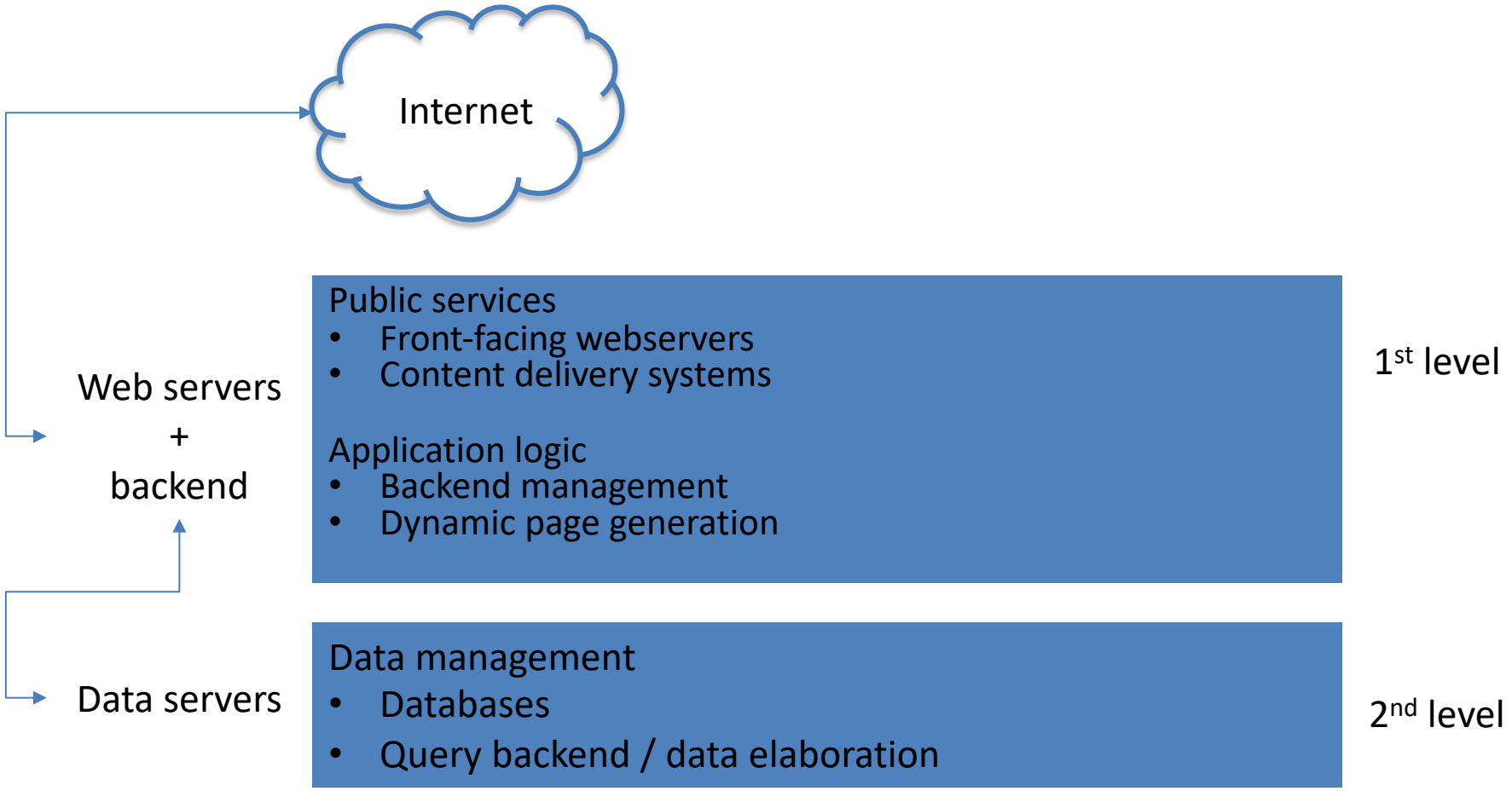


Advanced network topologies

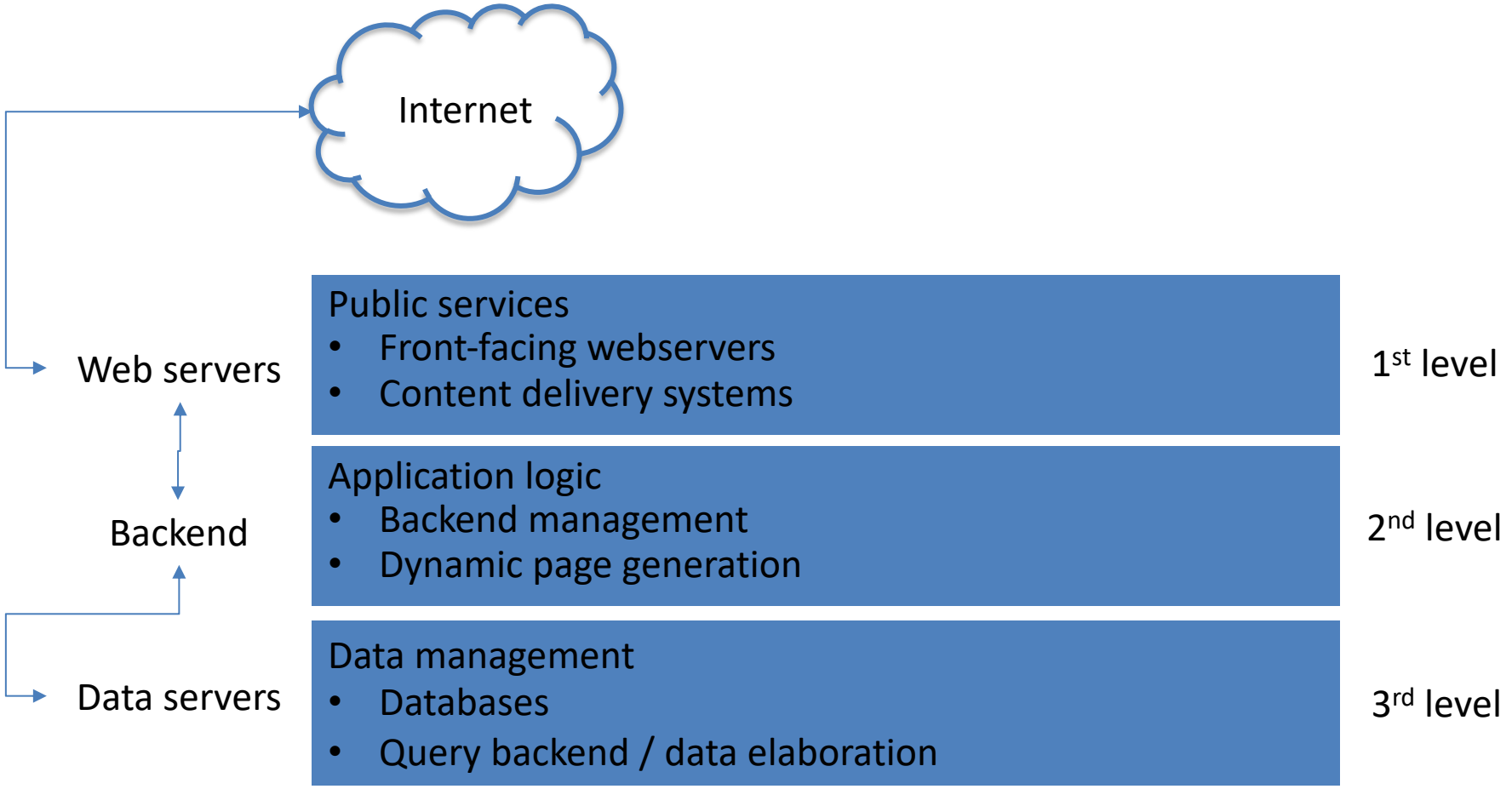
- Single/Double bastion topologies are adequate only when mapped to a significant *separation of networks*
- Good network separation allows for
  - Better management of firewall rules
  - Higher control on incoming traffic
  - Higher overall security
  - Lower load on single appliances

UNIVERSITY OF TRENTO

Typical multi-level network applications

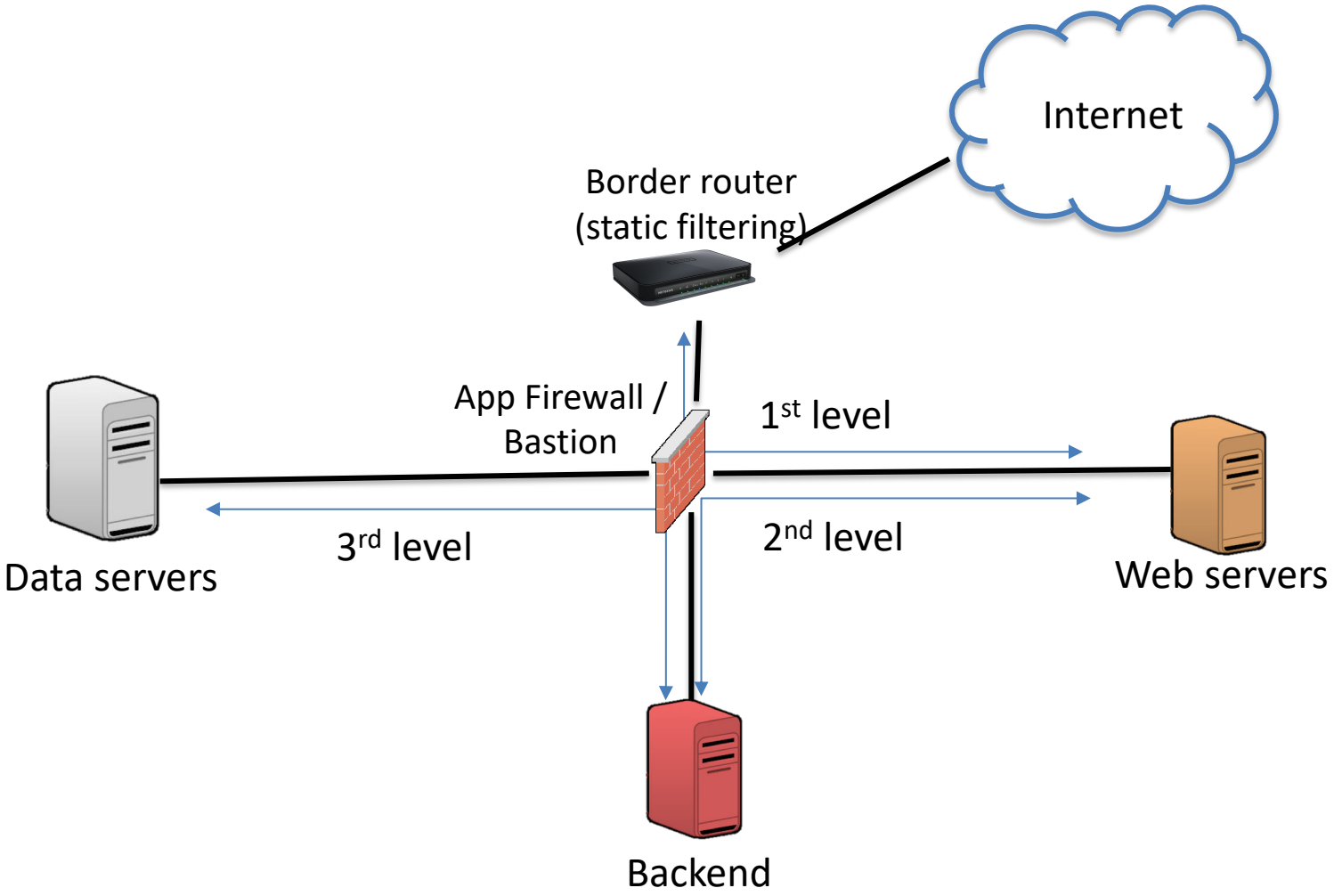


Separate network topology

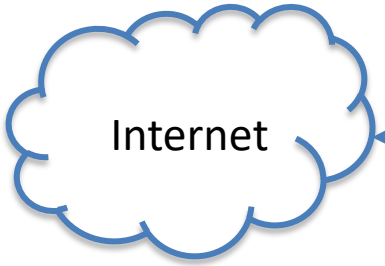


UNIVERSITY OF TRENTO

Separate network topology in practice – simple implementation



Divide et impera - Cascade firewalls

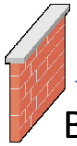


Border router (static filtering)

Web servers

Non-critical resources

Public services
- Front-facing webservers
- Content delivery systems



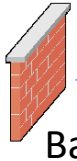
Bastion 1st level

- Outward packet filtering
  - e.g. implement client blacklist
- Light application control →Low on resources
  - Prevent DDoS, XSS.

Backend

Critical resources

Application logic
- Backend management
- Dynamic page generation



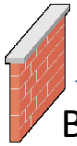
Bastion 2nd level

- Inward/Outward packet filtering
  - Allow management connections
- Strong application control
  - Deep packet inspection
  - Payload sanitisation

Data servers

Highly critical resources

Data management
- Databases
- Query backend / data elaboration



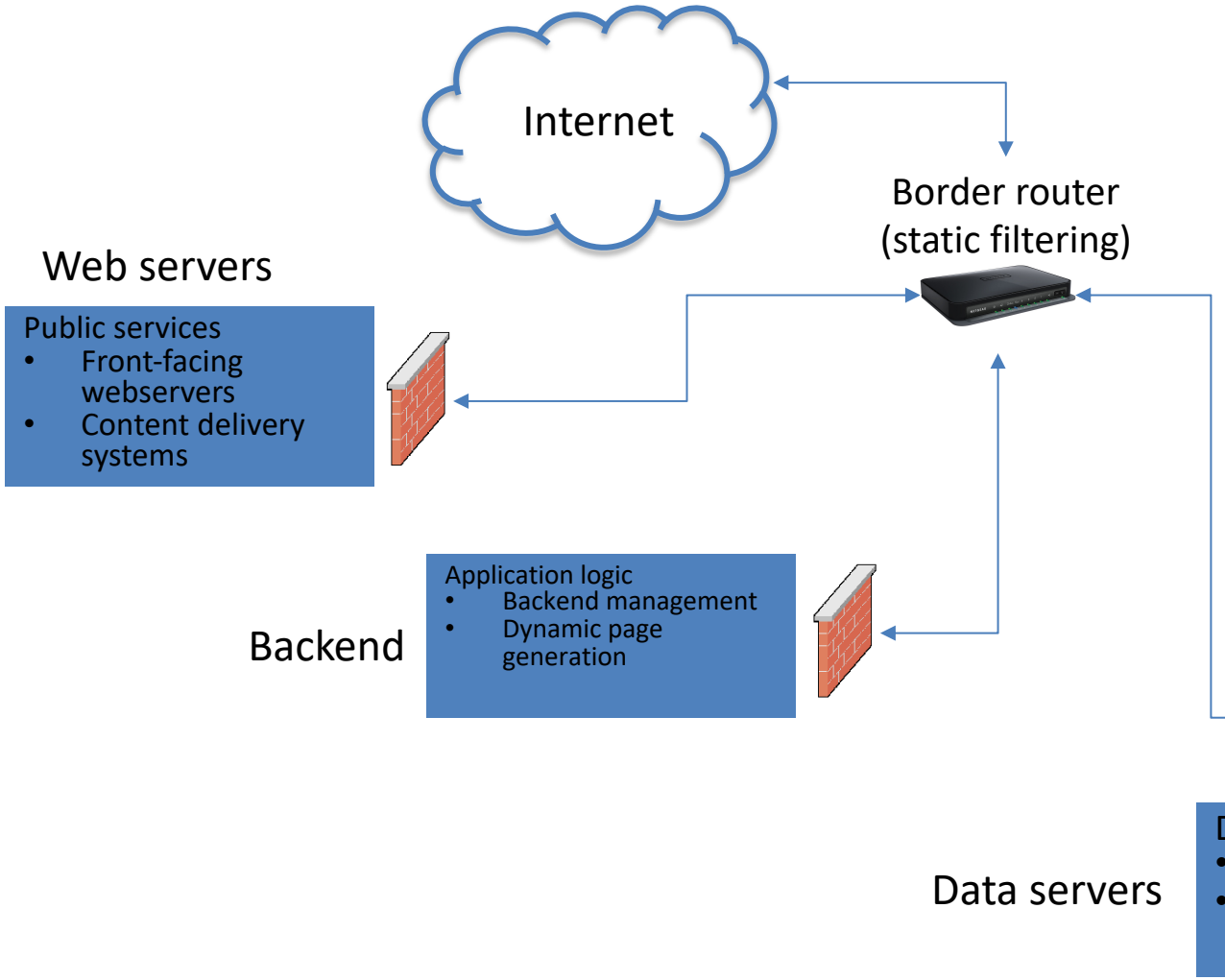
Bastion 3rd level

- Inward/Outward packet filtering
  - Allow management connections **from trusted network only**
- Strongest application control
  - e.g. Evaluate SQL queries before passing them to mySQL DB

Cascade firewalls - notes

- Inter-dependent firewall policies
- Each firewall must be configured considering functions needed at higher levels
  - E.g. firewall at level 1 must allow all packets eventually directed toward level 2 or 3
  - In complex networks this is unmanageable if network is not well configured
- Requires a good mixture of NAT/PAT policies, firewall configurations, and good separation of services
  - e.g. Hard to have effective NAT + firewalling for SSH services at both level 1 and level 3 → where should the packet go?
    - Remember incoming packet will always have address of outward-facing NAT interface toward port 22.
  - Each layer should ideally be in a different subnet
    - Firewall @ Layer 1: 192.168.1.0/24
    - Firewall @ Layer 2: 192.168.2.0/24, etc..
    - ✓ F1 Accept all traffic that needs to be forwarded to F2
- High design, management, maintenance costs
  - Introducing a new service at any level requires testing all configuration at lower levels

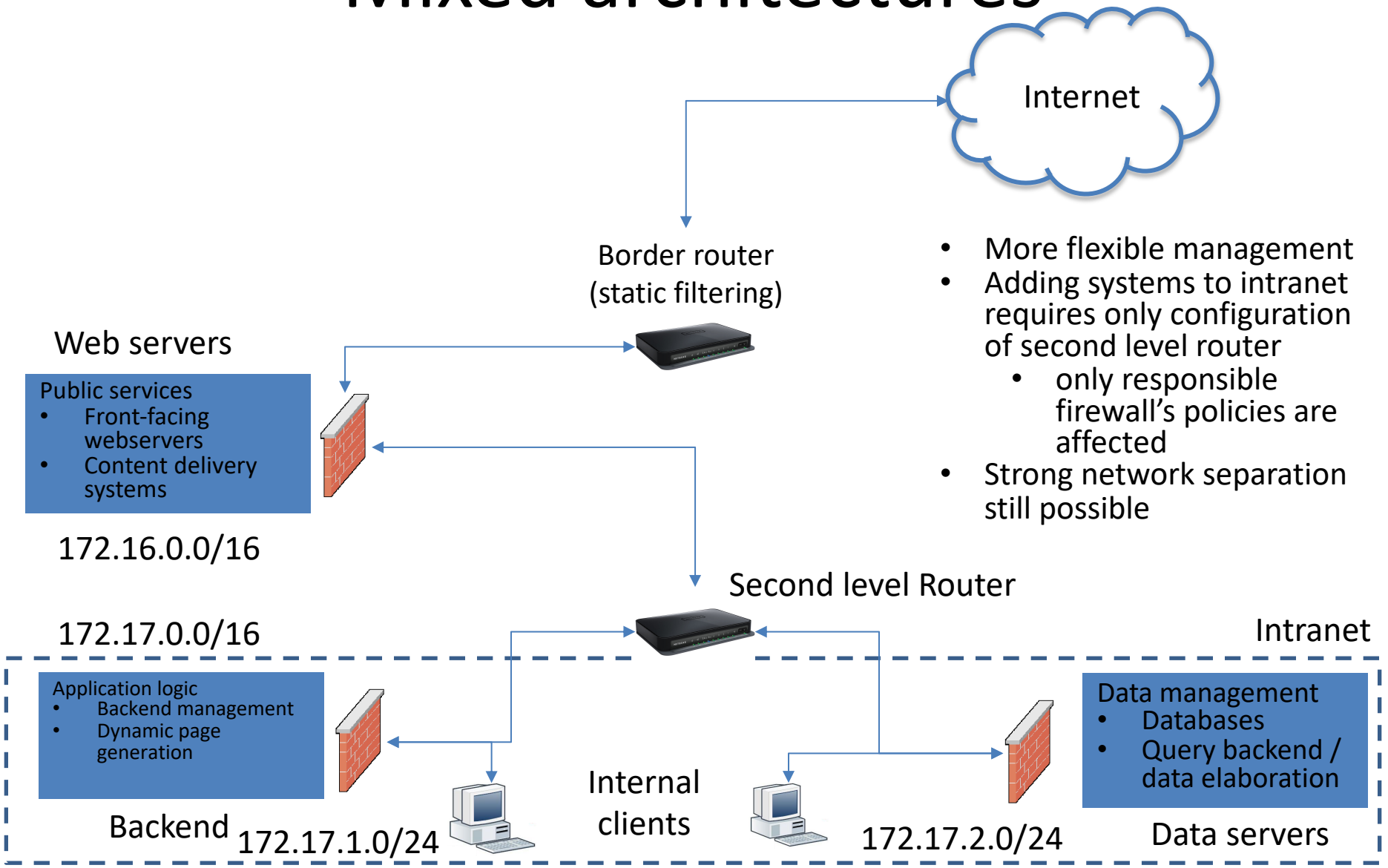
Divide et impera - Parallel firewalls



- Load distribution is on the border router
- One interface per firewall
- More complex management of internal connections
  - E.g. backend → web servers
  - all passes through router

UNIVERSITY OF TRENTO

Mixed architectures



- More flexible management
- Adding systems to intranet requires only configuration of second level router
  - only responsible firewall's policies are affected
- Strong network separation still possible