sicurezza delle reti
i firewall e l'esempio di netfiilter

vulnerabilità e minacce a livello rete, trasporto e applicativo

- rete/trasporto come veicolo per attacchi ai sistemi
  - l‘attacco può essere sferrato da molto lontano
  - elemento umano: email, web, ecc. l’utente permette l’entrata di virus, trojan, ecc.
- vulnerabilità dei protocolli
  - già visti all’inizio del corso: protocolli in chiaro e non autenticati, DoS, DDoS

firewalls

- sono apparecchiature che confinano (filtrano) selettivamente il traffico di rete
- network fw: layer3+4
  - stateful vs. stateless
- application fw: layer7
  - a.k.a deep packet inspection o applicative content inspection
- hardware vs. software
- personal fw vs. network fw
- possono avere molte altre funzionalità
  - nat, virtual private network, autenticazione utenti, ecc. spesso detti UTM

Unified Threat Management (UTM)

- evoluzione del concetto di firewall
- unisce in un unico dispositivo molte delle funzionalità di sicurezza relative alle reti
  - firewall
  - network intrusion detection/prevention
  - sicurezza delle email
    - antivirus, anti-spam
  - VPN termination
  - applicative content inspection
  - load balancing
- semplicità di gestione e riduzione dei costi
- vedremo molte delle funzionalità come se fossero apparati separati

soggetti, oggetti e diritti in un fw: un primo possibile punto di vista

- soggetto: un pacchetto/messaggio in ingresso
- oggetto: solo il firewall
- accesso: richiesta al firewall di essere instradato verso la destinazione
- diritti: per ciascun tipo di traffico in ingresso le destinazioni ammesse

- è un modo di vedere il fw molto concreto ma poco utile

il firewall come reference monitor

un firewall può essere visto come un reference monitor di rete

- soggetto: l'host sorgente che ha inviato il pacchetto/messaggio
- oggetto: l'host destinazione che riceverà il pacchetto/messaggio
- accesso: richiesta all'host destinazione di processare il pacchetto/messaggio inviato dall'host sorgente
  - caratterizzato dal valore di campi del pacchetto stesso
  - la semantica vera (cioè l'effetto sull'host destinazione) dell'accesso è data dal protocollo di rete e dal contesto in cui viene usato: il fw non ha bisogno di conoscerla
- diritti: categorie di traffico ammesso per la coppia di host

- permette di esprimere una policy come access matrix

stateless firewall

- regole di “packet filtering”
  - basate sulla quadrupla <saddr,sport,daddr,dport>
  - in pratica sono router configurati con delle access control list
    - detti anche “screening routers”
- non mantengono alcuno stato
- tipicamente quando si parla di firewall si fa riferimento a **firewall “stateful”**

stateful firewall

- si ricordano le “connessioni”
  - stato della connessione: new, established, ecc.
- ciascun pacchetto è assegnato ad una connessione
  - permettono regole in base allo stato della connessione
    - es. ammesso da *esterno* verso *interno* solo se connessione è established
  - verificano la correttezza del protocollo
    - es. syn ammesso solo per connessioni nuove
- possono modificare il traffico
  - es. per implementare schemi anti-SYNflood

connessioni

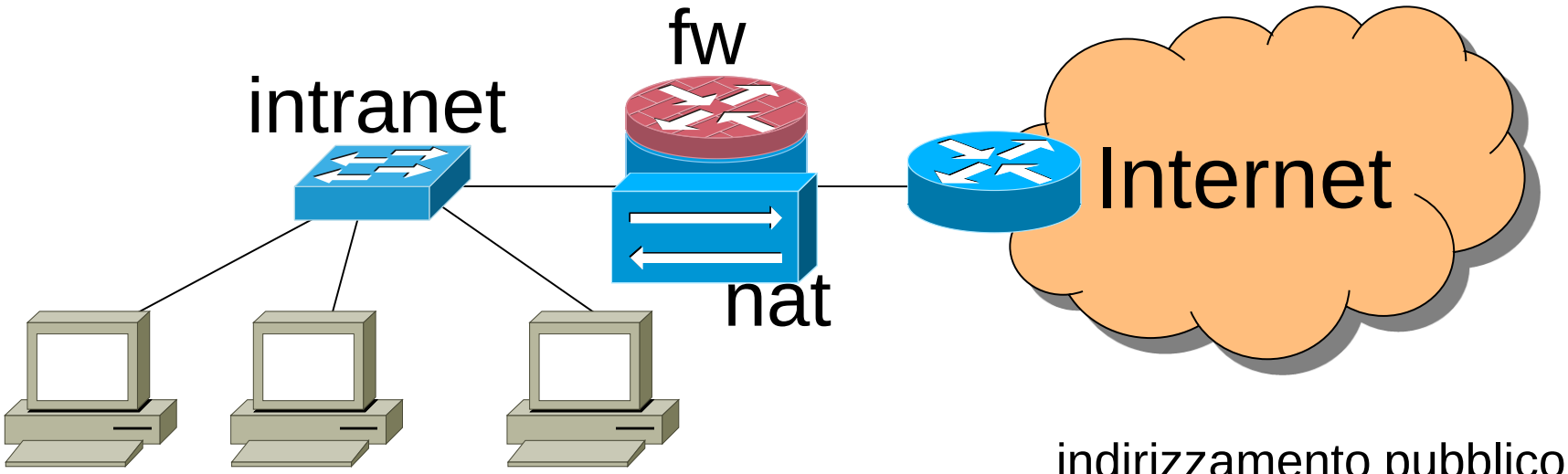
connessione: vari significati a seconda del contesto

- connessione per i protocolli, vedi standard:
  - TCP connesso, UDP non connesso
- connessione per il sistema operativo
  - il sistema operativo mantiene uno stato per i protocolli connessi
  - per il sistema operativo esiste una versione connessa di UDP

- **connessione per il firewall**
  - tipicamente identificata con la quadrupla <saddr,sport,daddr,dport> (o dalla coppia <saddr,daddr>)
  - **il primo pacchetto** con tale quadrupla **crea la connessione**
  - **gli altri pacchetti** con tale quadrupla sono **relativi alla connessione già creata**
  - es. per un firewall ping e le richieste DNS (su udp) creano “connessioni”

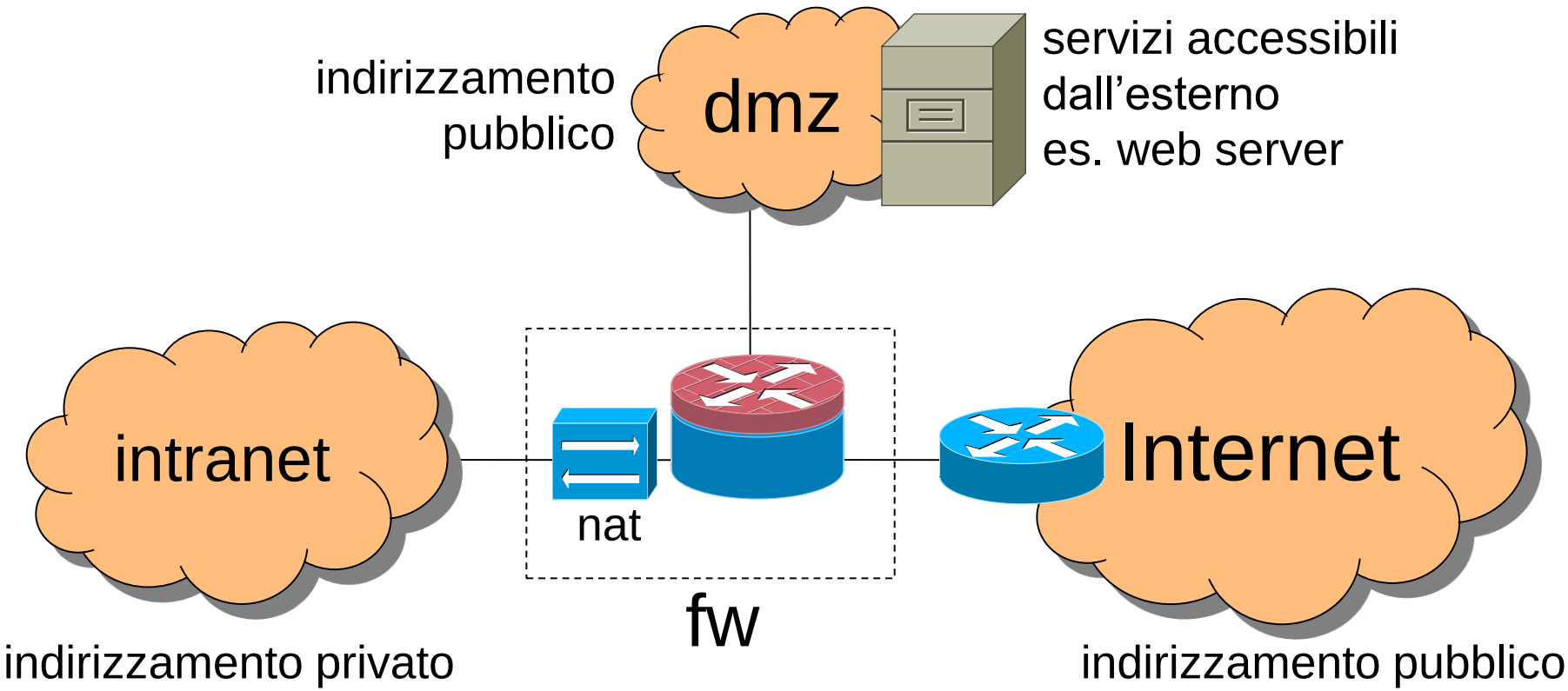
firewall, nat e intranet

- un semplice caso d’uso
  - isolare la intranet da Internet
  - in questa configurazione tipicamente il fw fa anche nat ma ciò non è strettamente necessario



dmz

- demiliterized zone (zona smilitarizzata) o perimeter network (rete perimetrale)
  - rete distinta sia da Internet che dalla intranet
  - gli host in tale rete sono in una classe di sicurezza distinta da quelli della intranet



dmz: esempio di policy

da \ a	host intranet	host dmz	host Internet
host intranet	all non si passa per il fw	richiesta dalla intranet si accede ai servizi della dmz	- dalla intranet non si accede a Internet
host dmz	risposta dalla dmz si risponde alle richieste della intranet	all non si passa per il fw	risposta dalla dmz si risponde alle richieste di Internet
host Internet	- da Internet non si accede alla intranet	richiesta da Internet si accede ai servizi della dmz	all non si passa per il fw

interpretazione per protocolli basati su tcp

- “richiesta”
  - un syn per una connessione NEW
- “risposta”
  - tutto ciò che è relativo ad una connessione ESTABLISHED (dopo il syn)

- facile da implementare in un firewall stateful

varianti

- intranet potrebbe dover accedere ad Internet
  - questo potrebbe rappresentare un grave problema di sicurezza perché Internet è una fonte di input non fidato
    - consigliato l'uso di application level gateway o applicative content inspection
- dmz potrebbe dover accedere a…
  - dns, per arricchire i log
    - fonte non fidata
  - servizi di altri fornitori
    - verso cui possiamo avere un certo grado di fiducia che si dovrebbe tradurre in una più o meno stringente configurazione del firewall

problemi

- la configurazione può essere molto complessa
  - facile fare errori
  - risoluzione DNS:
    - non fidata
    - quando farla?
    - inefficiente
  - configurazione deve essere fatta con indirizzi IP
    - o con nomi configurati staticamente

firewall e DDoS

- DDoS che saturano la banda
  - il firewall non aiuta
    - perché il target del DDoS è l'infrastruttura di rete
  - anycast e CDN sono approcci migliori ma costosi
    - esistono specifici servizi in cloud che fanno economia di scala
- DDoS sulle risorse del server nel caso **senza spoofing**
  - rilevamento in base all'attività delle sessioni (aging)
  - contrasto mediante white/black-list
    - gestibili in modo automatico dal firewall

syn-flood e syn-proxy

- **syn-flood**: DDoS che inviano tanti syn con **spoofing della sorgente**
- **syn-proxy**: un firewall che protegge il server da syn malevoli e da syn-flood
- come funziona
  - quando arriva un syn: il fw risponde syn-ack e non inoltra il syn al server
  - quando arriva l'ack, il fw riconosce la sessione come buona e apre una sessione tcp con il server
  - da ora in poi il fw fa da proxy tra le due sessioni
  - il fw è progettato per scalare sulle connessioni "mezze aperte" (per cui solo il syn è stato ricevuto)

syn cookies

- **syn cookies:** contromisura per attacchi syn-flood
- implementata su server o in fw con funzione di syn-proxy
- una delle poche soluzioni in grado di discriminare il traffico lecito da quello non lecito in maniera automatica
- **obiettivo: rispondere al syn senza mantenere stato**
- questo permette di scalare arbitrariamente rispetto al numero delle connessioni mezze aperte

- implementato in Linux

  echo 1 > /proc/sys/net/ipv4/tcp_syncookies

syn cookies

- lo stato è codificato nel sequence number di tcp
- per le connessioni «buone» il server ottiene nella risposta lo stato che avrebbe dovuto mantenere
  - il syn+ack contiene il sequence number +1
- **contrasto a ack malevoli**
  - la codifica è firmata in modo da poter riconoscere gli ack genuini
  - lo schema sarebbe ovviamente facilmente attaccabile senza questa contromisura
- il campo sequence number è di 32 bit
  - lo spazio è poco: deve contenere una firma + altro
  - la firma è quindi piuttosto debole
  - comunque un attacco (es. brute force) riesce a far passare solo una frazione minima di pacchetti
- nota che se l'attaccante risponde con un ack corretto, non può fare spoofing e si può applicare la contromisura delle white/black-list

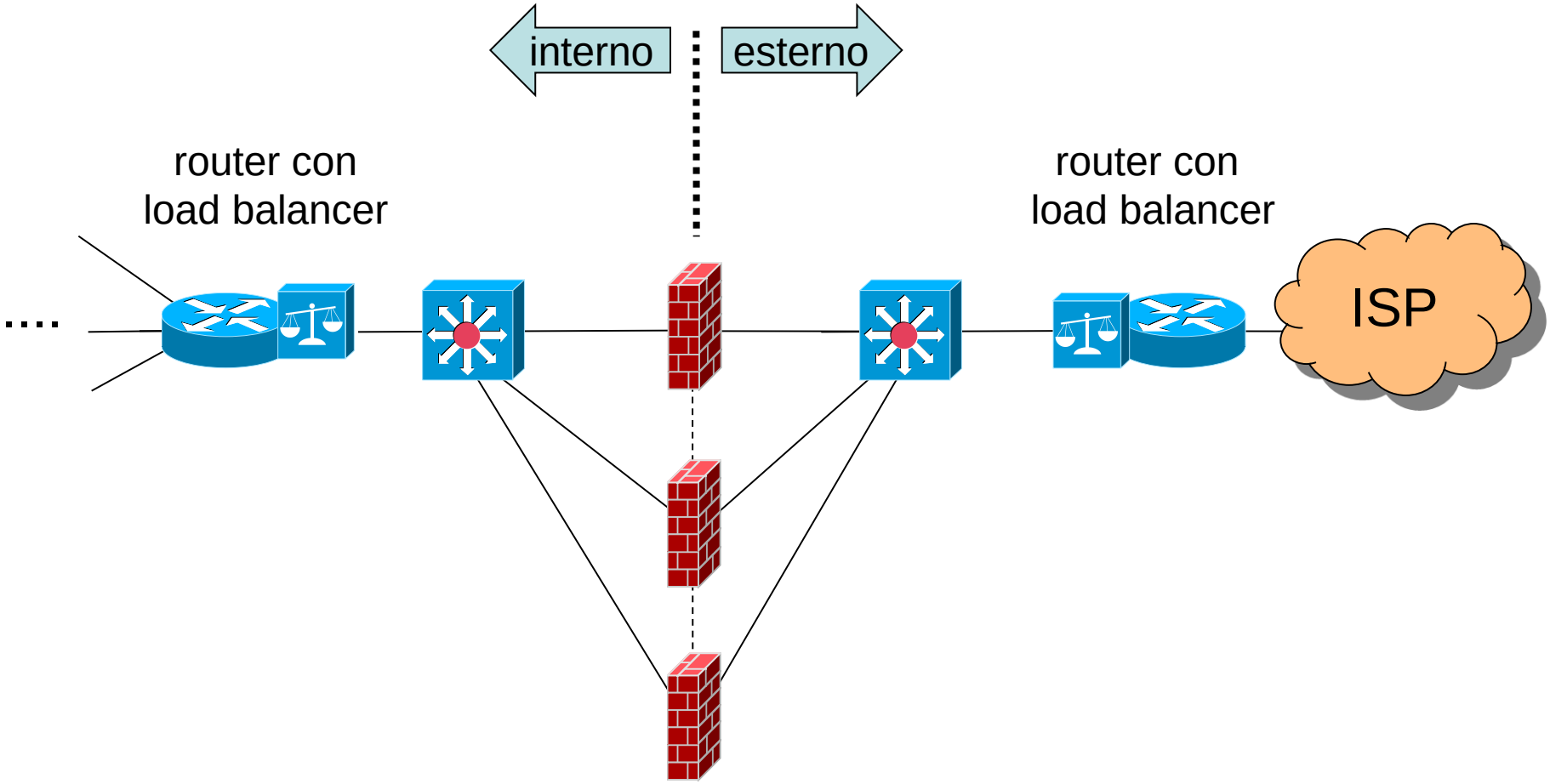
syn cookies

- limitato supporto a tcp options
- limitato supporto per maximum “segment size”

firewalls: prestazioni e affidabilità

- i fw spesso sono
  - un **collo di bottiglia** per le prestazioni
  - un **single point of failure**
- spesso utilizzati in configurazione “cluster” con tutti gli elementi attivi
  - **bilanciamento del carico**
  - **ridondanza**
- sono necessari load balancer e switch aggiuntivi

una architettura d'esempio



firewall e condivisione dello stato...

- se i load balancer inviano i pacchetti ad un firewall a caso...
- ... tutti i firewall devono vedere lo stesso stato delle connessioni!

- necessità di...
  - un collegamento tra i firewall
  - protocollo ad-hoc per la notifica delle nuove connessioni agli altri firewall
- si tratta di soluzioni proprietarie

architetture senza condivisione di stato

- strategia alternativa: i pacchetti di una stessa connessione sono sempre trattati da uno stesso firewall
- il load balancer non può fare scelte casuali
- ciascun fw deve vedere il traffico di una connessione in entrambe le direzioni
- in qualche modo i load balancer devono fare scelte coordinate

architetture senza condivisione di stato: coordinamento tra i load balancer

prima possibilità

- ciascun load balancer si ricorda il firewall assegnato alla connessione
  - ... e usa tale fw per i prossimi pacchetti

- richiede memoria lineare con il numero di connessioni
  - poco male la ram costa poco
- richiede di accedere alla ram at wire speed!
  - ...e la cache costa tanto!

architetture senza condivisione di stato
coordinamento tra i load balancer

seconda possibilità

- i load balancer adottano lo stesso algoritmo di scelta, tale che...
  - dipende solo dal pacchetto (es. da src, dest)
  - deterministico
- ...ma usato in maniera «speculare»
  - LB esterno: fw=hash(src_ip, dest_ip)
  - LB interno: fw=hash(dest_ip, src_ip)
  - infatti i pacchetti di ritorno hanno i src e dest invertiti
- no ram, no cache, solo cpu!
  - facile da realizzare in hardware (ASIC)

architetture senza condivisione di stato
coordinamento tra i load balancer

caveat

- il load balancer assegna l'intera connessione ad un fw, ma…
- ...alcuni protocolli coinvolgono più sessioni tcp
  - es. ftp

configurazioni high-availability (HA)

- quanto down-time siamo disposti a sopportare?
  - dipende dal business (service level agreement, penali, ecc.)
- le configurazioni tolleranti ai guasti per mezzo di elementi ridondati si dicono in “fail over” o “high-availability” (HA)
- i single point of failure vanno evitati
  - altrimenti esiste un fault per cui si ha down immediato del sistema
  - vogliamo invece avere del tempo per permettere ad un tecnico di ripristinare lo stato inziale senza che ci sia un down
    - un limitato degrado del servizio potrebbe essere ammesso nel frattempo (dipende dallo SLA)

HA e analisi del rischio

l'high-availability è necessaria quando...

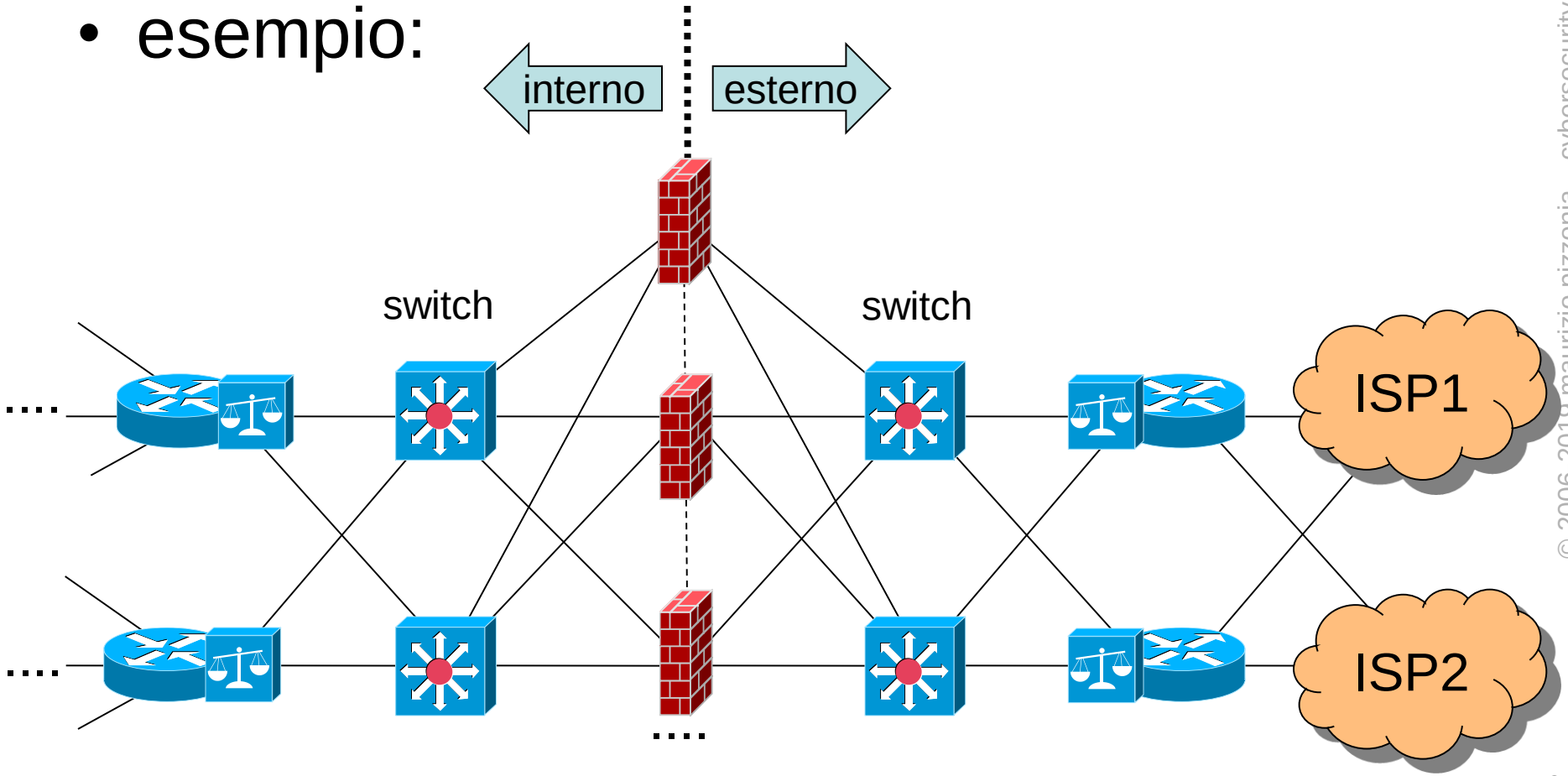
- gli SLA non prevedono il tempo per la sostituzione di uno degli elementi prima che scatti la penale
  - cioè esiste un rischio di pagare penali
- la penale (impatto) è sufficientemente alta da essere un rischio da mitigare
  - è possibile calcolare la spessa attesa per penali se riusciamo a stimare la probabilità di fault
  - i clienti che richiedono la penale potrebbero essere molti (es. hosting)
- il costo della configurazione HA dovrebbe essere inferiore al costo atteso derivante dal rischio di pagare penali

full HA

- in un cluster di fw, load balancer e switch sono dei «single point of failure»
- per una full high-avaliability bisogna ridondare tutto!

full high-availability

- ciascun elemento dell'architettura deve essere ridondato
- esempio:



HA: tempi di reazione

- in caso di guasto di un singolo elemento la rete di deve riconfiguare automaticamente
- OSPF ha tempi di convergenza molto rapidi
  - deve girare sui load balancer e sui firewall
  - convergenza in ~200ms (fonte cisco)
- soluzioni di livello 2 (802.1w rapid spanning tree) sono più lente
  - ~1s (fonte cisco)

HA: costi

- le configurazioni HA sono più complesse e più costose
- capex (capital expenses)
  - apparati ridondati
  - deployment più complesso
  - sistema di alert sui fault
- opex (operating expense)
  - personale (reperibilità, skill specifici)
  - più apparati da manutenere
  - più spazio da affittare nel datacenter

(intermezzo: gw ridondato)

- principio: no single point of failure in lan
- soluzioni tipiche per il default gateway di una lan
  - quando si rompe corri ad accendere il fail-over (!)
  - fai partecipare ciascuna macchina della lan al protocollo di routing (!!!!)
  - Virtual Routing Redoundancy Protocol, rfc 5798
- VRRP
  - un indirizzo ip *A* per un «router virtuale»
  - il router virtuale è composto da, almeno, due router fisici
  - solo uno dei due risponde all'arp request per *A* : il *master*
  - il master manda un heartbeat periodico agli altri router
  - quando il master muore, uno degli altri prende il suo posto
    - c'è un meccanismo di elezione come per il root bridge nello spanning tree degli switch
- HSRP: analogo, proprietario cisco

linux netfilter

- sistema di moduli Linux che realizza un fw
  - packet filter
  - stateful
  - nat
  - sistema di access list (chains) organizzate per funzionalità (tables)
  - tables
    - filter
      - chains: INPUT, OUTPUT, FORWARD
    - nat
      - chains: PREROUTING, POSTROUTING, OUTPUT
  - si possono definire delle “user-defined chain”
  - ciascuna tabella viene gestita da un modulo del kernel

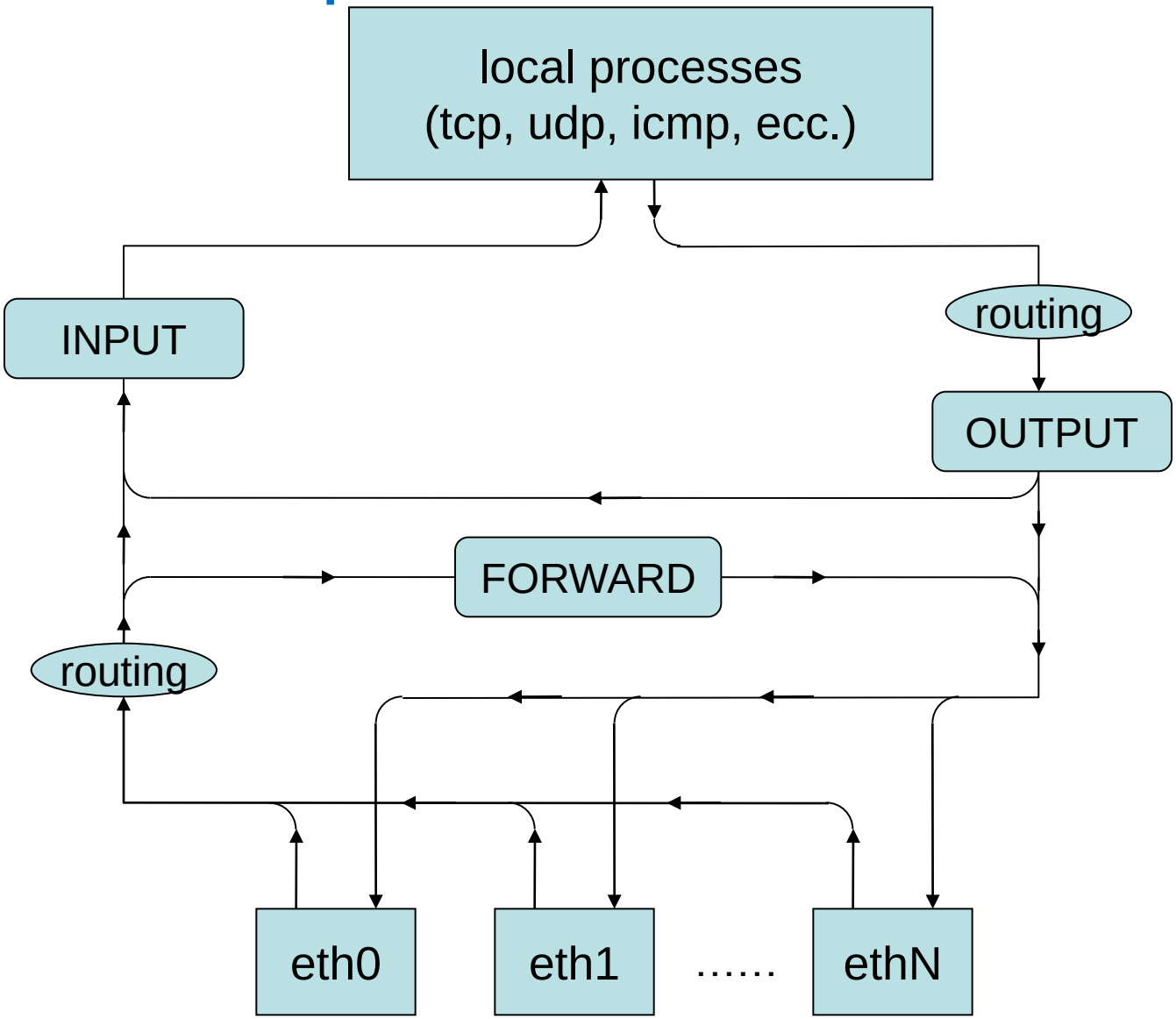
linux netfilter: chains (acl)

- ciascuna chain ha una sequenza di regole e un target
  - il target specifica cosa fare del pacchetto se questo non ha attivato alcuna regola
- target possibili
  - ACCEPT (pacchetto ok, lascia passare)
  - DROP (pacchetto droppato)
  - REJECT (come drop ma invia un icmp di errore al mittente)
  - <chain name> (salta alla chain specificata, come una chiamata a procedura, utile con “user-defined chain”)
  - RETURN (ritorna alla chain chiamante)
  - ...

linux netfilter: firewall rules

- una regola ha dei parametri e un target
  - i parametri specificano per quali pacchetti la regola viene attivata
  - il target specifica cosa fare
- parametri
  - protocollo (-p ip/tcp/udp/)
  - source/destination address (-s/-d [!]x.x.x.x/x)
  - source/destination port (--sport/--dport port)
  - input/output interface (-i/-o ethN)
  - tcp flags syn=1 e ack=0 (--syn)
  - e altri

chains e flusso pacchetti per la «filter table»



linux netfilter: connection tracking

- netfilter tiene traccia delle “connessioni”
  - per tcp e udp: coppia (non ordinata) di coppie {<addr, port>, <addr, port>}
  - per icmp echo: coppia di indirizzi {addr, addr}
- a ciascun pacchetto è associata una connessione (eventualmente ne viene creata una nuova)
  - questo viene fatto appena un pacchetto entra in netfilter da una interfaccia o dai processi locali
  - le connessioni muoiono per timeout o per protocollo (fin/ack, rst)
- stato del pacchetto
  - NEW: primo pacchetto della connessione
  - ESTABLISHED: pacchetto di una connessione già esistente
  - RELATED: nuova connessione associata ad una precedente (ftp, icmp errors)
  - INVALID: impossibile associare una connessione (es. icmp error non associati ad alcuna connessione)

linux netfilter: extended matches

- l'opzione -m <nome> permette di attivare molti altri tipi di regole
  - **state**: filtro in base allo stato (NEW, ESTABLISHED, RELATED, INVALID)
    - **conntrak** è una versione estesa
  - **mac**: filtro in base agli indirizzi mac di livello 2
  - **limit**: filtro in base alla frequenza di arrivo
    - anti DoS, anti scanning ecc.
  - **iprange**: es. 192.168.1.13-192.168.2.19
  - e molti altri

netfilter: comandi

- iptables
  - modifica e mostra la configurazione
    - -L mostra la configurazione
    - --flush cancella la configurazione
    - -A <chain> aggiunge in coda a <chain> un regola
    - ecc.
- iptables-save
  - dump della configurazione attuale
- iptables-restore
  - carica la configurazione da un dump fatto con iptables-save (più efficiente che molte chiamate a iptables)

esempio: nessun filtro

- **iptables --flush**
- **iptables-save**

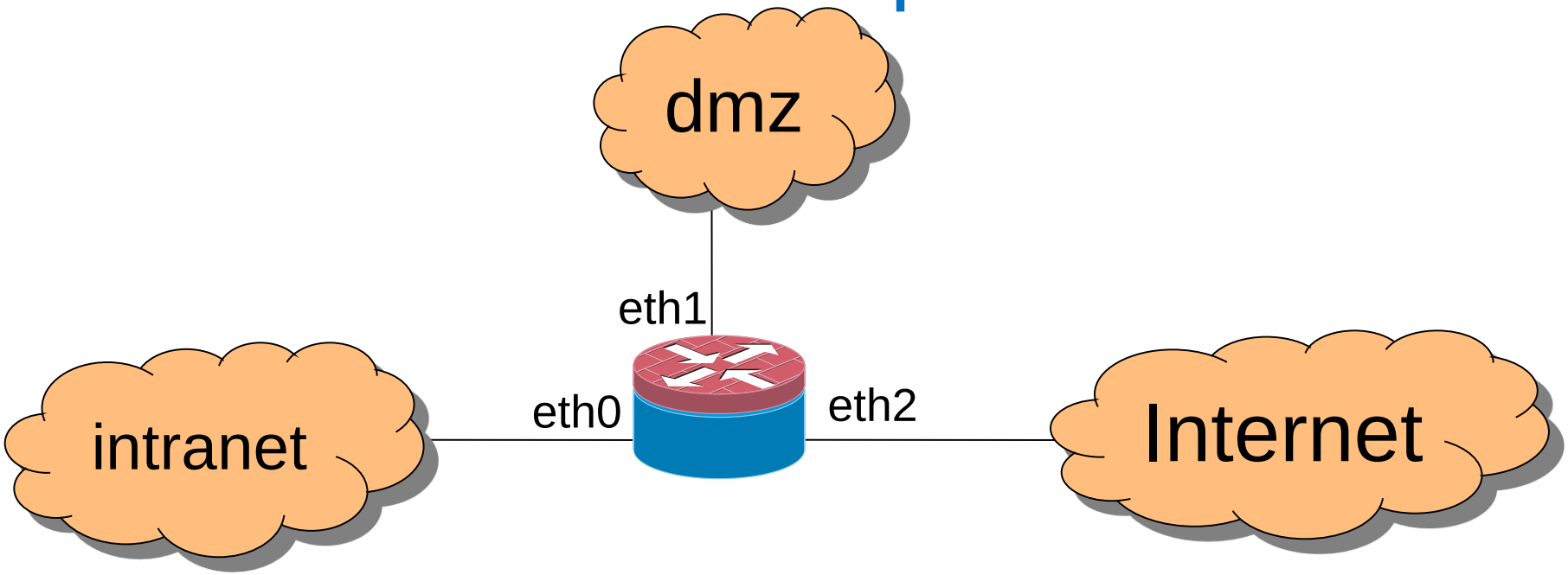
```
# Generated by iptables-save v1.3.3 on Fri Dec  8 17:58:19 2006
*filter
:INPUT ACCEPT [37:4620]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [19:1352]
COMMIT
# Completed on Fri Dec  8 17:58:19 2006
```

esempio: un semplice «personal firewall»

- `iptables -P INPUT DROP`
- `iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`

```
# Generated by iptables-save v1.3.3 on Fri Dec  8 18:26:51 2006
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [14735:2397896]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Fri Dec  8 18:26:51 2006
```

implementazione della politica dmz di esempio




```
*filter
:INPUT ACCEPT [64:3448]
:FORWARD DROP [13:858]
:OUTPUT ACCEPT [409:37987]
-A FORWARD -i eth0 -o eth1 -m state --state NEW -j ACCEPT
-A FORWARD -i eth2 -o eth1 -m state --state NEW -j ACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

nftables

- iptables è in via di dismissione
- **nftables** sarà il nuovo comando per configurare netfilter
  - sintassi più simile a quella di un router
  - sintassi più semplice
  - un solo comando per tutte le feature
  - supporta matching di *set*
    - permette di scalare sul numero di prefissi e porte su cui fare matching