

# APPUNTI TEORIA INTELLIGENT AND SECURE NETWORKS

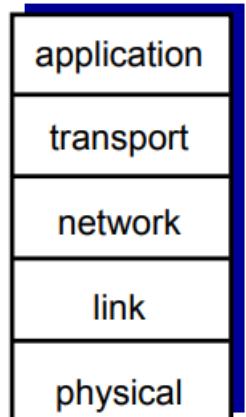
Credits By  
Engineering Team

## TCP/IP Recap

### Stack protocollare di internet

Abbiamo 5 livelli principali

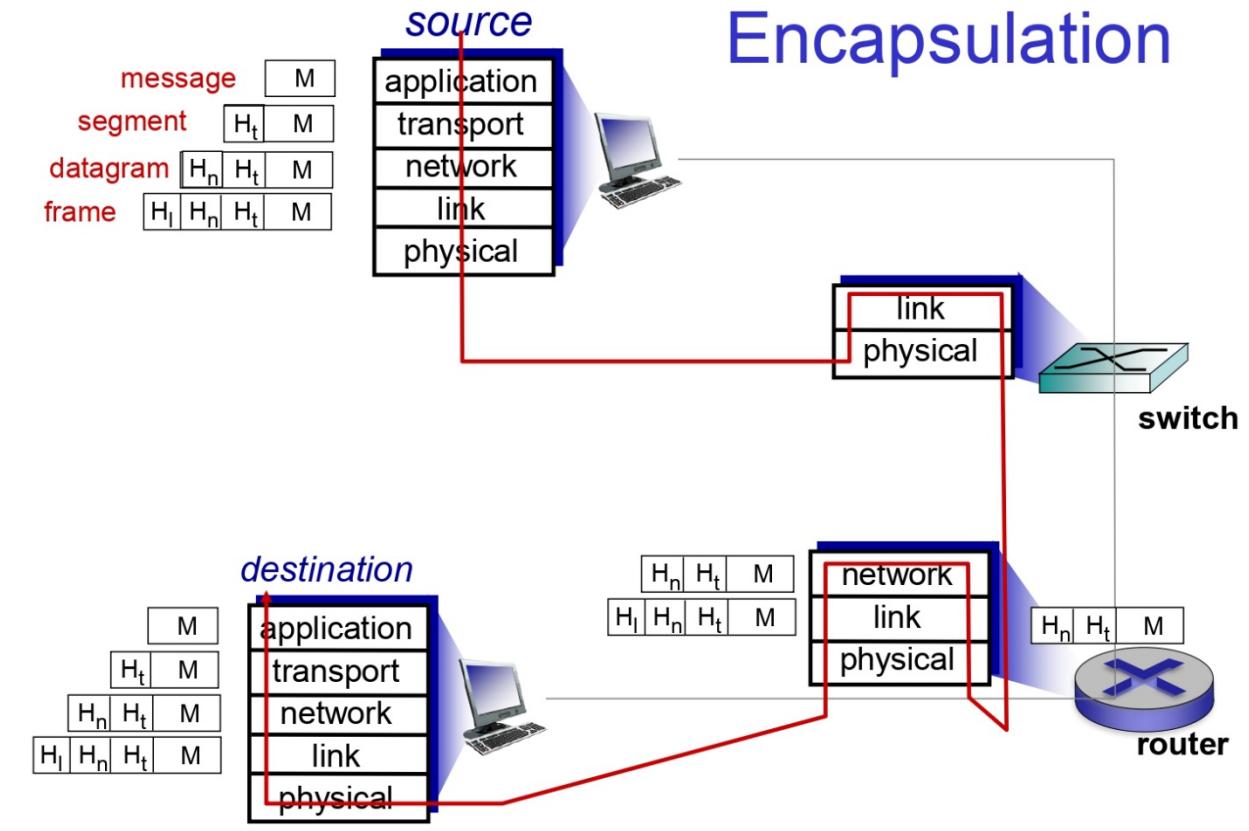
- **Strato di applicazione:** definisce il formato dei messaggi scambiati tra le singole applicazioni. Rete di supporto alle applicazioni.
  - FTP, SMTP, HTTP (protocollo più utilizzato, usato anche per tutto il traffico video, rispetta i vincoli del REST)  
(\*vedi interfaccia REST per conoscenza)
- **Strato di trasporto:** trasferimento dei dati da processo a processo; vede solo mittente e destinatario, non tiene conto di tutti i nodi intermedi tra l'origine e la destinazione. Lo scopo è quello di fornire all'applicazione un canale virtuale con delle caratteristiche.  
Analoga del condominio: numero civico rappresenta l'indirizzo dello strato 3 (IP) mentre il numero di porta dello strato 4 rappresenta l'interno del condominio.
  - TCP (trasporta il 70% del traffico internet, offre un canale affidabile e complesso), UDP (offre un canale snello, è l'essenza del best effort, cioè, fa del suo meglio per trasportare i pacchetti senza fornire alcuna garanzia di prestazione)
- **Strato di rete:** instradamento dei datagrammi dal mittente al destinatario (in internet una rete complessa è internet, ovvero una rete realizzata mediante interconnessione di altre reti). Lavora dal nodo sorgente al nodo destinazione tenendo conto di tutti i dispositivi che sono in mezzo.
  - IP, protocolli di routing
- **Strato di collegamento:** aggrega i bit in unità informative ed applica, a seconda del tipo di collegamento, le tecniche per l'affidabilità sul collegamento, quindi controllo dell'errore, correzione dell'errore in anticipo, ...  
Lavora sul collegamento.  
Trasferimento dei dati tra elementi di rete vicini
  - Ethernet, 802.11 (WiFi), PPP  
Gli indirizzi ethernet e wifi hanno la stessa struttura: 12 cifre esadecimali
- **Strato fisico:** trasferimento di bit da un capo all'altro di un collegamento, può essere wired o wireless. Lavora sul collegamento.



### Incapsulamento

Le reti funzionano tramite il principio dell'incapsulamento. Un messaggio generato dall'applicazione, per essere trasportato da un nodo sorgente a un nodo destinatario, deve

essere incapsulato attraverso tutti gli strati protocollari. Essere incapsulato significa che ogni strato protocollare aggiunge al messaggio originale una intestazione che rappresenta le istruzioni che devono essere eseguite dal protocollo di quello strato.



H: header

Gli switch vengono attraversati in maniera trasparente, cioè non modificano il pacchetto neanche allo strato di collegamento, però ispezionano lo strato di collegamento per decidere su quale porta debba essere inviato il pacchetto.

Invece i ruoter, che hanno 3 strati protocollari, fanno un parziale decapsulamento, cioè tolgono l'intestazione dello strato di collegamento, viene passato lo strato di rete che analizza l'intestazione dello strato di rete e decide su quale porta spedire il pacchetto. Il pacchetto quando sarà spedito vedrà aggiungere di nuovo l'intestazione dello strato di collegamento e poi verrà consegnato a destinazione.

## Cosa è un'applicazione di rete

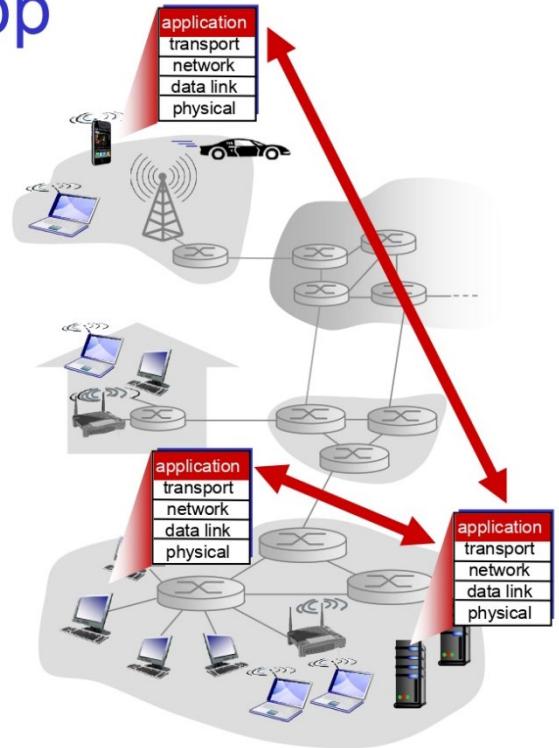
Un'applicazione di rete è per definizione un'applicazione distribuita che:

- gira su sistemi diversi connessi in rete
- comunica in rete
  - ad esempio, software del server web comunica con software del browser

→ Le applicazioni girano SOLO su sistemi terminali!  
(I sistemi terminali non sono solo i nostri dispositivi ma sono anche i server che chiedono i servizi.)

I dispositivi core della rete (router e switch) non eseguono codice che contribuisce alle applicazioni.

Ci possono essere le cosiddette middlebox, cioè dei dispositivi intermedi che hanno anch'essi una funzione applicativa. Rappresentano un incubo per i gestori della rete perché rompono la visione da estremo a estremo. Le middlebox possono essere controllate dallo strato 3 a salire. Es: un firewall agisce allo strato 3, un NAT agisce allo strato 3, un proxy agisce allo strato applicativo.

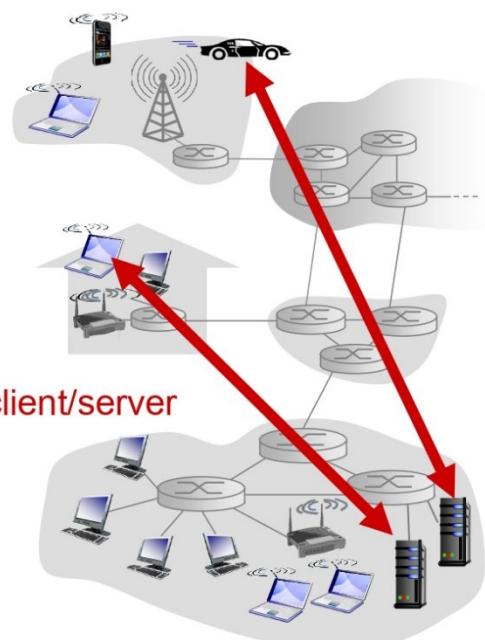


## Architettura client-server

Il paradigma client-server è quello più diffuso tra le applicazioni.

Server:

- sempre accesso per essere sempre pronto a rispondere alle richieste
- deve avere un indirizzo fisso, sia a strato 3 (indirizzo IP) che a strato 4 (sempre stessa porta lato server)
- i servizi offerti dai server devono essere in grado di scalare, cioè di aumentare le risorse di calcolo a propria disposizione man mano che aumentano le richieste; questo si realizza tramite piattaforme virtualizzate.



Client:

- devono poter comunicare con il server
- la loro connessione può essere intermittente

- possono avere un indirizzo IP dinamico
- non c'è mai interazione client-client, non c'è mai una connessione diretta tra loro

## Comunicazione tra processi

Un processo è un programma che gira all'interno di un host.

- All'interno dello stesso host, due processi possono comunicare attraverso comunicazioni **inter-processo** (definite per sistema operativo)
- Quando gli host sono su nodi differenti la comunicazione tra i processi avviene utilizzando un protocollo di rete per scambiare **messaggi** tra nodi diversi

Processo del client: processo che avvia la comunicazione.

Processo del server: processo che aspetta di essere contattato.

Le applicazioni con architetture P2P (peer-to-peer) hanno processi client e processi server. In queste applicazioni i nodi agiscono sia da client che da server contemporaneamente e questo permette la comunicazione diretta tra due entità, quindi senza entità di intermediazione.

## I protocolli applicativi definiscono

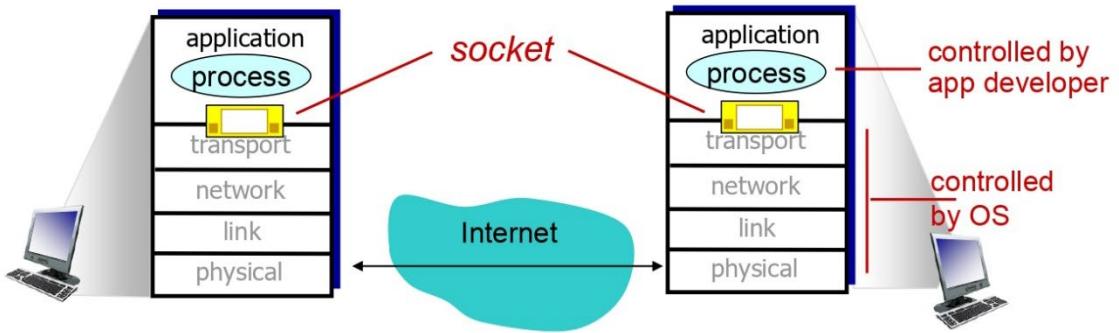
- I tipi di messaggi scambiati
  - es: richiesta, risposta
- La sintassi dei messaggi
  - quali campi ci sono e come vengono definiti i campi
- La semantica dei messaggi
  - che cosa significano quei campi
- Le regole: che cosa fare quando invio un messaggio o quando ricevo un messaggio
  - tra le regole assume un ruolo importante il timing, cioè la temporizzazione per lo scambio dei messaggi
- Protocolli aperti
  - definiti negli standard di internet (RFC, Request for Comments)
  - consente l'interoperabilità
  - es: HTTP, SMTP
- Protocolli proprietari
  - es: Skype

## Sockets

Gli indirizzi di strato 3 accoppiati con gli indirizzi di strato 4 rappresentano il cosiddetto socket, ovvero una porta che interconnette i processi con la rete.

Il processo invia/riceve messaggi al/dal suo socket.

Un socket è una concatenazione di due stringhe: la prima stringa è l'indirizzo IP e la seconda è l'indirizzo di porta. C'è un socket sul client e un socket sul server. A seconda del tipo di protocollo di strato 4 per definire una comunicazione avremo bisogno di entrambi, cioè della coppia di socket: questo è il caso del TCP; oppure solo il server destinatario: questo è il caso dell'UDP.



## Processi di indirizzamento

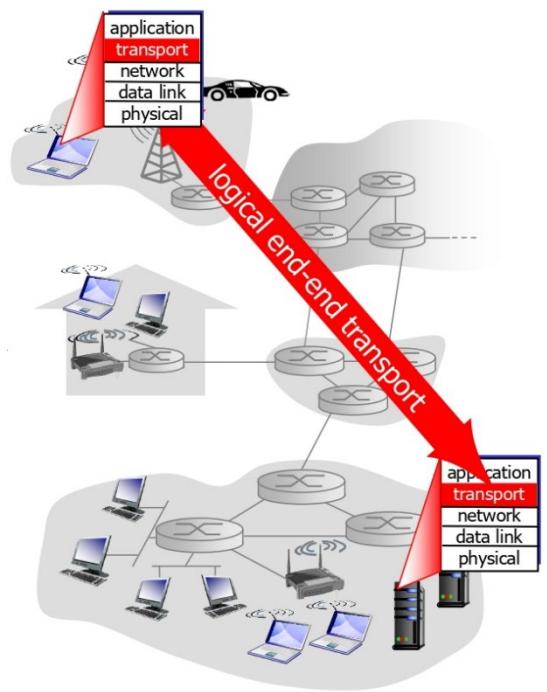
- per ricevere messaggi, un processo deve avere un identificatore
- l'identificatore include sia l'indirizzo IP sia i numeri di porta associati al processo sull'host
- esempi di numeri di porta:
  - server HTTP: 80
  - server mail: 25
- per inviare un messaggio HTTP al server web gaia.cs.umass.edu :
  - indirizzo IP: 128.119.245.12
  - numero di porta: 80
- un dispositivo host ha un unico indirizzo IP a 32 bit
- un indirizzo IP di un host non può identificare un processo che è in esecuzione sullo stesso host perché diversi processi possono essere in esecuzione sullo stesso host

## Servizi e protocolli di trasporto

Forniscono una comunicazione logica tra i processi applicativi in esecuzione su host diversi.

I messaggi applicativi possono essere molto grandi e quando il messaggio viene passato allo strato di trasporto, questo strato lo divide in un certo numero di identità informative più piccole, chiamate segmenti e che vengono incapsulate nei messaggi dello strato di trasporto. Dal ricevitore viene fatta l'operazione duale: vengono quindi presi i differenti segmenti e vengono riassemblati per ricostruire il messaggio a strato applicativo. Questo viene fatto sia dal TCP che dall'UDP.

La differenza fondamentale è che se perdo un segmento con UDP posso dire addio al messaggio perché il segmento non verrà mai recuperato, mentre se perdo un segmento con il TCP, il TCP stesso prova a ritrasmetterlo.



## Strato di trasporto vs strato di rete

- Strato di rete: comunicazione logica tra host
- Strato di trasporto: comunicazione logica tra processi; si basa e migliora i servizi dello strato di rete

## Che servizio offre uno strato di trasporto per un'applicazione?

- Affidabilità dei dati
  - alcune app (ad es. trasferimento dei file, transazioni web) richiedono il 100% di affidabilità sul trasferimento dei dati
  - altre app (ad es. audio) possono tollerare qualche perdita  
➔ Solo il TCP garantisce l'affidabilità
- Rapidità con cui le unità informative vengono elaborate e spedite
  - alcune app (ad es. telefonia di internet, giochi interattivi) richiedono un ritardo basso per essere “efficaci”  
➔ L'UDP garantisce la rapidità (il timing)
- Portata
  - alcune app (ad es. multimedia) richiedono un minimo di portata per essere “efficaci”
  - altre app (“elastic apps”) possono usare qualsiasi portata che hanno  
➔ L'UDP garantisce una portata più elevata

- Sicurezza
  - Crittografia, integrità dei dati, ...
  - ➔ Né TCP né UDP garantiscono la sicurezza. Servono dei meccanismi aggiuntivi (tipo TLS Transport Layer Security, SSL Secure Sockets Layer) che offrono sicurezza ma solo alle applicazioni. Altrimenti IPSEC se voglio sicurezza a strato 3 (VPN, che rappresenta però anche un altro ostacolo al monitoraggio della rete).

## App di internet: mappatura delle applicazioni sui protocolli di trasporto

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

## Servizi internet dei protocolli di trasporto

TCP:

- Trasporto affidabile tra il processo di invio e di ricezione
- Controllo del flusso: il mittente non farà sovrapporre il ricevitore
- Controllo della congestione: accelerazione del mittente quando la rete è in sovraccarico
- Non prevede: rapidità, portata minima garantita, sicurezza
- Orientato alla connessione: richiede una configurazione (un saluto, il three-way handshake) tra i processi del client e del server. I flag SYN servono infatti per sintonizzare mittente e destinatario e per definire l'intero che rappresenta il numero dei byte che ci serve per assicurare la corretta consegna dei dati senza errori,

UDP:

- Trasferimento dei dati inaffidabile tra il processo mittente e ricevitore
- Non prevede: affidabilità, controllo del flusso, controllo della congestione, rapidità, portata garantita, sicurezza o configurazione della connessione

## Protezione del TCP

TCP & UDP:

- Nessuna crittografia

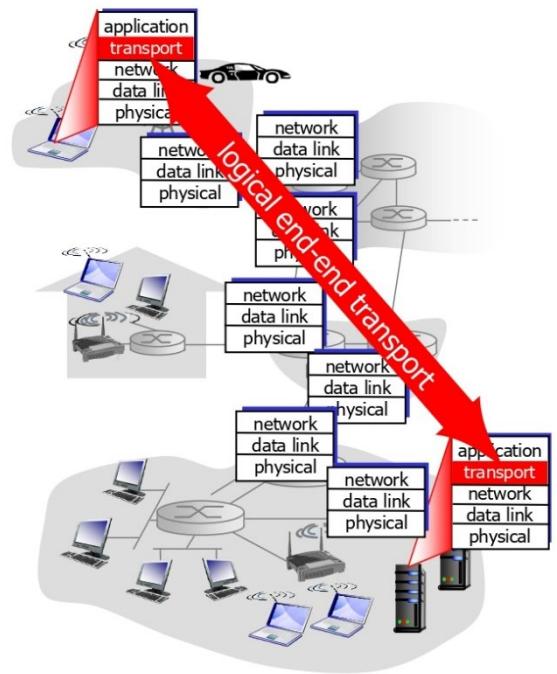
- Password in chiaro inviate nel socket di internet

SSL:

- Fornisce una connessione TCP crittografata
- Integrità dei dati, cioè affidabilità
- Autenticazione del punto finale
- È allo strato applicativo: le applicazioni usano librerie SSL che “parlano” con il TCP
- SSL socket API: password in chiaro inviate nel socket di internet crittografato

## Protocolli dello strato di trasporto di internet

- Consegna affidabile e in ordine (TCP)
- Consegna inaffidabile, non ordinata (UDP)
- Servizi non disponibili:
  - garanzie sui ritardi
  - garanzie di larghezza di banda



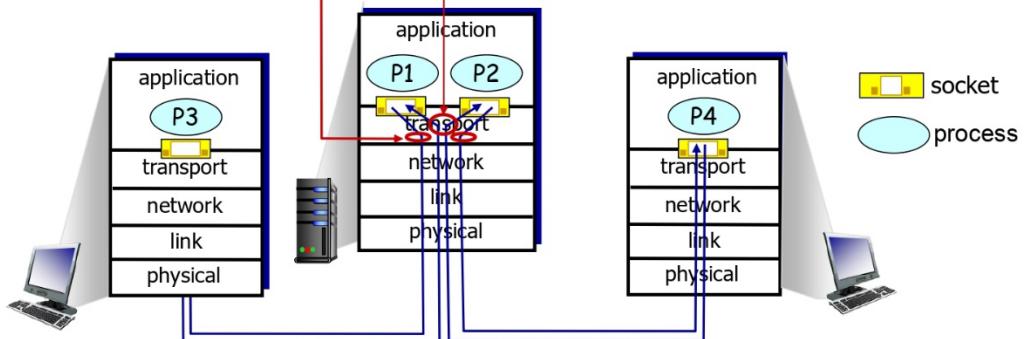
## Multiplazione/Demultiplazione

*multiplexing at sender:*

handle data from multiple sockets, add transport header (later used for demultiplexing)

*demultiplexing at receiver:*

use header info to deliver received segments to correct socket



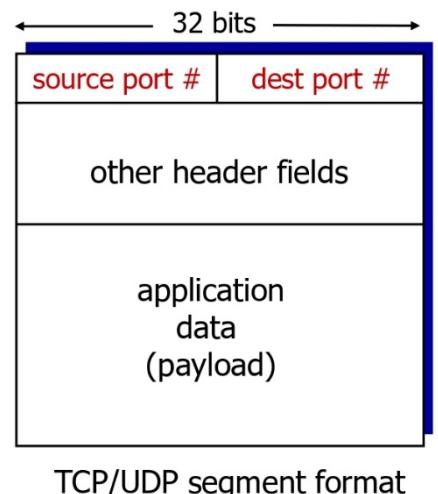
L'applicazione definisce una porta mittente e attraverso questa porta invia i dati verso un'applicazione remota che ha la sua porta destinazione già acquistata. Per utilizzare i socket ho bisogni di informazioni sull'indirizzo IP e sulla porta locale e remota.

In questo esempio abbiamo la multiplazione al mittente perché due applicazioni diverse, P<sub>1</sub> e P<sub>2</sub>, utilizzano lo stesso strato di trasporto per inviare i messaggi e le risposte sono la demultiplazione al ricevitore perché lo stesso strato di trasporto riesce a gestire i messaggi incoming che poi devono essere consegnati a P<sub>1</sub> e a P<sub>2</sub> sulla base del numero di porta di destinazione che viene specificato nei messaggi inviati da P<sub>3</sub> e da P<sub>4</sub>, entrambi remote.

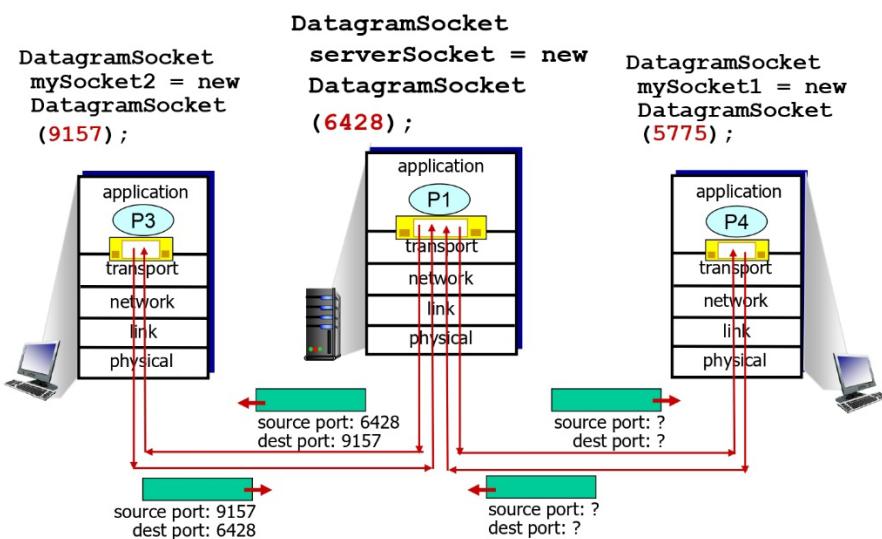
Questo meccanismo utilizza il numero di porta mittente e destinatario.

## Come funziona la demultiplazione

- l'host riceve i datagrammi IP
  - ogni datagramma ha un indirizzo IP mittente e un indirizzo IP destinatario
  - ogni datagramma porta un segmento dello strato di trasporto
  - ogni segmento ha il numero di porta mittente e destinatario
- l'host usa l'indirizzo IP e il numero di porta per indirizzare il segmento verso il giusto socket

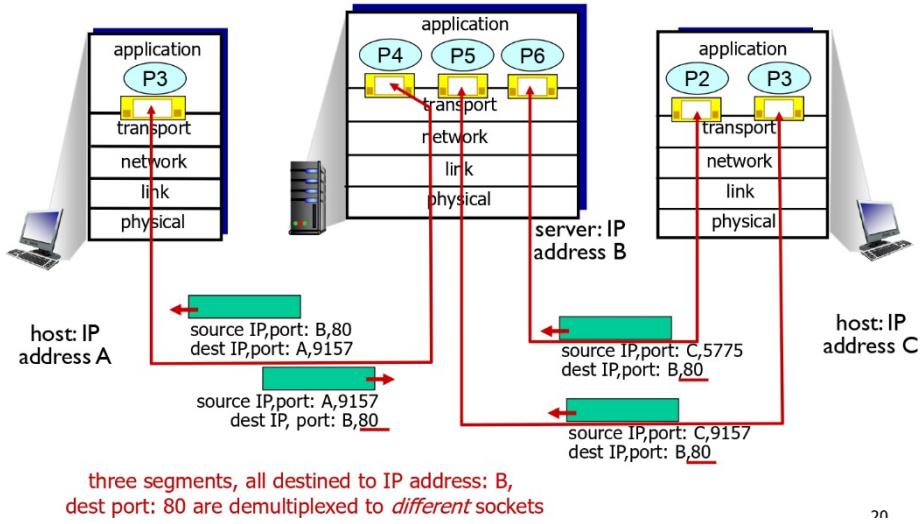


## Demultiplazione senza connessione



In questo caso il socket è unico e risponde anche ad altre applicazioni. Grazie al socket unico avrò risposte veloci.

## Demultiplazione orientata alla connessione

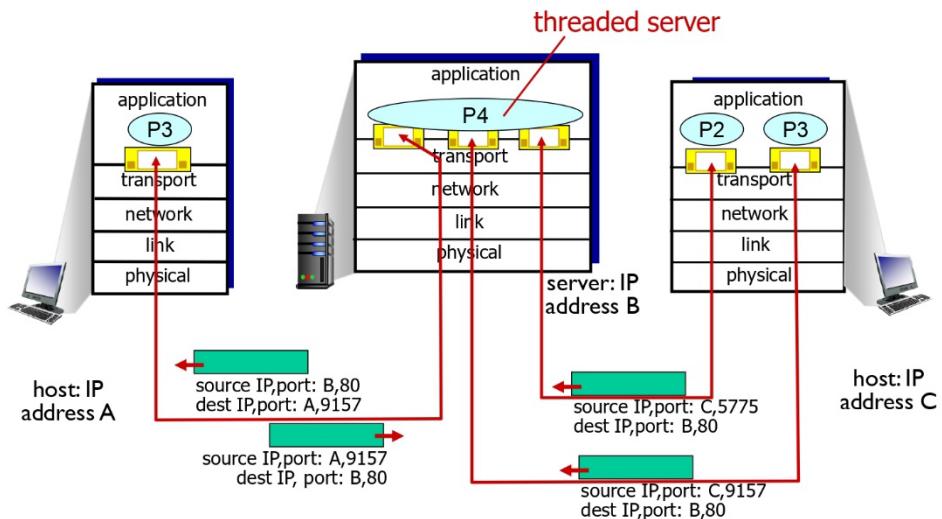


20

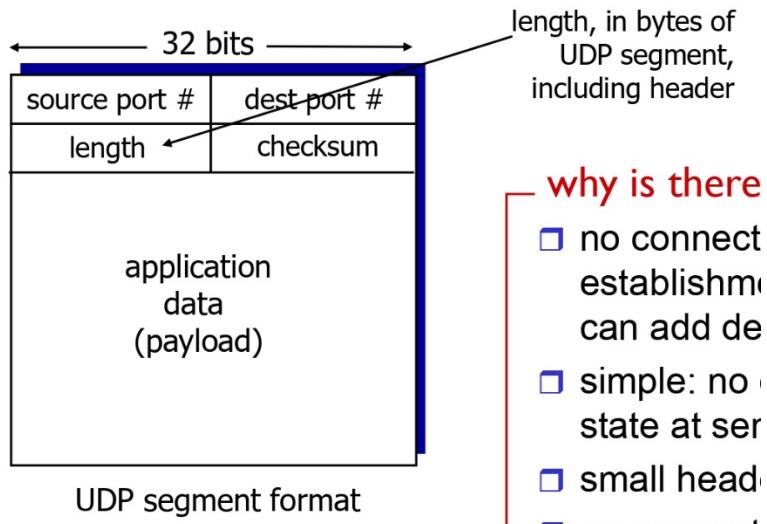
In questo caso abbiamo la stessa porta di destinazione ma viene creato un socket separato.

In questo caso quando ho molte richieste si creano molti processi e il moltiplicarsi dei processi può andare ad esaurire le risorse di calcolo di un nodo: questo è il modo in cui si fanno tipicamente attacchi DoS o DDoS.

Per gli ambienti cloud, man mano che aumentano il numero di socket può essere necessario andare a duplicare la macchina virtuale per evitare che ci siano situazioni di stallo, di mancanza di responsività da parte del servizio. L'osservazione di un server da un punto di vista delle sue prestazioni può dar luogo sia a tecniche che rilevino o prevengano le intrusioni o che rilevino e prevengano le situazioni di sovraccarico. Nel primo caso bisognerà notificare e, nei casi migliori, anche bloccare i tipi di attacco; nel secondo caso dovrò avere delle tecniche predittive che mi facciano capire che il numero di richieste sta progressivamente salendo e sulla base di previsioni più o meno accurate dovrò decidere quando aumentare il numero di istanze che realizzano il servizio. È importante perché in cloud le istanze (cioè le macchine virtuali) si pagano e quindi avere dei meccanismi intelligenti permette di evitare spreco di risorse di calcolo e finanziarie.



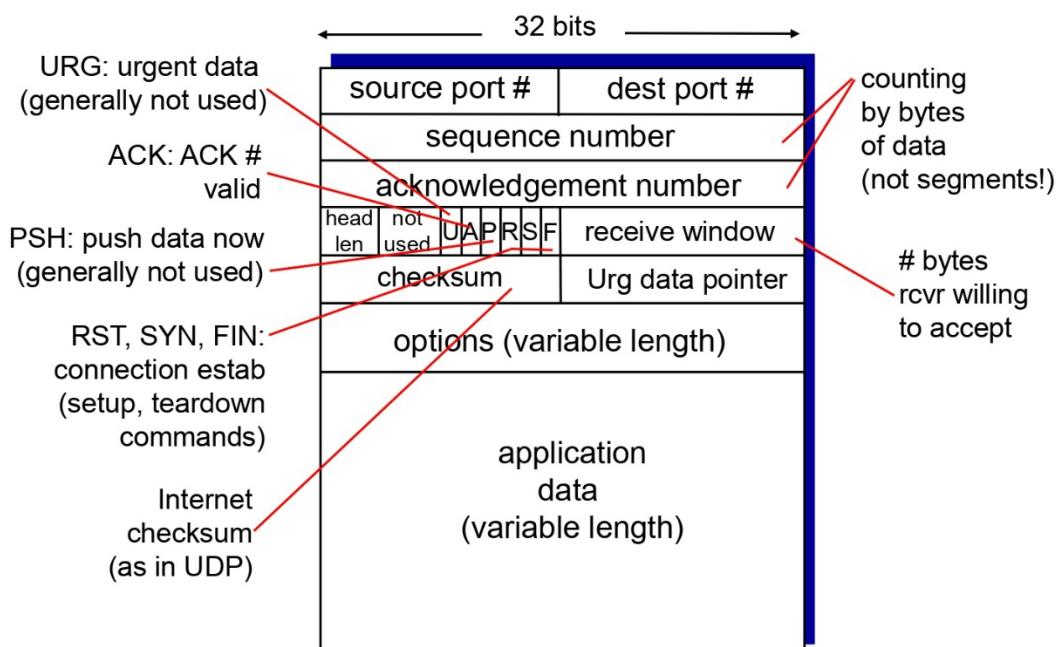
## UDP: intestazione del segmento



UDP:

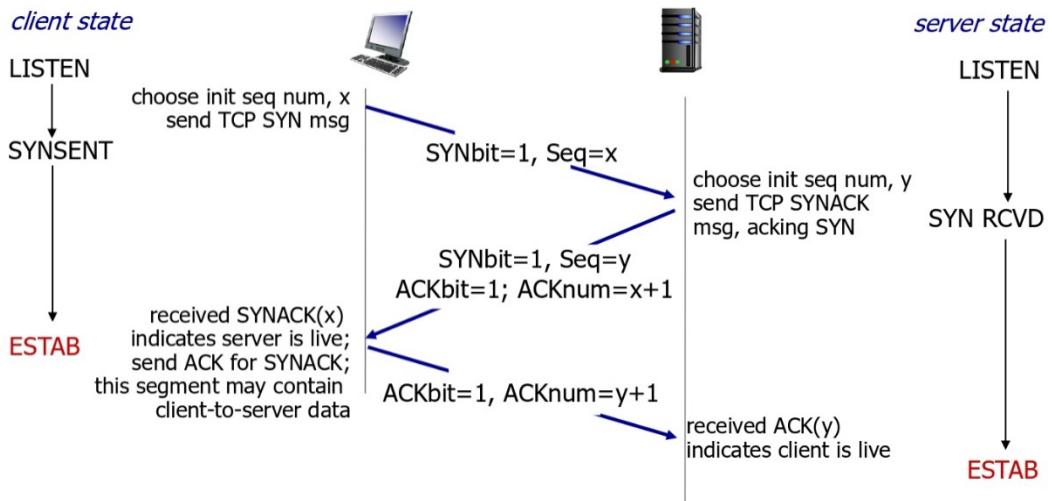
- Non serve stabilire una connessione (che può aggiungere ritardo)
- Dimensioni ridotte dell'intestazione
- Nessun controllo di congestione: UDP può essere veloce come desiderato

## Struttura del segmento TCP

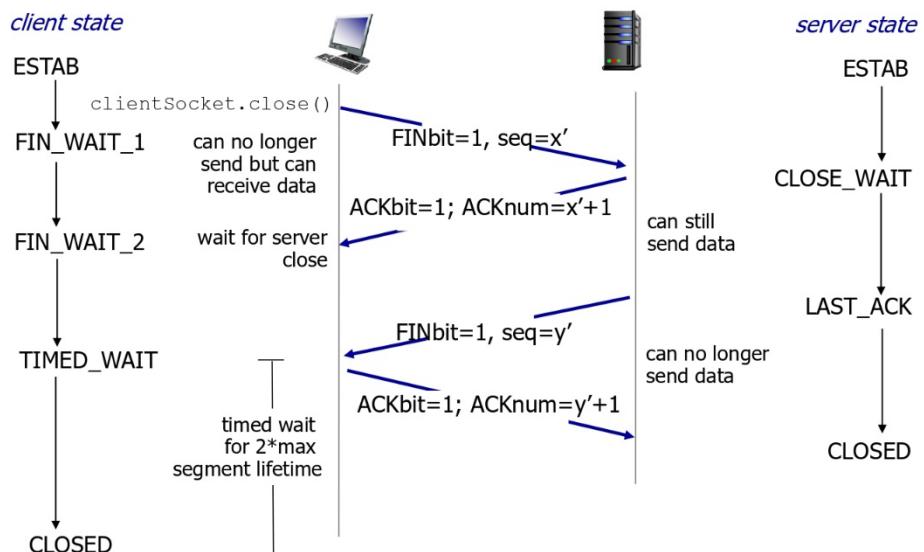


Il flag SYN indica una connessione che si apre; il flag FIN indica una connessione che si chiude.

## TCP 3-way handshake



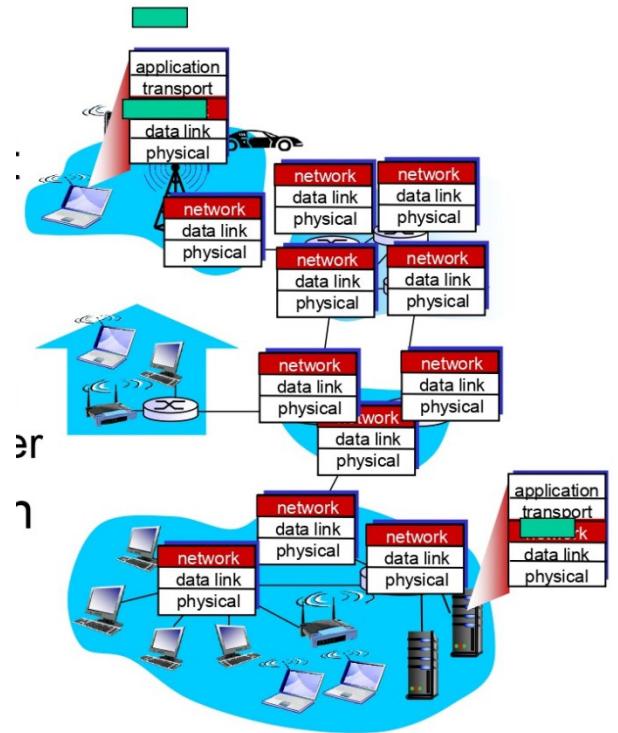
## TCP: chiusura di una connessione



## Strato di rete

- Trasporta i messaggi (segmenti) da un punto di origine a un punto di destinazione
  - lato mittente incapsula i segmenti in datagrammi
  - lato destinazione consegna i segmenti allo strato di trasporto
- Protocolli dello strato di rete in ogni host, router
  - tutti i dispositivi di rete che hanno lo strato di rete (cioè il protocollo IP) analizzano i pacchetti che hanno intestazione IP!! (Questo perché ogni nodo deve effettuare l'instradamento dei pacchetti)

 : switch (strato 2)  
 : router (strato 3)



## Due funzioni chiave dello strato di rete

- Funzione decisionale: routing (instradamento): determinare il percorso dei pacchetti da sorgente a destinazione
- Funzione attuativa: forwarding (inoltro): il pacchetto viene spostato dalla porta di ingresso alla porta di uscita

Una volta portato il pacchetto sulla porta di uscita lo strato 3 ha finito, per effettuare la trasmissione si passa allo strato 2.

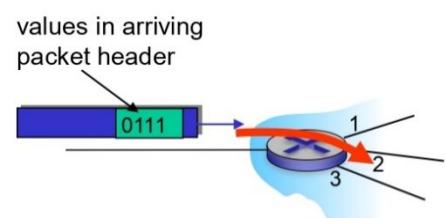
Analogia del viaggio: la pianificazione è l'equivalente della funzione decisionale mentre l'attraversamento di un aeroporto o di una stazione è la funzione attuativa.

## Strato di rete: piano dati e piano controllo

Piano dati:

- È un'astrazione che rappresenta le funzioni di controllo che servono a trasferire i dati di utente
- Locale
- Determina come il datagramma in arrivo sulla porta di ingresso del router è inoltrato alla porta di uscita del router
- Funzione attuativa

Piano di controllo:



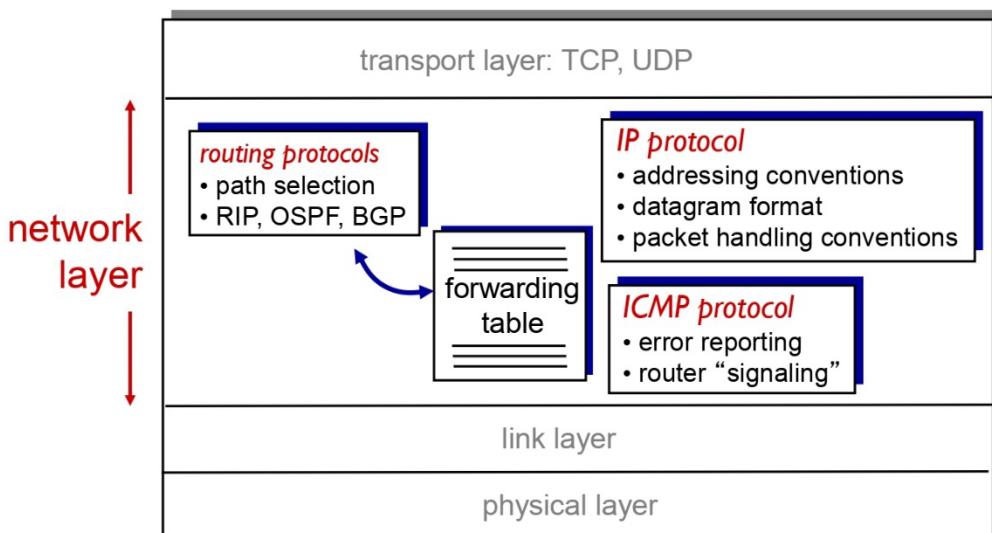
- Permette al piano dati di funzionare
- Logica a livello di rete
- Determina come il datagramma viene instradato tra i routers lungo il percorso end-end da host mittente a host destinatario
- Due approcci del piano di controllo:
  - Algoritmi di instradamento tradizionali: implementati nei routers
  - Reti definite dal software (SDN): implementati nei servers (remoti)

Esempi: allo strato 4 una funzione del piano di controllo è il controllo di flusso (avviene attraverso lo specifico campo del pacchetto che è la receive window) oppure i riscontri (avvengono attraverso due specifici campi del pacchetto: il flag A, Acknowledgement, e il numero di sequenza del riscontro). Per quanto riguarda il controllo TCP, il piano di controllo avviene in banda, cioè nello stesso pacchetto ci sono sia il piano dati che il piano di controllo.

Nel protocollo IP il piano di controllo è quello che popola le tabelle di instradamento dei nodi, il piano dati usa queste tabelle. OSPF è il classico protocollo di instradamento e i dati di quel protocollo sono dati di controllo perché servono solo a formulare le tabelle di routing.

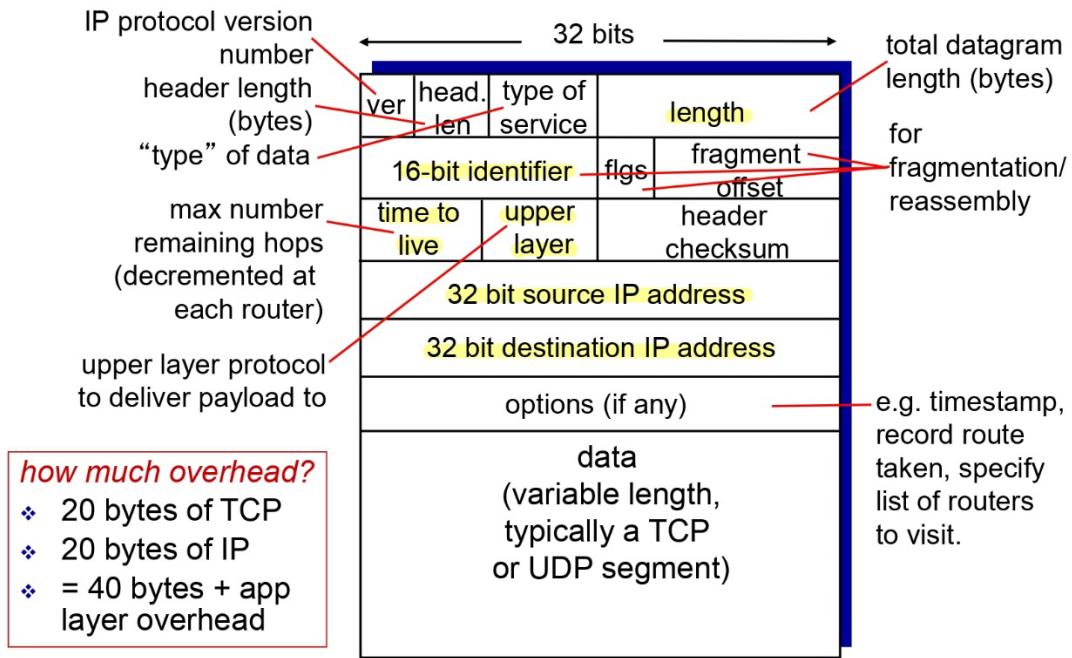
Il piano di controllo di uno switch è lo spanning tree: protocollo che serve per decidere quali collegamenti “spegnere” in modo tale da evitare percorsi chiusi.

## Lo strato di rete di Internet



Esempio di ICMP (protocollo di messaggistica): ping.

## Formato del datagramma IP



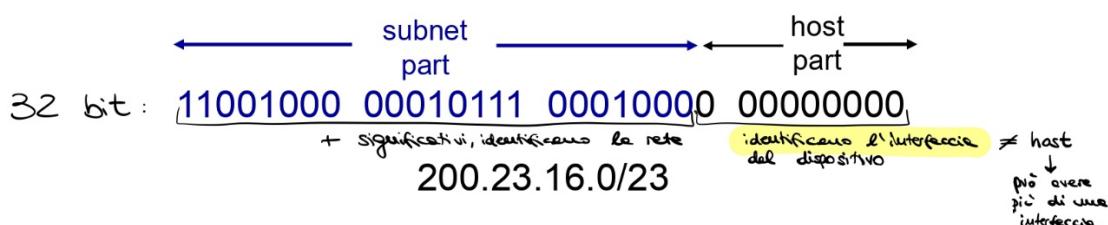
Sono importanti da ricordare i campi evidenziati.

## Indirizzamento IP: CIDR

Inizialmente per gestire gli indirizzi in internet veniva usate classi statiche, poi è stato visto che questa gestione era altamente inefficiente e quindi è stato definito un modo più fluido per definire la subnet, cioè la porzione di indirizzo che identifica la rete. Questo schema prende il nome di CIDR.

# CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
  - address format:  $a.b.c.d/x$ , where x is # bits in subnet portion of address



Agli host non possono assegnare la prima e l'ultima configurazione: cioè quando ho la host part tutta a 0 quello è il nome della rete e quando ho la host part tutta a 1 quello è l'indirizzo di broadcast (che permette di inviare lo stesso pacchetto a tutti gli indirizzi della rete).

## Longest prefix matching

Sulla base dell'organizzazione CIDR gli algoritmi di routing (che utilizzano la tabella di routing per definire l'interfaccia di uscita) effettuano questa procedura.

Longest prefix matching: quando si cerca la voce della tabella di routing per dare l'indirizzo di destinazione si utilizza il prefisso dell'indirizzo più lungo che corrisponde all'indirizzo di destinazione.

Destination Address Range	Link interface
11001000 00010111 00010**** *****	0
11001000 00010111 000110000 *****	1
11001000 00010111 00011**** *****	2
otherwise	3

Tipicamente le tabelle di routing hanno almeno 4 entry, esempio:

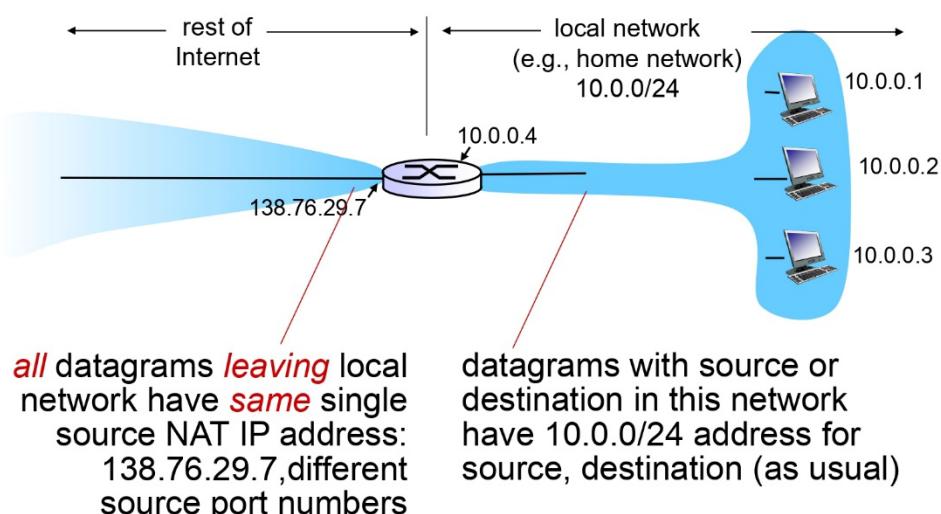
Destinazione/netmask	Gateway	Interfaccia
141.250.0.0/24	141.250.41.1	Etho

## NAT: network address translation

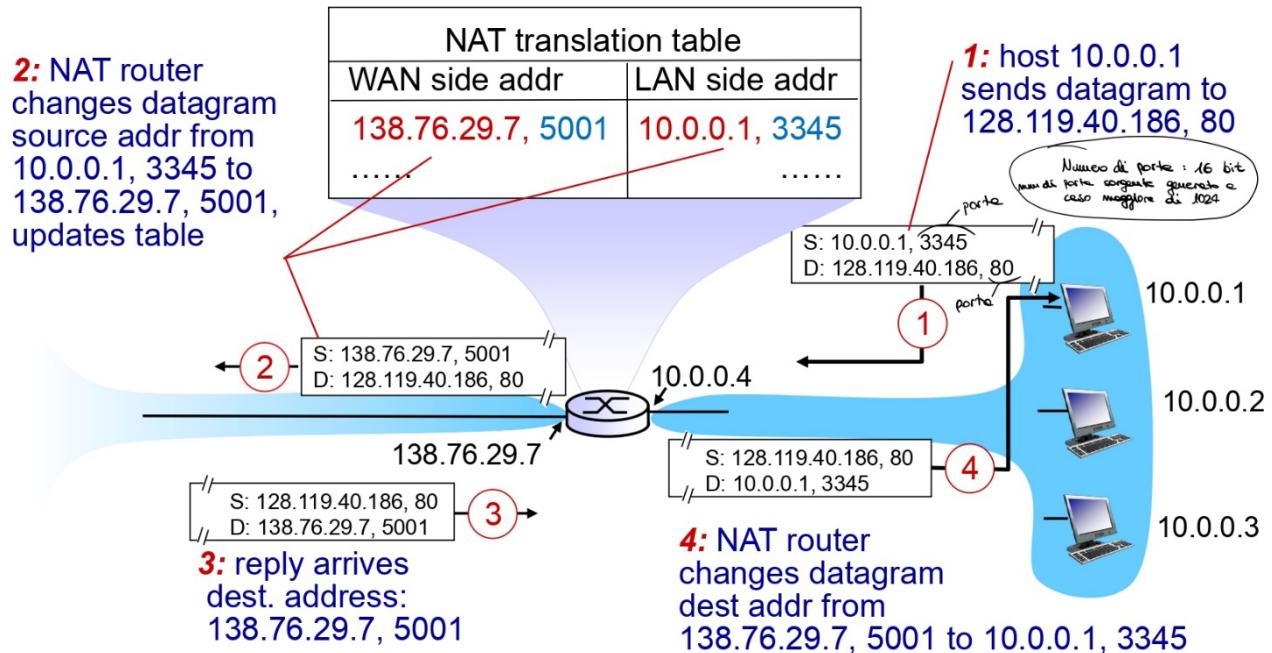
La funzionalità del NAT è quella di nascondere dietro un indirizzo pubblico (nell'immagine il 138.76.29.7) un insieme di indirizzi privati. Questo si fa per permettere alle organizzazioni di acquisire un indirizzo pubblico, di solito quello del proprio gateway, e di gestire con questo unico indirizzo una rete di più calcolatori.

Si è inventato il NAT per far fronte al rapido esaurimento degli indirizzi IP. Poi sono intervenuti due fattori:

- 1) Il NAT fornisce un livello di protezione più elevata rispetto a una configurazione piatta in cui gli indirizzi IP dei calcolatori sono esposti
- 2) Erogazione più oculata degli indirizzi IP visto che si stavano esaurendo



Il NAT effettua un'operazione di mascheramento, quindi è molto importante per chi osserva sapere se c'è un NAT e sapere chi lo gestisce. Ecco come funziona il NAT:



Il router ha un indirizzo pubblico a sinistra e un indirizzo privato a destra.

Ricordiamo che esistono 3 blocchi di indirizzi privati:

- 172.16
- 192.168
- 10

Il numero di porta mittente è un numero casuale maggiore di 1024. I numeri di porta sono di 16 bit. I primi 1024 numeri di porta sono assegnati ad indirizzi noti.

Se riesco ad accedere al pacchetto di livello applicativo e anche se vedo un unico dispositivo ma gli id all'interno dell'applicazione hanno nomi diversi ho fatto un'operazione di inferenza e ho avrò capito che quello in realtà è un NAT che “nasconde” dispositivi diversi. Questo andando ad ispezionare i protocolli applicativi; di solito però non è sempre possibile perché ci sono i protocolli che garantiscono la sicurezza.

Se si hanno a disposizione sufficienti indirizzi IP pubblici è chiaramente meglio evitare di mettere un NAT, visto che costituisce un collo di bottiglia a livello prestazionale.

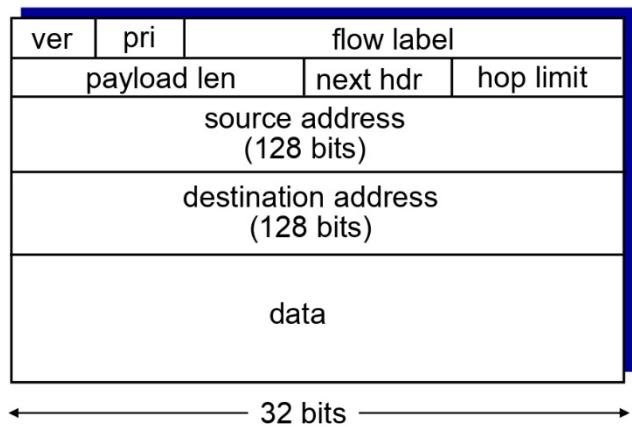
## Formato del datagramma IPv6

Priority: identifica la priorità tra i datagrammi nel flusso

Flow label: identifica i datagrammi nello stesso "flusso"

Next header: identifica il protocollo di strato superiore per i dati

L'intestazione di IPv6 è variabile, non la so dire a priori (l'intestazione di IPv4 è invece sempre di 20 byte)



## Indirizzi MAC e ARP

- Indirizzi di strato 4: 16-bit
- Indirizzi di strato 3 (indirizzi IP): 32-bit
  - indirizzo per l'interfaccia a strato di rete
  - usato per l'inoltro a strato 3 (strato di rete)
- Indirizzi di strato 2 (indirizzi MAC, o LAN o fisici o Ethernet): 48-bit
  - sono utilizzati localmente per ottenere frame da un'interfaccia a un'altra interfaccia fisicamente connessa
  - nella maggior parte delle reti LANs l'indirizzo MAC è scritto nella ROM della scheda di rete, a volte è anche impostabile tramite software
  - esempio: 6A-2F-BB-76-09-AD
    - notazione esadecimale (base 16) (ogni numero rappresenta 4 bits)
    - i primi 24 bits, cioè 3 bytes, identificano il costruttore/modello della scheda: OUI
    - gli altri 24 bits identificano la singola scheda

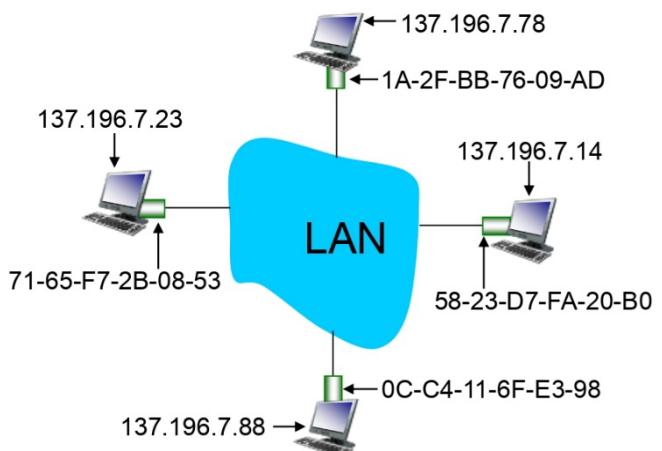
## ARP: Address Resolution Protocol

Il protocollo ARP serve per chiedere l'indirizzo MAC del dispositivo remoto quando devo inviare un pacchetto

Arp table: ogni nodo IP (host, router) sulla LAN ha una tabella

- indirizzo IP/MAC mappato per alcuni nodi della LAN:

<indirizzo IP; indirizzo MAC; TTL>



- TTL (Time To Live): tempo dopo il quale verrà dimenticata la mappatura per l'indirizzo (tipicamente 20 minuti)

Quando si conosce l'indirizzo IP e non quello MAC per consegnare un pacchetto viene utilizzato questo protocollo ARP: viene inviata una richiesta in broadcast alla rete chiedendo il MAC, dato l'IP; il dispositivo con quell'indirizzo IP risponderà in unicast indicando il proprio MAC e questo verrà segnato sulla tabella.

Lo switch ha un algoritmo di autoapprendimento con delle tabelle che si compilano da sole, diverse dalle tabelle di instradamento dello strato 3.

## Switches vs routers

Entrambi sono store-and-forward (cioè analizzano il datagramma prima sulla porta di ingresso e poi lo passano sulla porta di destinazione):

- Routers: dispositivi a strato di rete (esaminano le intestazioni a strato di rete)
- Switches: dispositivi a strato di collegamento (esaminano le intestazioni a strato di collegamento)

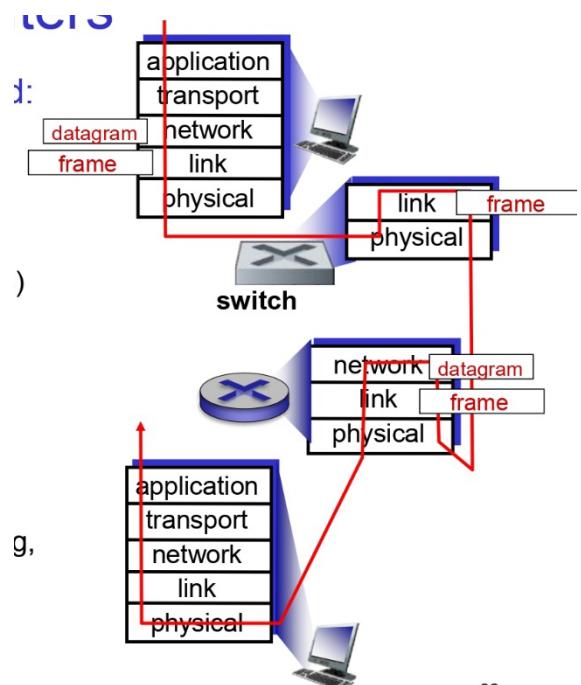
Entrambi hanno le tabelle di inoltro:

- Routers: completano le tabelle usando algoritmi di routing, indirizzi IP (3 modi per compilare le tabelle: instradamento, controllo remoto o utente umano)
- Switches: imparano le tabelle di inoltro usando il flooding, l'apprendimento, gli indirizzi MAC

Differenza:

- Routers: devono essere configurati
- Switches: si configurano da soli (nei casi base)

Differenza: i routers hanno su porte diverse indirizzi che appartengono a reti diverse; di fatto, i routers isolano gli indirizzi di broadcast (e di super broadcast, cioè tutti 1), mentre gli switches no.



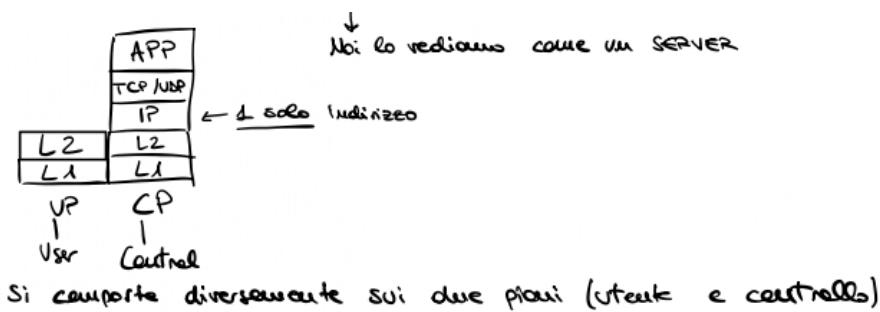
## Come si configura uno switch

Gli switch più costosi possono essere anche configurati.

Le VLAN (VirtualLAN) servono per creare più reti virtuali su una rete fisica e segregare il traffico su una certa porzione della rete di strato 2.

Le VLAN si creano configurando lo switch. Poi in ogni rete virtuale lo switch avrà il suo meccanismo di autoapprendimento.

Lo switch si configura mediante la separazione UserPlane e ControlPlane. Sul piano di utente sarà sempre un dispositivo di strato 2, mentre sul piano di controllo si comporta come se fosse un host.



Il piano di utente trasporta i nostri dati ed è quello che si va ad ispezionare facendo operazioni di network management; il piano di controllo è quello invece che si utilizza per configurare alcune operazioni di network management.

## Network management and SNMP

### Network Management (NM)

- Introduzione al network management: motivazione e componenti principali
- Soluzioni SNMP
- Evoluzione delle soluzioni di network managements:
  - NETCONF/RESTCONF
  - YANG
  - YAML/JSON/XML

### Cosa è il network management?

Il NM è la gestione di un particolare tipo di sistema autonomo (chiamato rete) composto da migliaia di componenti hardware/software che interagiscono tra loro.

Data la cardinalità degli elementi che interagiscono e l'eterogeneità di queste interazioni viene considerato un sistema complesso e ha quindi bisogno di opportune pratiche per essere monitorato e controllato.

Esempi di sistemi che necessitano monitoraggio e controllo: aereo, impianto nucleare, rete ferroviaria, ...

Definizione (non va imparata a memoria): “Il network management include la distribuzione, l'integrazione e il coordinamento di elementi hardware, software e umani per monitorare, testare, interrogare, configurare, analizzare, valutare e controllare la rete e le risorse elementari per soddisfare i requisiti di real-time, prestazioni operative e qualità di servizio ad un COSTO RAGIONEVOLE.”

Esempi di compiti NM da gestire:

- Rilevamento del guasto di una scheda di rete su un host, un router o uno switch
- Monitoraggio degli host e/o dei dispositivi di rete
- Monitoraggio del traffico per una corretta distribuzione delle risorse (un corretto dimensionamento del sistema)
- Rilevamento di cambiamenti rapidi nelle tabelle di routing
- Monitoraggio dei Service Level Agreements (SLAs), cioè i contratti che un operatore stabilisce con i propri clienti
- Rilevamento delle intrusioni

Più formalmente, l'ISO ha definito il modello del network management utile per collocare scenari aneddotici di cui sopra in un modo più strutturato.

L'ISO (l'organizzazione internazionale che definisce gli standard) ha raggruppato tutti questi compiti in una sigla che prende il nome di FCAPS (Fault management, Configuration management, Accounting management, Performance management, Security management).

- **Performance management:** l'obiettivo è quantificare, misurare, segnalare, analizzare e controllare le prestazioni dei diversi componenti di rete. Questi componenti includono singoli dispositivi e astrazioni end-to-end, come un percorso attraverso la rete.
- **Fault management:** l'obiettivo è registrare, rilevare e rispondere alle condizioni di guasto nella rete. Il confine tra fault management e performance management non è ben delineato. Si può pensare al fault management come alla gestione immediata di guasti temporanei della rete, mentre la performance management ha una visione a lungo termine per fornire livelli accettabili di prestazioni.
- **Configuration management:** consente di tenere traccia di quali dispositivi si trovano sulla rete gestita e delle configurazioni hardware e software di questi dispositivi.
- **Accounting management:** consente di specificare, registrare e controllare l'utente e il dispositivo di accesso alle risorse di rete. Gestisce quindi i permessi degli utenti (chi può accedere dove e a che cosa, a quale macchina e a quali applicativi o a quale macchina e a quale interfaccia) e assegna i privilegi di accesso alle risorse.
- **Security management:** l'obiettivo è controllare l'accesso alle risorse di rete secondo una politica ben definita. La distribuzione delle chiavi o l'utilizzo dei firewalls per monitorare e controllare i punti di accesso esterni alla propria rete sono due componenti fondamentali.

In realtà tutte queste sigle non rappresentano dei compartimenti stagni, ma sono fortemente interlacciate l'una con l'altra, ad esempio un problema di prestazioni potrebbe essere anche un problema di configurazione.

In generale, tutte le azioni che vanno sotto l'area del network management sono azioni a ciclo chiuso e la principale azione con cui inizia tutto è quella dell'osservazione del sistema. Una volta presi i dati possiamo analizzarli per capire se il sistema funziona come vogliamo oppure no. Nel caso funziona correttamente, lasceremo il sistema nello stato corrente e continueremo a monitorarlo; quindi, da Analize chiudiamo il cerchio passando su Observe, nel caso in cui, invece, lo stato del sistema o, meglio, l'elaborazione della nostra analisi ci dice che c'è qualcosa che non è in linea con le nostre aspettative (come: un guasto, un problema di accounting o prestazioni della sicurezza) è necessario impiegare un certo numero di azioni che dovrebbero risolvere questo problema. Successivamente, il ciclo si chiude, perché bisogna continuare a osservare il sistema per capire se le azioni che abbiamo messo in atto hanno effettivamente risolto il problema o hanno mitigato/peggiorato la situazione.

Negli ultimi anni si parla di AIops (Artificial Intelligence Operation), ovvero l'operatività del sistema guidata dall'utilizzo di intelligenza artificiale.

È un ciclo chiuso quello dell'intelligenza artificiale? No, dipende da caso a caso. In alcuni casi l'intelligenza artificiale può essere utilizzata solo per analizzare il sistema, in altri casi, anche per decidere quali sono le azioni da introdurre, dipende dai singoli casi d'uso, dalla tecnologia e dai dati che si hanno a disposizione. Sicuramente, l'intelligenza artificiale non viene utilizzata in fase di osservazione, anche se non è vietato utilizzare degli schemi avanzati, ad esempio per andare a campionare lo stato del sistema invece di fare un monitoraggio in modo continuo.

La prima vera e propria parte del corso si concentrerà sulla observation del sistema. Un sistema viene detto osservabile se si riesce a stimare in maniera completa lo stato del sistema, a partire dall'osservazione delle sue uscite. Sulla base dell'uscita possiamo cambiare gli ingressi per pilotare il sistema come vogliamo, per ottenere le uscite desiderate, questo perché un sistema osservabile è anche controllabile, cioè, c'è proprietà di dualità.

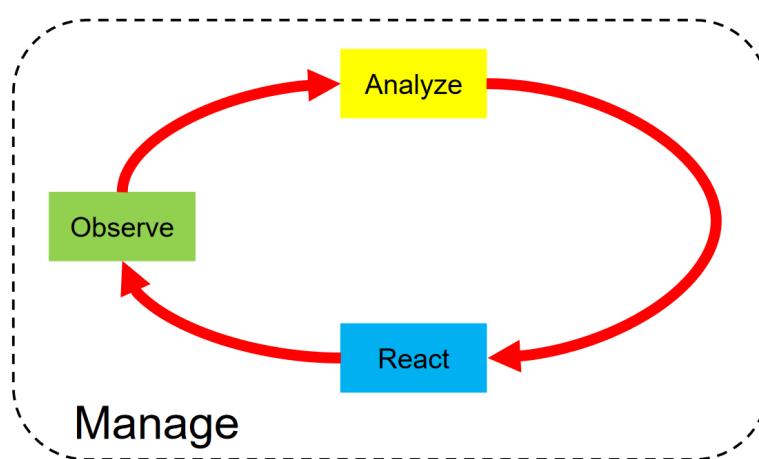
Una rete però non è lineare e non è invariante. Dunque, possiamo permetterci di osservare solo le uscite del sistema? E soprattutto, che cosa intendiamo per uscita di una rete? Inizialmente, si può pensare a tutti i servizi ma la rete è divisa in livelli protocollari. Quindi i servizi di quali livelli protocollari andiamo ad esaminare? Anche qui non esiste una risposta, caso per caso bisogna andare a capire cosa osservare e quali tecnologie utilizzare per effettuare questa osservazione, anche se dei principi di guida generali esistono. Cominciamo con il dire che per osservare il sistema ho bisogno di piazzare delle sonde, dispositivi hardware, software o entrambi che permettono di estrarre dei dati dal sistema. Dove bisogna posizionare queste sonde? Dipende dal nostro problema e dipende da quello che noi andiamo a osservare.

**Esempi:** un fornitore di servizi cloud, dovrà andare a monitorare lo stato di salute dei container o delle macchine virtuali che andrà a vendere o in generale di quello che la piattaforma mette a disposizione agli utenti. Se invece ospita servizi web, dovrà

andare a monitorare che i server in servizio siano effettivamente funzionanti e che le risposte alle richieste che vengono dall'esterno siano erogate in tempi ragionevoli. Se sono un operatore di rete devo andare a osservare il mio sistema per assicurarmi che le rotte che vengono utilizzate all'interno della mia rete siano quelle che mi permettono di minimizzare i costi e che non ci siano percorsi chiusi in cui le informazioni vengono perse.

Esistono vari tipi di modalità di osservazione e tutti dipendono dallo strato protocollare in cui noi andiamo a osservarle. La cosa fondamentale è che, quando parliamo di network management non ci possiamo solo concentrare sull'uscita del sistema. Quindi, ad esempio, se io sto operando un'infrastruttura di rete, non posso vedere solo se i servizi funzionano bene (navigazione web, posta elettronica, messaggistica) ma dovrò anche andare a controllare lo stato di salute dei miei dispositivi. Per quanto riguarda l'analisi, dipende da quali dati abbiamo a disposizione e dal tipo di dato (log in formato testuale, valori numerici). Queste informazioni servono a capire quale tecnica di intelligenza artificiale dobbiamo utilizzare.

**Esempio:** se dobbiamo analizzare i log di un sito web in cui è scritto quale oggetto del sito web è stato recuperato, a che ora, da quale indirizzo IP e da quale browser, dovrò utilizzare delle tecniche che sono diverse dal monitoraggio dello stato di salute di un router in cui andò a vedere se tutte le interfacce sono accese, se tutte le ventole funzionano correttamente.



## Struttura del Network Management

La struttura classica del network management si basa sul concetto di gestore o entità di gestione e dispositivo gestito, dove il dispositivo **non è necessariamente un componente hardware**; quindi un dispositivo può essere anche un database.

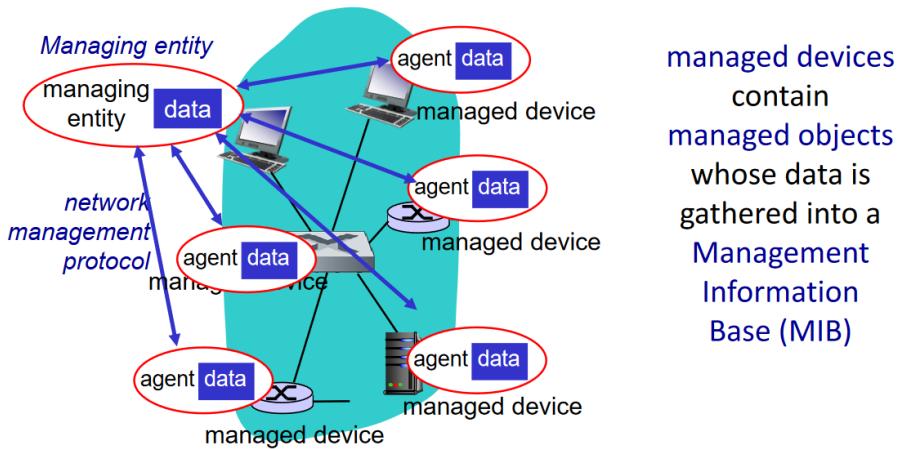
In questa figura, abbiamo: dei dispositivi, dei PC, un server in basso a destra, uno switch al centro e un router al centro a destra; ognuno dei quali mette a disposizione

un certo numero di informazioni. Nel protocollo **SNMP** (Simple Network Management Protocol) che è l'infrastruttura di base per l'osservazione di un sistema (**solo per l'osservazione e in minima parte per la reazione, cioè per cambiare la situazione del sistema**), le informazioni sono organizzate in oggetti. Un dispositivo gestito contiene un certo numero di oggetti e questi oggetti contengono i propri dati che vengono prodotti, aggiornati e mantenuti all'interno dell'oggetto stesso in un database che prende il nome di MIB (Management Information Base). Per osservarlo ho bisogno di un'entità di gestione che mi permetta di recuperare questi oggetti e li mantenga anch'esso in un MIB locale che però rappresenta il più possibile lo stato della mia rete. Affinché io possa recuperare i dati relativi agli oggetti ho bisogno che sui dispositivi gestiti vengano eseguiti dei componenti software specifici che prendono il nome di **Agenti**. Questi agenti vengono interrogati da un protocollo di gestione e il risultato viene raccolto e gestito in una entità di gestione. Quindi ho un insieme distribuito di informazioni che tipicamente sono gestite in modo più o meno diverso da vari agenti che devo interrogare per recuperare questi dati.

La modalità di gestione dipende dal protocollo che utilizzo e da come è strutturato il MIB. Se tutti i dispositivi sono compatibili con il protocollo SNMP, utilizzerò il protocollo SNMP per andare a recuperare questi dati e per memorizzarli in un MIB centralizzato. Se invece alcuni di questi dispositivi utilizzano SNMP e altri dispositivi non lo utilizzano oppure (**caso più comune**) parte dei dati non sono tra quelli gestiti da SNMP e quindi ho bisogno di meccanismi aggiuntivi per catturare e gestire i dati, allora avrò bisogno di un'infrastruttura eterogenea che sia in grado di recuperare dati potenzialmente eterogenei, aggregarli, normalizzarli, memorizzarli, analizzarli e così via.

## Infrastructure for network management

definitions:



## Standard del Network Management

Il primo tentativo di mettere in piedi un protocollo per la gestione dei dispositivi di Internet prende il nome di CMIP (Common Management Information Protocol). Questo protocollo ha fatto la fine della pila protocollare ISO-OSI, cioè è stata inglobata da quella TCP, perché più semplice. Il protocollo che si è imposto è il protocollo SNMP, dove la S sta per **Simple**. Esso, non viene utilizzato solo per il network management ma viene utilizzato in generale per la gestione di dispositivi in rete, quindi anche per la gestione dei servizi. SNMP è utilizzato tutt'ora, ad esempio, viene impiegato per interrogare lo stato delle stampanti o di altri dispositivi di rete. Nel tempo però è diventato un pochino più complesso, data la mancanza di estensioni di sicurezza, aggiunte nella versione corrente che è la 3 ed è lo standard per il network management.

Il protocollo SNMP è semplice perché è un protocollo senza connessione e quindi viene trasportato dal protocollo di trasporto UDP. Questo è il secondo macro- esempio che vedete tra protocolli importanti che viaggiano non su TCP ma UDP. L'altro è ovviamente il DNS su cui si basa tutto il funzionamento di Internet. Quando parliamo di SNMP, non parliamo solo del protocollo, ma anche della struttura del database, ovvero la base di dati dove vengono mantenute le informazioni di network management. Le informazioni che vengono mantenute sono particolarmente eterogenee, di solito sono: stringhe o interi. Le stringhe, ad esempio, sono utilizzate per identificare le informazioni di stato come: acceso, spento, caratteristiche di una ventola, un contatore, la versione software).

---

## Network management standards

### OSI CMIP

- Common Management Information Protocol
- designed 1980's: *the* unifying net management standard
- too slowly standardized

### SNMP: Simple Network Management Protocol

- Internet roots (SGMP)
  - Simple gateway monitoring protocol
- started simple
- deployed, adopted rapidly
  - growth: size, complexity
- currently: SNMP v3
  - de facto* network management standard

## MIB, SMI e Sicurezza

Possono esserci diverse tipologie di informazioni, alcune riguardano la configurazione e altre riguardano lo stato corrente. Il linguaggio utilizzato per definire come devono essere strutturati gli oggetti all'interno del MIB è **SMI** (struttura dell'informazione di gestione) che non è un componente di SNMP. L'obiettivo di SMI è quello di definire la semantica e la sintassi dei dati in maniera non ambigua. Quello che ne esce fuori è qualcosa di semplice, ma verboso e ridondante. All'interno di ogni oggetto avremo: il tipo di dato, uno stato e la semantica (significato dell'oggetto stesso).

## SNMP overview: 4 key parts

### Management information base (MIB):

- distributed information store of network management data
  - Counters, state information, software version, etc

### Structure of Management Information (SMI):

- data definition language for MIB objects

### SNMP protocol

- convey manager<=>managed object info, commands

### Security and administration capabilities

- major addition in SNMPv3

Un altro concetto molto importante è il **modulo**, oggetti simili vengono raggruppati all'interno di un unico modulo e all'interno di un dispositivo ci possono essere più moduli.

I tipi di dati di base sono gli interi che possono essere a 32 bit con o senza segno, le stringhe, gli identificativi e altri tipi di dati più particolari. Ad esempio: gli indirizzi IP (che sono gestiti come indirizzi IP e non come stringhe, quindi 32 bit), i contatori (che possono essere a 32 o 64), i cruscotti e poi il tipo di dato opaco, all'interno del quale si può scrivere qualunque cosa e il suo significato non è noto a priori a SMI. Dunque, all'interno del MIB, avremo uno o più moduli che avrà un suo identificativo e all'interno di ogni modulo ci sono un certo numero di oggetti che sono simili per semantica, cioè che hanno significati simili.

# SMI: data definition language

## Purpose:

- syntax, semantics of management data well-defined, unambiguous
- base data types:
  - straightforward, boring
- OBJECT-TYPE
  - data type, status, semantics of managed object
- MODULE-IDENTITY
  - groups related objects into MIB module

## Basic Data Types

INTEGER
Integer32
Unsigned32
OCTET STRING
OBJECT IDENTIFIER
IPaddress
Counter32
Counter64
Gauge32
TimeTicks
Opaque

Di seguito, viene mostrato un esempio di protocollo IP. Il modulo è quello di IP e l'oggetto è il numero **ipInDelivers**. Ogni oggetto ha: la descrizione del suo tipo (in questo caso un contatore a 32 bit), le operazioni che posso fare sull'oggetto (in questo caso, `readOnly`, ovvero, posso leggere solo senza poter modificare), lo stato e una descrizione informale che definisce l'oggetto.

La notazione **ip 9** in basso a sinistra ci dice che l'oggetto `ipInDelivers` fa parte del modulo IP e che nella gerarchia del database è il nono oggetto. Nel modulo invece abbiamo le informazioni su chi ha creato il modulo, l'ultima volta che è stato aggiornato, una descrizione e la revisione del modulo. Anche in questo caso abbiamo una notazione di tipo gerarchica, che è **mib:2 48**, quindi è il quarantottesimo modulo del MIB 2.

### OBJECT-TYPE: ipInDelivers

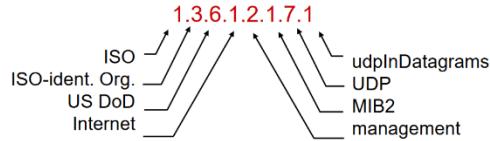
ipInDelivers OBJECT TYPE  
SYNTAX Counter32  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
“The total number of input datagrams successfully delivered to IP user-protocols (including ICMP)”  
::= { ip 9 }

### MODULE-IDENTITY:

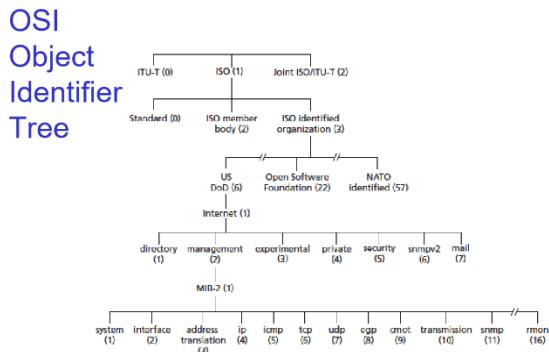
ipMIB  
ipMIB MODULE-IDENTITY  
LAST-UPDATED “941101000Z”  
ORGANIZATION “IETF SNPv2 Working Group”  
CONTACT-INFO  
“ Keith McCloghrie .....

DESCRIPTION  
“The MIB module for managing IP and ICMP implementations, but excluding their management of IP routes.”  
REVISION “019331000Z”  
.....  
::= {mib-2 48}

Per ciascun oggetto del modulo abbiamo un identificativo numerico. Questo identificativo numerico è standardizzato da un ente sovranazionale che si chiama ISO (International Standard Organization) e ha una struttura che rappresenta la navigazione all'interno di un albero.



Quindi, si ha la possibilità, attraverso SNMP, di interrogare il modulo sistema che contiene esclusivamente le informazioni generiche del nostro sistema, oppure le statistiche per singola interfaccia o le statistiche per protocollo.



## Funzionamento Protocollo SNMP

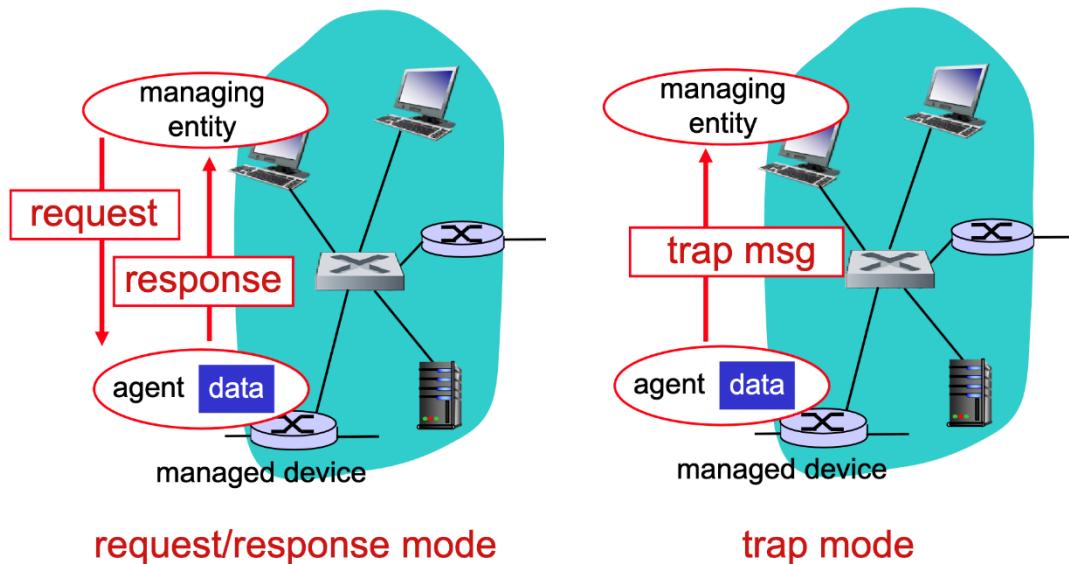
Il protocollo SNMP offre due modalità di interazione tra il gestore e gli agenti: **request-response** e **notify**. La prima modalità è quella classica in cui il gestore chiede informazioni e l'agente una volta recuperata l'informazione la invia con la risposta, la seconda utilizza la **trap**.

L'uso della prima modalità, comporta che ogni qualvolta che il gestore vuole un'informazione, deve chiederla e quindi avrà bisogno di due messaggi. Se si volesse monitorare un determinato fenomeno potrei dover inviare un numero molto elevato di messaggi per andare a intercettare quando quel fenomeno avviene o quando quel fenomeno sta accadendo.

Per evitare di intasare la rete con molte richieste SNMP, posso lanciare una trappola (trap), ovvero, un evento con una condizione che, quando verificata deve inviare un messaggio. La condizione da verificare può essere anche quella di analizzare un semplice time e in questo caso invece di richiedere il messaggio dello stato del mio sistema una volta ogni minuto, istanzio la trap e sarà lei che una volta al minuto mi fornirà l'informazione.

# SNMP protocol

Two ways to convey MIB info, commands:



## Tipi di Messaggio (SNMP)

1. **GetRequest:** Messaggio più semplice, va dal manager (Mgr) all'agente;
2. **GetNextRequest, GetBulkRequest:** servono per inviare richieste in continuazione una dietro l'altra, anche qui si va da gestore ad agente gestito;
3. **InformRequest:** permette ai gestori di inviarsi dei messaggi;
4. **SetRequest:** messaggio per configurare il sistema, sempre da Mgr ad agente. Esempio: voglio cambiare il nome del sistema, voglio cambiare il valore di una soglia di un algoritmo di instradamento;
5. **Response:** tipicamente va da agente a gestore. Il gestore invia la richiesta GetRequest e l'agente invia la response;
6. **Trap:** notifica che viene inviata nel caso in cui si verifichi una determinata condizione, in questo caso da agente a gestore.

# SNMP: message types

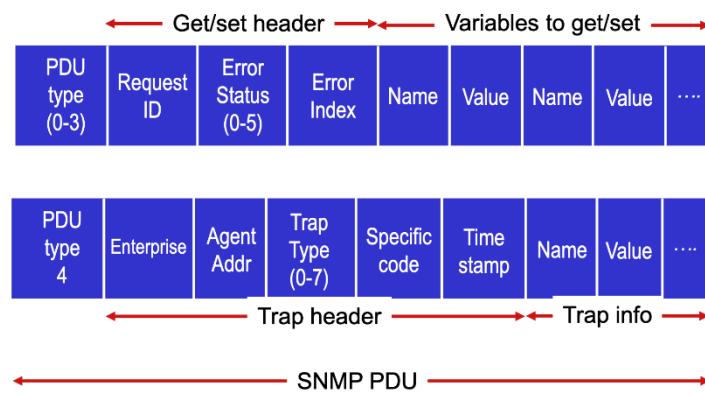
<u>Message type</u>	<u>Function</u>
GetRequest	Mgr-to-agent: “get me data”
GetNextRequest	(instance, next in list, block)
GetBulkRequest	
InformRequest	Mgr-to-Mgr: here’s MIB value
SetRequest	Mgr-to-agent: set MIB value
Response	Agent-to-mgr: value, response to Request
Trap	Agent-to-mgr: inform manager of exceptional event

## Formato dei Messaggi (SNMP)

Nella prima parte del messaggio SNMP troviamo l'intestazione del pacchetto. All'interno dell'intestazione abbiamo dei campi per capire se è una richiesta o una risposta e poi ci sono un numero variabile di variabili get/set, perché con un messaggio posso chiedere più informazioni.

Nello schema della Trap invece abbiamo un'intestazione più articolata, incluso il tipo di trap, le informazioni che costituiscono il nome e il valore dell'oggetto che viene monitorato.

## SNMP: message formats



## Sicurezza e Amministrazione

Poiché non sono state definite delle estensioni di sicurezza per il protocollo UDP, possiamo applicare e implementare soluzioni all'interno del protocollo applicativo stesso (ovvero nel SNMP).

- **Cifratura:** effettuata tramite un meccanismo a chiave comune, dove il mittente e il destinatario possono condividere la stessa chiave. Lo standard prevede l'utilizzo della cifratura DES che non ha un livello di sicurezza molto elevato.
- **Autenticazione:** Per la protezione e l'integrità dei messaggi viene utilizzato il MIC, un hash crittografico. Al pacchetto M viene agganciato un hash crittografico che dipende non solo dal contenuto del pacchetto ma anche dalla chiave segreta relativa all'hash.
- **Protezione contro attacchi in playback:** gli attacchi di playback avvengono quando uno stesso pacchetto viene inviato più volte per forzare il sistema ad avere lo stesso comportamento. Sono degli attacchi che possono dare origine a situazioni di Denial Of Service, dato che una qualsiasi entità potrà ricevere migliaia di richieste e produrre migliaia di risposte, sovraccaricando il sistema. Per proteggersi da questi attacchi bisogna rendere univoca ogni richiesta per far sì che la cattura di una richiesta, anche cifrata, non possa essere copiata e inviata nuovamente su un nuovo pacchetto in quanto il sistema si accorgerebbe che quella richiesta è già stata evasa e quindi non darà luogo ad una risposta. Questo meccanismo sfrutta però il numero sequenza, caratteristica tipica dei protocolli con connessione. Dato che SNMP è senza connessione al posto del numero di sequenza utiliziamo un contatore, basato sull'ultima volta in cui il software del ricevitore è stato riavviato (ovviamente non sarà estremamente preciso perché dipende dall'orologio del ricevitore).
- **Controllo di accesso:** c'è un database all'interno del MIB, in cui è presente un elenco degli utenti che possono accedere.

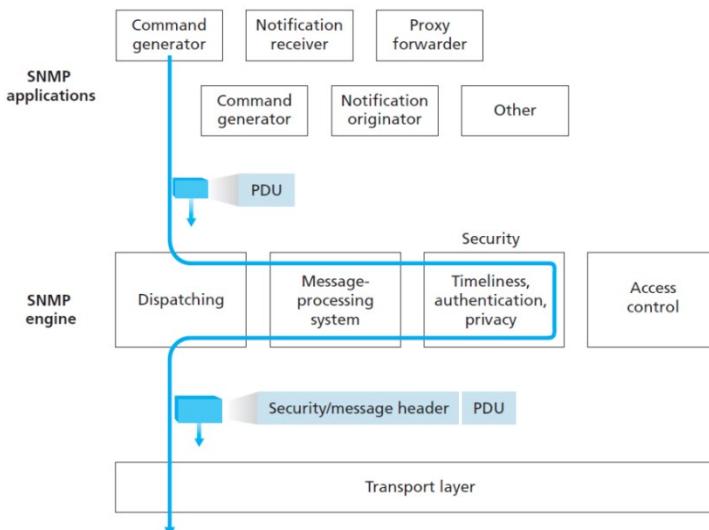
# SNMP security and administration

- **encryption**: DES-encrypt SNMP message
  - DES is a shared key system
- **authentication**: compute, send MIC(m,k):
  - compute hash (MIC) over message (m) using secret shared key (k)
- **protection against playback**: use nonce
  - sender includes a value in each message based on a counter in the receiver, reflecting the time since the last reboot of the receiver's NM SW
- **view-based access control**:
  - SNMP entity maintains database of access rights, policies for various users
  - database itself accessible as managed object!

23

Come possiamo notare tutte queste misure sono abbastanza deboli: nella cifratura viene utilizzato il DES, l'autenticazione non prevede la cifratura dell'hash ma semplicemente la generazione di un hash concatenando il messaggio con la chiave, la protezione contro un attacco di playback utilizza un'informazione che può essere dedotta e infine il controllo di accesso viene basato sull'identità ma senza particolari misure di sicurezza.

Il meccanismo di base su come funziona l'estensione di sicurezza di SNMP è il seguente:

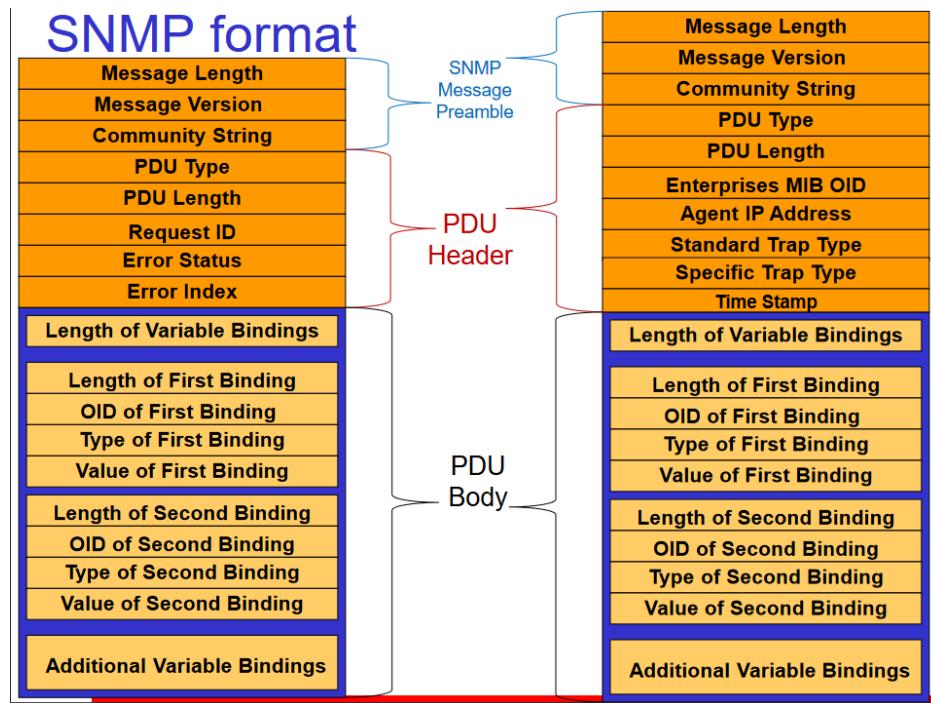


In alto c'è la comunicazione SNMP (quella del gestore). Ad esempio, si invia un comando di GetRequest e viene generata una PDU che entra all'interno del protocollo

SNMP e a cui verranno aggiunte le varie caratteristiche di sicurezza. Una volta aggiunte viene inviato al protocollo di trasporto UDP.

## FORMATO PACCHETTI SNMP

Nel dettaglio i pacchetti SNMP hanno: una parte in comune che è quella che ci dice quando è lungo il messaggio, la versione del protocollo, un'altra informazione comune (la cosiddetta community string) e poi ci sono le informazioni che dipendono dal fatto che il messaggio può essere di richiesta/risposta (SINISTRA) oppure una trap (DESTRA). Infine, ci sono i vari oggetti dentro.



## PROBLEMA DI PRESENTAZIONE

Dopo aver definito come gli oggetti sono memorizzati all'interno del database dell'SNMP, bisogna definire come questi messaggi vengono trasportati dentro il protocollo perché chiaramente all'interno del messaggio SNMP non avrò tutta quella struttura verbosa che ho dentro il MIB. Quindi bisogna risolvere il problema di presentazione, ovvero come trasportare e organizzare i miei dati.

Possibili soluzioni:

- Il mittente apprende il formato del destinatario, traduce il messaggio nel formato del destinatario e lo invia;
- Il mittente invia il messaggio. Il destinatario apprende il formato del mittente e traduce nel formato locale.
- Il mittente traduce il formato indipendente dall'host e si invia il messaggio. Il destinatario traduce nel formato locale.

## FORMATO ASN.1

Il formato ASN.1 (Abstract Syntax Notation) è uno standard ISO utilizzato nei messaggi che viaggiano in rete e definisce i dati nei costruttori degli oggetti, come l'SMI utilizzato nel MIB, ma in maniera molto meno verboso.

L'ASN definisce il cosiddetto BER (Basic Encoding Rules) che ci dice come devono essere formattare le informazioni che viaggiano in rete. In particolare, questo standard è adottato dal protocollo SNMP. Si utilizza la codifica TLV (Tipo-Lunghezza-Valore) per capire quando un blocco di un pacchetto SNMP inizia e finisce.

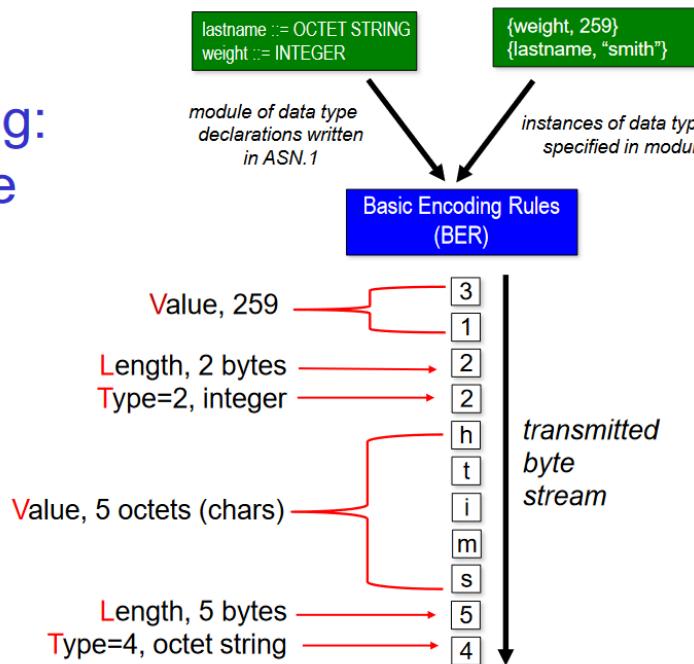
## TLV Encoding

*Idea:* transmitted data is self-identifying

- T: data type, one of ASN.1-defined types
- L: length of data in bytes
- V: value of data, encoded according to ASN.1 standard

Tag	Value	Type
1		Boolean
2		Integer
3		Bitstring
4		Octet string
5		Null
6		Object Identifier
9		Real

# TLV encoding: example



Abbiamo capito che il Network Management è molto importante perché ci permette di ottimizzare la gestione di sistemi molto complessi e di ottimizzare i costi.

Inoltre, non esiste una regola universale per andare a configurare un sistema di gestione.

## Network management

- extremely important: 80% of network “cost”
- ASN.1 for data description
- SNMP protocol as a tool for conveying information

## Network management: more art than science

- what to measure/monitor
- how to respond to failures?
- alarm correlation/filtering?

In generale il Network Management è più artigianale che una scienza vera e propria, perché di volta in volta bisogna definire che cosa monitorare e cosa misurare. Un altro aspetto importante del NW è come rispondere ad un problema durante un’analisi, che può essere ovviamente di varia natura (dispositivo che si rompe, una presa elettrica

staccata per sbaglio o altro). In questi casi è importante capire come identificare e come estrarre le informazioni dai dati che mi permettono di identificare il problema, e bisogna correlare i vari allarmi per capire il più rapidamente possibile qual è il problema.

Un grande limite del protocollo SNMP e di altri protocolli simili è la possibilità di poter collezionare solo statistiche aggregate basate su contatori, ovvero, informazioni binarie (come: acceso, spento, funzionante, non funzionante). Nonostante i contatori siano molto importanti, non riescono a darci una serie di informazioni che possono esserci utili, per esempio, nel caso di un attacco hacker.

Inoltre, SNMP che è basato su UDP, dal punto di vista della tempistica abbiamo tempi di risposta abbastanza blandi. Di norma il monitoraggio che viene fatto è nell'ordine dei secondi o dei minuti, inoltre, bisogna stare attenti, perché dobbiamo bilanciare la necessità di osservare il sistema con quella di evitare il sovraccarico dato dal traffico di osservazione.

Infine, SNMP è di tipo passivo. Non prevede il monitoraggio attivo della rete, quindi, vado a interrogare un dispositivo senza testarlo. Esistono invece altri tool che prevedono il monitoraggio attivo della rete come **traceroute**, **ping** o **nmap**.

## Open issues

- Only passive measurements are possible with those solutions
  - SNMP and similar protocols can collect only aggregate statistics based on counters
  - It is not suitable for true real-time monitoring
    - large time scales, order of seconds
- Use well-known networking tools for active measurements
  - Ping, traceroute, nmap
  - Limited extent/information

## NETCONF e RESTCONF

NETCONF e RESTCONF sono due paradigmi nati per la configurazione della rete ma che possono essere utilizzati anche per recuperare informazioni sullo stato della rete.

La principale differenza tra i due paradigmi è il modo in cui interagiscono con l'agente remoto, mentre dal punto di vista dell'architettura di sistema abbiamo sempre un meccanismo di richiesta e risposta.

In SNMP la notifica è chiamata **trap**, mentre in altri protocolli prende tipicamente il nome di **notifica** o **sottoscrizione**. Avremo quindi un'entità centralizzata che raccoglie i dati e per quanto riguarda NETCONF/RESTCONF si occupa anche di pushare le configurazioni.

Per quanto riguarda i protocolli utilizzati dai due paradigmi cambia solo il protocollo utilizzato per interfacciarsi con l'AGENT (protocollo a strato applicativo): NETCONF utilizza un protocollo RPC, quindi la ME è il cliente che chiede di eseguire la funzione su un server remoto che gira sull'AGENT (SNMP lavora allo stesso modo). Per quanto riguarda l'interfaccia/API orientata al paradigma REST, l'accento non è posto sull'esecuzione di una funzione remota, ma piuttosto sul modificare una risorsa remota tramite operazioni di tipo CRUD.

### Operazioni CRUD

C -> Create Metodo POST

R -> Read Metodo GET

U -> Update Metodo PUT

D -> Delete Metodo DELETE

Quindi un API realizzata seguendo i dettami del REST le operazioni fondamentali sono le 4 appena viste e vengono eseguite su risorse remote. L'API REST si appoggia su protocollo HTTP.

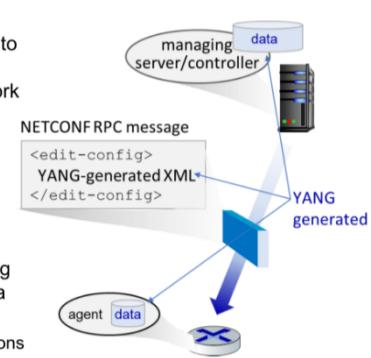
Quindi la differenza principale tra NETCONF/RESTCONF e SNMP non è lo schema di base, in quanto entrambi dispongono dei metodi **Request/Response** e **Notify**, ma piuttosto il protocollo utilizzato: SNMP usa SNMP mentre sia RPC(NETCONF) sia REST (RESTCONF) utilizzano HTTP/HTTPS.

Un ulteriore differenza riguarda la memorizzazione dei dati:

- **SNMP:** i dati vengono memorizzati internamente in base al formato SMI che definisce la struttura dei dati in moduli composti da oggetti. Un modulo tipicamente è un'astrazione di livello superiore: può essere un'interfaccia, un protocollo o un pezzo di un nostro componente.
- **NETCONF/RESTCONF:** utilizzano il protocollo YANG come formalismo per la memorizzazione dei dati.

### YANG

- Data modeling language used to specify structure, syntax, semantics of NETCONF network management data
  - built-in data types, like SMI
- XML document describing device, capabilities can be generated from YANG description
- Can express constraints among data that must be satisfied by a valid NETCONF configuration
  - ensure NETCONF configurations satisfy correctness, consistency constraints



Per quanto riguarda il trasferimento invece:

- **SNMP**: utilizza il protocollo SNMP e i dati sono formattati secondo il formato ASN.1 o Abstract Syntax Notation 1;
- **NETCONF/RESTCONF**: utilizzano tipicamente il protocollo HTTP/HTTPS e i dati sono formattati attraverso XML oppure JSON.

Viste le considerazioni precedenti è evidente che NETCONF/RESTCONF utilizzano protocolli più maneggevoli.

In SNMP ho sempre a che fare con gli OID, che sono standardizzati ISO ma rimangono comunque una tecnologia proprietaria. In altre parole, mentre ci sono delle parti dell'albero che sono comuni a tutti, quindi per esempio tutti i dispositivi che utilizzano il protocollo UDP avranno gli stessi oggetti, ovvero un modulo comune UDP e gli oggetti che rappresentano i parametri UDP di SNMP. Che cosa succede se invece vado considerare un router della Cisco di uno specifico modello? I costruttori sono autorizzati ad aggiungere parti dell'albero delle gerarchie, ci saranno quindi delle parti "private" che dipendono dal singolo costruttore. Quindi ogni azienda manifatturiera è autorizzata, non solo ad utilizzare gli OID già definiti, ma ne può aggiungere di nuove, quindi se per esempio Cisco vuole aggiungere un nuovo campo per il monitoraggio della rete è libra di farlo.

Quindi come fa un amministratore di rete se vuole visualizzare questo nuovo campo?

Deve disporre di una libreria aggiornata contenente gli OID di quel dispositivo di quel costruttore, in quanto il costruttore può decidere di aggiungere degli OID solo per alcuni modelli di dispositivo.

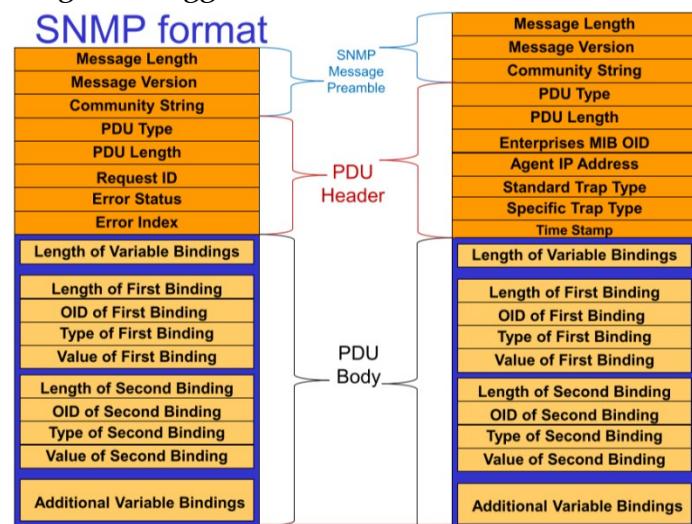
Di conseguenza per interpretare i codici presenti nei messaggi SNMP abbiamo necessariamente bisogno di avere le librerie degli OID aggiornate.

Il corpo dei messaggi SNMP è composto da una struttura che si ripete in cui sono presenti i campi TLV e OID.

Volendo astrarre a più alto livello: OID ci fornisce la semantica mentre il TLV ci dà la sintassi.

SNMP sfrutta un approccio non nuovissimo a differenza di NETCONF/RESTCONF che si appoggiano tipicamente al protocollo HTTP, risultando più maneggevoli e

potenti. Nonostante esista quindi più software per questi ultimi, non esistono grandi basi di dati come per SNMP.



La cosa interessante del RESTCONF è la flessibilità nella formattazione dei dati, che non necessariamente devono essere memorizzati in formato XML, il quale risulta essere piuttosto verboso anche se facilmente parsabili dalla macchina e leggibile dall'uomo. È più semplice invece utilizzare il formato JSON quando si utilizza l'approccio RESTCONF, questo tipo di formato ci permette di modificare oggetti remoti tramite le 4 operazioni fondamentali CRUD del REST. Inoltre, il formato JSON, oltre ad essere più leggero risulta anche molto più leggibili in quanto non ha il meccanismo verboso dei tag di apertura e chiusura caratteristico di XML.

Principi del paradigma **RESTful** (Representational State Transfer):

- **Client-Server** -> Interazioni basate su meccanismo Request-Response
- **Interfacce uniformi** -> per accedere alle risorse in maniera non ambigua tramite URI (o Uniform Resource Identifier)
- **Transazioni Stateless** -> Ogni transazione è atomica, ovvero inizia e finisce con la transazione stessa.
- **Cacheable** -> Dati ricevuti possono essere memorizzati dai client per evitare di richiederle in continuazione
- **Layered System** -> Un sistema può essere diviso in livelli, il livello frontale nasconde la complessità di tutti gli altri livelli (principio di base su cui si basano i sistemi web)

Un sistema REST può utilizzare i dati in formato JSON e XML. Il JSON è basato su una struttura chiamata

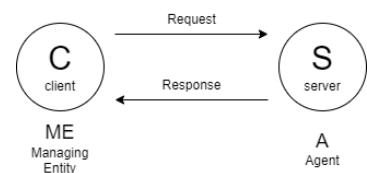
Documento, al cui interno abbiamo una struttura chiave valore in cui al valore può essere associato un numero, una stringa, un vettore o altri json, con la possibilità di avere una struttura annidata e facilmente leggibile.

### The JSON language

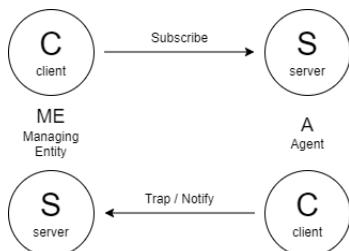
```
{  
    "givenname": "Mario",  
    "lastname": "Rossi",  
    "age": 45,  
    "employed": true,  
    "salary": 1200.00,  
    "phones": ["0243434", "064334343"],  
    "office": [  
        {"name": "A", "street": "Zamboni", "number": 7},  
        {"name": "B", "street": "Irnerio", "number": 49}  
    ]  
}
```

### Cosa accomuna questi due tipi di approccio (SNMP vs RESTCONF/NETCONF)?

Questi approcci sono non soltanto passivi, ma è anche possibile interrogare o farsi notificare dal sistema, statistiche di tipo aggregato, tramite uno schema Client-Server invertito.



Nel caso di Request-Response la ME o (Managing Entity) è il client e A (Agente) è il server.



Invece nel caso in cui si utilizza il meccanismo di trap notify si invertono i ruoli, la ME assume prima il ruolo di Client e A di server quando si sottoscrive (subscribe) una notifica o si installa una trap sul A. Da questo momento in poi si invertono i ruoli: ME diventa server e A Client. Così avremo le Trap/Notifiche lanciate dall'agente verso la ME.

Sia SNMP che i paradigmi NETCONF/RESTCONF supportano entrambi le tipologie di comunicazione, quindi di fatto i componenti software per il Network Management devono implementare sia la componente Client che la componente Server.

Quindi meccanismi simili, implicano limitazioni simili. Posso quindi ricevere informazioni di stato (principalmente binarie) oppure posso avere dei contatori aggregati. Entrambi questi approcci sono basati su una struttura che lega gli oggetti che devo modificare/monitorare. Quindi ho sempre un MIB (una base di dati) con una struttura dati che è nota a priori ed è l'unica utilizzabile, che posso andare a modificare in scrittura oppure ad interrogare la lettura. Come già detto dal punto di vista numerico posso avere solo contatori aggregati o informazioni di stato semplici.

## **ESEMPIO**

Accesso SSH è considerato parte del mondo Netconf in poiché quando si entra con SSH su un dispositivo, si apre la configurazione, si cambia e poi si salva la modifica, che è equivalente a caricare un file in cui è indicata la modifica alla configurazione.

Tutte le misure viste fino ad ora sono di tipo passivo, le informazioni catturate sono principalmente di tipo aggregate, e questa limitazione è dovuta alle basi di dati a cui si appoggia questo schema. Per superare questa limitazione devo utilizzare degli schemi di cattura che vadano oltre la cattura aggregata, lo step successivo al contare i pacchetti che passano su un'interfaccia, è il monitoraggio i flussi che passano sull'interfaccia.

Monitorare i flussi significa creare un record per tutte le connessioni TCP e tutti i flussi UDP, è chiaro che non esistono dei veri e propri flussi UDP però posso mappare una richiesta con una risposta, quindi posso creare una specie di meta-flusso.

## **ESEMPIO**

Teams usa sempre le stesse porte src e dst per l'audio e il video anche se utilizza UDP.

Quindi se si vuole monitorare chi sta accedendo e non soltanto quanto traffico passa, devo trovare una soluzione che faccia un passo ulteriore, ovvero per che permetta di monitorare tutti i flussi che passano attraverso una interfaccia.

Questo chiaramente è interessante per i dispositivi di rete, come switch e router, poiché sui sistemi terminali non ci sono i flussi che attraversano l'interfaccia, ma solo flussi in ingresso e in uscita, che tipicamente sono mappati con i servizi che girano sulla macchina, quindi nel caso di sistemi terminali, dobbiamo andare ad analizzare non tanto i flussi, quanto i log dei servizi che stanno girando sulla macchina.

Quindi ciò che è necessario è complementare i dati di SNMP con altri dati eterogenei e possibilmente con un maggiore contenuto informativo e una maggiore complessità.

Se voglio fare un ulteriore passo in avanti?

La cattura di un flusso infatti è un record che ci dice che il flusso è iniziato ad una determinata ora, quale indirizzo mittente e quello destinatario, i numeri di porte e numero di pacchetti e bytes. È quindi ancora un'informazione aggregata anche se più di dettaglio rispetto al numero di bytes che attraversano un'interfaccia. La si considera aggregata perché non ci dà una visione temporale di quello che sta succedendo.

Per avere dei dati completi anche da un punto di vista temporale devo fare una cattura di tipo TCMDUMP o WIRESHARK che mi permette di effettuare una cattura in tempo reale.

Come in SNMP, anche in questo caso i dati andranno memorizzati e trasferiti, in quanto siamo sempre in una architettura Client-Server in cui una parte produce dati e una li consuma. C'è quindi la necessità di fare particolare attenzione a quanti dati vado a generare per evitare che la rete venga sommersa dal traffico di gestione della rete stessa.

Tutte le tecniche visto fino ad ora: SNMP, statistiche sui flussi o cattura dei singoli pacchetti sono tutte tecniche di tipo passivo. Esistono però anche tecniche classificate come attive, nello specifico si parla di tecnica attiva quando questa inietta del traffico in rete, riuscendo ad acquisire informazioni che le tecniche passive non riescono ad acquisire.

Se per esempio sono interessato a capire se un sistema remoto, non sotto il mio controllo è attivo, posso eseguire un Ping. Se invece è sotto il mio controllo posso vedere tramite NETCONF e RESTCONF. Se voglio capire che strada fa una richiesta tra mittente e destinatario posso usare traceroute, se voglio conoscere le porte di un determinato sistema, posso utilizzare NMAP.

Tuttavia, questi tool attivi, tipicamente hanno lo svantaggio che le informazioni fornite sono comunque limitate, esistono quindi degli altri tool ancora più sofisticati che riguardano la programmazione attiva della rete. Nello specifico questi dispositivi più moderni permettono creare dei programmi in C ed iniettarli nella macchina, facendoli eseguire in una sandbox del kernel. Questi programmi permettono di effettuare sia operazioni di osservazioni, ovvero monitorare statistiche del traffico particolari, sia di applicare delle azioni.

Riprendendo lo scopo del Network Management: osservare, analizzare e agire; queste soluzioni di programmabilità ci permettono di automatizzare alcuni processi di osservabilità e la reazione ad eventuali situazioni di problemi o malfunzionamenti.

Per quanto riguarda l'analisi invece, non possiamo pensare di svolgere un'analisi completa utilizzando un programmino in C. L'analisi va comunque fatta online o offline, dall'umano o automatizzata. Posso quindi osservare le statistiche e se trovo un problema che sono in grado di risolvere tramite configurazione, utilizzerò i metodi standard, se invece de fare un'operazione più sofisticata posso pensare di pushare nel dispositivo un ulteriore software; è ovvio che un software di data programmability può aiutare nel caso di load balancing, a dirottare il traffico, a reagire ad un attacco hacker ma se una ventola del router si è bloccata, non ci si può fare niente, quindi non è una soluzione per tutti i problemi.

Telemetria di rete è solo passiva, quindi utile solo per l'osservabilità, anche se esistono degli schemi che permettono di renderla attiva. Si parla di tecnica attiva perché per effettuare questa osservazione in modo più dettagliato si attaccano delle etichette sui pacchetti e facendo ciò si va modificare il traffico, si può quindi parlare di schemi attivi. D'altro canto, questa tecnica ha il problema che tende a creare volumi di traffico estremamente rilevante poiché va a creare traffico per ogni dispositivo che il pacchetto attraversa. Vengono quindi utilizzati solo quando estremamente necessario, per evitare di intasare la rete con il traffico di management, perché permettono delle operazioni che altrimenti non sarebbero possibili.

# Monitoraggio del Traffico

Stiamo spostando l'attenzione dal monitoraggio della rete al monitoraggio del traffico (monitoraggio del traffico è una parte del monitoraggio della rete). Monitorare la rete significa anche conoscere lo stato di salute dei suoi dispositivi mentre monitorare il traffico significa monitorare il traffico che attraversa i dispositivi della rete.

A seconda del livello di dettaglio con cui sono interessato ad osservare il traffico, in generale o in uno specifico istante di tempo o uno specifico tipo di traffico

## ESEMPIO

Se volessi vedere anomalie in caso di malfunzionamento della risoluzione del nome di dominio, potrei essere interessato a vedere se il traffico DNS, ovvero quello che traduce le richieste mnemoniche in indirizzi IP. Posso capire se questo traffico va e viene da un server legittimo oppure se qualcuno si è impossessato del nostro server e sta dirottando le richieste verso una macchina controllata da terzi, in questo caso specifico, quindi, sono interessato al solo monitoraggio di uno specifico protocollo.

Quando si parla di traffico monitoring si deve subito entrare nell'ottica che potrebbe esserci necessita di gestire dati di tipo eterogenei, probabilmente raccolti da protocolli diversi, con formati diversi e che devono essere amalgamati insieme per essere analizzati in toto, oppure è possibile analizzarli a compartimenti stagni e incrociare i risultati anche se a volte questo tipo di analisi non funziona.

La sfida principale dell'analisi guidata dai dati è quella di raccogliere dati sufficienti per ricostruire eventi rari senza raccogliere una quantità di dati tale da rendere impraticabili le interrogazioni.

Le **query** sono le richieste che mi permettono di elaborare le statistiche di traffico che vado a raccogliere.

Risulta quindi che collazionare i dati è abbastanza semplice e visto che esistono svariate soluzioni come TCPDUMP o SNMP. Invece quando si tratta di estrarre informazioni dai dati la situazione non è semplice allo stesso modo e può risultare piuttosto complicato.

Prendiamo per esempio uno specifico aspetto del network management: la security.

Ricordiamo che il **Network Management** ha 5 declinazioni principali (FCAPS):

- **Fault**
- **Configuration**
- **Accounting**
- **Performance**
- **Security**

Andando considerare nello specifico la security: una complicazione aggiuntiva deriva dal fatto che di attacchi ce ne sono tanti e in continuazione; infatti, la maggior parte degli attacchi sono abbastanza innocui, come ad esempio lo spamming.

Essendo questi attacchi continui e risulta difficile all'interno di un certo numero di attacchi individuare le vere minacce, che sono poche e difficili da identificare.

Risulta quindi essenziale distinguere un **attacco** da una **minaccia**, in quanto di attacchi ce ne sono in continuazione ma non tutti rappresentano una reale minaccia.

La maggior parte degli attacchi risultano innocui in quanto necessitano la collaborazione dell'utente mentre quelli che non richiedono la collaborazione dell'utente e possono creare danni se non arrestati, rappresentano delle vere e proprie minacce.

Tutto ciò che è stato appena detto vale anche per altri dei 5 domini del Network Management.

Prendendo per esempio per le prestazioni (P - Performance), può succedere che le prestazioni delle macchine non siano costanti, o in altre parole, che ci siano delle oscillazioni sui livelli di servizio che la macchina offre. Questo comportamento è abbastanza normale nel caso in cui si tratti di un fenomeno sporadico; diventa un problema serio solo nel caso in cui tutti i tempi d'attesa vadano ben oltre un tempo accettabile.

Essendo il traffico di rete molto ripetitivo può essere suddiviso in macrocategorie di traffico come ad esempio e-mail, traffico video, accessi ai file, accessi alle risorse web e negli ultimi anni una quota significativa di traffico è associata alle videoconferenze.

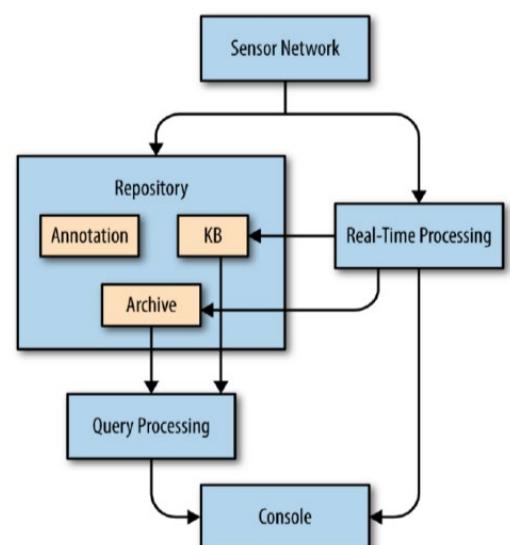
Un altro esempio di attacco spesso innocuo è lo scanning, ovvero una tipologia di attacco attiva con cui utenti remoti scansionano tutte le porte di strato di una macchina per vedere se sono aperte e se la macchina scansionata risponde.

Di solito questi attacchi avvengono in continuazione e non rappresentano una minaccia, a patto che si assuma di avere che il sistema sia ben protetto e non un colabrodo.

Riepilogando, le vere minacce sono poche e ben nascoste all'interno di un mare di traffico legittimo o comunque innocuo, può essere quindi abbastanza complicato identificarle in tempo reale. La rilevazione real-time risulta essenziale in quanto dovrei essere in grado di fermare una minaccia prima che questa crei dei disservizi. Nei casi in cui è troppo tardi per prevenire la minaccia, potrebbe essere necessario mettere offline la rete e identificare il problema analizzandola.

## Schema per Network Management e analisi del traffico

- **Sonde:** sono deputate alla raccolta di dati, non di informazioni e si parla di rete di sensori. Queste sonde permettono di avere informazioni sulla salute dei dispositivi o di catturare statistiche di traffico.
- **Repository:** archivio OFFLINE dove il traffico viene memorizzato.
- **Query processing:** è un meccanismo di elaborazione del traffico che interagisce con il repository recuperando i dati che gli interessano con delle query ed eventuali filtri. Si occupa inoltre di elaborare i dati recuperati e di presentarli ad una console.



- **Console:** può essere lo schermo dell'amministratore di rete o semplicemente un report che viene generato e archiviato/invia a un altro software o ad un utente umano.

In parallelo alla catena di sinistra, tutta relativa all'offline, dove i dati vengono elaborati quando ci sono risorse di calcolo a disposizione, abbiamo:

- **Real-time processing:** i dati raccolti dai sensori, oltre che essere memorizzati, possono essere inviati a questo componente che si occupa di elaborarli in tempo reale. Le elaborazioni ottenute, a questo punto, possono essere sia inviate alla console che memorizzate nel repository, per essere utilizzate in futuro.

### **NOTA**

Il real-time processing potrebbe non essere presente in tutti i sistemi, quando è presente viene spesso associato al firewall. D'altro canto, potrebbe pure essere possibile che il sistema memorizzi in repository solamente dei dati processati in tempo reale.

Queste scelte dipendono da quale dominio dei 5 (FCAPS) stiamo esplorando, dalle potenzialità dei dispositivi e dalle skills del l'amministratore di rete; non esiste quindi una regola predefinita.

## **REPOSITORY**

Il repository è dove vado a salvare i dati non in stream, ovvero dove i dati non passano e basta ma vengono anche archiviati. Per decidere come costruire un repository è necessario considerare 3 aspetti fondamentali:

### **1. Orizzonte temporale**

I dati relativi a quale orizzonte temporale devo salvare? Ore, giorni, settimane, mesi, anni, o tutti.

L'orizzonte temporale è fondamentale nell'andare a dimensionare lo storage associato a questa funzione. Se per esempio voglio salvare tutti dati degli ultimi 20 anni devo tenere in conto che periodicamente dovrò aggiornare il mio sistema di storage. Di solito il sistema di storage sfrutta sempre una doppia tecnologia, una veloce dove leggo e scrivo i dati più nuovi, in quanto sono quelli che più probabilmente saranno oggetti di query; e parte di storage più lenta dove andrò ad archiviare i dati più vecchi.

Chiaramente, più la memoria è veloce e più è costosa, bisogna quindi dimensionare lo storage e le due tecnologie in base alle proprie necessità.

Inoltre, devo garantire che il costo del Network Management (che comprende anche i costi relativi allo storage) non deve impatti troppo sul costo complessivo del servizio.

### **2. Tipi di dati**

Esistono delle macrocategorie di dati che posso memorizzare e sono:

#### **a. Dati relativi al traffico di rete**

Posso raccogliere statistiche di tipo SNMP, statistiche relative ai flussi (tramite soluzione Net Flow) oppure qualcosa di più voluminoso come le catture.

Sappiamo però che, quando si va a catturare il traffico, si possono applicare filtri di cattura per raccogliere solo le informazioni utili all'analisi del traffico.

Ad esempio, posso utilizzare le statistiche di flusso che contengono indirizzi IP, porte, pacchetti e byte; e poi le statistiche del traffico DNS, che rappresentano un traffico moderato, circa un record per flusso. Posso quindi associare la query DNS che ha preceduto l'installazione del flusso con il flusso stesso, andando così a mescolare due tipi di dati, ovviamente catturati con due sonde diverse.

**b. Dati relativi all'informazioni di sistema**

È importante non confondere le informazioni di sistema con i log di sistema, in quanto i primi riportano le statistiche sulla salute del sistema, mentre i secondi riportano i log dei servizi posti a strato applicativo.

**c. Dati relativi ai log di sistema**

In questo caso posso considerare vari livelli di log, nello specifico, generalmente i log si dividono in INFO, DEBUG, WARNING ed ERROR. Quelli a più alto livello sono INFO e DEBUG, per i quali per ogni richiesta, sia che questa generi un problema sia che non lo faccia, viene scritto un record. ERROR invece è il più limitato e scrive una riga nel file di log solo quando si verifica un errore.

### 3. Ogni quanto riconsiderare l'aggiornamento del sistema di storage

Quando si parla di aggiornamento non si fa riferimento solo ad acquistare nuova memoria ma anche a quale utilizzare e a come gestire le risorse che si hanno a disposizione. Oltre ovviamente a considerare le nuove esigenze, come ad esempio la necessità di aumentare le velocità di trasferimento su una determinata interfaccia.

Tutto quello appena visto rientra nell'ambito di *configurazione* del Network Management, è importanti infatti monitorare il sistema per vedere se la configurazione predisposta è aderente all'obbiettivo progettuale prestabilito. Una cosa fondamentale che distingue i repository per i dati di traffico dai classici database relazionali è che questi archivi di dati non si aggiornano. Gli utenti possono transazioni di sola lettura o cancellazione, mentre solo le sonde (che possono essere log di sistema e servizi, catture di traffico e informazioni sul sistema) possono scrivere nuovi record. Quindi, una volta che transazione ha avuto luogo non è possibile modificare il record, il dato rimane immutato; risulta quindi molto diverso dalle transazioni di un classico database relazionale.

Quindi delle 4 proprietà ACID dei database relazionali:

**A – Atomicità      C – Consistenza      I – Isolamento      D – Durabilità**

La *consistenza* non è importante perché nessun transazione va a modificare i dati. È vero che la transazione è *atomica* e che quindi o ha effetto o non ha effetto, e nel caso in cui vada a buon fine scrive un nuovo record.

#### **In che cosa consiste la transazione?**

La transazione da parte dei sensori genericamente detti, è data da operazioni di scrittura oppure di sola lettura da parte degli utenti, tramite il query processor. Sensori diversi possono scrivere in parallelo e la *durabilità* deve valere.

È possibile quindi osservare che viene meno uno dei requisiti dei DB relazioni e di conseguenza non è strettamente necessario utilizzare un DB relazionale, ho quindi maggiore flessibilità su come implementare la base di dati.

## **Come è fatto il repository?**

Il repository tipicamente ha 3 sottocomponenti di cui una sola è necessaria: l'archivio, in questo componente vengono scritti i dati.

Come già visto non è necessario mantenere tutti i dati, alcuni di questi possono essere sostituiti con le loro elaborazioni, ma elaborare un dato è diverso da modificarlo con una transazione.

## **Cosa è un'annotazione?**

Le annotazioni sono un modo per aumentare i dati con una sorgente diversa, che non dipende dalle sonde che catturano traffico; un esempio classico è quello della geolocalizzazione oppure per quanto riguarda la sicurezza, l'aggiunta di ulteriori basi dati che contengono le firme digitali di attacchi noti. Queste sono informazioni che vengono dall'esterno, tipicamente non inserite manualmente ma recuperate, come per esempio le utility di geolocalizzazioni che mappano un indirizzo IP con la posizione geografica.

## **Che differenza c'è tra i dati dell'archivio e i dati dalle annotazioni?**

Mentre i dati dell'archivio sono immutabili, i dati delle annotazioni non necessariamente sono immutabili, ad esempio l'indirizzo IP, non necessariamente è immutabile, basta prendere come esempio l'assegnazione degli indirizzi IP tramite server DHCP all'interno dell'ateneo di Perugia. Una volta che un utente si logga, acquisisce un indirizzo IP, ma nel momento in cui si collega questo indirizzo IP torna nel pool di indirizzi liberi e potrebbe quindi essere assegnato ad un nuovo utente che farà il login. Le annotazioni possono quindi essere modificate.

## **Le annotazioni vanno combinate con i dati dell'archivio, formando una sola base di dati oppure no?**

Dipende, anche da come si ha intenzione di costruire le query, si potrebbe infatti decidere di svincolarsi dalle annotazioni oppure avere tutto in un'unica base di dati e rendere più facile il processo.

## **Knowledge Base**

Le KB sono dei dati che vengono generati dal personale dell'organizzazione e riguardano la loro esperienza pregressa, sono principalmente dei dati manuali, quindi non generati da tool esterni. Tipicamente si tratta di dati non strutturati e in piccole quantità, proprio perché generati manualmente e legati all'esperienza dell'amministratore di rete.

Non sono necessariamente presenti, come potrebbero non essere presenti le annotazioni. Potrei anche fare riferimento ad un servizio esterno che mi fornisce le annotazioni quando vado a fare le query. Quindi non salvare i dati delle annotazioni all'interno del mio repository ma ricercarle quando ne ho bisogno.

In questo caso, il rischio è di avere delle annotazioni che sono valide solo per istante attuale e non per l'istante a cui la traccia di traffico si riferisce, quindi potrei avere annotazioni sbagliate, non affidabili o non presenti. Questo poiché alcune annotazioni potrebbero non avere associata una finestra temporale di validità, risultano quindi inaffidabili. Potrebbero anche risultare sbagliate poiché chi le ha generate non ha utilizzato un processo di funzione corretto o ancora più semplicemente sto facendo riferimento a dati troppo distanti nel tempo e non più presenti in memoria.

Nell'archivio quindi metto i dati di evento, dove con evento si intende qualcosa di estremamente generico come, per esempio, la generazione di un record SNMP, un pacchetto catturato, una statistica di traffico aggiornata, un antivirus che genera un report di una minaccia; un evento è quindi qualcosa di molto generico che può essere sincrono o asincrono.

**03/10**

### **Recap:**

Non tutte le proprietà acide dell'archivio sono necessarie: in particolare la C di consistenza non è necessaria perché i record che vengono prodotti durante il monitoraggio della rete non vengono alterati. Una volta che sono stati scritti rimangono lì, al massimo possono essere, sul lunghissimo periodo, cancellati per essere sostituiti con una versione aggregata dei record stessi, ma non vengono cancellati.

L'annotazione invece è un processo di autodetection, quindi di arricchimento dei dati che tipicamente avviene attraverso dei tool esterni, ad esempio il tool di geolocalizzazione o attraverso il DNS. Quando si catturano le informazioni tipicamente si hanno informazioni sugli indirizzi IP, ma non sulla notazione mnemonica del corrispondente dell'indirizzo IP. Se invece si ha come tool di annotazione un tool che fa il mapping tra l'indirizzo IP e l'indirizzo mnemonico si hanno entrambi gli indirizzi: questo può essere d'aiuto per capire qual è l'host noto al quale si fa riferimento.

Infine, KB, Knowledge Base, è di solito una banca dati piuttosto limitata e che tipicamente è prodotta dall'amministratore di rete sulla base della sua esperienza. Possono essere quindi informazioni aggiuntive tipicamente non strutturate, gestite su base personale, che servono anch'esse ad aumentare le informazioni catturate nell'archivio ma che hanno una valenza locale e che sono comunque preziose. Ad esempio, l'amministratore di rete può appuntarsi all'interno del database che un certo tipo di traffico che potrebbe sembrare anomalo in realtà non è da considerare anomalo perché è relativo ad uno specifico servizio a cui accedono gli utenti di quel dominio.

Abbiamo poi due blocchi entrambi relativi all'elaborazione dei dati: elaborazione delle query ed elaborazione real time. Sono blocchi simili ma formalmente ben distinti.

### **Elaborazione delle query**

Il sistema di elaborazione delle query è un ambiente di sviluppo che supporta la sintesi dei dati da fornire dati contestuali. Il sistema può elaborare dati da più posizioni per sintetizzarli.

Questa è l'elaborazione associata alle richieste che vengono fatte all'archivio. Parliamo di richieste fatte sempre offline, quindi non si effettuano richieste sul traffico che sta fluendo dentro il sistema di monitoraggio della rete. Prima scrivo dati e poi li elaboro.

Due cose sono fondamentali nell'elaborazione delle query:

- La **velocità** con cui le query vengono eseguite: è importante che le query diano un risultato in un tempo utile

- Il **livello di concorrenza** sia del motore che effettua le query sia del repository a cui il motore accede deve essere **significativo**: in modo tale che più utenti possano effettuare query distinte

Questo non dipende solo dal motore di erogazione delle query ma anche da come i dati sono memorizzati nell'archivio. Tipicamente i database relazionali non sono competitivi rispetto a quelli non relazioni e per questo non si utilizzano i database basati su SQL (sequel) ma i database di tipo NoSQL.

→ Ci sono quindi dei meccanismi dove i dati vengono archiviati in maniera più libera e che permettono una maggiore concorrenza e una maggiore velocità nell'accesso dei dati.

## Elaborazione real time

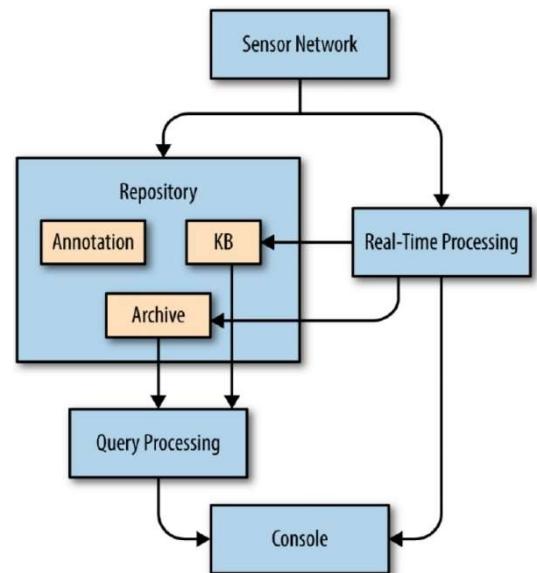
L'elaborazione in tempo reale consiste nell'elaborare i dati o prima che i dati vengano scritti sull'archivio o in parallelo alla loro scrittura sull'archivio.

Infatti, dalla sorgente dei dati, che sono le nostre sonde (i sensori), i dati possono finire nel repository direttamente oppure passare per l'elaborazione real time. A seconda degli scenari posso avere l'uno o l'altro o entrambi. Possono avere quindi uno scenario in cui i dati sono sia salvati grezzi dentro il nostro archivio, direttamente come arrivano, sia salvati pre-elaborati. Se li salvo pre-elaborati ho tipicamente una versione più snella e compatta e da cui ho già estratto un'informazione dalla mole di dati. È più utile perché permette di effettuare le query in maniera più semplice.

Sarebbe quindi da preferire uno scenario in cui esistono sia i dati pre-elaborati sia i dati grezzi perché in qualsiasi preelaborazione perderò sempre qualche informazione presente invece sui dati grezzi. Se quindi ho i dati pre-elaborati affiancati dai dati grezzi ho la possibilità di fare le query su entrambi i tipi di dati, anche per valutare se il meccanismo di elaborazione real time è efficiente oppure produce dei dati con informazione troppo compressa e quindi perdita di informazione stessa.

Il pre-processing può semplicemente scrivere i dati nell'archivio o nella KB oppure avere anche una funzione di exporter del risultato di questa elaborazione in tempo reale verso una possibile console di controllo.

La console di controllo può essere sia associata all'utente umano sia essere un software che poi interagisce con altri software.



L'elaborazione in tempo reale consiste in tutte le analisi eseguite durante l'arrivo dei dati:

- corrispondenza diretta della firma,
- analisi ad alte prestazioni,
- generazione del file di registro.

Di norma, l'elaborazione in tempo reale dovrebbe essere distinta dall'elaborazione delle query:

- i dati in tempo reale dovrebbero idealmente riassumere o elaborare i dati per ridurre la quantità di elaborazione delle query necessaria

## Rete di sensori

Sono i sensori di rete, cioè le sonde che andiamo a mettere in opera per catturare le informazioni sul traffico di rete e che poi sono quelle che dovremo andare ad elaborare per estrarre informazioni.

Possiamo avere sonde di 3 tipi per la raccolta dei dati:

- relativi alla rete (network sensing):
  - sonde associate ai NIDS (Network Intrusion Detection System), sonde associate ai firewall, sonde relative ai flussi, sonde che catturano i pacchetti (tcpdump)

**NIDS** è un sistema messo in opera per rilevare potenziali intrusioni all'interno della rete ed è funzionalmente distinto dal **firewall**, che invece rappresenta una barriera che discrimina quali pacchetti possono passare e quali no. Il NIDS osserva invece i pacchetti che passano o che tentano di passare e determina se c'è in corso un attacco e se quell'attacco è di poco conto o se rappresenta una minaccia. I ruoli sono quindi ben distinti.

- relativi agli host (sistemi terminali) (host-based sensing):
  - AV (AntiVirus), HIDS (Host-based Intrusion Detection System), syslog
- relativi ai servizi (servizi in esecuzione su un determinato host che eroga un servizio, tipicamente i server) (service sensing):
  - syslog, HTTP logs, output di un sistema (assumendo che un sistema sia un servizio), log di un server DNS, e tutti i log dei servizi in esecuzione

I log hanno tipicamente strutture molto diverse l'uno dall'altro; quindi ogni servizio può generare informazioni aggregate in maniera diversa e il livello di dettaglio delle informazioni è tipicamente configurabile.

Tutti questi schemi sono di tipo passivo, cioè non vanno ad alterare il traffico che passa in rete. Il rilevamento attivo non è integrato nella rete di sensori.

Esistono poi degli schemi di tipo attivo (Ping, traceroute, nmap) che non è che alterano il traffico ma che generano traffico di modestissima entità, che viene trasportato attraverso la stessa rete che vado a monitorare e proprio utilizzando l'output di questi tool possono generare ulteriori informazioni aggiunte poi all'archivio.

**Problemi di progettazione:** quando progetto una rete, un sistema di monitoraggio delle informazioni ho bisogno di rispondere a queste domande prima ancora di cominciare a progettarlo.

Quali sono i dati associati al monitoraggio del traffico? Sono di entità modesta o di entità trascurabile? Se sono di entità trascurabile non avranno impatto sulla rete e possono essere trasportati “in banda”, insieme al traffico monitorato; se invece la loro quantità può essere significativa sarebbe meglio effettuare il trasporto dei dati fuori banda, cioè su un’infrastruttura parallela, tipicamente virtuale (VLAN utilizzate nella maggior parte dei casi). Il traffico associato al monitoraggio viene segregato nelle VLAN che di solito hanno una priorità inferiore rispetto a quella del traffico stesso.

Qual è l’impatto dello storage? Se si ha una generazione di una grande quantità di dati, i dati grezzi vengono lasciati nelle sonde e vengono trasferiti verso tool di osservazione centrale solo quando richiesto. Normalmente all’interno della sonda è previsto un modulo di preelaborazione che condivide solo i dati pre-elaborati. Quindi in molti casi questi due componenti sono associati insieme: ho un modulo che cattura il traffico e che lo elabora localmente e che invia al repository solo il risultato della preelaborazione. In alcuni casi però i dati grezzi catturati possono essere salvati localmente sulla sonda e vengono recuperati secondo necessità (questa cosa tipicamente non si fa sui router perché troppo costosa).

Come faccio a capire che la sonda sta funzionando bene? Potrei effettuare un monitoraggio molto blando per vedere cosa succede all’interno della sonda. Avere un minimo di capacità di monitoraggio non solo del traffico ma anche dei dispositivi che catturano il traffico è fondamentale per capire se quello che sto osservando è affidabile oppure no. Tutti gli schemi di monitoraggio del traffico NON prevedono di default di inserire il controllo della sonda stessa, bisognerà utilizzare qualcosa di parallelo e consumare altre risorse.

Un monitoraggio efficace richiederà di destreggiarsi tra più sonde di diverso tipo, che trattano i dati in modo diverso.

Per categorizzare le sonde di solito si usano 3 attributi: (CHIESTI COSTAMTEMENTE AGLI ORALI!!)

- **Vantaggio:** rappresenta l'impatto che ha la posizione topologica della sonda all'interno della rete sulle informazioni che vengono catturate (non tanto sui dati quanto sulle informazioni che riesco ad estrarre dai dati).  
Le sonde con punti di vista diversi vedranno parti diverse dello stesso evento.
- **Dominio:** ha a che fare con l'informazione che viene catturata dalla sonda perché sensori diversi possono catturare informazioni diverse. Quindi sensori con lo stesso vantaggio, cioè, posizionati nello stesso punto della rete, ma con domini diversi vedono parti diverse della stessa informazione (di solito complementari, molto spesso parzialmente sovrapposte). È fondamentale cercare di coprire il nostro spazio delle informazioni con il minor numero possibile di sensori perché i sensori costano in termini economici ma soprattutto dal punto di vista computazionale e per analizzare i dati che generano.

Esempio: supponiamo di avere due sensori su uno stesso dispositivo, su un server web: uno raccoglie i log del server web e l'altro è un antivirus. È chiaro che i log dell'antivirus sono separati dai log del server web perché uno è relativo all'host-based sensing, l'antivirus, mentre i log del server web sono relativi al service sensing.

- **Azione:** ci dice come la sonda effettua il report delle informazioni.  
Può essere una reportistica base in cui vengono generati report relativi a tutto il traffico che passa in un certo sensore e su un certo dominio oppure può essere una reportistica ad eventi (esempio di evento: ventola del processore che si blocca, scadenza di un timer, rilevamento di una intrusione nel sistema).  
Ci possono essere infine delle azioni attive, che non hanno a che fare solo con la reportistica, ma che possono includere l'elaborazione e il controllo del dispositivo stesso. Cioè, posso implicare un'azione retroattiva che è scatenata dal processo di monitoraggio.

## Perché questa classificazione

La classificazione viene fatta per permetterci di capire a cosa serve il sensore e quello che riesce a vedere.

Fornisce un modo per scomporre e classificare i sensori in base al modo in cui gestiscono i dati.

- Il dominio, andando a distinguere le diverse informazioni che può catturare un sensore, ci permette di caratterizzare dove i dati sono collezionati e come.

- Il vantaggio ci informa su come il posizionamento del sensore influisce sulla cattura del sensore stesso, indipendentemente dal dominio.
- L'azione descrive in dettaglio il modo in cui il sensore manipola effettivamente i dati.

L'insieme dei 3 attributi ci permette di capire quanto le informazioni collezionate attraverso una sonda siano valide. Proprio perché il risultato delle conclusioni di un analista (software o persona) possono essere più o meno valide. La validità si riferisce alla forza degli argomenti. Un argomento ha forza quando c'è una connessione diretta e logica tra le informazioni estratte dai dati e le conclusioni. Quando questo legame è debole posso avere delle conclusioni non ragionevoli e non affidabili. Quindi, a seconda del tipo di sensore che utilizzo posso giungere a delle conclusioni più o meno forti.

Esempio: se utilizzo un sensore che cattura solo le informazioni di strato 2 (indirizzi MAC) su uno switch posso dedurre molto poco sul livello di sovraccarico di un server web attaccato a quello switch. Posso vedere che ci sono molti pacchetti inviati al server ma non so che tipo di pacchetti sono, vedo solo il traffico di pacchetti. Quindi se ho come vantaggio lo switch di fronte al server, come dominio lo strato 2 e come azione la reportistica base, riesco ad estrarre ben poche informazioni relativo al processo di servizio del server web e quindi la forza dei miei argomenti è estremamente bassa perché la validità delle mie conclusioni è modesta.

## Dominio

I domini tipicamente sono 4 e sono sempre gli stessi: rete, servizio, host e caratterizzazione di dominio di tipo attiva.

Le **sorgenti di dati** per ogni tipo di dominio sono:

- Rete: catture dei pacchetti (PCAP è tipicamente il formato in cui vengono salvati i dati catturati con tcpdump, tshark o wireshark) o catture delle informazioni relative ai flussi dei pacchetti (NetFlow è il formato in cui vengono salvati i metadati relativi ai flussi) o loro elaborazioni.

Non necessariamente devo salvare report NetFlow o intere catture PCAP, posso anche effettuare una preelaborazione di questi dati e avere quella come informazione, però la sorgente dati è quella.

(Bisogna sempre distinguere tra sorgente dati e informazione fornita: l'informazione è associata al dominio)

- Servizio: logs (log relativi ai vari servizi che vado a monitorare)
- Host: stato del sistema o meccanismi di allerta preconfezionati
- Attivo: processo di scansione, più in generale di interazione con la rete (esempio: ping, traceroute, nmap)

Dal punto di vista della temporizzazione, **timing**, potrò avere delle statistiche di tipo:

- Real-time
- Basate sui pacchetti
- Basate sugli eventi (tipicamente richieste di servizio)
- Asincrono
- Eseguite quando l'utente decide di eseguirle (attraverso interazione diretta o interazione programmata)

Per ogni sorgente di dati posso ricavare una certa **identità**:

- IP
- MAC
- (Numero di porta per il dominio di rete)
- IDs basati sui servizi
- UUID (Universal Unique IDentifier): è una stringona che tipicamente è univoca perché è molto bassa la probabilità che vengano generate due stringhe uguali con lo stesso formato, anche se la probabilità di collisione non è 0. Usato ad esempio dal sistema operativo per etichettare gli hard disk della macchina.

Domain	Data sources	Timing	Identity
Network	PCAP, NetFlow	Real-time, packet-based	IP, MAC
Service	Logs	Real-time, event-based	IP, Service-based IDs
Host	System state, signature alerts	Asynchronous	IP, MAC, UUID
Active	Scanning	User-driven	IP, Service-based IDs

(+ esempio fatto in aula con disegno)

## Vantaggio

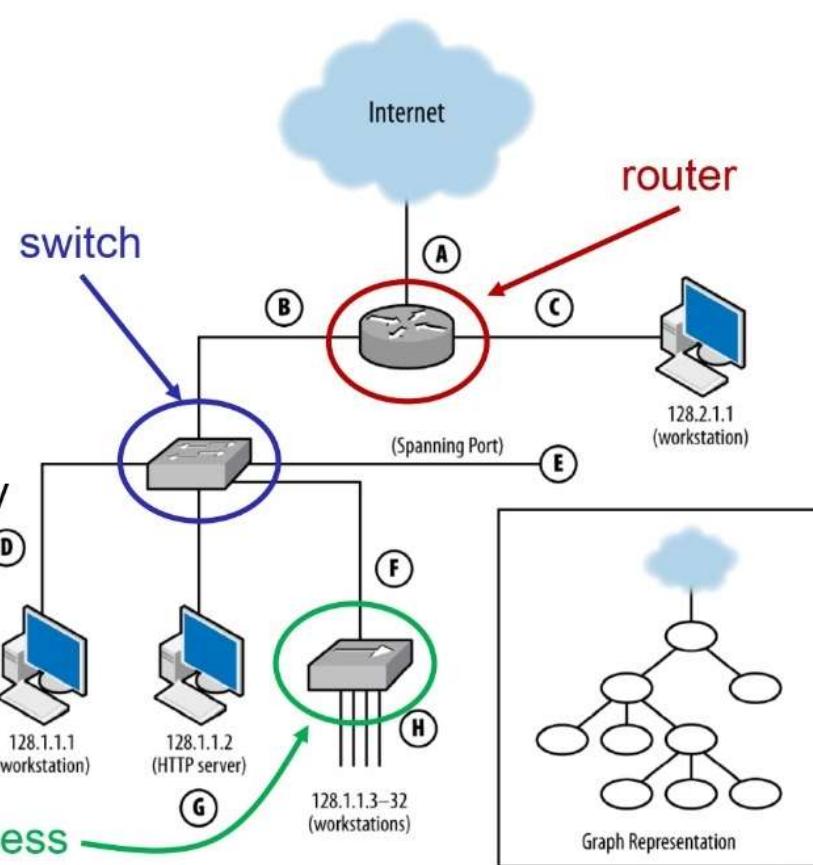
Il vantaggio spesso è legato a quello che il nostro sensore riesce ad osservare. Il dominio chiaramente non è quella cosa che il sensore o la sonda riesce a catturare, ma rappresenta un attributo o dimensione diversa della sonda.

Esempio: se consideriamo una sonda che cattura informazioni di rete (i pacchetti), quando parliamo del vantaggio consideriamo tutte sonde dello stesso tipo, cioè in grado di catturare pacchetti, però pensate in punti diversi della rete. Il piazzamento influenza i pacchetti che riesco a catturare. Assumiamo che un collegamento sia capace di registrare tutto il traffico che lo attraversa. Il vantaggio è importante perché a seconda di dove vado ad osservare vedrò cose diverse, che dipendono esclusivamente dalla posizione della sonda.

» -

or's  
!  
es the  
that  
will be  
observe  
will be  
record any  
at  
that link  
a  
ion

hub/wireless



Il primo dispositivo che andiamo ad analizzare è un router, il secondo dispositivo è uno switch e l'ultimo è un hub.

Realisticamente gli hub non esistono più. Un hub è un dispositivo che riceve un pacchetto su una porta e lo replica su tutte le porte tranne quella su cui è stato ricevuto. Questi dispositivi sono ormai stati rimpiazzati ampiamente dagli switch, che sono dispositivi intelligenti che effettuano queste operazioni fino a quando, sulla base del

processo di autoapprendimento, non hanno capito dove smaltire il pacchetto ricevuto sulla porta di ingresso.

L'hub è presente sulla slide perché è assimilabile a un nodo di tipo wireless. Il canale wireless è per definizione condiviso e quindi broadcast. Manteniamo quindi la nozione di hub perché è funzionalmente equivalente a quella di un access point (nel caso di wi-fi) o di una stazione radio base (nel caso delle reti cellulari 4G, 5G).

Quando decidiamo dove piazzare una sonda dobbiamo seguire un processo che ci permette di determinare qual è il punto migliore per piazzarla. Prima di tutto dovremo **acquisire la mappa** della rete, che può essere astratta, a grafo, o una mappa fisica. Entrambe hanno vantaggi e svantaggi: quella astratta è più semplice da maneggiare, quella fisica ci fornisce maggiori informazioni.

Una volta acquisita la mappa, dobbiamo **determinare i potenziali punti di vantaggio**, cioè dove all'interno della mappa posso veramente piazzare una sonda. Non necessariamente posso piazzare sonde dappertutto, ad esempio non posso piazzarle su dispositivi privati che usano una rete aziendale, su server obsoleti o su switch e router di tipo consumer.

Sulla base delle prime due informazioni, la mappa e i punti dove posso applicare le sonde, dobbiamo determinare **quante sonde** mi servono **per effettuare una copertura**, cioè, mettere in opera il numero minimo di sonde che mi permette di osservare tutta la rete a un determinato livello protocollare. È praticamente impossibile avere una copertura completa senza sovrapposizioni, e le sovrapposizioni costano.

A questo punto esaminiamo le lettere del grafo, dalla A alla H per capire i potenziali vantaggi che ho piazzando una sonda su ciascuno di quei link.

Supponiamo di piazzare una sonda che vada a monitorare il link numero A (o equivalentemente la porta del router numero A). Riesco a monitorare le interazioni della rete in forma aggregata con la rete esterna (internet). Se il calcolatore 128.2.1.1 effettua una richiesta al server HTTP 128.1.1.2 non riesco a vedere questa interazione. Posso decidere allora di andare a considerare una sonda associata alla porta numero B, sempre sul router. Questa porta mi permette di vedere tutto il traffico scambiato dalla parte della rete in basso a sinistra verso internet o verso la workstation tutta a destra.

Se voglio quindi vedere tutto quello che passa per il router devo mettere 3 sonde: 1 sul link A, 1 su B e 1 su C e nonostante questo il traffico servito dallo switch è limitato alla parte inferiore della rete non riesco a vederlo.

Se metto una sonda su D osservo tutto ciò che va e viene da quella workstation, esattamente lo stesso che accade con C.

Stessa cosa per il nodo G, riesco a monitorare solo l'attività del server web.

Il punto E, o spanning port, è un punto di monitoraggio esterno agganciato a uno switch o a un router (tipicamente a uno switch). Gli switch possono avere capacità di monitoraggio intrinseche, e per far questo hanno bisogno di risorse di calcolo (RAM e processore). Gli switch effettuano quindi le proprie operazioni di commutazione (spostamento del pacchetto da un'interfaccia di ingresso a una o più interfaccia di uscita) utilizzando le risorse di calcolo associate ciascuna a una scheda di ingresso e di uscita. Oltre a queste hanno tipicamente un processore general purpose e una RAM general purpose che servono per effettuare operazioni di management (es: gestione SNMP, gestione aggiornamenti, inizializzazione del dispositivo e così via). Per effettuare il monitoraggio all'interno dello switch, avendo risorse limitate, devo delocalizzare il monitoraggio a un punto esterno dello switch stesso. Lo switch mette a disposizione una porta di mirroring o spanning, all'interno della quale può essere copiato tutto il traffico che attraversa alcune interfacce. Il traffico è distinto in traffico di input e di output: importante perché uno stesso pacchetto lo vedo sulla porta di mirroring due volte (in input su una porta e in output su un'altra porta) e quindi per ovviare a questo faccio la cattura solo sul traffico in input o solo sul traffico in output. Chiaramente se il traffico è eccessivo riesco a copiare solo parte del traffico, il resto verrà scartato.

I moduli esterni in cui fare il monitoraggio dello switch non sono in genere di proprietà dello switch stesso ma sono dei server che catturano il traffico e lo elaborano.

Infine, se monitoro il link H, wireless, vedrò tutto quello che passa sull'interfaccia H ma non vedrò quello che passa sull'interfaccia F, e viceversa.

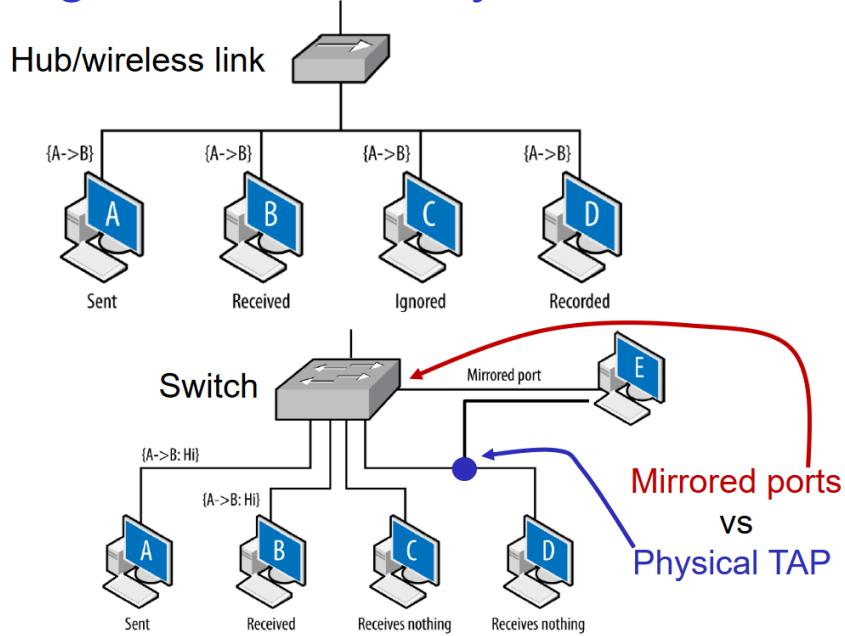
➔ A seconda del punto in cui vado a piazzare la sonda sarò in grado di acquisire informazioni diverse, indipendentemente dal dominio che caratterizza la sonda.

**Recap:** abbiamo visto qual è l'impatto dei punti di vantaggio su quello che io riesco a catturare. In questo schema abbiamo detto che vado a catturare su un collegamento. Ma catturare su un collegamento spesso non ci permette di avere una visione completa del contenuto di quello che catturo.

Se ad esempio catturo sul punto E, quindi configuro uno switch in modo da avere una porta in cui vado a copiare tutti i pacchetti che entrano su un sottoinsieme selezionato di porte, quello che potrò vedere sono, al massimo, i pacchetti e le intestazioni di strato 2,3 e 4 che sono sicuramente molto importanti ma non riesco a vedere il payload dello strato applicativo. Questo perché la maggior parte dei protocolli applicativi utilizzano la cifratura e senza la chiave di cifratura, che varia da sessione a sessione, non si può accedere alle informazioni dentro il payload e mi dovrò accontentare delle informazioni "esterne" al contenuto applicativo dei pacchetti. Noi sappiamo che le librerie SSL che implementano il transport layer security o secure socket layer, in realtà lavorano solo su TCP. Tuttavia, esistono una serie di soluzioni che vengono utilizzate a livello applicativo per cifrare anche pacchetti UDP e se non ho a disposizione la chiave della sessione non riuscirò a catturare il payload del pacchetto (UDP). Se però sono in possesso della chiave di sessione sarò in grado di decodificare anche il payload dello strato applicativo. Visto che tipicamente questa possibilità non la ho, tranne in casi particolari, quello che succede è che non si effettua la cattura di un intero pacchetto ma solo dei primi byte in cui ci sono le intestazioni, infatti, è inutile catturare ad esempio 1500 byte di un pacchetto e tenermeli, quando sono sufficienti solo 64/80 byte per ottenere tutte le informazioni che riesco a decodificare. Un caso notevole è quello delle reti wireless, in cui questa caratteristica è ancora più marcata. Nelle reti wireless l'unica informazione a cui ho accesso è quella di strato 2, quindi se sto catturando sul punto H e quello non è un HUB (che stanno sparando dalla circolazione) se ho un accesso wireless riesco a vedere solo le informazioni di strato 2 perché la cifratura viene apposta tra lo strato 2 e lo strato 3 e questo significa che i pacchetti IP sono già cifrati e non avrò accesso neanche alle informazioni di strato 4 (trasporto), né tantomeno a quelle dello strato applicativo. In alcuni casi, per quanto riguarda le reti locali (wi-fi) è possibile accedere alla chiave di cifratura, come ad esempio con un'infrastruttura personale che si ha in casa. Per quanto riguarda la cattura in G, viene fatta sul server e non sul link. Fare la cattura sul server ci permette di avere i pacchetti già decodificati. Quindi in generale bisogna: acquisire la mappa logica e fisica della rete, capire dove piazzare le sonde e infine determinare il numero di sonde che ci permettono di catturare tutto ciò che mi serve.

## Vantaggi strato di collegamento

### Vantage at data link layer



Analizzando la figura in alto, nel caso di un hub/rete wireless, è indifferente su quale porta posiziono il sensore: in qualunque caso catturo tutto il traffico.

Con le precisazioni fatte nel **recap**, possiamo vedere quella situazione lì. Quindi se il terminale A invia del traffico al terminale B, il terminale C riceve comunque il traffico inviato da A verso B, ma lo ignorerà. Se invece posizioniamo una stazione di monitoraggio sul terminale D o il terminale D è esclusivamente una stazione di monitoraggio riceverà esattamente quello che viene inviato da A e da B.

In realtà, il meccanismo non è così semplice, perché mentre quella figura vale al 100%, nel caso dell'hub non vale al 100% nel caso di una rete wireless. Nel caso di wi-fi bisogna fare attenzione alle onde radio in questione. Nelle tecnologie moderne vengono utilizzate le cosiddette antenne direttive che ci permettono di concentrare la potenza in una specifica direzione. Questo vale per le ultime versioni del Wi-fi e per i dispositivi moderni.

Mentre nel caso dello switch esso risolve i problemi dell'hub. Tipicamente lo switch non ha a disposizione la possibilità di installarci una sonda e generalmente sono di due tipi: managed e unmanaged. Gli switch unmanaged sono quelli con cui non

possiamo farci niente e non possiamo configurarli, mentre quelli managed sono quelli che possono essere configurati e a cui possiamo attivare il mirroring delle porte. Lo switch, generalmente, non può effettuare la cattura a meno che non sia realizzato tramite un dispositivo virtuale.

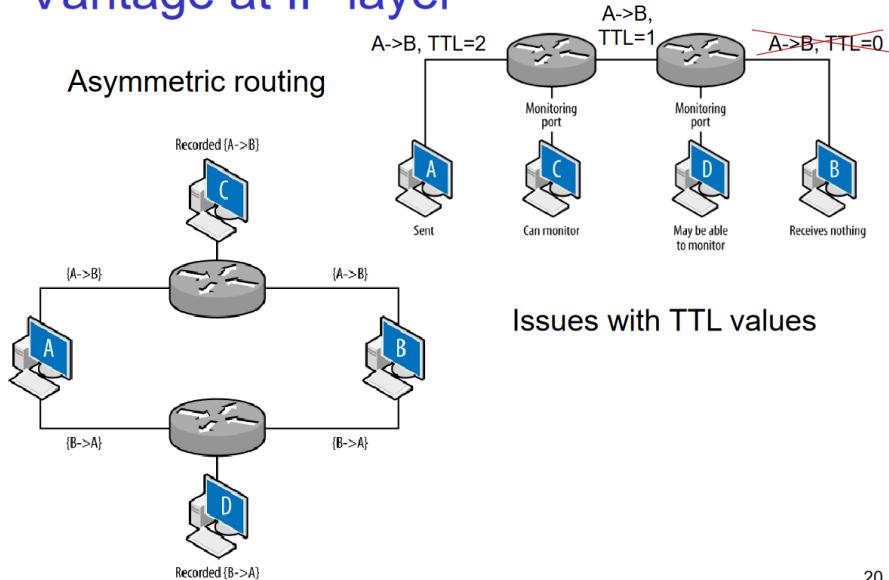
Nel caso di switch fisico ho due possibilità:

- Utilizzare una mirrored port (o spanning port) per duplicare il traffico che voglio analizzare. Quindi se voglio vedere il pacchetto che il terminale A invia a B, non posso mettermi né su C, né su D. In questo modo posso dire allo switch che sono interessato a copiare tutti i pacchetti che vengono da A e da B per vedere tutto il traffico scambiato. Ovviamente in base al verso che viene scelto verrà perso del traffico.
- Se lo switch raggiunge il numero massimo di porte che possono andare in mirroring o non supporta il mirroring si possono utilizzare le **Physical Tap**. Esso è un dispositivo fisico, tipicamente a più porte, che copia fisicamente tutto il traffico che passa dalla porta 1 alla porta 2 e ad esempio lo scrive sulla porta 3. Le Physical Tap vengono posizionate vicino allo switch e si collega il cavo che collegava il dispositivo D (stazione di monitoraggio) allo switch, alla physical tap. Quindi avremo un collegato che collega il dispositivo D con la Physical Tap e la Physical Tap al Router.

Quindi, abbiamo capito che se voglio catturare del traffico allo strato 2, ho bisogno di interagire con lo switch o bypassare lo switch attraverso un dispositivo fisico di cattura. Nel caso wireless invece ho bisogno di agganciarmi alla stazione radio base/access point o mettere in opera una stazione di cattura che funziona bene tutte le volte che è molto vicino alla stazione radio base/access point, poiché tutte le comunicazioni sono mediate da loro (access point/stazione base) che ricevono i segnali e li ritrasmettono, e se la stazione di monitoraggio ci è vicino riesco a catturare la maggior parte del traffico.

## Vantaggi strato di Rete (IP)

### Vantage at IP layer



20

Il primo problema da affrontare è capire se l'instradamento del traffico è simmetrico (come spesso accade) o asimmetrico. Nel caso in cui è asimmetrico potrebbe non essere sufficiente piazzare una sonda su uno dei router del cammino, perché il cammino di andata e il cammino di ritorno possono essere diversi. Quindi, se io sono interessato soltanto ad un verso specifico del traffico non ci sono problemi, ma se sono interessato alla sessione quindi ad entrambi i versi devo stare attento che il mio instradamento sia asimmetrico oppure devo fare in modo di piazzare una sonda in una parte del percorso condiviso dall'up-link e dal down-link. L'esempio in figura in basso a sinistra è estremo; entrambi i calcolatori A e B sono connessi a due router e l'instradamento è asimmetrico. Quindi se io metto una stazione di monitoraggio in C vedrò solo il traffico da A a B, diversamente se la metto in D vedrò solo il traffico da B ad A. Un ulteriore problema si ha con il TTL perché ogni router abbassa di 1 il TTL del pacchetto e quindi si potrebbe non catturare il pacchetto se il TTL arriva a zero oppure posso avere problemi legati alle tabelle d'instradamento (mettendo il sensore su un collegamento che i router considerano non buono, non vedrò nulla là dentro) e al loro malfunzionamento (se iniziano a flippare le routing table, il sensore che pensavo di aver messo nel punto giusto ora non lo è più).

## MIDDLEBOXES

Il problema principale non sono tanto i percorsi asimmetrici o le problematiche derivanti dal TTL, quanto la presenza delle middleboxes. Le middleboxes sono scatole che stanno nel mezzo che fanno tutte quelle funzioni che tipicamente non dovrebbero essere fatte da un router. Nel corso degli anni, da quando è nato Internet si è visto come queste funzioni (decisionali e attuative dei router) sono spesso insufficienti e quindi sono state aggiunti ai dispositivi intermedi(router) un notevole numero di funzioni che permettono di erogare dei servizi più avanzati ma che allo stesso tempo complicano in maniera significativa il compito del monitoraggio. In particolare, quando un router non si comporta come un dispositivo piatto di strato 3, ovvero instrada i pacchetti e basta ma svolge altre funzioni, possiamo avere i seguenti problemi:

- **problemi di identità:** non sempre riesco a ricostruire i processi di mapping interni e perdo la possibilità di ricostruire il percorso/la sorgente dei dati;
- **problemi di causalità:** non tutte le richieste che arrivano da un lato della middlebox vengono osservate dall'altro lato;
- **problemi di aggregazione:** la stessa identità può essere utilizzata per identificare più individui contemporaneamente.
- **problemi di consistenza:** l'identità degli utenti può cambiare nel tempo e devo stare attento a quello che vedo e a quando lo vedo;
- **problemi di cifratura:** alcuni dati potrebbero essere criptati (Livello IP). Allora i tools di analisi dei pacchetti/flussi potrebbero non funzionare.

## ○ Identity

- In some situations, the identity of individuals is not discernible because the information used to identify them has been remapped across boundaries

## ○ Causality

- Information after the middlebox boundary does not necessarily follow the sequence before the middlebox boundary

## ○ Aggregation

- The same identity may be used for multiple individuals simultaneously

## ○ Consistency

- The same identity can change over the duration of the investigation

## ○ Encryption

- When traffic is contained within an encrypted envelope, deep packet inspection and other tools that rely on payload examination will not work

Andiamo a esaminare una per una le funzionalità tipiche delle middleboxes per capire quali tipi di problemi causano.

❖ **Problema del NAT:** nel caso del NAT io posso avere vari schemi di natting (di mascheramento) e lo schema più classico, quello che si utilizza nelle piccole installazioni, è quello dei molti a uno.

- **Identità:** Quindi, ho un router che non agisce solo da router ma fa anche da NAT, a cui sono attaccati differenti client. In questo caso ho un problema d'Identità, perché guardando il traffico non sono in grado di sapere né l'identità dei singoli individui, né la porta mittente che stanno utilizzando. Questo a causa del NAPT;
- **Aggregazione:** Ho un problema di aggregazione perché tutti questi terminali, una volta che il loro traffico ha attraversato il NAT, vedranno rimappato l'indirizzo mittente dei propri pacchetti con l'indirizzo esterno del router che fa da NAT; quindi, un unico indirizzo viene utilizzato per mascherarne N. In questo caso il termine esatto è NAPT (Network Address and Port Translation) perché viene mappata la combinazione indirizzo

IP/porta mittente con indirizzo IP/ porta mittente dell'interfaccia esterna del router.

- **Consistenza:** ho un problema di consistenza perché nel tempo l'associazione tra indirizzo del client e l'indirizzo del NAT può variare. Questo non accade nel caso di mapping molti a uno, ma in alcuni casi potrei avere un pull di indirizzi e andare a rimappare le sessioni su un certo numero di indirizzi che ho a disposizione nei confronti della rete. Tipicamente questo accade quando il router è connesso ad altri due nodi diversi e in questo caso la mia sessione tra client e server la prima volta che viene effettuata sarà mappata su un collegamento, mentre la seconda volta (dopo 2minuti ad esempio) una sessione analoga potrebbe essere mappata attraverso un altro indirizzo, portando un problema di consistenza.

❖ **Problema del DHCP:** il DHCP ci dice che in alcune situazioni l'identità degli individui non è discendibile perché ho un'operazione di remapping. In realtà l'operazione non è esattamente di remapping ma il problema è dovuto allo stesso tempo da un problema di **identità e consistenza**, perché in presenza di DHCP uno stesso nodo può accedere ad una rete in momenti diversi e potrà avere un indirizzo IP diverso;

❖ **Problema del Load Balancer:** il load balancer serve a smaltire delle richieste attraverso più server. Ad esempio, potrei avere un numero di client C<sub>1</sub>...C<sub>n</sub>, che accedono ad internet e arrivati ad una infrastruttura di servizio, io dirò al mio router che guardo le richieste e queste richieste vengono inviate ai server S<sub>1</sub>, S<sub>2</sub> o S<sub>3</sub> ad esempio. P.S: il load balancer non elabora la richiesta, guarda solo il pacchetto e quindi riesce a smaltire più velocemente.

- **Casualità:** ho un problema di casualità con il load balancer perché vedo tutte le richieste e se vado ad ispezionare tutte le richieste di uno specifico collegamento quello che vedo a monte è diverso da quello che vedo a valle perché il load balancer splitta le richieste su un numero potenzialmente variabile di istanze di servizio (cioè server).
- **Consistenza:** problema di consistenza perché in un momento di basso traffico potrei avere un'unica istanza di servizio e quindi il load balancer è come se fosse inattivo: tutto quello che vedo a monte è uguale a tutto quello che vedo a valle. Ma in un momento in cui traffico arriva al picco potrei avere che quello che vedo su uno specifico link a valle è un'informazione diversa di quello che vedo su uno specifico link a monte.

❖ **Problema del Proxy:** il proxy ha gli stessi difetti dell'load balancer, con qualcuno in più. Supponiamo di avere un sistema dove ho un proxy aziendale, il proxy può girare sia sul router oppure il router può essere configurato per ridirezionare determinate richieste di tipo applicativo ad un'altra macchina che fa da Proxy. Quello che io vedo in ingresso guardando in maniera aggregata questo

link può essere diverso da quello che monitoro da un'altra parte della rete, assumendo di avere il proxy.

- **Identità:** la richiesta al server remoto viene fatta dal proxy, perché le richieste del client vengono terminate sul proxy a strato applicativo e poi il proxy fa le richieste. Quindi, chi fa la richiesta dopo la middlebox è diverso da chi fa la richiesta prima della middlebox;
  - **Casualità:** non tutte le richieste che vengono fatte prima corrispondono alle richieste che vengono rigirate dopo. Questo perché se il proxy ha il vostro contenuto ve lo serve in maniera diretta (Non capito molto);
  - **Aggregazione:** N richieste possono dare origine ad una richiesta in uscita. Questo perché se richiedo ad esempio la home page della gazzetta dello sport e il mio proxy non la ha, la andrà a recuperare sul sito la metterà in memoria. Successivamente, il secondo utente che richiederà la home page della gazzetta verrà servito direttamente dal proxy, così come il terzo, il quarto etc...
  - **Consistenza:** l'identità cambia nel tempo, perché non sono sempre gli stessi utenti ad innescare le richieste.
- ❖ **Problema delle VPN:** Supponiamo di avere un certo numero di client, connessi ad un router che accedono ad internet. Le VPN possono essere fatte in due modalità diverse:
- **Road Warrior:** (ovvero quella fatta quando utilizziamo il client per creare una VPN verso un server remoto e a quel punto possiamo accedere alle risorse della rete remota) in cui il client acquisisce un ulteriore indirizzo virtuale della VPN compatibile con la rete remota alla quale si collega. In questa modalità non ho molti problemi, ho solo il problema della cifratura perché l'indirizzo con cui contatto il VPN gateway è il mio indirizzo reale.
  - **Gateway to Gateway:** Il problema maggiore si ha quando la cifratura non la fa il client ma i router. Questo accade quando ho un'infrastruttura distribuita e ad esempio ho dei server in parte ospitati da una parte e in parte ospitati da un'altra e per far vedere questi server come se fossero tutti su una LAN utilizzo una VPN, di cui però i client e i server non devono necessariamente essere al corrente e quindi si crea un link cifrato tra le interfacce dei router che agiscono sia come router che come VPN gateway, e quello che viene cifrato è il collegamento IP esterno. Quindi, tutti i pacchetti che vanno da un nodo ad un altro verranno mascherati tra gli indirizzi IP mittente di un'interfaccia e gli indirizzi IP destinazione dell'altra interfaccia, così da creare un tunnel cifrato a strato IP o applicativo. Questo causa svariati problemi come: problema di identità perché ho un remapping, di aggregazione perché N indirizzi IP vengono visti come un unico indirizzo IP, di consistenza perché indirizzi diversi vengono mappati sullo stesso indirizzo ma poi ci possono essere VPN diverse e quindi

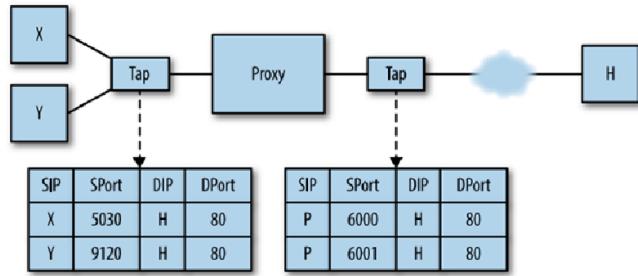
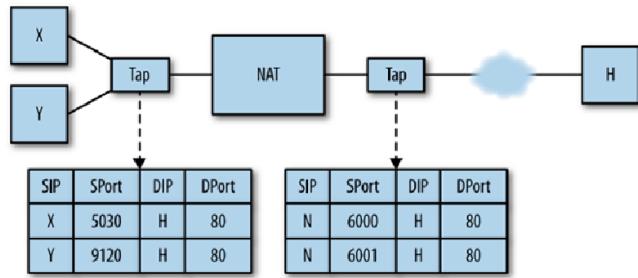
possono avere in momenti diversi identificativi diversi e infine il problema della cifratura che crea tutti gli altri problemi quando viene applicata questa modalità (chiamata anche Router to Router o VPN Gateway).

	Identity	Causality	Aggregation	Consistency	Encryption
NAT	X		X	X	
DHCP	X			X	
Load balancer		X		X	
Proxy	X	X	X	X	
VPN	X		X	X	X

In questi due schemi, vediamo come bypassare questi problemi (caso del NAT e del Proxy). La soluzione è estremamente semplice, per bypassare il problema ho bisogno non di osservare un unico link, quello a monte o quello a valle, ma ho bisogno di osservali entrambi. Bisogna mettere un rubinetto a cui attacco la mia sonda sia prima che dopo, così da poter capire se il meccanismo di NAT o di Proxy stia funzionando correttamente. Nel caso del NAT e del Proxy per capire se stiano funzionando bisogna mettere una sonda a monte e a valle e se riesco e se se posso vado ad analizzare i log del NAT e del Proxy (un plus molto utile). Altrimenti, se vado ad osservare una parte sola non avrò una visione chiara di quello che sta accadendo. Quindi, solo visionando l'ingresso e l'uscita riesco a capire se il mapping funziona e riesco a bypassare i vari problemi citati in precedenza. Ovviamente questo comporta un onere maggiorare dal punto di vista dell'elaborazione, archiviazione e trasmissione del traffico, soprattutto se queste operazioni non sono fatte in locale ma su un nodo dislocato.

P.S Chiaramente nel caso del DHCP che è un caso leggermente diverso, essendo un servizio, basta monitorare i log del server DHCP.

## Vantage at IP layer



## Recap:

Nella scorsa lezione abbiamo finito di vedere il secondo attributo che caratterizza un dispositivo che viene utilizzato come sonda per il traffico di rete ovvero il VANTAGGIO. Due lezioni fa abbiamo visto il concetto di dominio ovvero che cosa viene catturato. Nelle ultime due lezioni abbiamo approfondito il concetto di vantaggio, ovvero come la posizione della sonda all'interno della topologia sia logica che fisica della rete, influenza i dati che andiamo ad acquisire quindi anche l'informazione che riusciremo ad estrarre dai dati.

Un terzo attributo che dobbiamo considerare per le nostre sonde è l'attributo che va sotto il nome di **azione**. L'azione di una sonda descrive come il sensore interagisce con i dati che raccoglie, e in base ciò possiamo classificare i sensori in 3 macrocategorie: sensori che restituiscono un report generico, sensori che forniscono le indicazioni di occorrenza di un evento e sensori che oltre ad acquisire informazioni in caso di occorrenza di un evento sono in grado di controllare il traffico catturato ed eseguire delle operazioni in risposta all'evento scatenante.

Partiamo quindi dal primo caso:

### - Report Sensor

Il report è il modo più semplice per interagire con il traffico, di fatto, quando un sensore genera dei report generare un file che di fatto riporta tutto il traffico. Le caratteristiche del traffico catturato variano a seconda del tipo di report, possiamo infatti suddividere ulteriormente i report in 3 categorie:

- Report TCPDUMP ovvero un file *.pcap* che può essere analizzato da wireshark o tshark per estrarre informazioni; questo file riporta tutti i pacchetti che vengono catturati dalla sonda oppure tutti i pacchetti che rispettano eventuali filtri di catturare della sonda.
- Report Netflow, ovvero un report che si approccia all'analisi del traffico ad un livello di astrazione più elevato rispetto a quello dei singoli pacchetti, permettendo di raggruppare pacchetti "omogenei" nel concetto di **flusso**. Un flusso, quindi, è un insieme di metadati che rappresenta un gruppo di pacchetti che hanno caratteristiche comuni. Anche un sensore che genera un report Netflow, raccoglie le statistiche per tutti i flussi che sono catturati, indipendentemente dal fatto che abbiano una qualche rilevanza statistica o no.
- Server Log, ovvero un report associato a tutte le richieste che il server ha ricevuto, questa tipologia di report risulta essere molto utile in quanto i dati collezionati da questi tipi di sonde sono i più completi che possano avere. D'altro canto, un log potrebbe avere necessità di una dose molto significativa di post-processing per estrarre poi l'informazione di interesse.

Se non è presente uno specifico target nell'analisi dei dati, generare un report omnicomprensivo è sicuramente una buona soluzione, in quanto permette di effettuare analisi molto diverse. Per esempio è possibile sviluppare delle signature, tipicamente associate a dei metadati di pacchetti o di flussi, che rappresentano degli attacchi informatici; per far ciò è richiesta una pesante elaborazione oltre alla cattura dei pacchetti,

infatti nel momento in cui viene riconosciuto un attacco che si presenta sempre con le stesse caratteristiche è possibile definire una signature, o in altre parole identificare un insieme di metadati che caratterizzano un attacco, in modo che la prossima volta che questi metadati vengono riconosciuti sarà possibile individuare un attacco prima che avvenga o comunque in una primissima fase di attacco. Lo stesso concetto è applicabile a tutti e 5 gli ambiti del Network Management.

#### - **Event Sensor**

Il secondo tipo di sonda è il cosiddetto sensore ad evento. A differenza del sensore a report che cattura tutto il traffico indistintamente, un sensore ad evento riesce a distinguere tra eventi di interesse e eventi di non interesse, andando quindi a catturare solo i pacchetti legati ad un determinato evento, quando questo si verifica. Questo approccio può essere utile per sviluppare algoritmi intelligenti che anche osservando tutto il traffico, riescono a generare dei report più sintetici legati soltanto a delle condizioni che individuate a priori. Se per esempio, in una situazione in cui sono presenti dei server, sia virtuali che fisici, si può realizzare un sensore che genera un report solo quando si ha un intervallo temporale di 20 secondi con un consumo di risorse medio superiore al 70%, questa situazione infatti potrebbe indicare un sovraccarico dei server. Allo stesso modo, si possono utilizzare delle signature di attacchi informatici noti, precedentemente generate con un report completo, per configurare dei sensori che analizzando il traffico riescano a generare dei report mirati che mostrino soltanto quando un determinato evento si va a verificare, come per esempio il riconoscimento della signature di un attacco imminente. Per quanto riguarda la sicurezza, questi meccanismi sono implementati nei sistemi IDS (Intrusion Detection System) e nei sistemi antivirus. Tipicamente i sistemi antivirus non generano un messaggio quando è presente del traffico legittimo ma soltanto quando viene rilevato un potenziale virus, allo stesso modo gli IDS, generano un segnale di allarme quando avviene un'intrusione e non per ogni pacchetto che arriva.

#### - **Control Sensor**

I sensori che esercitano un'azione di controllo lavorano allo stesso modo dei sensori ad evento, ma riescono anche a mettere in piedi delle azioni sulle basi degli eventi rilevati, ovviamente in base agli eventi che vado a rilevare avrò delle azioni differenti. Quindi azioni in domini diversi potrei aver bisogno di sensori di controllo diversi. Infatti, se ci sono dei sensori che guardano soltanto lo strato 2 avrò delle azioni relative solo allo strato 2, mentre se ci sono dei sensori che analizzano solo i log, avrò delle azioni che riguardano solo i server che generano quei log. Al solito il concetto di dominio e azione, quindi la relazione di controllo è strettamente legata al dominio, ovvero al tipo di informazione che vado a catturare. Esempi di sensori che possono esercitare un controllo sulla base delle informazioni estratte. Un esempio di sensore di controllo è un IPS (Intrusion Prevention System) che è un'estensione del IDS e permette non solo di rilevare una potenziale minaccia ma anche di mettere in campo una contromisura per evitare che la minaccia possa diventare dannosa. Per quanto riguarda i firewall invece, nascono per esercitare una operazione di protezione e restrizione dell'accesso alla rete, quindi di fatto sono dei dispositivi statici configurati per riconoscere alcune situazioni indesiderate e preconfigurati per bloccarle. Queste operazioni di bloccaggio si possono attuare, per esempio, tramite la config di IPtables che possono impedire il passaggio dei pacchetti sia in ingresso che in uscita, anche quelle che transitano solamente per la macchina.

La situazione illustrata nella parte alta della slide è una situazione di traffico normale, mentre la situazione in basso è la situazione di traffico di rete associata ad un attacco informatico. In questo caso specifico si tratterà il caso degli attacchi informatici ma le stesse tecniche si possono applicare ad altri dominio del Network Management (FCAPS).

Partendo da sensore in alto a sinistra ovvero dalla sonda di tipo R, questa analizza il traffico e che genera dei report completi. In questo modo, il calcolatore che interagisce con internet vedrà lo stesso traffico che vede la sonda, quindi la sonda andrà periodicamente a generare dei report dai quali, una volta analizzati, sarà possibile determinare lo stato del traffico. Il report, quindi, cattura solo delle informazioni ma non da giudizi.

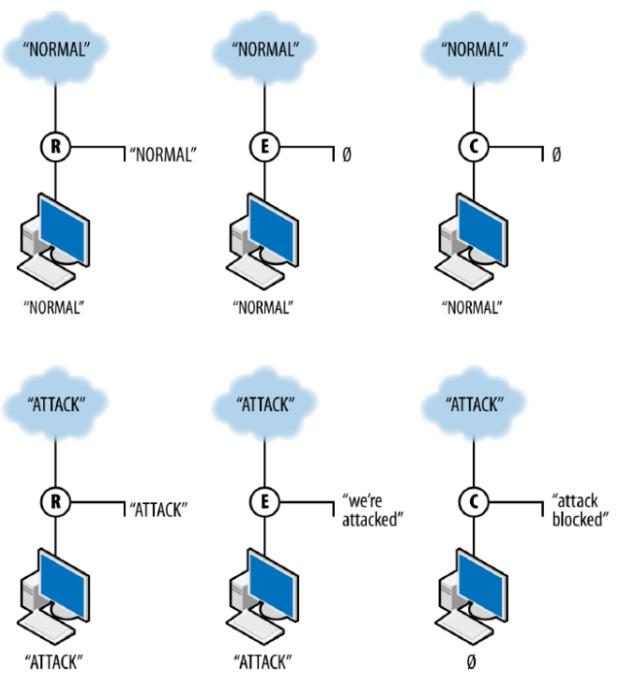
Sempre sulla riga in alto, al centro è presente la stessa situazione ma con un sensore ad eventi. Anche in questo caso è un sensore passivo, che vede lo stesso traffico che viene scambiato tra internet e il calcolatore. Ha come output l'insieme vuoto, in quanto il sensore è configurato per generare dati in output in caso di occorrenza di determinati eventi, nell'esempio in questione, il sensore genera dati in uscita se si presenta un problema di sicurezza, ma essendo il traffico normale non genera nulla. Il sensore con l'etichetta C è un sensore sia passivo che attivo, anche se in questa situazione di traffico normale lavora solo come sensore passivo, non effettua quindi alcuna azione di controllo e non genera report.

Nella parte in basso della figura vengono mostrati i 3 tipi di sensore in una situazione di attacco che viene effettuato da qualcuno che sta in rete rispetto al calcolatore che invece subisce l'attacco. Nel primo caso utilizzando una sonda che effettua solo un report, si ottengono delle informazioni, che solo una volta elaborate mi diranno che sto subendo un attacco. In questo caso quindi il calcolatore subisce un attacco, in quanto il sensore è passivo e al massimo genera dei report che una volta analizzati permettono di identificare se c'è stato un attacco. Nel secondo caso, il sensore configurato ad evento, genera uno specifico messaggio che avvisa di essere sotto attacco, quindi si parla sempre di sensore passivo, ma in questo caso quando il calcolatore subisce l'attacco il sensore invierà questo messaggio che avvisa dell'attacco in corso. Rispetto al primo tipo di sensore in cui l'attacco è identificabile solo dopo il post-processing del report, questo sensore genera esso stesso l'informazione che identifica l'attacco.

Nel terzo caso, il sensore che rileva l'attacco può agire in due modi:

- L'attacco viene rilevato ma non è possibile bloccarlo
- L'attacco viene rilevato e sono disponibili contromisure per bloccare l'attacco, il sensore si comporta quindi come un IPS.

Quindi anche in questo caso il sensore reagisce a degli eventi ma riesce a mettere in campo delle contromisure.



## **NOTA – Perché non si usano sempre i sensori C?**

I sensori di controllo costano di più e non sempre si è in grado di avere un tipo di sensore che mette in piedi le contromisure tutte le volte che servirebbe. Un ulteriore motivo, che è non meno importante, è che se ho un sensore configurato per rilevare gli eventi e bloccare potenzialmente degli eventi indesiderati, o per reagire a delle situazioni indesiderate, è vero che avrò notifica che mi segnala le azioni compiute dal sensore, come tipicamente l'antivirus con un popup indica di aver bloccato un virus, ma un sensore di questo tipo non genera il report contenente le statistiche complete del traffico, che risultano essenziali nel caso di attacchi zero-day.

Gli attacchi zero-day infatti sono degli attacchi mai visti prima ed è quindi impossibile rendersi conto di un attacco se si utilizzano solamente sensori di tipo E e C. Questa tipologia di attacco è identificabile solo tramite post processing di un report completo del traffico e dopo aver subito molteplici attacchi dello stesso tipo. Lo stesso discorso si può applicare agli altri ambiti del Network Management, come una situazione di malfunzionamento mai vista, soprattutto a seguito di un aggiornamento software. Quindi volendo costruire un sistema di sonde più funzionale possibile, la situazione ideale sarebbe quella in cui, oltre ad un sistema di reportistica che catturi tutti il traffico e permetta un'analisi sul lungo periodo, sia presente anche un sistema potenzialmente attivo in grado di fornire un'analisi più immediata.

È importante quindi non sottovalutare i sensori che generano l'archivio della piattaforma per l'elaborazione dei dati in fase di post-processing, soprattutto quando si ha la necessità di identificare situazioni mai viste prima.

## **Formato dei dati**

### **Tracce RAW (File PCAP)**

Le tracce RAW sono quelle catturate con TSHARK o TCPDUMP con o senza filtri di cattura, i quali permettono di limitare la quantità di informazioni che vengono catturate, evitando che la quantità di traffico generata sia troppo abbondante e il sistema di monitoraggio non sia in grado di gestirla.

Le tracce di tipo RAW, generalmente, contengono solo l'intestazione dei pacchetti catturati in quanto la maggior parte dei pacchetti in rete sono cifrati e di conseguenza risulta inutile catturare il payload cifrato di un pacchetto.

### **Eventi (Pre-processed Data)**

Si parla di eventi quando si fa riferimento statistiche di rete elaborate. Per evitare confusione è importante fare una precisazione: quando si fa riferimento ai dati grezzi (tracce RAW) si parla di report, ma bisogna parlare di report anche quando si fa riferimento a dati pre-processati, come per esempio tramite l'utilizzo di linguaggi di scripting, in quanto questi dati sono ancora completi anche se ne viene nascosta una parte e mostrata sono quella di interesse. Si parla invece di evento quando i dati raccolti vengono elaborati in tempo reale o l'elaborazione viene svolta prendendo in esame solamente i dati di interesse, andando quindi a perdere tutte le altre informazioni sul traffico che non interessa l'evento in questione.

## LOG

Quando si parla di log si fa riferimento ai log di tutti i tipi di server (web server, server proxy, ...). I log a loro volta possono essere di due tipologie: completi (o RAW) oppure pre-elaborati. Quelli completi molte informazioni, come per esempio informazioni sul client, sul ip e sulla geolocalizzazione. Se invece si parla di log pre-elaborati si fa riferimento a dei dati in cui viene eliminato tutto ciò che non è considerato utile e lasciato solo ciò che si ritiene fondamentale a seguito di una pre-elaborazione.

È importante sottolineare che, quando si parla di log, si parla sempre di pre-elaborazione. In alcuni casi i server hanno la possibilità di generare dei log configurabili tramite i cosiddetti log-level.

Generalmente i log sono configurabili su 4 livelli: Info, Normal, Warn ed Error; nel caso in cui si configurino solo i livelli Warn ed Error, il log si comporterà come un sensore ad eventi, in quanto andrà a notificare e mostrare le statistiche solo nel caso in cui si presenti un determinato evento che faccia scaturire la generazione del log. È importante tenere a mente che quando si parla di log bisogna sempre essere consapevoli del processo che li genera e se questo risulta configurabile.

## Flow-based Information

Per le informazioni organizzate per flusso abbiamo come standard Netflow, acronimo di network flow, è allo stesso tempo un protocollo, un formato dati, un architettura di monitoring e un applicazione in senso lato (si occupa di catturare metadati relativi al traffico IP).

## Netflow

Netflow è stato originariamente sviluppato da Cisco e poi standardizzato dal Network engineering task force.

Quando si ad identificare un insieme di pacchetti sotto il concetto di flusso, non salvano le caratteristiche dei singoli pacchetti ma si lavora ad un livello di astrazione più elevato, andando quindi a salvare solo le caratteristiche del flusso che vengono raccolte in un gruppo di parametri e dati che prende il nome di metadati. Si parla di metadati in quanto non sono i veri e propri dati contenuti dai pacchetti ma sono dati estratti dai pacchetti, che di fatto ripartano i dati comuni ai pacchetti.

Tipicamente i metadati estratti sono: indirizzo ip sorgente e destinazione, informazioni sull'instradamento, sistema autonomo sorgente e destinazione, porte strato 4 sorgente e destinazione, protocollo strato 4, numero di pacchetti, numero medio di KB per pacchetti, KB associati al flusso e next hop (ovvero l'informazione su dove tutti i pacchetti associati al flusso saranno inviati al prossimo salto).

Si può suddividere Netflow in 3 componenti principali, di cui due obbligatorie e una terza opzionale:

## - **Flow Exporter (Dispositivo di rete)**

Tipicamente uno switch o un router in grado di generare delle statistiche Netflow, quindi un dispositivo che viene attraversato dal traffico (stiamo parlando di sistemi terminali) riesce ad identificare all'interno del traffico che smaltisce un determinato numero di flussi e generare per ognuno di questi dei metadati che vengono poi esportati tramite un protocollo standardizzato.

Il protocollo di network monitoring utilizzato, sfrutta UDP come protocollo di trasporto verso le altre entità che prendono nomi di collettori (relazione molti a uno, pochi collettori, collezionano statistiche di un elevato numero di dispositivi Netflow)

## - **Flow Collector**

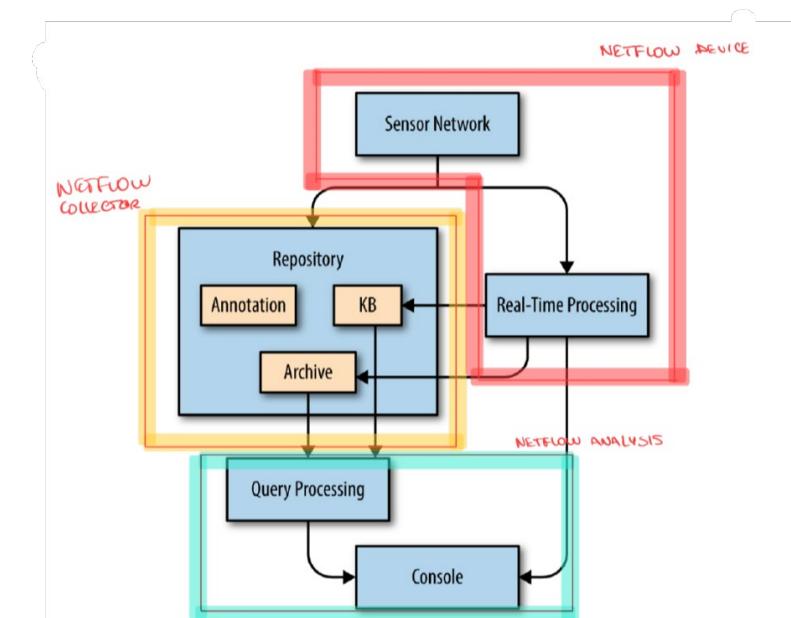
I metadati che sono raccolti dai collettori (non ci stanno tracce di pacchetti, infatti i dispositivi in tempo reale estraggono i metadati dal pacchetto, quindi il contenuto dei singoli pacchetti è perso, in quanto qui si guarda solo il concetto di flusso. Tutte le informazioni per identificare quali pacchetti appartengono ad un determinato flusso, si ottengono non dall'analisi del pacchetto ma dal numero di porta e dal protocollo di trasporto).

## - **Flow Analyzer (Componente Opzionale)**

Una volta che i metadati sono memorizzati su uno o più collettori, questi dati possono essere recuperati da una console di analisi che può estrarre delle statistiche aggregate.

Riprendendo il diagramma in figura si può mappare Netflow all'interno di questa architettura del tutto generica

- **Sensor network**: dispositivo che osserva il traffico e esporta i metà dati di Netflow.-
- **Real-time processing** è embedded all'interno della rete dei sensori, poiché il sensore osserva il traffico (i pacchetti) in tempo reale, ne effettua una elaborazione e la spedisce al collettore
- **Repository**: collettore dove vengono inviati non questi dati ma il risultato dell'elaborazione in tempo reale.
- **Query processing** e **Console** sono i componenti che si occupano di effettuare la Netflow analysis.



Quindi la cattura del traffico e l'analisi in tempo reale vengono effettuate direttamente sul dispositivo di rete in quanto il real-time processing avviene all'interno del dispositivo, a questo punto il dispositivo invia al collettore soltanto il risultato dell'elaborazione. È possibile infine interrogare il repository per andare a vedere quali sono queste informazioni raccolte da Netflow.

#### **NOTA – a che cosa serve Netflow?**

È vero che si perdono informazioni a livello di pacchetto, però è utile in casi come le reti ad alta velocità, dove sono presenti un gran numero di pacchetti e sarebbe impossibile archiviarli tutti se non con uno storage infinito.

#### **NOTA – Cosa mi permette di vedere Netflow?**

Tramite Netflow è possibile identificare quanti flussi un dispositivo di rete trasporta, è importante per capire se una rete è attraversata da pochi flussi a banda molto elevata (cosiddetti elefanti) o tanti flussi a banda esigua (o durata limitata) (chiamati topolini) oppure un mix di entrambi. Andando ad analizzare le informazioni offerte da Netflow, sicuramente si possono avere delle informazioni sulla portata della rete e sul livello di congestione del traffico, è possibile vedere per flusso ma potenzialmente anche tramite una vista aggregata. Se per esempio, in un nodo della rete è presente un numero anormale di tentativi di accesso su un determinato servizio caratterizzato da flussi di breve durata, posso accorgermi che è in corso un attacco DOS che tenta di mandare in saturazione le risorse di calcolo o di rete di un dispositivo.

## **SFlow**

Esiste approccio alternativo a Netflow, ma dello stesso tipo, che si chiama sflow, dove la S sta per sampling. La differenza fondamentale tra Netflow e sflow è che Netflow genera le statistiche guardando tutti i pacchetti del flusso, sflow invece non osserva tutti i pacchetti del flusso ma effettua un campionamento con una frequenza calcolata tramite un rapporto predefinito, di pacchetti osservati su pacchetti campionati. Dal momento in cui un pacchetto viene campionato, questo viene analizzato integralmente, in questo modo SFlow permette di vedere e acquisire informazioni che Netflow non permette di catturare. Netflow è orientato agli strati 3 e 4, quindi per esempio non tiene conto degli indirizzi di strato 2; SFlow invece analizza tutto il pacchetto, ma decide a priori ogni quanti pacchetti osservati deve campionare. SFlow genera quindi informazioni simili a Netflow, in alcuni casi anche più accurate, proprio perché contengono più informazioni. Dal punto di vista dei metadati generati sono abbastanza simili. Netflow è stato standardizzato da Cisco e ne esistono approcci molto simili generati da dispositivi di altri costruttori come Jflow e Netstream (rispettivamente Juniper Network e Huawei). In generale tutti questi dispositivi sono compatibili con Netflow e se vado ad utilizzare protocolli proprietari in reti in cui ho tutti dispositivi di quel costruttore, potrò avere informazioni aggiuntive. Nel corso degli anni il Netflow è arrivato alla versione 9, mentre è stato standardizzato dalla Network Engineering Task Force con la versione 10 chiamata IPFIX (ovvero una versione vendor independent). Tutti i dispositivi sono quindi compatibili con la IPFIX e la versione 5 di Netflow.

## Flusso

Un flusso è definito in Netflow come un insieme di pacchetti con lo stesso indirizzo di strato 3 e 4 raggruppati nel tempo. Questo meccanismo ricorda le connessioni TCP, quindi si può immaginare il flusso di Netflow come una sorta di approssimazione della connessione TCP. Per definire una connessione TCP è necessario andare ad intercettare i pacchetti del three-way handshake e osservare i numeri di sequenza iniziali generati casualmente. In questo modo è possibile stabilire se un pacchetto appartiene ad un flusso in base al numero di sequenza superiore o inferiore a quello stabilito in fase di handshake. Ovviamente bisogna tenere conto che i numeri di sequenza vengono realizzati tramite un numero finito di bit e sono quindi circolari, una volta finite le combinazioni si riparte da 0. Quindi se ho un numero di sequenza inferiore a quello iniziale deve essere trascorso parecchio tempo ed essere compatibile con la velocità di trasferimento dei bytes. A differenza di TCP, Netflow non utilizza i numeri di sequenza, poiché sarebbe estremamente complicato e di conseguenza dispendioso a livello di risorse di calcolo. Tipicamente i router e switch effettuano operazione di istradamento e commutazione attraverso dei processori collocati sulle schede di rete, hanno poi hanno un'architettura di calcolo general-purpose che serve e a gestire il dispositivo e fase altre operazioni tra cui quella di generare le statistiche utilizzando la cpu e la ram associate al dispositivo e non alle schede di rete. Quindi l'operazione di catturare delle statistiche Netflow deve essere fatta su dispositivi che non abbiano un consumo di risorse di calcolo troppo elevate altrimenti non sarebbe possibile effettuare la cattura in maniera corretta. Netflow raggruppa nel tempo tutti i pacchetti che hanno gli stessi indirizzi di strato 3 e 4. Da questa definizione ne deriva che i flussi Netflow sono unidirezionali anche se protocolli come TCP sono intrinsecamente bidirezionali. Per fare un esempio, durante il Three-Way Handshake, TCP apre una connessione in un verso e una in un'altra, mentre UDP non ha neanche il concetto di flusso quindi Netflow non è compatibile al 100% né con i flussi tcp né con i datagrammi udp, ma nonostante questo può essere applicato ad entrambi. Una classica sessione TCP sarà associata a due flussi unidirezionali Netflow, per raggruppare insieme questi due flussi che hanno in comune solo gli indirizzi di strato 3 e 4 invertiti, è possibile sfruttare la piattaforma di analisi. Quando si procede a fare l'analisi sulla piattaforma, si estraggono le informazioni in fase di post-processing andando anche ad identificare i flussi bidirezionali, i quali hanno in comune ip sorgente e destinazione, porte di strato 3 e 4, campo protocollo di strato 3 e 4, TOS (o il COS nel caso di ipv6) e Input Interface Port, ovvero la porta di ingresso dove i pacchetti sono ricevuti. Riprendendo la considerazione iniziale, ovvero quella per cui posso esserci flussi molto grandi che durano ore e flussi che durano qualche frazione di secondo. Mentre per i secondi non ci sono grossi problemi, se per esempio stiamo vedendo un film su una piattaforma di streaming avremo un flusso http attraverso il quale, senza mai chiuderlo, si fanno continuamente richieste alla piattaforma, si richiede un blocco di dati alla volta, per tutta la durata del film. Quindi la sessione dura all'incirca quanto il film, quindi in teoria il nostro export (dispositivo di rete che cattura i dati) dovrebbe aspettare 2 ore prima di esportare il flusso; ma non funziona esattamente così.

Il flusso viene chiuso ed esportato per una delle seguenti cause:

- **Flag FIN e RESET**

Quando osservo i flag FIN o RESET all'interno di un flusso TCP. Quindi se ho aperto il flusso TCP perché ho visto un pacchetto di SYN e poi uno di FIN, so che il flusso è stato

aperto e anche chiuso, quindi posso esportare i metadati associati a questo flusso che ha terminato la sua esistenza.

#### - Timer Inattività

Si imposta un timer che indica la durata di tempo massima di inattività di flusso, quando il timer scade il flusso viene interrotto ed esportato. Questi meccanismo, a differenza del primo funziona sia con TCP che UDP.

#### - Timer per flussi Long-Liver

L'ultimo caso è quello dei cosiddetti flussi elefanti o long liver, questa tipologia può riguardare sia un flusso TCP che UDP. Se quando il timer scade, il flusso è ancora attivo, si chiude la sessione di cattura e si inviano i metadati al collettore, contestualmente si apre una nuova sessione per acquisire i nuovi dati. Quindi si utilizza sempre un meccanismo basato sul tempo, se il flusso è rimasto vivo per un tempo pari a quello massimo si chiude la finestra di cattura e se ne apre un'altra. Questi metadati verranno inviati al collettore, ed una eventuale piattaforma di analisi dovrà riunire insieme flussi diversi che in realtà appartengono allo stesso flusso reale.

## Netflow Versione 5

I metadati che caratterizzano la definizione di flusso per Netflow v5 sono quelli descritti in figura. Tramite i seguenti dati è possibile calcolare la durata e i byte totali del flusso, ma anche il bit rate e informazioni sull'instradamento dei pacchetti tra sistemi (protocollo BGP). Come già visto Netflow è un'architettura di monitoraggio del traffico, un formato dei dati e un protocollo, in particolare, utilizza un protocollo proprietario costruito sopra UDP per veicolare le informazioni dall'exporter al collettore. Netflow versione 5 viene ormai considerata un pochino primitiva perché ad esempio, la versione 5 non riesce a gestire bene tutte le problematiche che abbiamo visto relative alla presenza delle cosiddette middle box, quindi se è presente un NAT Netflow potrebbe andare in confusione.

Bytes	Name	Description
0–3	srcaddr	Source IP address
4–7	dstaddr	Destination IP address
8–11	nexthop	Address of the next hop on the router
12–13	input	SNMP index of the input interface
14–15	output	SNMP index of the output interface
16–19	packets	Packets in the flow
20–23	dOctets	Number of layer 3 bytes in the flow
24–27	first	sysuptime at flow start <sup>a</sup>
28–31	last	sysuptime at the time of receipt of the last flow's packet
32–33	srcport	TCP/UDP source port
34–35	dstport	TCP/UDP destination port, ICMP type, and code
36	pad1	Padding
37	tcp_flags	Cumulative OR of all TCP flags in the flow
38	prot	IP protocol
39	tos	IP type of service
40–41	src_as	Autonomous system number (ASN) of source
42–43	dst_as	ASN of destination
44	src_mask	Source address prefix mask
45	dst_mask	Destination address prefix mask
46–47	pad2	Padding bytes

<sup>a</sup> This value is relative to the router's system uptime.

## Netflow Versione 9 e IPFIX (v10)

Nelle nuove versioni di Netflow, la 9 standardizzata da Cisco e la 10 da NETF permettono di catturare anche informazioni relative alle middle box, in particolare la versione 9 è la prima ad essere considerata template-based ovvero la prima versione mette a disposizione un template, ovvero una maschera di informazioni, tra le quali l'operatore di rete può decidere quali andare ad esportare verso il collettore. In particolare, la versione 10, ha un supporto completo ad IPV6 e dispone anche di campi speciali per i dispositivi che agiscono da NAT, fanno quindi vedere indirizzi IP tenendo conto che è presenti un meccanismo di natting, quindi non fanno vedere gli indirizzi mascherati, ma tutto.

## Netflow Export Techniques

Per generare i report Netflow posso utilizzare due approcci: tramite un dispositivo compatibile Netflow oppure, in caso contrario si può sfruttare una soluzione basata sul mirroring delle porte, funzionalità presente sia nei dispositivi di strato 2 che 3. Quindi se il dispositivo non supporta Netflow la soluzione è quella di selezionare le porte di ingresso su cui si vogliono generare le statistiche, ridirezionare il traffico associato a queste porte sperando che non causino overflow sulla porta di mirroring ed agganciare sulla porta di mirroring una stazione di cattura, tipicamente un server Linux, configurato per ricevere il traffico e generare le statistiche Netflow. Il principale svantaggio di questo approccio è che, se il dispositivo che intendo osservare ha un numero elevato di porte, potendo fare il mirroring di un numero limitato di porte potrebbe non essere possibile vedere tutti i flussi che passano nel dispositivo, si dovrà quindi andare a scegliere quali porte andare ad osservare. Una cosa classica da evitare assolutamente è registrare come flussi i pacchetti Netflow inviati dal dispositivo stesso o da quello associato, al collettore. Netflow è supportata da un sacco di dispositivi, che tipicamente sono di fascia abbastanza elevata, ma può anche succedere che i modelli più vecchi vadano in sofferenza a causa delle catture Netflow troppo pesanti. Non riuscendo a stare dietro a tutti i pacchetti che attraversano il dispositivo, alcuni pacchetti sfuggono e i metadati catturati ad esempio il numero di pacchetti (e il numero di byte), potrebbero essere considerate statistiche non particolarmente affidabili poiché non riesco ad osservare più tutti i pacchetti, che vengono parzialmente scartati dall'elaborazione del processore. Per ovviare a questo problema si potrebbe utilizzare un sistema di cattura a campionamento, come ad esempio Sflow. Delle volte i dati catturati ed elaborati tramite Netflow non sono abbastanza, è quindi buona norma, quando possibile, combinare i dati Netflow con i log dei server e delle middle-box.

Abbiamo visto che NetFlow è un protocollo che permette di catturare delle statistiche aggregate per flusso, dove flusso è definito in maniera unidirezionale e le statistiche aggregate sono ottenute andando ad esaminare tutti i pacchetti che attraversano un router o uno switch, in toto oppure focalizzandosi su una specifica interfaccia.

## sFlow

sFlow è un meccanismo alternativo il cui obiettivo è molto simile a NetFlow. Quello che cambia sono le modalità operative: mentre netflow esamina tutti i pacchetti per generare le statistiche del flusso, sFlow invece effettua le stesse stime ma basandosi su un processo di campionamento; quindi, non va ad analizzare tutti i pacchetti, ma va ad analizzare un sottoinsieme di questo.

sFlow sta per “sampled flow”, quindi “flusso campionato” e a differenza di NetFlow include anche le informazioni allo strato 2, non solo allo strato 3 e 4.

Il protocollo sFlow è stato originariamente sviluppato da InMon Corp e attualmente è mantenuto dal consorzio che mantiene anche il sito web sFlow.org

Fornisce un mezzo per esportare pacchetti troncati, insieme a contatori di interfaccia ai fini del monitoraggio della rete.

A differenza di NetFlow, si basa sul campionamento e non utilizza l'intero flusso. Si adatta meglio, ma potrebbe perdere alcune funzionalità di un flusso.

Per quanto riguarda NetFlow consideriamo che la sonda includa sia la capacità di sensing sia la capacità di preelaborazione e infatti all'interno del repository non scrivo il risultato del sensing ma il risultato dell'elaborazione fatta in locale (all'interno del router, dello switch o della stazione di monitoraggio).

Nel caso di sFlow la sonda effettua semplicemente un campionamento del pacchetto e, una volta che il pacchetto è stato selezionato per essere un campione, è il pacchetto stesso in modalità troncata (cioè solo con l'header) che viene inviato al repository. Poi all'interno del repository ci sarà la fase di elaborazione per archiviare il metadato. I singoli campioni vengono quindi inviati a una sorta di real-time processing, agganciato però al repository, e questi pacchetti campionati vengono elaborati e vengono salvate le statistiche.

All'interno del repository perciò non avremo i campioni dei pacchetti troncati, ma le statistiche che riesco ad estrarre dai pacchetti troncati.

I dispositivi devono riuscire ad identificare i pacchetti che devono essere inviati come “campioni troncati” al modulo di elaborazione che sta prima del repository. Insieme ai

pacchetti troncati devono inviare le statistiche relative al processo di campionamento; le statistiche sono dei semplici contatori (ad esempio dirò quanti pacchetti sono passati e quanti pacchetti ho campionato, in modo tale che chi elabora i pacchetti ha un'idea di quello che sta vedendo dal punto di vista quantitativo).

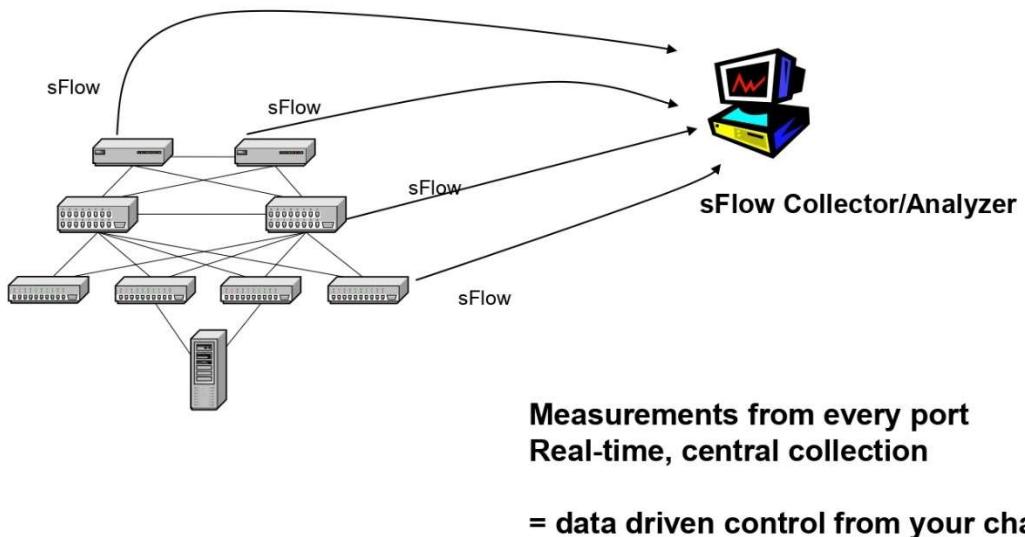
Vantaggio rispetto a NetFlow: riesco ad avere informazioni aggiuntive, che sono quelle di strato 2, e poi siccome posso definire in maniera arbitraria la rate di campionamento  $r$  (dati  $N$  pacchetti che passano, quanti ne vado ad osservare) riesco a stare dietro ai processi in maniera più affidabile

$$r = \frac{N}{n} > 1 , \text{ con } N \text{ numero di pacchetti totali e } n \text{ numero di pacchetti che campiono}$$

più è alto  $r$  e più il processo scala bene

Svantaggio rispetto a NetFlow: perdo informazioni ogni volta che non considero tutti i pacchetti

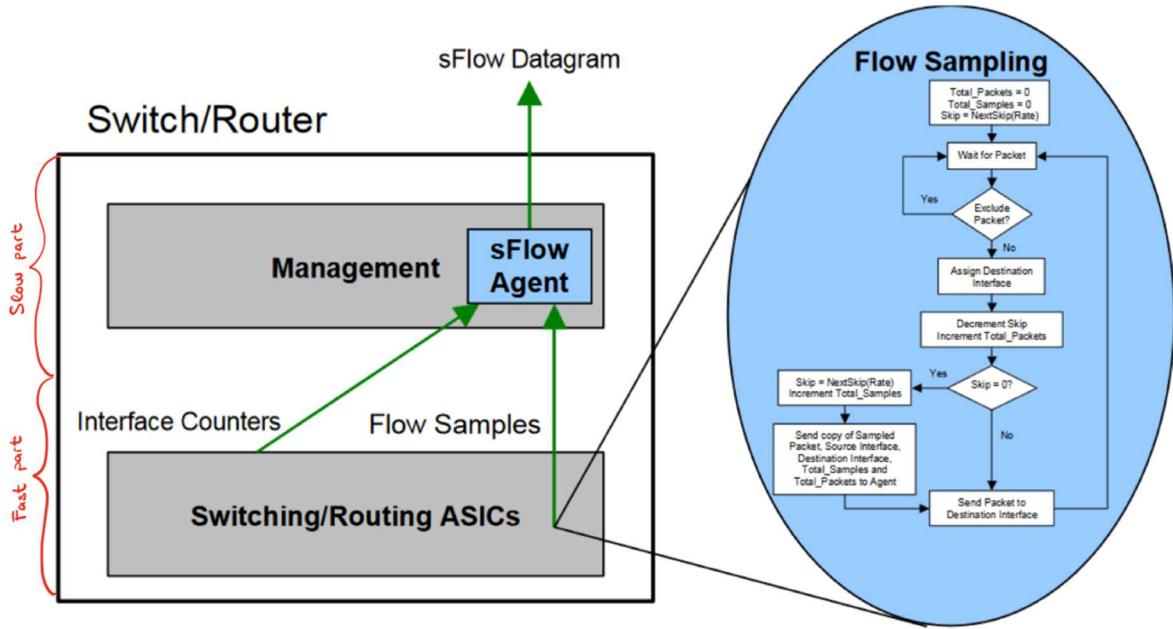
## Raccolta sFlow



Il meccanismo è sempre lo stesso: ho una serie di sensori che inviano il tutto a un collettore che poi può avere o meno la capacità di analisi.

## Come funziona sFlow

L'algoritmo che andiamo ad eseguire all'interno di un dispositivo che utilizza sFlow per catturare statistiche è il seguente.



Qui c'è la distinzione all'interno delle cosiddette network appliance (all'interno dei dispositivi di rete) tra fast part e slow part.

La parte sotto è fast perché è una matrice di commutazione realizzata in hub, cioè quella che sposta ad altissima velocità i pacchetti tra l'interfaccia di ingresso e l'interfaccia di uscita.

La matrice di commutazione è un oggetto distribuito, composto dai processori che stanno sulle schede di rete in input, da un processore centrale e dai processori che stanno sulle schede di rete in output. Tipicamente le rotte più frequenti vengono caricate direttamente nelle cash dei processori delle schede di rete perché permette di velocizzare molto l'attraversamento del nodo.

Questa parte veloce ha quindi a che fare con routing e forwarding.

Esiste poi una slow part che ha a che fare con la gestione del dispositivo. I protocolli di instradamento servono per aggiornare le tabelle di instradamento, e questo avviene sulla parte lenta. Poi una volta che la tabella è stata aggiornata questa viene caricata sulla fast part.

(NB: sFlow, come NetFlow, è allo stesso tempo un protocollo, un formato dati e un'architettura di rete)

Nella parte veloce viene eseguito l'algoritmo che sta sull'ovale celeste: vediamo come funziona.

Inizializziamo 3 contatori: contatore dei pacchetti totale, contatore dei pacchetti campionati e contatore della rate.

Il valore di skip iniziale viene impostato pari a r ( $r=N/n$ ); il valore tipico di r è 400, cioè significa che ogni 400 pacchetti che passano ne prendo 1.

Una volta inizializzato tutto entro nel loop. Aspetto il pacchetto, quando arriva vado all'elemento decisionale: il pacchetto lo considero o no? Potrei avere delle regole che determinano l'esclusione di un pacchetto dal meccanismo sFlow, indipendentemente dal campionamento.

I pacchetti esclusi non concorrono all'aggiornamento dei contatori.

Se invece il pacchetto è tra quelli campionabili, assegno l'interfaccia di destinazione, faccio skip - (quindi r diventa r-1) e incremento il numero totale di pacchetti.

Controllo se skip è arrivata a 0, se non è 0 mando il pacchetto all'interfaccia di destinazione.

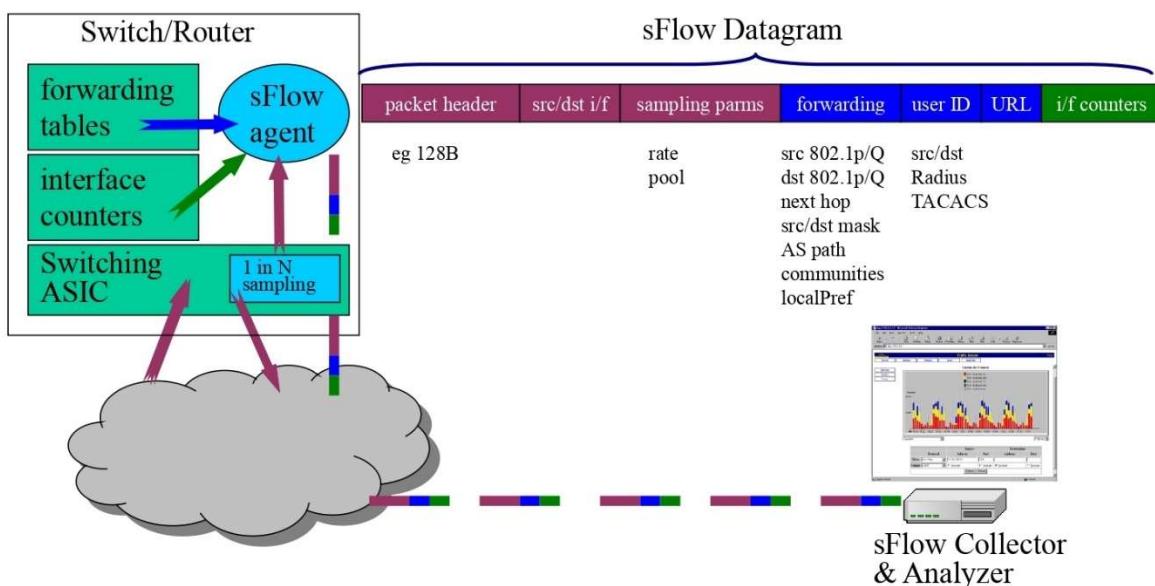
Questa cosa si ripete fino a quando un nuovo pacchetto sarà considerato per il campionamento ma nel momento in cui andrò a decrementare la variabile skip, questa passerà da 1 a 0. Entro quindi nel ramo di sinistra, reinizializzo skip a r e incremento il numero dei pacchetti campionati.

Prendo il pacchetto campionato troncato (cioè prendo l'intestazione), mi segno l'interfaccia su cui ho ricevuto il pacchetto, l'interfaccia su cui lo devo smaltire, il contatore del numero totale di campioni, il contatore del numero totale di pacchetti campionabili e lo invio al mio agente, cioè al processo che gira nella parte slow del router e che viene eseguito dal processore general purpose del router.

L'agente sFlow invia il pacchetto e tutti i contatori al collettore.

➔ Campionamento di tipo deterministico.

Non è un campionamento di tipo statistico, ogni r pacchetti ne pescò 1.



A questo punto l'agente invia al collettore:

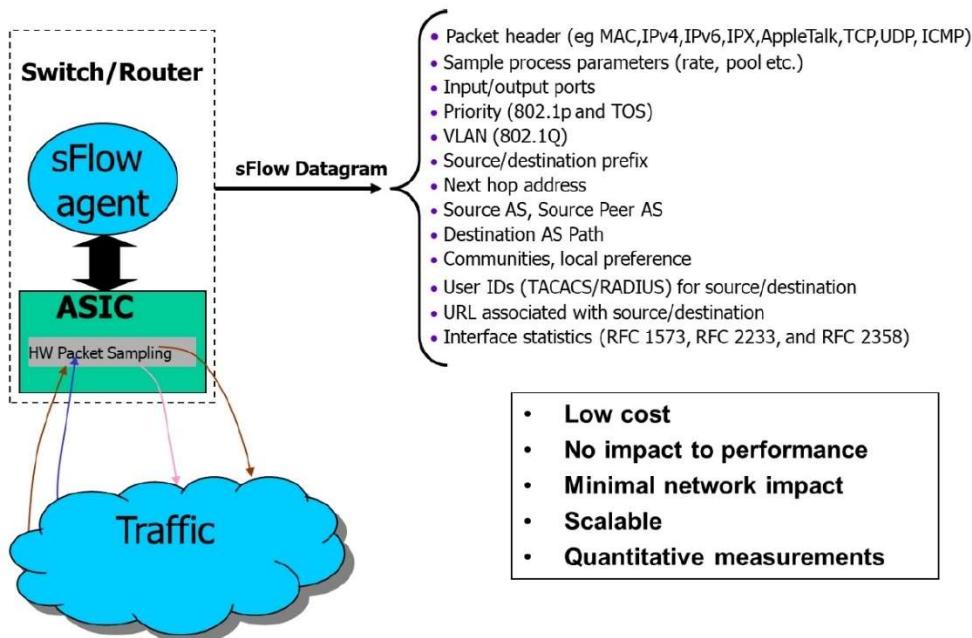
- il pacchetto troncato (l'header)
- le interfacce sorgente e destinazione del router
- i parametri di campionamento (ad esempio la rate, cioè ogni quanti pacchetti ne viene estratto uno, o il pool, cioè le regole che ci dicono quali pacchetti considero o meno per il campionamento)

Fino a qui queste informazioni vengono direttamente insieme al pacchetto

- dalle tabelle di inoltro l'agente poi riceve tutte le seguenti informazioni: VLAN di provenienza, VLAN di ritrasmissione, il prossimo salto, la netmask della sorgente e della destinazione, il percorso TCP e altri parametri relativi al routing
- Nel caso in cui il mittente o il destinatario appartengono a un sistema su cui ho il controllo è utile avere l'ID del mittente o del destinatario
- Stessa cosa se a questi utenti è associato un URL di servizio
- i contatori di interfaccia, tra cui il numero totale di pacchetti che ho osservato e il numero totale di pacchetti che ho campionato

Il collettore sFlow riceve questi dati, estrae le statistiche e le prepara per essere visualizzate attraverso opportune query.

## sFlow report



Questo è un formato diverso per riportare in modo più semplice da visualizzare le informazioni associate al pacchetto.

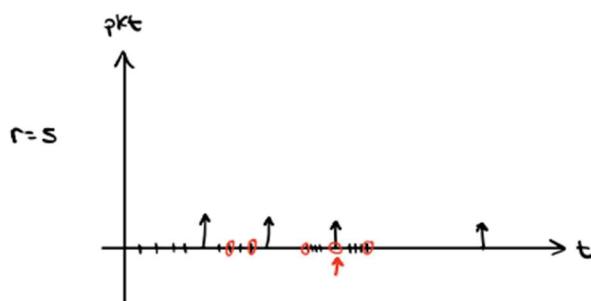
Poi abbiamo i vantaggi di sFlow rispetto a NetFlow:

- basso costo
- impatto minimale sulle prestazioni del nodo perché, se l'impatto comincia a crescere basta aumentare il rapporto  $r$
- impatto minimale sulla rete: questo accade perché dobbiamo inviare i pacchetti; mentre con netflow ho un timer configurabile dall'amministratore di rete per esportare solo le statistiche (scritte in forma binaria all'interno di un pacchetto UDP) che hanno quindi un impatto sulla rete praticamente nullo, in questo caso invece devo inviare un pacchetto alla rete. Invio solo l'intestazione e altre informazioni, non il pacchetto completo, però è sicuramente più grande rispetto al report di netflow. Ho quindi un impatto minimale perché il grosso del pacchetto è il payload (che non ho), oppure la frequenza dei pacchetti, ma quella posso regolarla a piacere aumentando il rapporto di campionamento  $r$
- scalabile: aumento o diminuisco  $r$  a seconda della necessità
- permette di fare delle misure di tipo quantitativo, esattamente come NetFlow

Cosa perdiamo rispetto a NetFlow?

Non posso stabilire l'inizio di un flusso con sFlow perché posso dirlo solo se intercetto un pacchetto di SYN tra quelli che verranno campionati.

$r = 5$  significa che ogni 5 pacchetti ne campiono uno



Supponiamo di avere un flusso (identificato con i pallini rossi): fino a che non intercetto un pacchetto di quel flusso per me il flusso non esiste, e quindi la stima dell'inizio del flusso potrebbe essere assai poco affidabile.

Chiaramente per  $r = 1$  diventa affidabile.

La stessa cosa per la fine di un flusso: deciderò che un flusso è finito dopo che saranno passati  $n$  pacchetti campionati e non avrò più visto nessun campione che appartiene a quel flusso.

Perdo quindi: istante di inizio e istante di fine del flusso.

Le altre grandezze che andiamo a stimare sono il numero di pacchetti del flusso e il numero di byte del flusso.

VEDI PDF con calcoli sulle stime

- ➔ Abbiamo visto che sFlow ha vantaggi significativi dal punto di vista delle prestazioni e del consumo di risorse da parte dei dispositivi di rete, tanto che la maggior parte dei dispositivi che utilizzano sFlow non ricorrono a stazioni di monitoraggio esterne ma utilizzano direttamente un campionamento interno al dispositivo proprio perché sono intrinsecamente più scalabili. Questa scalabilità si paga con un aumento dell'incertezza di quello che vado a stimare: viene introdotta aleatorietà ma sono in grado di stimare il valore di questa aleatorietà.
- ➔ Differenza fondamentale: NetFlow osserva i pacchetti ed estrae le statistiche. sFlow osserva un sottoinsieme dei pacchetti, cioè effettua un campionamento, e poi estrae le statistiche.

Supponiamo che il mio flusso consista di:

N: # pkt totali  
M: # pkt sampled (campionati)  
C: # pkt sampled flow x

→ Come prima metrica voglio stimare quanto il numero di pacchetti che riesco a stimare con l'uso di sFlow sia affidabile: sono interessato al valore  $\hat{N}_c$

$$N_c : \# \text{pkt totali flow x} \quad N_c = N \cdot p$$

$\hat{N}_c$  stima di  $N_c$

Lo voglio sapere quanto la stima sia vicina al valore reale

Il nostro ambiente di lavoro va ad utilizzare una variabile aleatoria bernulliana: il numero di tentativi è  $M$  e la variabile è di conteggio, e infatti, per ogni pacchetto che osservo posso avere solo due risultati: è un pacchetto del mio flusso o non è un pacchetto del mio flusso

$$P(c) = \binom{M}{C} p^C (1-p)^{M-C} \quad \mu = M \cdot p \text{ media} \quad \sigma^2 = M \cdot p \cdot (1-p) \text{ varianza}$$

$p$  è la porzione di pacchetti, tra tutti quelli che considero, che appartengono al flusso in questione (flusso x)

$p$  è una quantità che appartiene alla popolazione, che non osservo in maniera completa, ma osservo campionandola → quindi  $p$  non lo conosco

costruiamo uno stimatore  $\hat{p} = \frac{C}{M}$  e dimostriamo che è un buon stimatore di  $p$

$$E[\hat{p}] = E\left[\frac{C}{M}\right] = \frac{1}{M} E[C] = \frac{1}{M} M \cdot p = p \quad \rightarrow \text{il valore atteso dello stimatore è uno stimatore NON polarizzato di } p$$

$\xrightarrow{\text{è la proporziona campionaria}}$

$M = \frac{N}{p}$  (e' un numero → lo porto fuori dall'operazione di media)

sotto media  
errore sistematico

$$\text{Var}(\hat{p}) + \text{Var}\left(\frac{C}{M}\right) = \frac{1}{M^2} \text{Var}(C) = \frac{1}{M^2} M \cdot p \cdot (1-p) = \frac{1}{M^2} \frac{N}{p} \cdot \frac{N}{p} \cdot (1-\frac{N}{p}) = \frac{1}{M} \left[ \frac{N}{p} \cdot (1-\frac{N}{p}) \right]$$

$$y = kx \quad E[y] = K \cdot E[x] \\ \text{Var}(y) = K^2 \cdot \text{Var}(x)$$

$$\text{Var}(\hat{p}) = \frac{1}{M-1} \left[ \frac{N}{p} \cdot (1-\frac{N}{p}) \right] \quad \text{per avere una varianza stimata NON POLARIZZATA}$$

$$\hat{N}_c = N \hat{p}$$

$$E[\hat{N}_c] = N \frac{\hat{p}}{p} + rN$$

$$\text{Var}(\hat{N}_c) = \text{Var}(N \hat{p}) = N^2 \text{Var}(\hat{p}) = N^2 \cdot \frac{1}{M-1} \left[ \frac{N}{p} \cdot (1-\frac{N}{p}) \right]$$

→ Tramite la varianza posso definire un intervallo di confidenza:  $[\hat{N}_c - 1,96 \sigma_{\hat{N}_c}; \hat{N}_c + 1,96 \sigma_{\hat{N}_c}] = I_{95\%}$ : un intervallo così costruito, nel 95% delle volte contiene il valore vero

$$\sigma_{\hat{N}_c} = \sqrt{\text{Var}(\hat{N}_c)}$$

Per quantificare meglio questa precisione si va a considerare l'errore normalizzato: diviso per  $\hat{N}_c$

$$\left[ 1 - 1,96 \frac{\sigma_{\hat{N}_c}}{\hat{N}_c}; 1 + 1,96 \frac{\sigma_{\hat{N}_c}}{\hat{N}_c} \right]$$

Errore normalizzato  
Lo se lo moltiplico ·100 ottengo l'errore percentuale

Audiamo a stimare l'errore normalizzato

$$1,96 \frac{\sqrt{\text{Var}(\hat{N}_c)}}{\hat{N}_c} = 1,96 \sqrt{\frac{N^2 \frac{1}{M-1} \left[ \frac{N}{p} \cdot (1-\frac{N}{p}) \right]}{\hat{N}_c^2}} = 1,96 \sqrt{\frac{N^2 \frac{1}{M-1} \left[ \frac{N}{p} \cdot (1-\frac{N}{p}) \right]}{N^2 \left( \frac{N}{p} \right)^2}} = 1,96 \sqrt{\left( \frac{1}{p} \right)^2 \frac{1}{M-1} \left( \frac{N}{p} \right) \left( 1 - \frac{N}{p} \right)}$$

$$= 1,96 \sqrt{\frac{1}{M-1} \cdot \frac{N}{p} \cdot \left( 1 - \frac{N}{p} \right)} = 1,96 \sqrt{\frac{1}{M-1} \cdot \frac{(N-1)}{p}} \xrightarrow{M \gg C} 1,96 \sqrt{\frac{1}{M} \cdot \frac{N}{p}} = 1,96 \sqrt{\frac{1}{p}} = E$$

Se  $C$  cresce  $E$  diminuisce, ma solo se contestualmente  $n$  cresce anche  $M$

errore  
normalizzato  
approssimato

→ Questa è la stima del numero di pacchetti

Assumiamo con buona approssimazione che il numero dei pacchetti sia indipendente dalle dimensioni dei pacchetti

## Vediamo ora le stime delle dimensione dei pacchetti:

$$\bar{b}_c = \# \text{ boxe}$$

$$\in [\bar{b}_c] = \in [\bar{b}_c \bar{n}_c] = \bar{b}_c \in [\bar{n}_c]$$

$\bar{b}_c$  è la dim. media dei pacchetti

$$\bar{b}_c = \frac{1}{c} \sum_{i=1}^c b_{ci}$$

$$I_{95\%} = [\bar{b}_c - 1,96 \sigma_{\bar{b}_c}; \bar{b}_c + 1,96 \sigma_{\bar{b}_c}]$$

$$\text{Var}(xy) = \text{Var}(x)\text{Var}(y) + \text{Var}(x)y^2 + \text{Var}(y)x^2 \leftarrow \text{Var}(x) \text{ SE E SOLO SE } x \text{ e } y \text{ sono indipendenti, e deriva dalla definizione di covarianza}$$

$$\text{Cov}(x,y) = \in [(x-\bar{x})(y-\bar{y})] = \in [\bar{xy} - \bar{x}\bar{y} - \bar{xy} + \bar{x}\bar{y}] =$$

$$= \in (xy) - \bar{xy} - \cancel{\bar{xy}} + \cancel{\bar{xy}} =$$

$$= \in (xy) - \bar{xy} \quad (= 0 \text{ se } x \text{ e } y \text{ sono indipendenti})$$

$$\text{Cov}(x^2, y^2) = \in (x^2 y^2) - \bar{x}^2 \bar{y}^2 \quad (= 0 \text{ se } x \text{ e } y \text{ sono indipendenti})$$

$$\text{Var}(xy) = \in [(xy)^2] - \in^2(xy) = \in [x^2 y^2] - \bar{x}^2 \bar{y}^2$$

$$\begin{aligned} \text{Var}(b_c) &= \frac{1}{c-1} \left[ \sum_{i=1}^c (b_{ci} - \bar{b}_c)^2 \right] = \frac{1}{c-1} \left[ \sum_{i=1}^c (b_{ci} - \frac{1}{c} \sum_{i=1}^c b_{ci})^2 \right] = \\ &= \frac{1}{c-1} \left[ \sum_{i=1}^c \left( b_{ci}^2 + \frac{1}{c^2} \left( \sum_{i=1}^c b_{ci} \right)^2 - 2 \frac{1}{c} b_{ci} \sum_{i=1}^c b_{ci} \right) \right] = \\ &= \frac{1}{c-1} \left[ \sum_{i=1}^c b_{ci}^2 + \frac{c}{c^2} \left( \sum_{i=1}^c b_{ci} \right)^2 - \frac{2}{c} \underbrace{\sum_{i=1}^c b_{ci} \sum_{i=1}^c b_{ci}}_{\left( \sum_{i=1}^c b_{ci} \right)^2} \right] = \\ &= \frac{1}{c-1} \left[ \sum_{i=1}^c b_{ci}^2 - \frac{1}{c} \left( \sum_{i=1}^c b_{ci} \right)^2 \right] \end{aligned}$$

$$\text{Var}(b_c) = \text{Var}(\bar{b}_c) \text{Var}(\bar{n}_c) + \text{Var}(\bar{b}_c) \cdot \in^2[\bar{n}_c] + \text{Var}(\bar{n}_c) \cdot \bar{b}_c^2$$

se la dim dei pacchetti è costante:  $\text{Var}(b_c) = 0$

$$I_{95\%} = [\bar{b}_c - 1,96 \sigma_{\bar{b}_c}; \bar{b}_c + 1,96 \sigma_{\bar{b}_c}]$$

FINE 11/10

**RECAP:** la scorsa volta abbiamo visto come soluzioni del traffico sFlow utilizzano il campionamento per estrarre i pacchetti dal flusso di traffico e per ridurre le statistiche del traffico stesso. In particolare, il numero di pacchetti appartenenti ad un flusso e il numero di byte appartenenti allo stesso flusso.

## Evoluzione delle tecniche di monitoraggio

NetFlow e SFlow condividono un certo numero di benefici, ovvero, riescono ad identificare un flusso a differenza dei tool basati sulla pura cattura che invece riescono solo a catturare i pacchetti e poi rimandano in post-processing il compito di identificare i flussi.

### I vantaggi e gli svantaggi di NetFlow, in particolare i suoi svantaggi.

NetFlow richiede risorse di calcolo addizionali, siano esse all'interno del dispositivo in cui gira NetFlow, siano esse dislocate in un dispositivo esterno che effettua l'acquisizione di dati successivamente rilanciati in mirroring ed elaborati tramite una soluzione di tipo NetFlow. Un altro problema classico di NetFlow è che non è un sistema real-time, per due motivi:

1. In primis, non è un sistema real-time, perché noi sappiamo che l'esportazione avviene solo quando un flusso è “giudicato” terminato e la terminazione di un flusso può avvenire attraverso l'intercettazione di messaggi di segnalazione esplicativi (come: reset del TCP oppure dopo un determinato periodo di inattività). Noi sappiamo che in caso di flussi che durano molto tempo, Netflow utilizza una modalità per cui il flusso viene “interrotto” e viene fatto l'export dei dati e nella finestra di misura successiva viene di nuovo reinizializzato un nuovo flusso. Sarà poi compito della piattaforma di monitoraggio andare a combinare questi sotto flussi che appartengono in realtà ad un unico flusso (ad esempio, ad un'unica sessione TCP o UDP).
2. Per quanto riguarda i sistemi di monitoraggio, essi ricevono in automatico i dati dai dispositivi e questi dati rimangono sui collettori. Successivamente una stazione di analisi deve andare a interrogare tramite il sistema di polling i collettori stessi per recuperare questi dati e analizzarli.

Quindi NetFlow non è adatto per sistemi in tempo reale e richiede risorse di calcolo significative. **Questi sono i principali svantaggi.**

Tra i principali vantaggi vediamo che NetFlow è una soluzione standard, quindi utilizzabile nella maggior parte dei dispositivi cosiddetti gestiti, nonché in dispositivi open source. Per quanto riguarda sFlow, le considerazioni sono simili. La differenza fondamentale è che le risorse di calcolo associate alle richieste possono essere significativamente minori. Questo perché ogni N pacchetti si cattureranno n pacchetti, quindi andando ad aggiustare il rapporto tra il numero di pacchetti che passano e il

numero di pacchetti che analizzo, si può avere una soluzione scalabile e sostenibile dal punto di vista computazionale. Tuttavia, sappiamo che questo campionamento introduce un'incertezza sulle grandezze che osservo.

Infatti, il suo svantaggio principale è associato proprio al suo vantaggio, cioè al campionamento, perché se il campionamento permette di rendere di avere una soluzione scalabile, allo stesso tempo, riduce l'accuratezza. Con sFlow abbiamo un'enorme incertezza nell'identificare quando il flusso inizia, perché per identificare l'inizio del flusso, devo andare a campionarne un pacchetto. L'altra cosa fondamentale è che le statistiche che calcolo sono affette da un'incertezza e c'è anche la possibilità che flussi small (non con pochi byte ma con una durata piccola) potrebbero essere completamente ignorati, semplicemente perché nessun pacchetto associato a questi flussi può essere campionato e quindi il flusso c'è ma non si vede (Effetto collaterale del campionamento).

## Monitoring techniques evolutions

- **sFlow** samples packets from flows passing through switches
  - sFlow takes a sample once every n packets with a configurable sampling rate  $1/n$ 
    - Then, the sampled packet is sent immediately to a monitoring system for further analysis
  - sFlow requires a smaller amount of computing resources
    - However, it has lower accuracy
    - Sampling methods may be unable to detect small flows, and it can miss network events such as spikes or anomalies

Abbiamo categorizzato due tipi di approcci alla condizione del traffico:

- **Tecniche passive:** che sono quelle più diffuse e più affidabili. Tra le tecniche passive possiamo considerare sia le tecniche basate sulla rilevazione dei flussi (sFlow, NetFlow), sia quelle basate sulla lettura dei pacchetti (tshark, tcpdump);
- **Tecniche attive:** che consistono nell'iniettare del traffico in rete e nell'analizzare gli effetti che questo traffico iniettato induce (ad esempio, il ping, traceroute o nmap), quindi l'utilizzo del traffico per andare a misurare i parametri della rete.

Nelle tecnologie classiche queste misure vengono effettuate attraverso un terminale o un dispositivo di misura. Dov'è che posso fare queste misure? Le faccio attraverso delle sonde che vanno posizionate o sui sistemi terminali (principalmente server ma anche client) oppure sui dispositivi intermedi che supportano questo tipo di misure.

Negli ultimi dieci anni si è affermata una nuova tecnologia di rete che si chiama SDN, che consiste nel centralizzare il piano di controllo dei dispositivi di rete. Quindi se noi abbiamo tre router nell'approccio classico, i router si scambiano delle informazioni tra di loro (ad esempio con OSPF, ISIS o PGP) e costruiscono in maniera distribuita delle tabelle di routing che siano il più possibile stabili. **Questo è l'approccio tradizionale.**

- New network technologies (SDN) are constrained because they are based on fixed-function SDN switches

Nell'approccio SDN (Software Defined Networking) quello che succede è che i nostri tre router non parlano più tra di loro ma sono controllati da un controllore esterno (potenzialmente anche remoto) che effettua le operazioni sul piano di controllo in maniera centralizzata e poi carica il risultato di queste operazioni. Quindi, se vogliamo fare un parallelo con i protocolli di instradamento, i router non parlano tra di loro e eseguono un protocollo tipo OSPF o PGP per determinare le tabelle di instradamento, ma le tabelle vengono calcolate dal controller che le carica all'interno dei nodi.

Il Controller non è un utente umano che se le scrive a mano, ma è presente un meccanismo per cui quando avviene un evento (per evento si intende l'arrivo di un pacchetto non previsto nelle sue regole) e uno dei nodi non sa cosa fare perché il controller non ancora glielo ha detto, allora, il nodo in questione prende il pacchetto e lo invia al controller chiedendogli qual è l'azione che deve fare. Successivamente, il controller determina l'azione da fare su quel pacchetto e la rimanda al nodo. Da quel momento in poi tutti i pacchetti di quel tipo saranno gestiti in quella maniera.

Di fatto il piano di controllo decisionale diventa sincronizzato su uno o più controller, mentre il piano dati, cioè dove passano i pacchetti, rimane quello di sempre. Quindi si ha il passaggio da un approccio distribuito in cui l'intelligenza è distribuita sui dispositivi di rete, ad un approccio centralizzato in cui i dispositivi di rete diventano tipicamente più veloci e più stupidi e dove l'intelligenza è spostata in un componente terzo. Il vantaggio è che non bisogna aggiungere intelligenza ai nodi e quindi posso disaccoppiare la tecnologia di costruzione di nodi sempre più veloci (10, 25, 40, 100 gigabit o anche terabit) dal piano di controllo, cioè dalle operazioni che devono essere fatte per gestire delle interfacce con matrice di computazione così veloci. Il vantaggio vero, però, risiede nel fatto che se in un approccio di questo tipo, voglio inserire un

nuovo protocollo (ad esempio inventato a Stanford), quello che devo fare è aspettare che il costruttore del dispositivo rilasci un nuovo firmware che includa quel protocollo. Se invece lo devo aggiungere in una rete senza SDN devo sviluppare un software da caricare solo sul controller e poi sarà il controller che quando arrivano pacchetti con certe caratteristiche dirà ai nodi cosa farci. Un controller può anche gestire dispositivi eterogenei, ad esempio dispositivi di costruttori diversi potranno avere delle regole in comune per la gestione.

Nel SDN è stata standardizzata l'interfaccia di comunicazione tra i nodi del dataplane e il controlplane. Il primo protocollo che è stato definito si chiama OpenFlow inventato nel 2004 a Stanford ed è quello che praticamente si usa in tutti i datacenter del mondo, a partire da Google.

Perché parliamo di SDN quando il programma del corso è VNCC (Virtual Network and Cloud Computing)?

Perché nel caso delle misure di rete il vantaggio di avere un piano di controllo centralizzato ci permette di configurare questi dispositivi stupidi in maniera tale da abilitare nuove funzioni senza che quelle funzioni debbano essere supportate dal costruttore del dispositivo. Quindi, ad esempio, se io non supporto il mirroring dei dispositivi, però sono compatibile con SDN, basta che il mio controllore carichi una regola che per alcuni tipi di pacchetti mi dice di crearne una copia e inviarla su una certa interfaccia e di fatto ho costruito un **mirroring custom**, sicuramente più efficiente rispetto al mirroring che mi dice che tutti i pacchetti sulla porta 6 dello switch 4 vanno rigirate sulla porta di mirroring che è la porta 8. In questo modo posso decidere che alcuni pacchetti vanno copiati e ridirezionati su una porta e altri pacchetti vanno copiati e ridirezionati su un'altra porta.

Questo mi permette di creare dei framework di misura particolarmente efficienti e adattabili alla topologia della rete e ai dispositivi che ho a disposizione.

Il grosso svantaggio di questi sistemi è che alcune volte la comunicazione tra dataplane e controlplane può diventare un collo di bottiglia e che alcune funzioni, tipicamente quelle un pochino più complesse sono obbligato a realizzarle nel controlplane perché il dataplane supporta solo alcune funzioni.

## In-band Network Telemetry

Quindi dov'è che posso avere una innovazione? Possono avere un'innovazione se invece di programmare in maniera custom il controlplane, arrivo a programmare in maniera custom il dataplane. Questo comporta che si torna a ridistribuire il framework e quindi a mettere delle funzioni intelligenti, però programmabili, all'interno dei dispositivi. Per farlo carico a caldo una routine in C da eseguire, questo si chiama **Network Programmability**. Quindi, sviluppo un pezzo di codice e lo carico all'interno del dispositivo e il dispositivo lo esegue a runtime all'interno del proprio

kernel senza spegnerlo e senza cambiare la sua configurazione. Questa branca del network monitoring prende il nome di **In-band Network Telemetry**.

Perché in banda? Perché questa tipologia, cioè questo processo di misura, viene fatto direttamente sui nodi mentre passa il traffico, quindi io aggiungo a dei nodi delle sonde (ma delle sonde a basso livello) che permettono al nodo di effettuare variazioni anche sofisticate, consumando poche risorse di calcolo e soprattutto raggiungendo la win rate. Il problema principale è il costo perché questi dispositivi sono diffusi ma non sono ancora standard.

Perché si chiama telemetria attiva? Perché per fare queste operazioni di Telemetria, quello che viene fatto è che vengono aggiunte delle intestazioni sui pacchetti e visto che stiamo modificando i pacchetti questo monitoraggio torna ad essere attivo e non più passivo. In realtà è un monitoraggio attivo ma di cui l'utente non ha percezione, perché questi dispositivi così come aggiungono delle intestazioni, le tolgon prima che il traffico arrivi ai nodi terminali, quindi tipicamente, il mittente e il destinatario non si accorgono che questi nodi hanno fatto questo processing. Nel frattempo, il risultato dell'elaborazione viene inviato ad un collettore che effettuerà le relative operazioni.

Siccome siamo tornati di nuovo sul dataplane, siamo di nuovo a operazioni che possono implementare bene questo meccanismo, con alcune limitazioni, perché è chiaro che i programmi che possono essere caricati su questi dispositivi hanno una dimensione limitata e quindi anche alcune funzionalità saranno limitate. Rimangono però programmi molto potenti che permettono di fare cose particolarmente avanzate nel campo della sicurezza, della gestione della rete e del monitoraggio della rete.

Ora la domanda è: Ma c'è proprio bisogno di questa cosa?

Abbiamo detto che NetFlow comincia a non essere più adeguato a reti ad altissima velocità perché richiede di elaborare troppi pacchetti e questo potrebbe avere un patto negativo sulle prestazioni del dispositivo o alternativamente generare risultati non affidabili. sFlow, d'altra parte, effettua il campionamento e quindi alcuni dati possono essere persi, in particolare, i dati relativi ai flussi di piccole entità che soprattutto all'interno dei data center sono la stragrande maggioranza (questo perché i servizi vengono costruiti con l'approccio dei microservizi: componenti diversi girano su macchine diverse o macchine virtuali diverse). Inoltre, abbiamo detto che sFlow può perdere i pacchetti e NetFlow non è adeguato al monitoraggio in tempo reale. In generale NetFlow ci dà una previsione astratta del flusso, da quando inizia a quando finisce, ma non dei singoli pacchetti che compongono il flusso, quindi, devo essere in grado di fare la telemetria del flusso, ovvero, tracciare tutti i nodi che il flusso attraversa, identificare l'interfaccia d'ingresso e l'interfaccia d'uscita, l'istante in cui arriva e quanti pacchetti trova in coda. Solo in questo modo riesco a capire se ad esempio c'è un collegamento che per qualche motivo risulta in alcuni momenti del ciclo di servizio sovraccarico.

## In-band Network Telemetry

- Programmable data plane has been proposed to change fixed-function switches to programmable ones
- In large and high-speed networks, monitoring methods have to provide real-time, fine-grained and end-to-end network information
  - Sometimes packet-level monitoring is needed
    - debugging in a multipath routing environment
    - debugging faulty interfaces that affect only a specific group of packets with the same characteristics
  - INT has recently been proposed to solve these problems, and it has quickly attracted attention
- In-band network telemetry combines data packet **forwarding** with network **measurement**

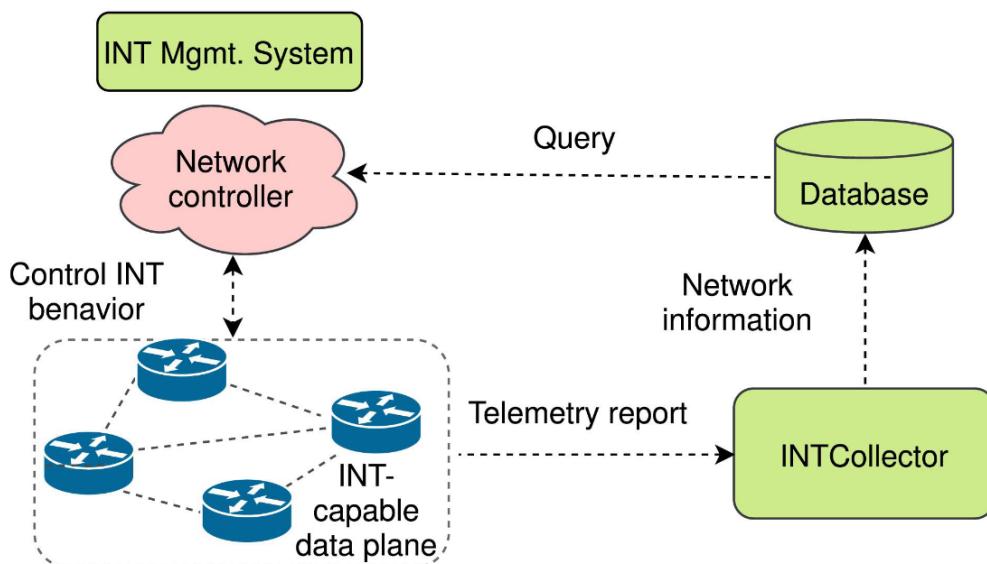
**In-band Network Telemetry collects the network status by inserting metadata into packet by switching nodes**

- In-band network telemetry data and user data usually **share** the same link or even the same packet

Quello in basso è lo schema di riferimento generale e INT sta per In-band Network Telemetry. Un sistema di gestione basata sulla telemetria attiva avrà certo numero di dispositivi (in basso a sinistra all'interno del riquadro tratteggiato). Questi dispositivi possono essere configurati uno per uno o possono essere controllati da remoto tramite tecniche SDN. Come abbiamo detto, questi dispositivi applicheranno delle etichette sui singoli pacchetti, sulla base delle regole che andremo a definire attraverso il sistema di controllo (schema simile ad iptables), quindi ci sarà una regola e tutti i pacchetti che matchano quella regola gli verrà attaccata un'etichetta. Quando un pacchetto con un'etichetta attraversa un nodo si hanno un certo numero di operazioni

che possono comportare la modifica, la rimozione, l'aggiunta di un'etichetta o l'invio di un messaggio di monitoraggio verso un collettore di dati. Il collettore, chiaramente fuori banda, non segue il percorso dei dati. Il/i collettore/i, poi, invierà i dati sul database e la piattaforma di gestione potrà recuperare quei dati attraverso le query al database e potrà capire che cosa succede.

## In-band Network Telemetry



**Esempio:** Supponiamo che il nodo a sinistra sia il nostro mittente e il nodo tutto a destra, sia il nostro destinatario. Questi due nodi si inviano un certo numero di pacchetti che saranno caratterizzati dall'intestazione e da un certo payload.

Cosa succede nel momento in cui questo pacchetto entra su uno switch di tipo INT e le sue caratteristiche, ad esempio, il numero di porta, gli indirizzi IP, eccetera matchano con una particolare regola?

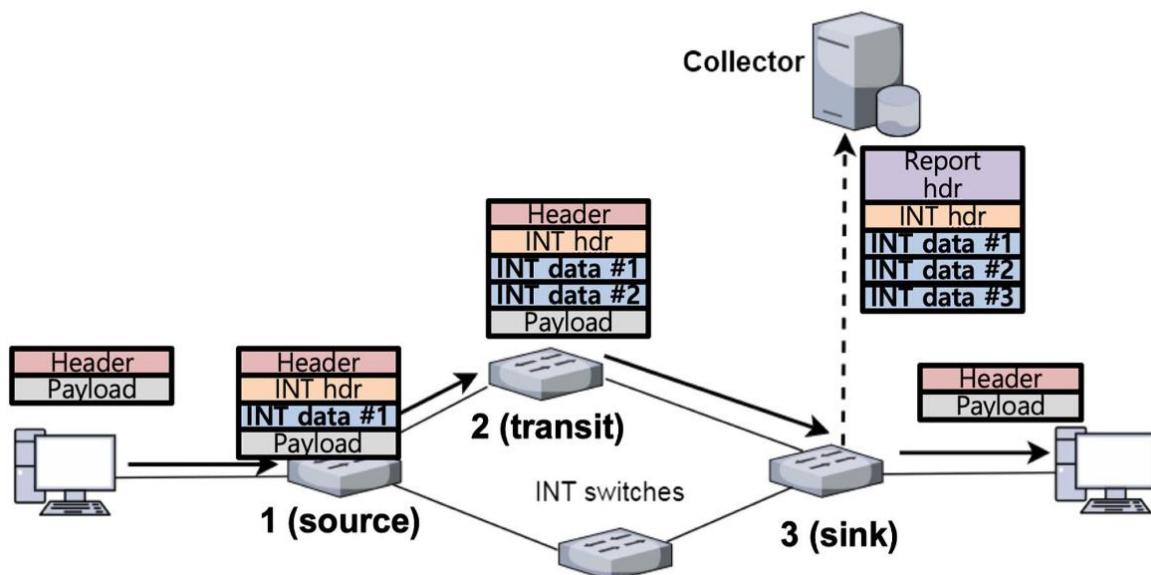
Lo switch aggiunge una ulteriore intestazione (INT hdr) e un certo numero di dati (INT data #1,2 etc.). I dati caratterizzano l'attraversamento di quello switch, ad esempio, il pacchetto è arrivato in questo istante temporale sull'interfaccia numero 6, verrà spedito sull'interfaccia numero 8, dove ha trovato 128 pacchetti in coda. Quindi inserisco questi dati tra l'intestazione originale del pacchetto e il payload del pacchetto, per questo è una tecnica attiva perché va a modificare il pacchetto. Questo è il nodo di **inserzione**.

Il secondo nodo, il due, è un nodo di transito. Esso, tipicamente va ad esaminare il pacchetto. In questo caso le caratteristiche del pacchetto determinano l'inserimento di un ulteriore set di dati o il fatto che c'è già un'intestazione INT determina

l'inserimento in un ulteriore set di dati. Sta di fatto che quello che succede è che anche il nodo due, mantenendo la stessa intestazione, aggiunge i suoi dati di telemetria. Infine, il pacchetto arriverà al terzo nodo che è il sink, ovvero il collettore (collettore per quanto riguarda la Telemetria, non il destinatario dell'informazione) che aggiungerà inizialmente al pacchetto i dati relativi al collettore (INT data #3), ma visto che riconosce di essere l'ultimo nodo, invia questi dati al collettore e riporta la situazione allo stato originale, togliendo i dati di telemetria e consegnando i dati a destinazione.

Quindi di fatto questi due nodi (quello più a sinistra e quello più a destra) che in questo disegno per semplicità sono attaccati alla stessa rete, ma che possono essere in parti opposte di Internet, non si accorgono che una rete intermedia o più rete intermedie applicano tecniche di telemetria andando a modificare il pacchetto. Questo perché queste tecniche di telemetria prevedono che il pacchetto venga riconsegnato inalterato dopo il nodo di uscita.

## In-band Network Telemetry



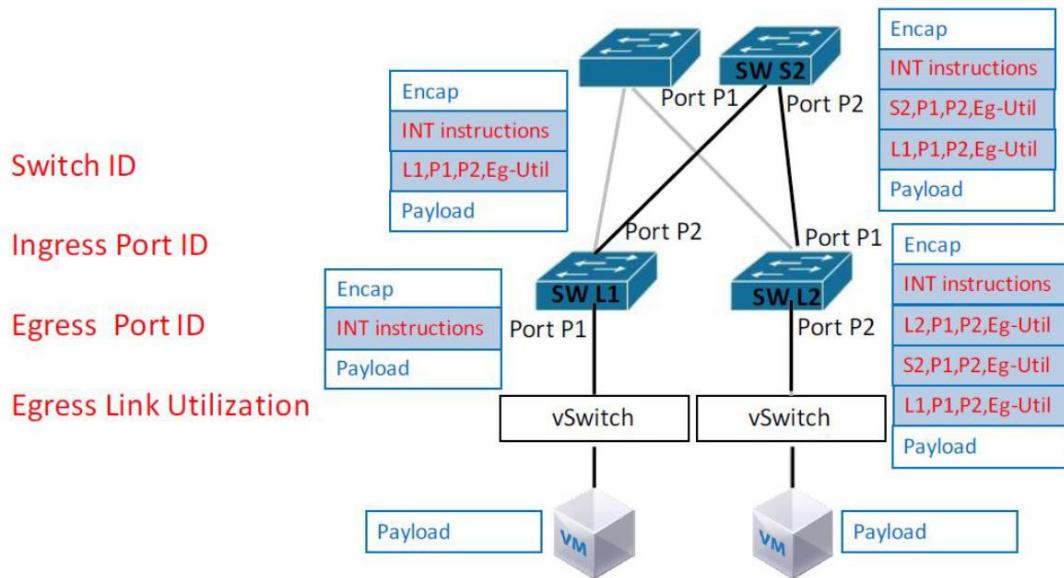
Le tipiche informazioni sono l'id del nodo che viene attraversato, l'id della porta d'ingresso, l'id della porta d'uscita e tipicamente l'utilizzazione della porta d'uscita, dove l'utilizzazione ci può dire quanti byte sono in coda oppure semplicemente nell'ultima finestra temporale quanti byte sono stati trasmessi diviso la capacità del limite di trasmissione. Quindi se ho un'interfaccia a 100 Gigabit dirò ad esempio che

negli ultimi due millisecondi sono stati trasmessi x Megabyte diviso 10 gigabit (Non capito molto il senso).

In questo disegno consideriamo uno scenario tipico da data center, all'interno del quale ho ad esempio due macchine virtuali che sono eseguite all'interno di due diversi server. Quindi all'interno di un server, avrò sia la macchina virtuale sia lo switch virtuale e il primo switch sarà quello che inserisce delle istruzioni e oltre a inserire le istruzioni, aggiungerà un certo numero di informazioni. Le informazioni sono ad esempio: L1 (id dello switch), P1 e P2 (porta d'ingresso e uscita) e utilizzo della porta d'uscita.

Quindi, i concetti di base sono abbastanza semplici; switch o dispositivi software o hardware che supportano schemi avanzati su dataplane, ovvero schemi che permettono di andare a fare un monitoraggio più fine del traffico che passa. Si capisce da subito che la soluzione non scala perché non posso applicare etichette diverse a tutti i pacchetti, altrimenti si va ad inondare la rete con molto traffico aggiuntivo. Questo perché aumenta la dimensione del pacchetto e se la dimensione aumenta devo scontrarmi con il problema della MTU (Maximum Transfert Unit) che è la dimensione massima del mio pacchetto. Se io ho un pacchetto di dimensione massima, non posso aggiungere le etichette perché non c'è spazio, quindi, al fine di supportare questo schema, potrei dover utilizzare una MTU all'interno dei dispositivi più bassa. Però noi sappiamo che l'efficienza di un collegamento tra i dispositivi è tanto più grande quanto più grande è la dimensione dei pacchetti. Una delle regole base del network monitoring o in generale della gestione delle reti, è che il processo di gestione non deve impattare in maniera significativa sulle prestazioni di quello che viene gestito, altrimenti non è sostenibile.

# In-band Network Telemetry

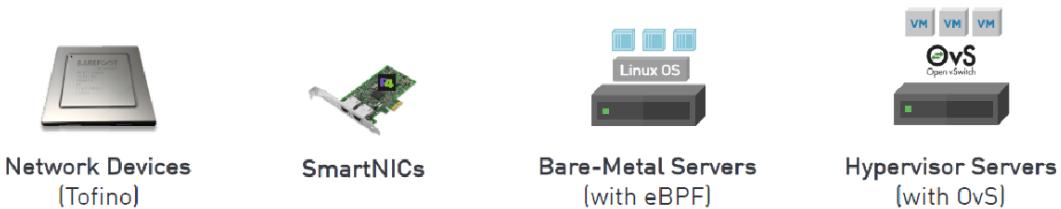


Le soluzioni di rete a disposizione sono:

- I dispositivi di rete che supportano in maniera nativa questo paradigma, quello del In-band Network Telemetry;
- Schede di rete che supportano il paradigma INT. Quindi ho dei server all'interno dei quali ho delle schede di rete programmabili e quindi vado a fare una programmazione non a livello di dispositivo ma a livello di rete. Potrei avere un server con due schede di rete programmabili e due schede di rete non programmabili, in questo caso, il traffico che passa su quelle programmabili potrà essere monitorato attraverso tecniche INT e quello che passa sulle altre schede potrà essere monitorato con tecniche più classiche;
- Server classici che non hanno schede che supportano la modalità INT ma supportano un nuovo paradigma di elaborazione dei pacchetti che si chiama eBPF (extended Berkeley Packet Filter), ovvero, strutture di filtraggio che stanno nel kernel di Unix e Linux associate alle prime Socket che sono state inventate all'università di Berkeley. Essi sono dei nuovi paradigmi che permettono di effettuare cose più sofisticate a livello di kernel e tra queste cose ci stanno anche le tecniche INT;
- Utilizzare un Server non in maniera “monolitica”, ma come hypervisor, ad esempio di macchine virtuali. Se come soluzione di rete per interconnettere queste macchine virtuali utilizzo OvS (Open virtual Switch), allora questo package che è open source supporta la telemetria di rete.

# In-band Network Telemetry

- It extends telemetry everywhere
  - Network devices with explicit INT support
  - Smart NICs/switch compliant with P4
  - Bare-metal servers with extended BPF
  - Bare-metal hypervisor



## Framework P4

P4 sta per Programming Protocol Independent Packet Processor. Esso permette di andare a caricare a runtime all'interno di dispositivi predisposti delle routine scritte in linguaggio C che effettuano l'elaborazione dei pacchetti. Tra le funzioni che posso essere caricate sono presenti anche delle funzioni di monitoraggio. Tipicamente (facendo riferimento al ciclo visto ad inizio corso: Observe, Analyze, React) con questi tool riesco sia a osservare che a reagire, mentre ad analizzare dipende, perché l'analisi e la decisione di cosa fare viene presa in una diversa sede, dato che questi tool hanno capacità di elaborazione spesso limitate. Quindi tipicamente riescono a fare molto velocemente sia azioni di monitoraggio dettagliate e customizzate, sia azioni di risposta a eventuali situazioni anomale o semplicemente per ottimizzare il funzionamento della rete.

Queste sono chiaramente delle soluzioni molto interessanti perché permettono di aggiungere delle funzioni che non sono supportate in maniera nativa da un costruttore. Ovviamente, ogni volta che un costruttore supporta una funzione nuova, questa fa lievitare il prezzo del dispositivo.

## Funzionamento Programma P4

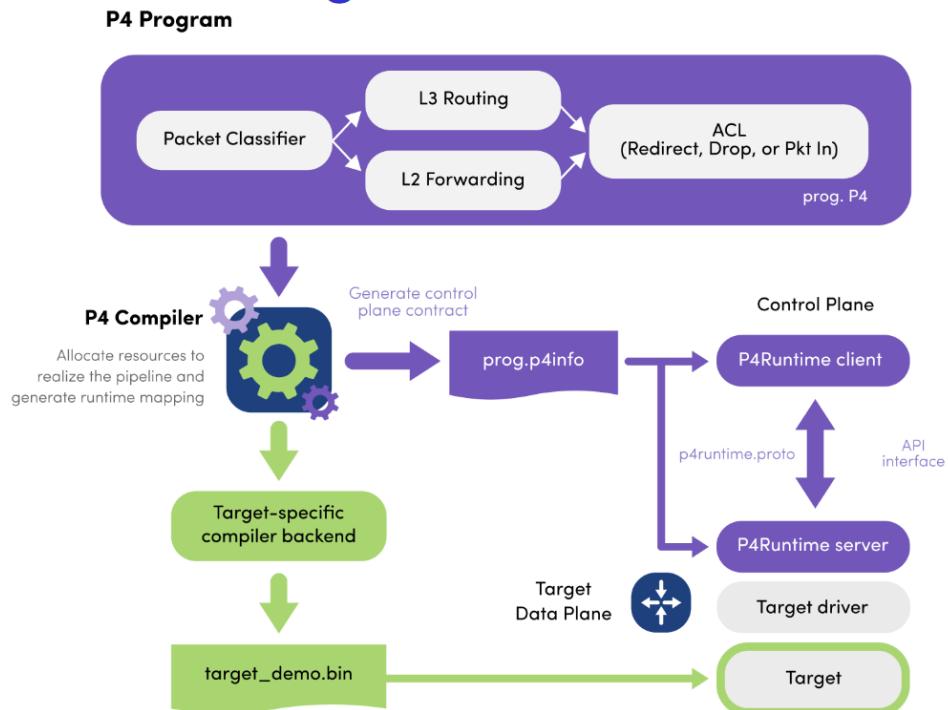
Inizialmente i pacchetti vengono classificati per capire se bisogna applicare una qualsiasi delle funzioni definite all'interno di P4 a quei pacchetti oppure no. Nel caso in cui ce ne sia bisogno, queste funzioni potranno riguardare sia il routing che il

forwarding (quindi lo spostamento dall'interfaccia d'ingresso all'interfaccia d'uscita) o potranno riguardare delle regole di tipo attivo, le cosiddette ACL (ad esempio: ridirezionare il pacchetto, scartare o inviare il pacchetto ad un controller remoto). Per caricare un programma di questo tipo quello che riesco a fare è creare un mio software, compilarlo e ottenere un modulo eseguibile. A questo punto il modulo eseguibile verrà caricato all'interno dello switch attraverso delle API dedicate e contemporaneamente, bisogna preparare il dispositivo ad accogliere il modulo e bisogna generare una sorta di contatto per il dataplane (un manifest che spiega in cosa consiste il modulo che andrà a caricare) e dopo aver caricato il modulo attraverso l'interfaccia, avrò che il mio dispositivo da quel momento in poi supporta quel programma.

Il vantaggio di questi framework è che tipicamente effettuano un controllo di compatibilità con il funzionamento del dispositivo, quindi di solito, quello che succede è che non è possibile andare a caricare un programma che per vostra inettitudine, va a compromettere per sempre il funzionamento del dispositivo. Alcune operazioni ovviamente non saranno ammesse e nel processo di compilazione viene fatta una verifica su potenziali azioni dannose o che compromettono la funzionalità del dispositivo. In quel caso la compilazione non ha buon esito.

Se ad esempio per errore si scrive nel codice un ciclo infinito, esso non passerà il test di validazione perché potrete dover spegnere il dispositivo per disabilitare il problema e quindi si avrà un'interruzione dei servizi.

## An interesting framework: P4



Potenzialmente l'elaborazione avviene attraverso una cosiddetta pipeline, quindi un certo numero di stage che sono:

1. **Parse iniziale**: la deserializzazione del pacchetto. Vengono estratti i bit dall'intestazione;
2. **Match action**: classificazione del pacchetto e l'identificazione dell'azione che sulla base di uno specifico match deve essere eseguita;
3. Inoltro verso l'interfaccia di uscita e accodamento;
4. Eventuale nuova azione da matchare sull'interfaccia d'uscita e serializzazione.

Quindi, posso codificare il pacchetto sia in ingresso che in uscita e per far funzionare P4 abbiamo bisogno di un dispositivo o di un dispositivo che lo supporti. Esistono anche degli emulatori software open source ma lo svantaggio dell'emulatore è la velocità.

## EBPF (extended Berkeley Packet Filter)

Se in una macchina non abbiamo nessuna scheda di rete che supporta P4 e vogliamo far girare tecniche di telemetria, una possibile opzione è eBPF. Esso consiste nel creare una sorta di macchina virtuale che gira dentro il kernel, all'interno del quale posso caricare programmi dallo user space in una modalità stile sandbox, in modo da avere una scatola isolata dove non posso fare danni e all'interno del quale posso parlare con lo user space.

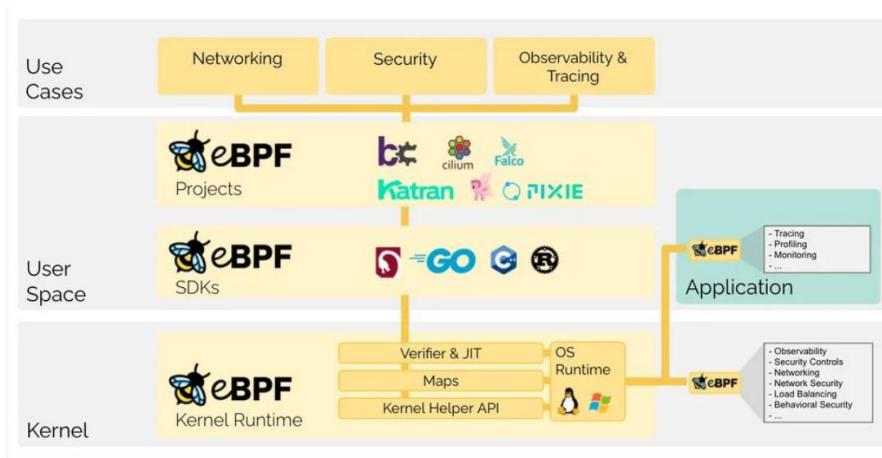
In che punto del kernel? Prima ancora che vengano eseguite le regole di iptables, ovvero prima del pre-routing. Per implementare un programma in eBPF posso utilizzare C, Go o Python perché ci sono delle librerie che convertono il programma in bytecode. Una volta fatta questa conversione viene effettuato un processo di verifica per evitare che il software comprometta il corretto funzionamento del calcolatore. Superata la verifica si fa il JIT (Just-In-Time) ovvero una compilazione runtime in cui il compilatore decide quale parte del bytecode andare a compilare per un'esecuzione più veloce quale invece interpretare per una esecuzione immediata anche se meno veloce.

Quindi una volta che ho il mio modulo all'interno del kernel, posso accedere a delle strutture dati che prendono il nome di mappe, che sono le strutture dati con cui i vari pezzettini del nostro software scambiano e mettono questi dati a disposizione allo user space. Il numero di funzioni che possono essere richiamate dentro il kernel è limitato. Questa funzione prende il nome di **Kernel Helper API**.

L'approccio di eBPF è un approccio molto moderno perché permette di fare cose che si potrebbero fare solo se fossimo degli sviluppatori del kernel e riuscissimo a convincere le maggiori distribuzioni di Linux di aggiungere la nostra funzione all'interno del kernel.

In questo caso, invece non si ha bisogno di creare una patch al kernel del sistema operativo per eseguire una funzione ma basta scrivere la funzione e caricarla a runtime senza neanche spegnere la macchina, testarla e se funziona caricarla. Dato che eBPF viene eseguito su un server, a seconda della potenza del calcolatore, potrei avere in parte delle capacità di analisi o capacità di osservazione ma sicuramente ho la capacità di reagire (ovvero di generare e caricare un nuovo software).

## A further option in Linux: eBPF



<https://ebpf.io/what-is-ebpf/>

## Formato pacchetto INT

I pacchetti INT sono costituiti da un'intestazione di molto semplice che contiene: la versione dei flag, un certo numero di contatore delle istruzioni, numero massimo di salti e i metadati. I metadati vengono inseriti da ciascun nodo che è attraverso in un framework in In-band Telemetry (in testa quello più recente e in coda il primo). Questa intestazione e questo payload (cioè i metadati) vengono inseriti tra l'intestazione e il payload del pacchetto che viene trasportato.

Quali sono i meccanismi che devono essere utilizzati all'interno dell'INT?

Bisogna raccogliere lo stato della rete all'interno del dataplane, quindi senza fare delle misure “orchestrated” con tecniche di piano di controllo, cioè fuori dal piano dati. Quindi devo di fatto stare dietro alla velocità del dataplane e bisogna effettuare un'osservazione in tempo reale così da avere una soluzione più flessibile possibile che possa adattarsi in maniera semplice a cambiamenti nel tempo.

Per stato della rete intendiamo: l'id dello switch e delle porte, l'utilizzazione con il numero di pacchetti che stanno sulla porta d'uscita, il tempo che ci metto per attraversare il dispositivo (**Hop Latency**), l'occupazione della coda di uscita e lo stato di congestione (ad esempio avere 128 pacchetti in coda può significare tutto e niente,

perché posso definire lo stato di congestione quando il numero di pacchetti in coda supera i 512 e se ne ho 128 non ho stato di congestione).

## INT collects network state

- Mechanism for collecting **network state** in the **dataplane**
  - As close to real-time as possible
  - At current and future line rates
  - With a framework that can adapt over time
- Examples of **network state**
  - Switch ID, Ingress/Egress Port ID
  - Egress Link Utilization
  - Hop Latency
  - Egress Queue Occupancy
  - Egress Queue Congestion Status
  - ....

66

## INT data

Qual è il percorso seguito dai miei pacchetti? Quali regole seguono i miei pacchetti? Con chi divido le mie code?

Abbiamo detto che ci possono essere dei fenomeni transitori, il cui impatto non è trascurabile e che sono difficili da identificare con soluzioni classiche orientate al flusso perché troppo astratte. Quindi per ritrovare ad esempio un microburst, devo sapere il pattern all'interno della rete dei miei pacchetti, devo sapere all'interno di ciascun nodo quante regole vengono eseguite sul pacchetto per capire, ad esempio, quali regole devo modificare all'interno di questi nodi per fare in modo di cambiare l'elaborazione che questi pacchetti subiscono e l'ultimo punto fondamentale è sapere all'interno del mio percorso con quali altri flussi sto condividendo il mio percorso o porzione del mio percorso.

Se ci si accorge che un'interfaccia d'uscita è congestionata, potrei avere due soluzioni a disposizione:

1. Cambio l'instradamento del mio flusso;

- Cambio l'instradamento ai flussi che attraversano lo stesso link, a seconda dello stato generale della rete. Quindi tipicamente vado a monitorare quando ho bisogno di effettuare un monitoraggio fine vado a monitorare più flussi.

## INT data

The network should answer these questions:

- Which path did my packet take?
- Which rules did my packet follow?
- Who did it share the queues with?

Real-time rich analytics:

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li><input type="radio"/> Anomalies           <ul style="list-style-type: none"> <li>• Congested flows</li> <li>• High hop/e2e latency</li> <li>• Path change/loop</li> </ul> </li> <li><input type="radio"/> Events           <ul style="list-style-type: none"> <li>• New flow/termination</li> <li>• Latency change</li> <li>• Unused switch/link</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li><input type="radio"/> Drop reports with rich metadata:           <ul style="list-style-type: none"> <li>• Timestamp</li> <li>• Drop reason</li> <li>• Packet 5-tuple/metadata</li> <li>• Switch ID</li> <li>• Ingress/egress port-ID</li> <li>• Queue ID</li> </ul> </li> </ul> |
|--|--|

67

## INT modes

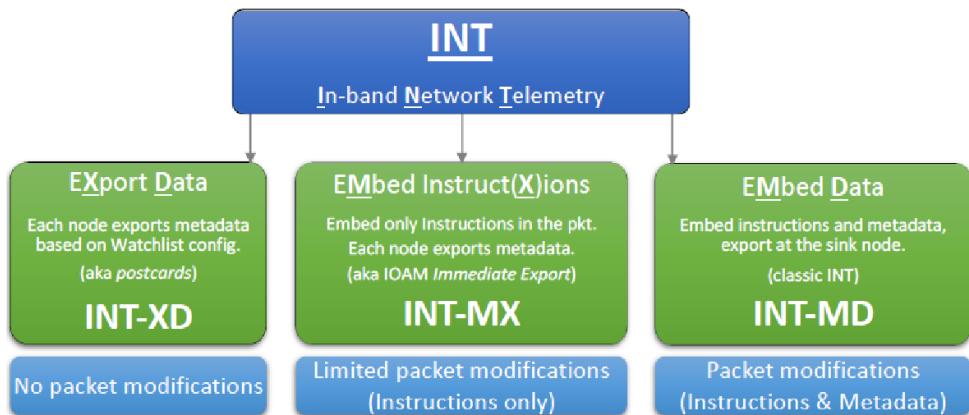
Per implementare la In-band Network Telemetry ho tre soluzioni, ognuna con vantaggi e svantaggi. La prima si chiama XD (export Data), la seconda si chiama MX (embed Instructions) e l'ultima si chiama MD (embed Data).

La prima non comporta una vera modifica dei pacchetti. All'interno del pacchetto vengono messe solo delle istruzioni, una cosiddetta Watchlist senza attaccare i metadati al pacchetto e sulla base della Watchlist si esportano una serie di informazioni.

Nel secondo caso, invece, vado a scrivere sul pacchetto solo le istruzioni e sulla base di queste istruzioni vengono catturate un certo numero di informazioni che vengono esportate verso un collettore. Quindi nel primo caso vado a configurare i nodi con la Watchlist, nel secondo caso non configuro i nodi ma scrivo sul pacchetto le istruzioni per i nodi.

Nel terzo caso, vado a scrivere direttamente i metadati sul pacchetto e quindi sarà solo l'ultimo pacchetto che è quello che rimuove le etichette a inviare i metadati al collettore.

## INT modes



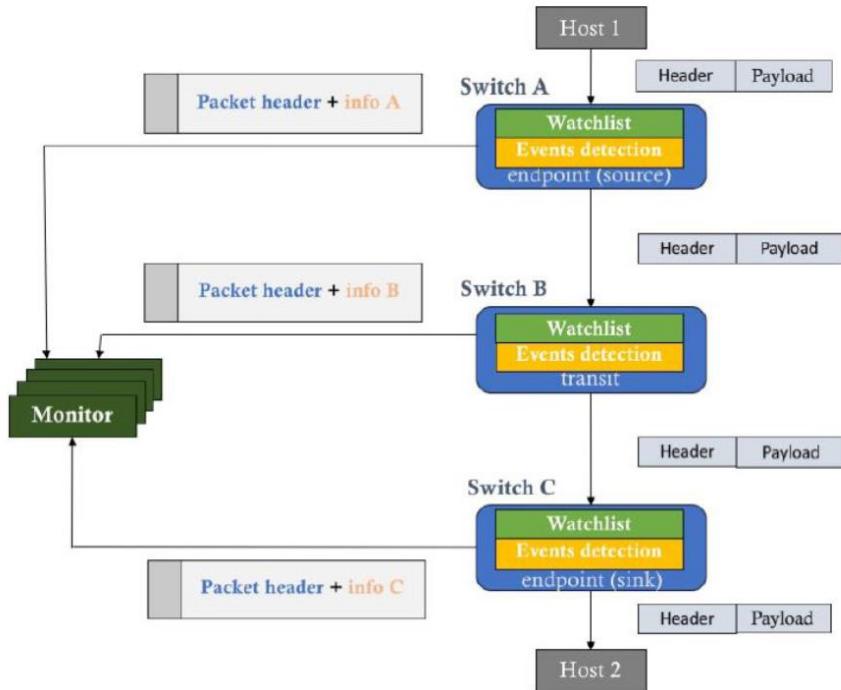
**Primo Schema XD:** Qui abbiamo detto che il pacchetto non viene modificato. Quello che vado a fare però è configurare all'interno dei miei nodi la Watchlist, quella **verde** e così facendo sto andando creare dei filtri che triggerano delle operazioni di monitoring. Quando mi arriva un pacchetto che matcha le istruzioni della Watchlist, genero un pacchetto dei metadati e invio un pacchetto che riporta: l'intestazione del pacchetto più i metadati del nodo info A (primo switch). Poi il pacchetto passa allo switch numero 2 e come prima se si matcha un'istruzione della watchlist invio l'intestazione più i metadati nel nodo B e infine lo switch C.

Le cose fondamentali sono: il percorso, le regole e con chi ho condiviso la coda. Tutte queste informazioni le posso dedurre perché si ha sia una copia dei metadati del nodo che ho attraversato, sia dell'intestazione del pacchetto. Quindi andando a trovare tutti i nodi che hanno inviato un messaggio con la stessa intestazione del pacchetto riesco a risalire al percorso che ha effettuato il pacchetto stesso e guardando dentro i metadati riesco a risalire: alle code di ingresso, all'interfaccia d'ingresso e di uscita e allo stato delle code. Il vantaggio in questo è che i pacchetti non vengono modificati e quindi il problema della INT non esiste. Questa potremmo classificarla come una soluzione passiva in cui viene sfruttata la programmabilità dei dispositivi e come svantaggio si ha che genera un'enorme quantità di dati di monitoraggio e per ogni dispositivo che attraverso genero un pacchetto di report.

Consideriamo ad esempio una sessione TCP (che è bidirezionale) molti pacchetti TCP saranno solo up (ovvero pacchetti vuoti con intestazione IP e TCP) e avremmo che per ogni pacchetto vuoto vado a generare anche i metadati che sono più grandi del

pacchetto che vado a monitorare. Una soluzione di questo tipo non modifica i pacchetti ma allo stesso tempo comporta un overhead sulla rete molto elevato e quindi lo posso utilizzare solo per brevi intervalli di tempo, altrimenti avrò un sovraccarico della rete enorme e vado a consumare più risorse di quante sono quelle che dovrei monitorare.

## INT-XD

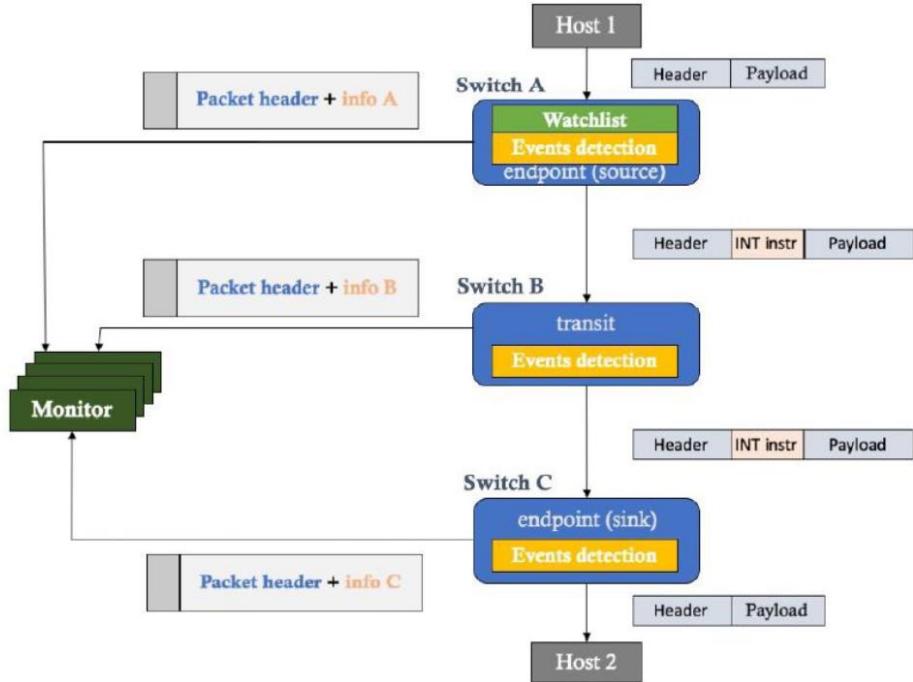


**Secondo Schema MX:** Creo la watchlist solo sul primo nodo. Che cosa fa? Questo è un modulo che quando il pacchetto matcha con alcune istruzioni di quel modulo, quindi con alcune condizioni, andrà a inserire una specifica istruzione sul pacchetto. Ciascun nodo non avrà bisogno di configurare la watchlist ma dovrà semplicemente eseguire le istruzioni che stanno scritte sul pacchetto. Le istruzioni, tipicamente sono molto limitate, perché non riguardano i metadati. Quindi, all'interno del pacchetto si aggiunge un'ulteriore intestazione e in questo modo ciascuno switch in grado di interpretare queste istituzioni invierà questi pacchetti.

Rispetto allo schema di prima abbiamo un vantaggio enorme perché in questo caso devo conoscere solo il primo nodo, nell'altro caso invece, devo conoscere tutti i nodi o devo aggiungere la watchlist ad ogni nodo. Con questo schema tutti gli altri nodi saranno semplicemente triggerati dalla presenza di queste istruzioni, mentre l'ultimo nodo, deve avere la capacità di capire di essere l'ultimo e dovrà muovere l'intestazione che non deve essere visibile all'host finale. Anche qui l'overhead è enorme perché ogni

nodo attraversato genera una copia dell'intestazione del pacchetto più i metadati. Nonostante questo, è più flessibile, perché devo conoscere solo il punto di ingresso del mio flusso.

## INT-MX



**Terzo Schema MD:** Metto la watchlist solo sul primo nodo e l'evento detector solo sull'ultimo nodo. In questo caso devo conoscere il primo e l'ultimo nodo. Con la watchlist non solo inserisco le istruzioni ma anche i metadati (che scrivo sul pacchetto). Lo switch B intermedio osserverà le istruzioni e aggiungerà i suoi metadati. Sull'ultimo switch devo mettere la rilevazione dell'evento, in questo caso l'evento è la fine del percorso e quindi devo rimuovere le informazioni che ho collezionato sul percorso, le devo inviare a un nodo di monitoring e devo ripristinare il pacchetto nella sua forma originale.

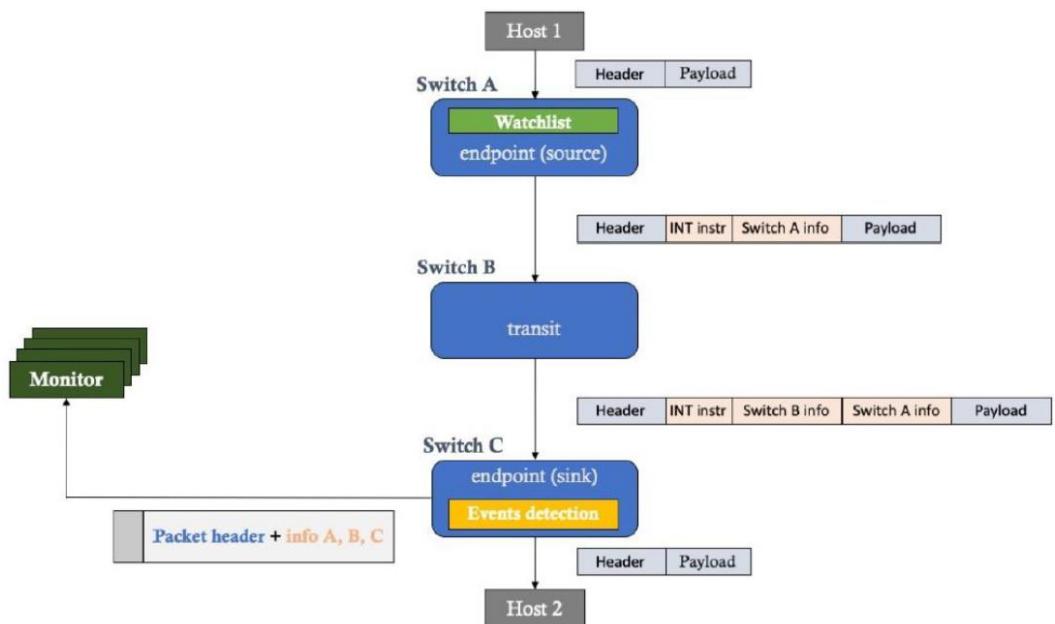
I vantaggi sono significativi; devo inviare un solo pacchetto di monitoraggio per ogni pacchetto che attraversa la mia rete. Questo pacchetto di monitoraggio potrebbe non essere piccolo.

Uno dei più grandi svantaggi è l'MTU. Bisogna avere la stima di quanto è grande la dimensione massima delle istruzioni più i metadati e bisogna diminuire l'MTU dei

dispositivi attraversati di conseguenza in modo da evitare che il TCP (ad esempio) non crei pacchetti che non permettano di aggiungere anche l'intestazione.

Un altro svantaggio è dato dal fatto che se il pacchetto non arriva mai al nodo finale, perdo tutto il monitoraggio. Quindi se ho qualche perdita io non vedrò nulla, come se il mio flusso non esistesse. Quindi è vero che questo schema è molto più efficiente dal punto di vista dell'impatto del traffico di monitoraggio, però è anche più sensibile alle perdite. Ad esempio, se ho un problema di congestione all'interno della rete e anche il traffico di monitoraggio venisse affetto è un problema.

## INT-MD



## INT disadvantages and challenges

- The payload ratio of the normal packet is reduced due to the encapsulation of telemetry instruction and metadata
  - Limitation of packet maximum transmission unit (MTU) caused by long telemetry path
- The construction, encapsulation, filling and extraction of telemetry instruction and metadata increase the processing burden of switch
- INT is sensitive to packet loss and cannot solve the problem of missing telemetry data due to packet loss

## INT disadvantages and challenges

- In-band network telemetry will consume part of the network bandwidth
- Most encapsulation protocols such as TCP contain checksum
  - The insertion of meta-measurements requires the checksum fields to be updated
    - It will increase the processing cost of the switch
- In-band network telemetry may generate a large amount of telemetry data
  - Total telemetry data length of a packet is proportional to the number of telemetry points
    - It may even exceed the size of the original packet

**Conclusioni:** In generale, la telemetria è uno strumento di debugging molto raffinato che può dare informazioni molto precise ma che va utilizzato con estrema attenzione, proprio perché abbiamo visto che la quantità di dati generati può essere molto importante soprattutto sulle reti ad alta velocità. Questa considerazione vale fino a un certo punto all'interno del data center, soprattutto se le regole di monitoraggio riguardano flussi che sono molto brevi. Quindi se osservo solo i flussi attraverso specifici microservizi, magari la quantità di traffico e di monitoraggio che vado a generare potrebbe essere più modesta, ma non è detto, dipende, dalla configurazione dell'instradamento in questi microservizi che può essere proprio uno dei problemi che

per qualche errore di configurazione può essere non ottimale e che attraverso queste tecniche si può cercare di risolvere.

## Cyber Security

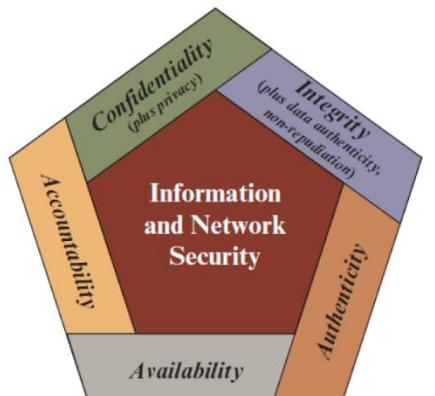
In questa sezione dedicata alla sicurezza informatica, si daranno per scontate le conoscenze di base, i paradigmi di cifratura a chiave asimmetrica e la differenza principale con gli schemi che utilizzano la chiave simmetrica. La cyber security è un ambito molto ampio, in questo caso si farà riferimento ad un sottoinsieme delle cyber security che va sotto il nome di sicurezza di rete. Verrà quindi trattato ciò che riguarda la sicurezza di un sistema informatico o dei sistemi di information technology. In questo caso quindi, la cyber security può essere definita come insieme delle azioni che servono per proteggere l'informazione, questa può essere immagazzinata, trasmessa ed elaborata da un sistema distribuito, tipicamente realizzato tramite una rete di calcolatori, ovvero dispositivi interconnessi da mezzi trasmissivi; chiaramente tutto questo include la stessa Internet. Per dare un'ulteriore definizione, la cyber security è l'insieme delle norme che servono per proteggere le informazioni.

La cyber security si basa su 5 concetti chiave definiti dal NIST (Network Institute for Standard Tecnology) e vanno sotto i nomi di Confidenzialità, Identità, Disponibilità, Autenticità e Accountability.

Il NIST ha definito tre caratteristiche fondamentali, quelle della triade CIA (Confidenzialità – Integrità - Availability), alle cui si aggiungono AA (Authenticity, Accountability)

Nell'ambito della Cyber Security è importante non solo la protezione dei dati e sistemi, ma anche e soprattutto come questa protezione viene erogata. Nella sicurezza di rete, differentemente da altri ambiti della Cyber Security, come ad esempio le tecniche di crittografia, c'è un enorme dislivello tra i requisiti, tipicamente facili da comprendere, e le azioni alle basi meccanismi necessari a rispettare tali requisiti, che tipicamente sono invece molto complicati e che spesso richiedono ragionamenti controidintuitivi. Nella sicurezza di rete quindi, si lavora sempre su questo doppio binario, in cui risulta semplice capire l'obiettivo, ma è tipicamente difficile capire come tale l'obiettivo può essere raggiunto. Per capire come raggiungere un determinato requisito è sempre tenere a mente, non solo le procedure/politiche organizzative, ma anche gli strumenti tecnici utilizzabili. Nello specifico, tra strumenti tecnici possiamo annoverare i protocolli di comunicazione sicura e la cifratura. In generale, quando si parla di sicurezza la componente umana riveste un ruolo particolare in quanto è da considerare uno dei principali elementi di debolezza. Il comportamento umano quindi è spesso una delle principali falle da tappare, e ciò può avvenire solo tramite la formazione. Nell'ambito della Cyber security, possiamo distinguere due "flavours": la sicurezza dell'informazione e la sicurezza di rete. Questi due aspetti sono profondamente diversi nonostante il forte legame che li unisce. Nello specifico, la sicurezza dell'informazione si occupa di preservare la confidenzialità, l'integrità e la disponibilità delle informazioni, ma anche di altre proprietà come l'autenticità, tracciabilità, la non ripudiabilità e affidabilità delle informazioni.

La sicurezza di rete invece, riguarda la protezione contro interventi di terzi che hanno come scopo di danneggiare il corretto funzionamento della rete, e quindi di garantire la fruibilità della rete anche a fronte di eventuali guasti o malfunzionamenti, sia volontari che non. I dettagli riguardanti il pentagono del NIST sono specificati nello standard FIPS 199 (o Standard for Security Categorization of Federal Information and Information System). Questo standard definisce gli obiettivi e che cosa si intende per perdita di sicurezza per ciascuno delle 5 caratteristiche definite dal NIST.



## C- Confidenzialità.

La confidenzialità, propriamente detta, può essere declinata in due modi diversi:

- **Confidenzialità dei Dati**

Questa assicura che assicura che persone non autorizzate non abbiano accesso a dati considerati privati o confidenziali. Questo è ciò che normalmente si assicura tramite cifratura

- **Privacy**

Tipicamente la privacy riguarda non solo chi ha accesso a determinate informazioni, ma anche quali sono queste informazioni, a chi vengono rivelate e da chi possono essere rivelate. Quindi non necessariamente un'informazione deve essere mantenuta sicura, semplicemente può essere privata, il proprietario dell'informazione vuole quindi controllare non solo a chi viene consegnata, ma anche da chi.

### NOTA [Svarione sulla Privacy]

Quindi dati che Tipicamente sono innocui, che non hanno. Nessun bisogno di essere cifrati, ma che l'individuo vuole controllare a chi vengano consegnati. Diciamo che in alcuni casi è evidente, la discrepanza tra quello che la normativa ci chiede e il comportamento degli individui, che spesso fanno in modo che la propria privacy sia violata attraverso l'uso delle tecnologie informatiche, spesso dai social media.

In Alcuni casi ci Sono delle situazioni paradossali in cui, ad esempio, un'utente nega il proprio consenso a tutta una serie di informazioni quando stabilisce dei contratti di fortuna di beni e servizi. E poi pubblica ogni dettaglio della propria Vita su un social network.

Questi comportamenti sono tipicamente quelli che in alcuni casi mettono in difficoltà chi deve erogare un servizio tra "sicuro", o meglio chi deve operare in una rete sicura. Perché è ovvio che noi non Possiamo impedire al detentore dell'informazioni di farci ciò che gli pare, però in alcuni casi i comportamenti rendono complesso assicurare quello che la normativa ci chiede.

In questo caso, verrà trattato il concetto di confidenzialità dei dati e non quello di privacy.

## NOTA

Una perdita di confidenzialità è la divulgazione non autorizzata di informazioni.

## I - Integrità

Anche il concetto di integrità può essere declinata In due modi diversi:

- **Integrità dei Dati**

Deve assicurare che i dati, sia quelli memorizzati, sia quelli trasmessi, debbano poter essere modificati o cancellati soltanto da coloro che hanno l'autorizzazione per farlo.

Questo concetto implica delle considerazioni ulteriori:

1. Autenticità dei Dati

Un oggetto digitale è autentico quando è realmente ciò che dice di essere; questa idea può applicata sia a dei dati classici, sia alle entità software

## 2. Non Ripudiabilità dei Dati.

La non ripudiabilità è molto legata all'autenticità dei dati, infatti una volta che un dato è autentico, non è possibile ripudiarlo, ovvero non è possibile far finta di non averlo ricevuto. Quindi una volta che si è ricevuto un dato e si hanno i mezzi tecnici per capire che è autentico, non è possibile rispedirlo indietro o cancellarlo facendo finta di non averlo ricevuto.

La non ripudiabilità si può assicurare facendo in modo che il destinatario del dato sia obbligato a segnalare al mittente che ha ricevuto quel determinato dato.

- L'integrità dei sistemi.

Valgono le stesse considerazioni fatte per l'integrità dei dati, ma in questo caso bisogna verificare che i sistemi, incluse la rete, siano modificate solo da chi ne ha l'autorizzazione.

### **NOTA**

Una perdita di integrità è la modifica o la distruzione non autorizzata di informazione

## A – Disponibilità

Un sistema garantisce la proprietà di disponibilità quando assicura che l'accesso al sistema non sia negato agli utenti autorizzati, nello specifico, la disponibilità non implica che il sistema funzioni correttamente ma solo che questo sia in stato di "UP and RUNNING".

Un sistema disponibile è accessibile con una certa percentuale, questa caratteristica fa spesso parte dei contratti di servizio. Le percentuali di disponibilità in genere sono molto alte, si fa spesso riferimento ai cosiddetti "five-nine" che letteralmente garantiscono la disponibilità del servizio nel 99,999% del tempo (generalmente una disponibilità inferiore al 95% è inaccettabile, perché significa che nel 5% nel tempo il servizio non sarà raggiungibile).

### **NOTA**

La perdita di disponibilità è l'interruzione dell'accesso o dell'utilizzo delle informazioni o di un sistema informativo.

## Obiettivi addizionali

### **A - Autenticità**

L'autenticità è la proprietà che verifica la genuinità e "l'affidabilità", ad esempio deve essere possibile verificare che un peer con cui si comunica, sia di fatto sia chi dice di essere. Questa proprietà garantisce la confidenza nella trasmissione di un messaggio o nell'identità di chi ha originato il messaggio.

Questa verifica d'identità, può avvenire sia quando il peer ha iniziato la comunicazione sia quando la comunicazione l'ha ricevuta. Ad esempio, quando si parla di home banking, non è il sito web della banca che ci effettua una chiamata al client, ma è quest'ultimo a richiamare il sito della banca, allo stesso tempo però si vuole essere sicuri che il sito web a cui sta accedendo sia veramente quello della banca e non quello di un impostore.

Si può quindi affermare che la proprietà di autenticità risulta fondamentale. Riprendendo l'esempio, anche la banca, per essere credibile ha bisogno di verificare l'identità del client che richeide la

connessione. Tipicamente questo implica che gli utenti e che le informazioni scambiate arrivino da una fonte sicura a fidata (o trusted).

## A - Tracciabilità

È l'operazione che associa ad un entità le azioni che vengono fatte da quella entità.

La tracciabilità o accountability ha due declinazioni particolari: una relativa alla sicurezza informatica e una relativa al mondo del business. La seconda è semplicemente la tracciabilità della nostra attività per fini di tariffazione.

L'altra motivazione, invece, ha a che fare con la sicurezza informatica. Tipicamente se si riesce a tracciare un sistema, anche se con un livello di sicurezza che non raggiunge il 100%, è più probabile capire quando si presenta un problema a da dove questo è stato originato.

La tracciabilità quindi, permette di capire attraverso un'operazione di reverse engineering, se un problema ha origine da un operazione umana, da un bug o semplicemente da un security hole.

Con "security hole" si fa riferimento ad una mancanza nei meccanismi di sicurezza, che non dipende da un mal funzionamento del sistema, ma che semplicemente non era mai stato ipotizzato che quella configurazione avrebbe potuto causare un accesso non autorizzato al sistema o a dei dati.

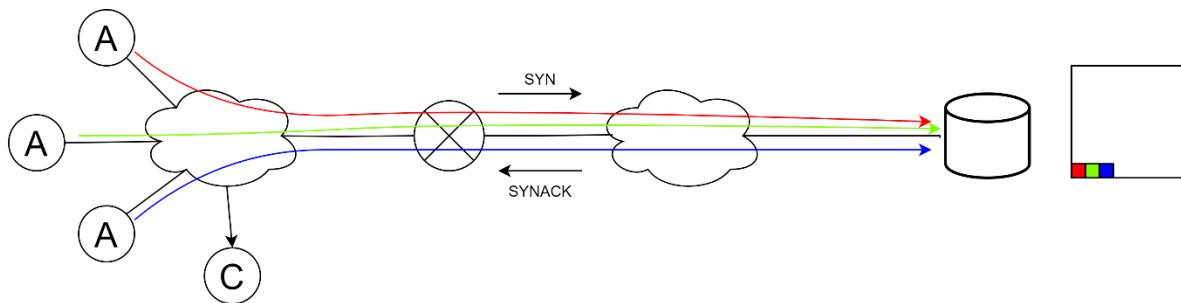
Per risolvere un problema, è fondamentale risalire alla fonte e riuscire a identificarla tramite operazione di tracciabilità. Quindi tracciare le attività degli utenti è fondamentale, sia che si tratti di utenti propri che di utenti terzi.

## Security Challenges

Per assicurare le 5 proprietà del pentagono CIA-AA e il loro sotto-oggettivi, tipicamente bisogna risolvere un certo numero di sfide, di seguito se ne vedranno le 10 più importanti.

La sicurezza non è semplice da assicurare, in quanto, mentre i requisiti e gli obiettivi sono chiari, basti pensare che spesso sono identificati da una singola parola, anche autoespliaviva. Differenti sono i criteri per raggiungere questi obiettivi, spesso infatti i meccanismi utilizzati per assicurare queste proprietà e i loro requisiti, sono complessi e richiedono ragionamenti non immediati. Volendo fare un paragone, è come quando si gioca allo sceriffo e ai banditi, in cui di solito il bandito pensa in maniera diversa da come pensa lo sceriffo. Nell'ambito della cyber security, quando bisogna proteggere un sistema da attacchi potenziali non si deve pensare come chi sta in difesa, ma piuttosto, si deve pensare come chi sta in attacco e verificare che tutte le possibilità di attacco siano chiuse. Ovviamente è impossibile pensare di chiudere tutte le vie di accesso per l'attaccante, però più si riesce a fare un'operazione di analisi dettagliata, migliore sarà il servizio di protezione (o di Network Information Security) che si riesce a garantire. Spesso le vulnerabilità di un sistema non sono così evidenti e risulta difficile, se non impossibile, bloccarle. Nella maggior parte dei casi le vulnerabilità vengono scoperte casualmente o dopo che sono già state utilizzate per compiere un attacco. In alcuni casi è possibile che quando venga rilevata una vulnerabilità, questa venga segnalata agli amministratori di sistema e notificata alla comunità internazionale. In questa situazione però, ne vengono a conoscenza non solo gli amministratori, anche potenziali attaccanti, e che quindi potrebbero "fare la corsa" a cercare il sistema che ancora non ha messo la patch alla vulnerabilità. Questo implica che tipicamente le procedure per fornire servizi di sicurezza, in particolare sicurezza di rete, siano contro-intuitivi. Volendo fare un esempio pratico sulla proprietà di disponibilità, che risulta essere una delle proprietà che più spesso si cerca in un sistema, in quanto compromettere la qualità di disponibilità non è necessario l'accesso non autorizzato nel sistema.

Tipicamente è presente una rete che offre un certo servizio, questa viene attaccata tramite uno o più router al resto del mondo dove si trovano i potenziali attaccanti.



Per quanto riguarda gli attacchi che puntano a minare la disponibilità, dobbiamo differenziare tra attacchi che vogliono minare la disponibilità della rete o dell'entità che offre il servizio. Entrambi i tipi di attacco ottengono lo stesso risultato, ovvero rendere i servizi indisponibili, ma differiscono nel procedimento. Si può sia sovraccaricare la rete di traffico in modo che le richieste legittime non arrivino mai all'entità che eroga il servizio, oppure si può sovraccaricare l'entità facendo sì che risponda in tempi e modi non accettabili. Si può quindi vedere che non c'è bisogno di pensare solo al servizio, ma si potrebbe anche lasciar perdere il servizio e puntare a sovraccaricare la rete che lo connette al resto del mondo, basta questo per rendere il servizio non disponibile. Per fare un esempio di attacco verso chi eroga il servizio. Gli attaccanti possono fare un attacco classico, che si basa su un minimo di conoscenza del TCP che di base è solo il protocollo più utilizzato a strato quattro per stabilire delle sessioni e trasportare i dati. Il protocollo TCP crea una sessione tra il mittente e il destinatario e quando la connessione viene aperta, vengono creati i cosiddetti socket. Si definisce come socket la coppia di indirizzo e porta di strato 4. Ad una sessione TCP è quindi applicata una coppia di socket ovvero indirizzo IP e porta mittente e indirizzo IP e porta di destinazione; la coppia di destinazione identifica la sessione TCP. Quindi ogni volta che viene creata una sessione TCP, sia il sistema di origine che il sistema di destinazione allocano delle risorse di calcolo, nello specifico, della RAM; il socket infatti, è un file all'interno del quale si leggono e scrivono informazioni e metadati dei dati scambiati. Per rendere indisponibile il servizio del sistema vittima, senza averne accesso, si possono tentare degli attacchi di tipo DDOS (Distributed DOS – Distributed Denial Of Service).

Si parla di Distributed DOS perché questi vengono svolti da fonti diverse, che ovviamente lavorano all'unisono. In questo modo, l'attacco è più difficile da identificare, o meglio, si identifica e si blocca ma con un po' di ritardo. Tornando all'esempio, data una sessione TCP richiedente una pagina di un server web. Questo server impiega una parte delle sue risorse per aprire un socket e quando il client invia una pacchetto SYN, il server risponde con un pacchetto SYN ACK, per poi concludere il 3 way handshake con l'ACK del client, questo però non arriverà mai, lasciando quella piccola parte di memoria, che il server aveva allocato, appesa. Se questo procedimento viene ripetuto per migliaia di volte e molto rapidamente, prima che scada il time-out che determina il tempo massimo, dopo il quale liberare le memorie, si potrebbe arrivare ad occupare praticamente tutta la memoria di questo server; anche nel caso in cui la memoria non arrivi a saturazione ci potrebbe comunque essere un forte rallentamento del servizio. In questo modo chiunque voglia accedere in maniera legittima, può vedere che la sua risposta non sarà mai servita, o comunque la risposta sarà molto rallentata; il servizio risulterà quindi indisponibile o poco fruibile. Anche nel caso della scarsa fruibilità, è di fatto indisponibile relativamente ai requisiti di qualità con cui il servizio deve essere garantito. Quando questo rallentamento diventa inaccettabile, si può dire di aver minato la disponibilità di un sistema, semplicemente sovraccaricandolo. Esistono delle tecniche per rilevare in modo precoce il tentativo di sovraccarico e per limitarlo, cercando in questo modo di evitare che gli utenti legittimi subiscano

dei malfunzionamenti del servizio. Supponendo di aver messo in piedi la soluzione per controbattere a un attacco di DOS (Queste considerazioni valgono per tutti i tipi di attacco), bisogna individuare in che punto della rete fisica e a che livello protocollare posizionarla. La scelta di questo a volte, può rallentare la messo in opera di meccanismi di sicurezza; solito un meccanismo di sicurezza è molto più oneroso che mettere in piedi un nuovo protocollo, perché utilizzare un nuovo protocollo è abbastanza semplice. Mentre nel caso di meccanismo di sicurezza, di solito ci sono delle operazioni accessorie che sono più complicate, tra le quali, ad esempio la gestione delle chiavi. Questo perché, nei meccanismi di sicurezza, è spesso necessario garantire l'accesso solo alle persone autorizzate e distribuire informazioni che siano segrete. Per effettuare la creazione, la distribuzione e la protezione di queste informazioni è fondamentale tenere in conto l'utente umano; anche avendo a disposizione un meccanismo estremamente affidabile, che garantisce l'unicità della mia chiave e un protocollo iper sicuro per distribuire la chiave, ma se l'utente umano copia la chiave su un post-it dell'ufficio, chiaramente ha reso inutile tutte le misure viste precedentemente perché non ha adeguatamente protetto l'informazione segreta. Quindi la formazione fa parte delle attività della sicurezza informatica. Inoltre, potrebbe essere necessario fare affidamento su protocolli che complicano lo sviluppo dei meccanismi di sicurezza. Ad esempio, ci potrebbero essere dei protocolli che guardano al tempo di risposta dei messaggi per cercare di identificare se la transazione sta avvenendo con un soggetto autorizzato o con un soggetto non autorizzato, che magari sta cercando di attuare un attacco DDOS. Questo implica che se un ritardo di risposta è dovuto semplicemente al fatto che la rete è un po' congestionata, si potrebbe andare a negare la comunicazione e quindi l'accesso del servizio anche a utenti legittimi che però stanno solamente sperimentando delle condizioni di rete leggermente differenti da quanto previsto.

Quindi tutte quelle situazioni che prevedono il monitoraggio dei tempi di risposta, di solito soffrono molto della variabilità dei ritardi introdotti dalle condizioni specifiche del dispositivo oppure dalle condizioni di congestione della rete che connette il dispositivo con il servizio che vogliamo proteggere.

Tornando all'esempio del Bandito-Sceriffo. Chi attacca, deve trovare una vulnerabilità, mentre chi protegge deve proteggere da tutte le vulnerabilità, ammesso che questo sia possibile. Il risultato è una lotta, chi attacca ha bisogno di sfruttare una sola vulnerabilità, mentre chi protegge, deve proteggere da tutte le vulnerabilità contemporaneamente. Tipicamente nell'ambito delle grandi infrastrutture e delle grandi organizzazioni, si tende a trascurare la sicurezza fino a quando non capitano incidenti di grandi dimensioni. A quel punto, si mettono in piedi tutta una serie di misure, spesso iper restrittive. Ma altrimenti le misure di sicurezza spesso sono molto blande, anche se questo ovviamente non vale per tutti; ad esempio, in ambiti industriali, nascono con misure di sicurezza molto avanzate. Anche se nella maggior parte dei casi non si dà così importanza al tema della sicurezza informatica, non tanto quando si parla di protezione dei dati ma piuttosto quando si tratta di sicurezza di rete. Perché per assicurare la sicurezza di rete è assicurare un monitoraggio costante, per controllare che il sistema funzioni, soprattutto dal punto di vista dei famosi 5 Punti FCAPS. Quindi un sistema non deve funzionare solo dal punto di vista sicurezza, ma anche da altri aspetti, di cui spesso quello più importante per l'utente che usufruisce del servizio è la performance. Di conseguenza la sicurezza viene spesso vista quasi come una scocciatura e si tende a trascurarla, fino a quando succede qualcosa che non permette più di trascurarla. In altri casi, la sicurezza viene vista esclusivamente come una voce di costo, in quanto richiede impegno e risorse finanziarie, perché è necessario mettere in piedi un'infrastruttura che effettui un monitoraggio costante. Se non è presente un sistema di monitoraggio, quello che succede nel sistema è che risulta impossibile accorgersi di un eventuale malfunzionamento, sia voluto che occasionale, fino a quando i danni non sono evidenti. Progettare il sistema direttamente con i meccanismi di sicurezza è più facile ed efficiente che prende un sistema che nasce non sicuro e applicargli una patch che lo renda il più sicuro possibile. Alla fine dei conti quindi, un sistema creato senza misure di protezione fin dall'inizio, risulterà più costoso e tipicamente meno efficace. Se invece si progettare un sistema con i

meccanismi di sicurezza integrati, l'efficacia è tipicamente molto superior. Prendendo il protocollo IP come esempio, la differenza principale tra IPv4 e IPv6 è che, mentre IPv6 Nasce con l'estensione di sicurezza, IPv4 ha una patch rappresentata da IPSEC. Allo stesso modo, per quanto riguarda lo strato 4, TCP nasce insicuro, senza nessun tipo di protezione e viene aggiunti SSL/TLS per rendere sicuri i socket (si pone quindi tra strato 4 e strato applicativo). In ultima analisi, molti utenti ma anche molti amministratori di rete, percepiscono la sicurezza solo come una scocciatura perché, come già detto, richiede soldi, impegno e di mettere in opera procedure che richiedono un po' più di lavoro. Di conseguenza vengono viste con una cosa che non serve a niente, ma che si deve fare perché imposto dalla legge; tutto ciò tende a rendere più semplice il compito di chi ha intenzioni malevole.

## Nomenclatura

- **Attacco**  
Azione che compromette la sicurezza di un sistema o dell'informazione Gestita da quel sistema.
- **Meccanismo di sicurezza**  
Processo progettato per rilevare, prevenire o recuperare da una situazione di attacco. Un meccanismo di sicurezza si può declinare in varie forme, esistono avere meccanismi che servono a rilevare gli attacchi e altri che servono a prevenire; siccome non si riesce a prevenire tutti gli attacchi che esistono, sono disponibili anche meccanismi per recuperare la piena funzionalità del sistema o il pieno accesso alle informazioni dopo che c'è stato un attacco ed è stato annullato o bloccato.
- **Servizio di sicurezza**  
È un servizio di comunicazione che migliora La sicurezza dei dati e dei sistemi che gestiscono quei dati. Questi servizi tipicamente utilizzano o più meccanismi di sicurezza, ciò significa che un servizio di sicurezza può utilizzare un meccanismo per rilevare gli attacchi, un altro per prevenirli e un altro ancora per il recovery dopo che l'attacco c'è stato.

## NOTA

Rilevare l'attacco, non significa accorgersi che il servizio non funziona più, ma significa rilevare un'attività malevola nelle fasi iniziali dell'attacco, cioè quando ancora non è stato effettuato (non è stato inflitto nessun danno all'informazione o all'infrastruttura che eroga un terminato servizio).

Prevenire invece, significa mettere in piedi tutti quei meccanismi di sicurezza che proprio non permettono neanche di tentare l'attacco.

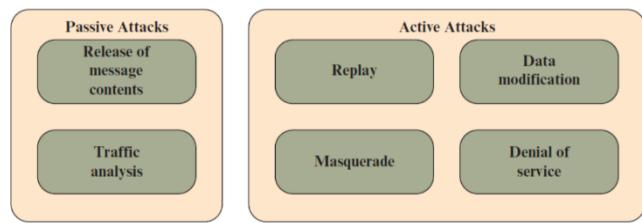
- **Minaccia**  
Evento o circostanza che tipicamente impatta in maniera negativa le informazioni o l'operatività di sistemi che gestiscono quella informazioni. Mentre l'attacco è un tipo di attività malevola, quindi effettuata malintenzionati che tentano di collezionare informazioni, negare l'accesso ai servizi, degradarli, distruggerli, o impedire l'accesso (la maggior parte degli attacchi sono "innocui"); le minacce invece sono poche e spesso causano un danno significativo.

## NOTA

Gli attacchi considerati "innocui" delle volte possono rappresentare un'attività preliminare per congegnare un attacco più importante.

## Attacchi di Sicurezza

Gli attacchi possono essere attivi o passivi, non c'è una pericolosità maggiore in quanto sono profondamente diversi. Mentre i primi sono difficili da prevenire, gli altri sono difficili da rilevare. Quindi, in generale, possono rappresentare entrambi un



problema molto significativo. Si definisce come passivo, un attacco che tenta di impara fare uso dell'informazione del sistema, senza alterare il sistema stesso. Mentre un attacco attivo, è un attacco che tenta di alterare le risorse di un sistema, incluse le informazioni stesse, e di minarne l'operatività. Gli attacchi passivi sono tipicamente di due tipologie: "rilascio del contenuto dei messaggi cifrati privati" e "analisi del traffico". Si parla di attacchi passivi, in quanto non includono un ruolo attivo dell'attaccante. Gli attacchi attivi invece, sono tipicamente categorizzabili in quattro tipologie: "replay", "data modification", "masquerade" e "denial of service".

### Attacco Passivo

Gli attacchi passivi Tipicamente sono attacchi basati sul sull'ascolto e il monitoraggio delle trasmissioni, sono molto semplici da attuare nel caso delle reti wireless, proprio perché una trasmissione wireless, per definizione, avviene su un mezzo che è condiviso; quindi un attacco passivo su una rete wireless, tecnicamente, non è illegale (È invece illegale trasmettere su una frequenza, quando non autorizzato). L'obiettivo degli attacchi passivi è ottenere informazioni che possono essere sia l'accesso al contenuto dei messaggi che vengono trasmessi, sia acquisire delle proprietà dei messaggi (es. header) che vengono trasmessi. Proprio perché gli attacchi passivi non implicano un ruolo attivo dell'attaccante sono estremamente difficili da rilevare, quindi, di solito l'unica cosa che si può fare è cercare di prevenire il più possibile. In alcuni casi un attacco "attivo" in alcuni casi è preceduto da un attacco passivo, ad esempio, l'attacco passivo potrebbe servire per identificare una chiave di cifratura o di accesso, ed una volta che si è in possesso della chiave di accesso, è possibile effettuare un attacco attivo di data modification. Quell'attacco, quindi, potrebbe essere composto in realtà da due tipi di attacchi: un primo attacco in cui acquisisco tramite la cattura del traffico, una determinata chiave e un secondo attacco in cui effettuo un attacco di replay o un attacco di data modification o anche di accesso non autorizzato. Questa procedura è tipica degli attacchi, ad esempio nelle reti wi-fi, mettendosi in ascolto della comunicazione di terzi con un Access Point, è possibile utilizzare gli strumenti open source, che sono pubblicamente disponibili, per dedurre delle chiavi; magari utilizzando il cosiddetto attacco dizionario, ovvero quello che prevedono la combinazione di stringhe note per dedurre chiavi, se la chiave debole, ovvero composta tramite regole facilmente identificabili, si può riuscire a risalire alla chiave di cifratura. Per prevenire che accada tutto questo è possibile mettere in piedi dei meccanismi che rendono difficile la deduzione della chiave da terzi. Esistono svariati meccanismi che si possono utilizzare per prevenire un attacco di questo tipo, ad esempio, è possibile andare ad effettuare tutta una serie di operazioni per di generare una chiave che non sia riconducibile a una composizione di stringhe note. Utilizzando, ad esempio, una stringa generata in maniera pseudo casuale, quella è sicuramente difficile da identificare tramite un attacco a dizionario. Inoltre, è molto importante essere sicuri che i protocolli che si utilizzano per la cifratura non abbiano delle vulnerabilità note. In caso di vulnerabilità note, delle volte se si utilizza una chiave molto lunga, la vulnerabilità viene fortemente mitigata; in altri casi invece l'unica soluzione è cambiare il metodo di cifratura. Ad esempio, se su una rete Wi-Fi utilizzo il protocollo di cifratura WEP, nonostante i 128 bit utilizzati rimane comunque vulnerabile, esistono infatti dei meccanismi sia attivi sia passivi che permettono tramite l'acquisizione di una significativa quantità di informazione, di identificare la chiave.

Se invece, sempre in una rete Wi-Fi si utilizzano i vari framework di WPA(1,2 o 3)(v2 e v3 hanno miglioramenti nella sicurezza) risulta particolarmente complesso per un attaccante identificare la chiave di cifratura. Questa tipologia di attacco, di solito è basata su attacchi a dizionario, ovvero sulla cattura dello scambio iniziale di informazione e sul tentativo di indovinare la chiave. Nello specifico, esistono degli strumenti open source, che una volta catturati i pacchetti dello scambio, generano delle chiavi attraverso la combinazione di dizionari. Tipicamente utilizzano dizionari per una specifica lingua combinati con qualche segno di punteggiatura e qualche numero. In questo modo si vanno a testare un certo numero di combinazioni che per quanto molto grande, è molto più piccolo di tutte le possibili combinazioni di lettere, numeri e segni di punteggiatura.

## NOTA

Un esempio di tool utilizzato per generazione di chiavi tramite dizionari è “Jack the ripper”.

Come già detto, gli attacchi passivi consistono nell'ascolto e nel monitoraggio di trasmissioni e mentre ciò risulta è particolarmente semplice nelle reti wireless, è più complicato quando si tratta di reti cablate; in questo caso c'è bisogno di punto di monitoraggio che acceda fisicamente ad un'infrastruttura di rete per posizionare una sonda ed effettuare il mirroring. Se tutta la comunicazione tra client e server avviene su rete cablata, l'attaccante deve per forza avere accesso alla rete fisica, può ottenere questo accesso in maniera diretta o forzata. Nel caso in cui l'attaccante riesca ad avere accesso ad infrastrutture di rete che svolgono un ruolo pubblico, potrebbe provare ad esportare delle rotte che globalmente alterano uno o più router di internet, al fine di deviare una parte del traffico sul dispositivo di rete controllato dall'utente malevolo. In questo modo l'attaccante sarebbe in grado di monitorare il traffico senza accedere fisicamente ai dispositivi terminali, ma semplicemente alterando le tabelle di routing.

## NOTA

Il routing su internet avviene tramite un processo distribuito tra sistemi autonomi, il routing avviene tramite protocollo BGP, cui i vari Autonomous System esportano i propri prefissi e decidono quale come instradare l'informazione. L'analisi del traffico è qualcosa di molto più difficile da rilevare, e tipicamente non punta ad acquisire il contenuto dei messaggi ma piuttosto a determinare il tipo di attività che c'è su una rete.

Perché si fa analisi del traffico?

Come già visto i meccanismi di sicurezza messi in opera assicurano, in modo molto difficile da corrompere, la confidenzialità delle informazioni, e di conseguenza è impossibile svolgere un attacco di tipo “release of content message”. D'altro canto, fare traffic analysis permette di vedere quando ci sono picchi di traffico, da quali utenti arrivano, con quale frequenza, che tipo di mole di dati è scambiata e da quali utenti. Perché se è vero che i payload dei protocolli applicativi sono cifrati, tipicamente l'intestazione di strato tre e quattro non è cifrata ed è quindi sempre possibile leggere l'intestazione IP e l'intestazione TCP (o UDP) se ho accesso ai messaggi scambiati. Se invece parliamo di un collegamento wireless cifrato, in questo caso si può avere accesso solo all'intestazione di strato 2, in quanto sia nelle reti cellulari che nelle reti wi-fi la cifratura è posta tra lo strato 2 e 3. Di conseguenza, non sono disponibili le informazioni sull'identità di chi sta comunicando a livello di indirizzo IP e di protocollo TCP(UDP). Non riesco quindi a capire se un pacchetto trasporta IPv4 o IPv6 e se dentro c'è un protocollo TCP o UDP. Nel caso di reti cellulari o wi-fi, le informazioni di strato 2 hanno comunque la loro rilevanza, in quanto permettono di determinare la posizione, l'identità pubblica (l'identità infatti dipende dal livello protocollare a cui viene pensata la cifratura) la frequenza e la lunghezza delle informazioni. Quindi, se si vuole pianificare un attacco DDOS,

un'analisi del traffico tipico può essere molto utile, in quanto permette di identificare quali sono i periodi di picco del traffico. Sapendo ciò, sarà possibile mascherare una attacco DDOS semplicemente avviando l'attacco durante l'ora di punta, in cui è normale vedere un aumento del traffico.

### Attacco Attivo

Gli attacchi attivi solitamente sono quelli che coinvolgono la modifano i dati, la creazione di falsi dati, l'accesso non autorizzato, la messa fuori servizio dell'accesso ai dati o dell'accesso al sistema.

Attacchi di questo tipo si possono suddividere in quattro macro categorie:

- attacchi di replay,
- attacchi di mascheramento
- attacchi di modifica dei messaggi
- attacchi di negazione del servizio (violazione della disponibilità).

Per definizione, gli attacchi attivi, implicano un'attività dell'attaccante e c'è quindi la necessità di poterli rilevare e recuperare la situazione precedente all'attacco. Quando di parla di prevenzione, è una chimera pensare di poter prevenire tutti i possibili attacchi, soprattutto quelli che sfruttano tecniche non ancora note. Di conseguenza la prevenzione assoluta è impossibile, ciò che si può fare è l'attività di tracciamento al fine di effettuare la rilevazione sia dell'attacco, che del problema (Accountability). Quindi, è bene distinguere le misure per rilevazione di un attacco, da tutto quell'insieme di meccanismi che dopo un attacco, permettono attraverso l'analisi dei record di capire da dove viene il problema. Ciò è molto importante poiché non conoscendo l'origine dell'attacco è possibile che questo si ripeta senza avere idea di come bloccarlo. È fondamentale tenere a mente che la prevenzione inseguiva la rilevazione, infatti solo una volta che l'attacco viene rilevato si posso acquisire informazioni sul sistema vulnerabile ed eventualmente mettere in piedi un meccanismo per prevenire un attacco della stessa tipologia. Infatti, tipicamente è molto difficile, se non impossibile, andare a prevenire attacchi che non sono mai stati tentati prima su nessun sistema, i cosiddetti attacchi 0-day. Questo tipo di attacchi, di solito sfruttano una vulnerabilità nuova e sono molto difficili da rilevare nel momento in cui chi scopre la vulnerabilità ha intenzioni malevoli. Se invece la vulnerabilità è nota, ma è "difficile" da patchare in maniera affidabile e tempestiva, in questo caso la rilevazione riesce con più facilità.

### **Attacco di replay**

È un attacco che consiste nella cattura passiva di alcuni dati e la ritrasmissione di questi in rete per produrre un effetto non autorizzato e ovviamente indesiderato. Ad esempio, se non riesco ad acquisire il contenuto del messaggio, ma riesco a catturare tutto il pacchetto e non sono presenti dei meccanismi contro gli attacchi di replay, posso rinviare in rete il messaggio catturato producendo un effetto indesiderato. Ad esempio, tra i vari attacchi che si possono fare in wi-fi, ci sono quelli di tipo "injection", nei quali si va a catturare un pacchetto wi-fi particolare e lo si replica in rete; questo è uno dei motivi che ha portato alla definizione degli standard di sicurezza odierni. Infatti, tra tutti i pacchetti che circolano c'è uno scambio particolarmente facile da identificare soprattutto nelle reti wireless: il pacchetto di ARP request. L'ARP request è facile da individuare in quanto a strato 2 avrà come mittente, il terminale che emette la request e come destinatario l'indirizzo di broadcast. L'ARP request serve a richiedere l'indirizzo MAC del dispositivo che detiene un determinato indirizzo IP.

Di conseguenza, il mittente e destinatario sono facili da identificare, e poiché l'ARP ha sempre la stessa lunghezza basata su una struttura fissa, è facilmente identificabile anche la lunghezza del messaggio. Nella richiesta, al posto dell'indirizzo MAC di destinazione ci sono tutti "0", quindi salvo dei campi particolari, del messaggio di ARP request si conosce quasi tutto. È vero che il messaggio è cifrato, ma un volta che il pacchetto viene replicato in rete, sarà sempre ritenuto un messaggio valido dal detentore del indirizzo IP, che risponderà con un ARP response (unicast). Se non è presente un meccanismo che prevenga un attacco di replay, è possibile catturare in rete un messaggio e iniettarlo 10 milioni di volte e ottenere 10 milioni di risposte. È importante notare che questo non implica la saturazione della rete in quanto i pacchetti ARP hanno priorità bassa. Ciò che realmente risulta essere pericoloso è dato dal fatto che tipicamente il destinatario del ARP request vede tutte richieste come diverse, di conseguenza questo risponde con messaggi "diversi", che vengono cifrati in maniera diversa. Tutto ciò è molto critico, perché se una informazione A viene cifrata in A1, A2, A3, ... , An; dove queste sono tutte versioni cifrate della stessa informazione in chiaro. Se vengono cifrate tutte con la stessa chiave ma con semi diversi, alla fine, siccome i semi normalmente vengono sempre scritti sul messaggio cifrato e siccome il seme deve variare ogni volta, questo viene anche scritto in chiaro, in modo tale che chi riceve il messaggio ne venga a conoscenza. Dalla parte del ricevitore, avendo già una chiave di cifratura, la si concatena e si ottiene una chiave di cifratura più grande per decifra il messaggio. Il problema si presenta quando l'utente malevolo è a conoscenza del messaggio, o meglio, l'attaccante sa che tutti questi messaggi cifrati che sono stati trasmessi sono corrispondenti tutti allo stesso messaggio, tramite delle tecniche di reverse engineering è possibile risalire quale era il contenuto del messaggio originale, e confrontandolo con la versione cifrata, risulta possibile dedurre la chiave di cifratura e quindi avere accesso al sistema. Un attacco di replay quindi, può essere innescato per fare in modo che un sistema remoto risponda in maniera non autorizzata o non voluta ad una richiesta, oppure per innescare un processo più complesso, ad esempio, l'acquisizione di una chiave di cifratura per violare la confidenzialità.

## EXTRA

- *Come è possibile prevenire un attacco passivo di "Release of Message Content"?*  
Mettendo in campo procedure di cifratura molto forti.
- *Come è possibile impedire i meccanismi di analisi del traffico?*  
Per quanto riguarda le reti di wireless si possono mettere in piedi solo meccanismi di cifratura, anche se negli ultimi anni è stato implementato un meccanismo di anonimizzazione del traffico, che consiste nella randomizzazione periodica dell'indirizzo MAC. Di conseguenza un terminale non utilizza sempre lo stesso indirizzo MAC.  
Questo meccanismo di protezione rende difficile l'erogazione di alcuni servizi, ad esempio, se ho delle colonie di monitoraggio del traffico pedonale, che non viene fatto attraverso le telecamere ma andando invece a tracciare l'attività dei terminali wi-fi, posso identificare i flussi di traffico pedonale semplicemente andando a vedere un indirizzo di strato 2 quante volte lo vedo e su quale colonnina, posso quindi determinare se ho un traffico significativo e su quale tratto.  
Se invece il flusso pedonale è più lungo e il terminale cambia l'indirizzo MAC durante il percorso, potrebbe sembrare che il terminale sia scomparso ed essere considerato come se fosse uscito fuori dalla zona di monitoraggio, anche se in realtà non è così, perché il terminale è sempre lo stesso, ma ha "cambiato l'identità di strato 2".  
Quindi, in alcuni casi, questi meccanismo di mascheramento potrebbero mettere in difficoltà un analisi del traffico di tipo malevola un'analisi con scopi legittimi.

## Recap

Abbiamo visto una prima classificazione degli attacchi informatici: questi attacchi possono essere classificati come passivi o attivi.

Gli attacchi passivi sono quelli che non implicano un'azione attiva (l'invio o la modifica di qualsiasi tipo del traffico) da parte dell'attaccante. Proprio per questo sono molto difficili da rilevare in quanto non lasciano traccia e l'unica cosa da fare è adattare delle configurazioni che li prevengano, o comunque delle azioni che impediscano ad un attacco di tipo passivo di andare a buon fine, di essere efficace.

Al contrario, gli attacchi attivi sono quelli che prevedono o l'invio di traffico oppure la modifica di qualsiasi tipo del traffico da parte di un attaccante. La cosa più fattibile è la rilevazione tempestiva dell'attacco (prima che l'attacco abbia portato a problemi seri all'interno della rete o del sistema), mentre una prevenzione a priori è praticamente impossibile perché le sfide a cui sono sottoposti gli attaccanti e chi progetta sistemi di sicurezza sono opposte. L'attaccante deve trovare una falla per portare a termine il suo attacco, chi progetta il sistema di sicurezza deve tappare tutte le falde esistenti e deve prevedere in anticipo eventuali possibili security hole, quindi buchi di sicurezza che permettono di effettuare un attacco a cui ancora nessuno ha pensato. Siccome i modi di pensare associati a eseguire l'attacco e alla difesa sono diversi è molto difficile a priori cercare di prevenire qualsiasi tipo di attacco.

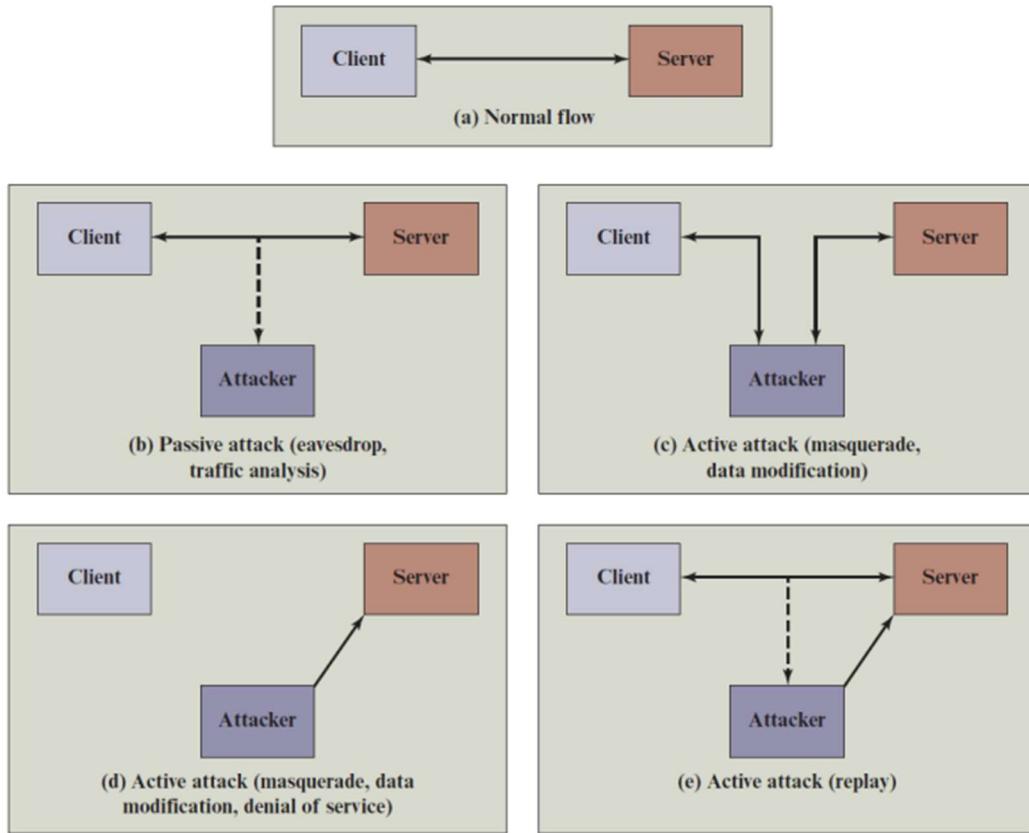
Per gli attacchi passivi la cosa è più ragionevole è la prevenzione, mentre per gli attacchi attivi la cosa più ragionevole è rilevazione tempestiva e poi la rilevazione di recupero, ovvero rilevare l'attacco e bloccarlo prima che abbia fatto danni significativi.

Nella realtà gli attacchi, soprattutto quelli più complessi, avranno sia una componente attiva sia una passiva. Saranno quindi inizialmente preceduti da una fase passiva di osservazione del traffico e di acquisizione delle informazioni che poi servono per effettuare la parte attiva dell'attacco.

I quattro macrotipi di attacco sono: replay (attacco di rilancio), di mascheramento (o impersonificazione), modifica dei messaggi e impedimento del servizio.

In realtà è molto difficile andare a categorizzare un attacco esclusivamente all'interno di questi 4, tipicamente è una combinazione di questi.

## Attacchi alla sicurezza



Nello schema (a) è rappresentato il flusso normale del traffico, quindi da un client a un server, assumendo che la maggior parte delle applicazioni che sono in rete obbediscono al paradigma client-server.

Negli attacchi passivi (b) la linea tratteggiata significa che l'attaccante acquisisce informazioni in maniera passiva, mentre la linea continua significa scambio di traffico vero e proprio. Quindi negli attacchi passivi, sia quelli di ascolto sia quelli di analisi del traffico, l'attaccante acquisisce il traffico senza farsi scoprire, in maniera passiva e senza modificarlo.

Nell'attacco attivo (e) di replicazione l'attaccante acquisisce del traffico, che può essere anche cifrato, e l'attaccante non necessariamente riesce a decodificare il traffico cifrato, può anche prendere il traffico cifrato così com'è e poi lo reinietta in rete. In questo esempio c'è client-server ma possiamo vederlo tipicamente come la rete a cui è attaccato il server. Questo perché sappiamo che gli effetti degli attacchi possono essere sia sulla rete sia sui sistemi terminali, non necessariamente sui sistemi terminali. La cybersecurity considera sia la sicurezza delle informazioni sia la sicurezza di rete perché le cose sono legate. Posso impedire ad esempio di accedere alle informazioni semplicemente bloccando il servizio di rete, senza andare a bloccare il sistema terminale. A seconda della configurazione del sistema terminale o della rete a cui il sistema terminale è attaccato reiniettare questo traffico può dar origine ad effetti indesiderati.

Effetti indesiderati: se il traffico fosse potenzialmente legittimo potrebbe dar luogo a delle risposte, se la rete invece è adeguatamente protetta è solo traffico di disturbo.

La prima misura per controbattere gli attacchi di replay è utilizzare i numeri di sequenza. Se catturo un pacchetto che ha un numero di sequenza e lo rilancio, il sistema terminale si accorge che quello è un pacchetto replicato. Se non lo posso modificare perché se lo vado a modificare ad esempio altero i campi che mi proteggono dalla modifica indesiderata dei pacchetti, il pacchetto non sarà più valido. Quindi se lo prendo così com'è e lo rilancio, un sistema terminale potrebbe riuscire ad accorgersi della modifica, poi dipende da dove viene fatta questa modifica.

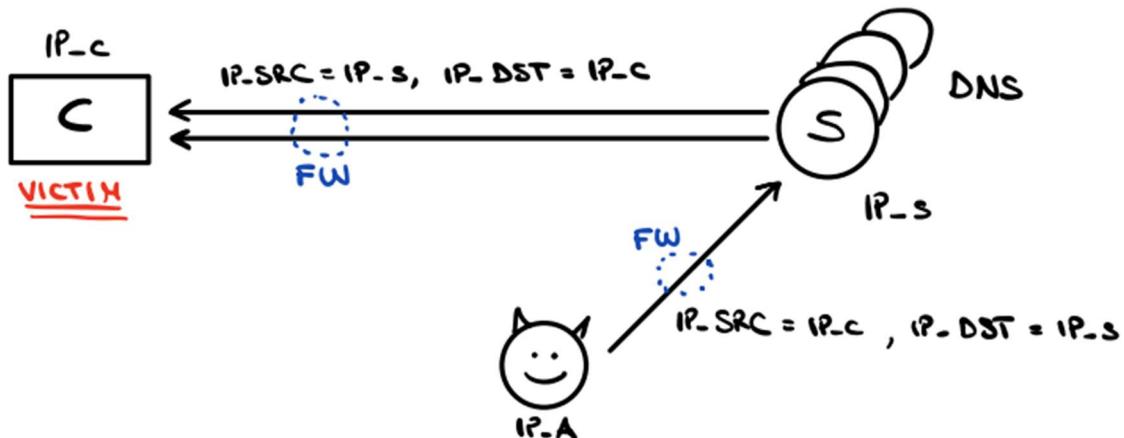
In alcuni casi è possibile effettuare questa operazione senza problemi, ad esempio, se reinolto porzioni di traffico udp, se non è implementato un meccanismo con un numero di sequenza a livello applicativo, il destinatario ogni volta risponderà alla stessa richiesta. Perché udp è un protocollo a datagramma, quindi non ha il concetto di sessione e non usa i numeri di sequenza e quindi tutte le applicazioni che usano udp sono potenzialmente sensibili ad attacchi di questo tipo a meno che non è l'applicazione stessa che usa i numeri di sequenza. È vero che abbiamo l'id del datagramma all'interno del protocollo ip che distingue tutti i datagrammi, però nessun router tiene memoria dei datagrammi già ricevuti. Quindi se invio un datagramma il router continuerà a inviarlo verso la destinazione. L'id del datagramma serve principalmente a gestire opportunamente le operazioni di frammentazione e riassemblaggio a destinazione.

Attacco di mascheramento o data modification o denial of service (d): l'attaccante invia del traffico. Non c'è l'attacco di replay perché non viene replicato nessun traffico precedentemente scambiato.

Gli attacchi di mascheramento o impersonificazione sono tutti gli attacchi in cui l'attaccante fa finta di essere un altro. In questo caso l'attaccante fa finta di essere un nuovo peer che sta inviando traffico in rete. Ad esempio, l'attaccante potrebbe utilizzare la tecnica dell'ip spoofing, quindi utilizzare un indirizzo ip diverso dal suo in modo tale che le risposte non arrivino a lui. È il classico attacco di denial of service.

Come faccio ad attaccare il mio client?

Consideriamo questo scenario:



L'attaccante può andare direttamente ad attaccare il client, ma magari sistemi di intrusion detection o intrusion protection del client potrebbero accorgersene.

La cosa più sicura è utilizzare del traffico originato da server legittimi.

Supponiamo che questo sia un server DNS ad alta capacità, cioè opportunamente ridondato per gestire grandi moli di traffico.

L'attaccante invierà del traffico, ad esempio una richiesta DNS (trasportata da udp) in cui l'ip della sorgente è  $IP_c$  e l'ip della destinazione è  $IP_s$ . L'attaccante sta impersonificando il client perché in questo modo, se il pacchetto riesce ad arrivare al server DNS, innescherà una risposta con  $IP\_SRC = IP_s$  e  $IP\_DST = IP_c$ .

Se effettuo questo attacco verso un numero molto elevato di server, ogni server vedrà una quantità di traffico modesta e quindi non si accorgerà, non avrà grossi problemi, e poi, se è un server DNS non risolverò sempre la stessa query ma potrei risolvere tutta una serie di siti well known, per cui so che quel DNS avrà la risposta pronta. Quindi mando la richiesta e ho la risposta. I server DNS si comporteranno come una batteria di attacchi di rilancio.

Di quanti attaccanti ho bisogno? Di uno solo, perché invierà a tutti questi server query di tipo DNS impersonificando (attacco di mascheramento) il client.

Il client è la nostra vittima, il traffico che gli arriverà sarà impegnativo e si troverà a dover gestire tutta una serie di risposte che sono del tutto legittime.

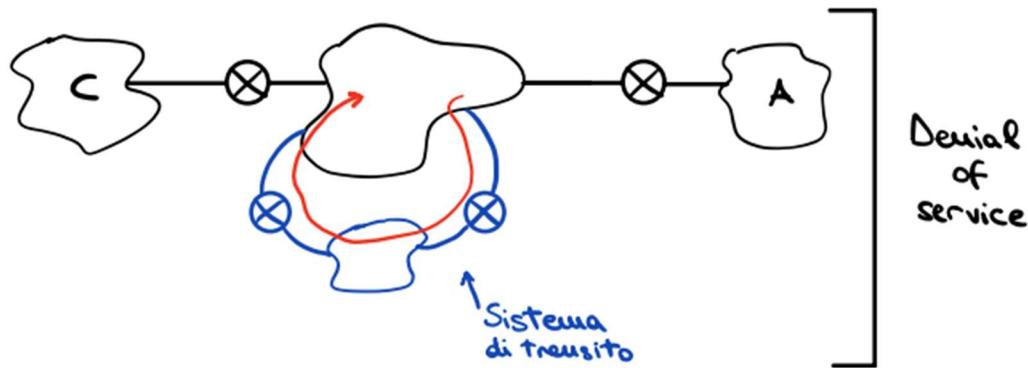
Come si blocca un attacco di questo tipo?

Supponendo che in mezzo ci sia un sistema di protezione, il firewall (FW), dovremo implementare una codifica che impedisca a delle risposte a cui non corrispondono precedenti richieste di attraversare il nostro muro. Questa è una prima cosa.

L'altra cosa che si può fare è prevenire che la propria rete sia utilizzata da un attaccante per fare operazioni di questo tipo. Si può limitare il numero delle reti che possono partecipare in maniera inconsapevole a un attacco di questo tipo.

Sarebbe opportuno configurare anche il firewall presente verso il client con opportune codifiche di ingresso, ovvero se il traffico di uscita ha un indirizzo ip che non è compatibile con le reti interne, il traffico non va inoltrato perché significa che c'è qualcosa che non va. Il messaggio viene scartato in maniera silente senza generare messaggi ICMP altrimenti l'obiettivo è proprio quello di inondare la rete con messaggi ICMP.

Queste due politiche si applicano solo a reti terminali. Esempio:



L'operazione da fare sul client vale per tutti quei sistemi che offrono dei servizi, perché se un sistema non offre dei servizi ci sarà già un firewall che permette il transito di pacchetti solo da dentro verso fuori, e non da fuori verso dentro.

Su un sistema di transito le cose sono più complicate: in questo caso la rete è configurata per trasportare traffico, e quindi è complesso decidere se del traffico è legittimo o meno.

Questo invece è un altro classico attacco di mascheramento o di man in the middle. L'attaccante cerca di impersonificare il client verso il server e di impersonare il server verso il client: in questo modo impedisce che ci sia un colloquio diretto tra i due e può aggiungere, cancellare o modificare il traffico, tutto questo come attacco di data modification (c). questi sono tanto più efficaci se non c'è il canale diretto, cioè se il canale originale viene bloccato e si obbliga il traffico a passare attraverso una terza parte. Contromisure: sono tutte quelle atte a rilevare la modifica dei dati e quindi tutte le tecniche che rientrano sotto l'obiettivo di data integrity o message integrity. Applico un hash crittografico per garantire l'integrità del messaggio. Prendo un pacchetto, calcolo l'hash, che è una funzione unidirezionale che per input di dimensioni variabili da sempre un output di dimensione fissa: il digest. Poi il digest lo cifro con la chiave privata del mittente, in modo tale che a destinazione le operazioni che vanno fatte per capire se il pacchetto è stato modificato sono le seguenti. Prendo il pacchetto in chiaro, prendo il digest, conosco tutti gli algoritmi, calcolo il digest. Quindi prendo il contenuto del pacchetto, lo passo alla funzione di hash e genero l'hash. Poi prendo l'hash crittografico cifrato con la chiave privata e lo decifro con la chiave pubblica del mittente. E controllo

se l'hash che ho ottenuto dal payload del pacchetto è uguale all'hash crittografico. Se sono uguali significa che nessun bit è stato modificato.

La soluzione tipica di questo problema del man in the middle è utilizzare una terza parte che si chiama entità o autorità di certificazione, che non fa altro che andare a firmare la chiave pubblica del client. Questo perché l'attaccante può usare la sua chiave pubblica e andare a modificare i dati. Se invece il client e il server hanno una chiave pubblica firmata da un'autorità di certificazione, di cui è nota all'interno dei calcolatori già dentro al sistema operativo la loro chiave pubblica, allora in quel caso, se ci si fida dell'autorità di certificazione automaticamente ci si fida anche di quelli di cui l'autorità di certificazione fa da garante.

La strategia per evitare la modifica dei dati è quella di utilizzare degli hash crittografici e delle firme digitali. Questo vale sia per il traffico sia per i dati.

(Esempio: Certificato X.509, è la chiave pubblica firmata)

Nel caso di data modification le cose cambiano a seconda del tipo di protocollo che andate ad utilizzare e si possono avere garanzie più o meno forti. Se si ha un protocollo basato sul concetto di sessione è possibile andare a proteggere l'intera sessione, oppure i dati, oppure il singolo messaggio. Se invece avete un protocollo a datagramma, cioè udp, è possibile solo inserire i meccanismi di data integrity direttamente nell'applicazione e quindi si può proteggere solo il singolo messaggio, niente di più.

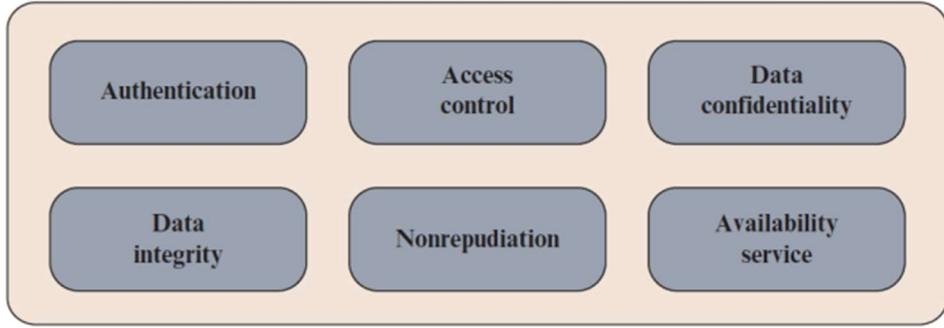
## Servizi di sicurezza

Per proteggermi dagli attacchi devo mettere in piedi un certo numero di servizi di sicurezza.

Un servizio di sicurezza cerca di soddisfare i requisiti di uno o più dei 5 obiettivi della sicurezza: confidenzialità, integrità, disponibilità, autenticità e accountability (CIA + AA).

I servizi di sicurezza (che sono quelli nel disegno: servizio di autenticazione, di controllo dell'accesso, confidenzialità dei dati, integrità dei dati, non ripudiabilità e servizi di disponibilità):

- implementano delle politiche di sicurezza, il cui compito è quello di soddisfare i requisiti dei nostri obiettivi;
- sono implementati da meccanismi di sicurezza.



Questi servizi, combinati, di solito non vanno mai da soli, hanno l'obiettivo di soddisfare i requisiti degli obiettivi di sicurezza.

#### **NUOVA SLIDE: 43 A FINAL RECAP (NON ANCORA CARICATA)**

Gli obiettivi di cybersecurity principali (confidenzialità, integrità, disponibilità, autenticità e accountability) danno luogo a un certo numero di requisiti e definiscono anche cosa si intende per perdite di sicurezza, ovvero quando i requisiti non sono soddisfatti.

Questi obiettivi li implementiamo attraverso le politiche di sicurezza. Le politiche di sicurezza vengono implementate attraverso i security service, che a loro volta sfruttano un certo numero di meccanismi di sicurezza.

Nell'implementazione dobbiamo fronteggiare un certo numero di sfide: ne abbiamo viste 10 ma possiamo classificarle sotto queste tre macrocategorie:

- gli attacchi (attivi o passivi)
- i criteri di usabilità (e in generale di disponibilità degli utenti ad accettare meccanismi di sicurezza). Questo perché sappiamo che in alcuni casi sono gli utenti che mal digeriscono delle politiche di sicurezza restrittive, in altri casi sono gli stessi operatori di rete che fino a quando non subiscono un danno significativo tendono a sottostimare l'importanza della sicurezza all'interno dei propri servizi o delle proprie reti oppure sono essi stessi che lo ritengono come ostacolo all'utilizzabilità dei servizi stessi
- motivi di tipo economici, CAPEX e OPEX (CAPEX: capital expenditure, OPEX: operational expenditure).

#### **Approfondimento su CAPEX e OPEX**

I CAPEX sono gli investimenti, cioè i soldi che si investono per dotare l'azienda di un nuovo prodotto o servizio. L'OPEX sono i soldi che bisogna investire per mantenerlo funzionante. Nel CAPEX, che di solito si fa una volta sola, ci sono tutti quei costi sono richiesti per mettere in opera il servizio, mentre nell'OPEX per mantenerlo funzionante. Per mettere in opera il servizio bisogna ad esempio acquistare i dispositivi e/o fare training al personale o assumere personale che li sappia usare. Oppure utilizzare

soluzioni open source o proprietarie sui dispositivi che si hanno già e formare delle persone già vostre per saperle utilizzare. Questa formazione può avere un costo esplicito (si fa un corso da qualche parte) o implicito (parte delle ore del lavoro di quelle persone saranno demandate alla formazione).

L'OPEX è quanto mi costa mantenere in piedi il servizio, cioè la formazione permanente del personale (corsi di aggiornamento che possono essere fisici, online) ed eventuali aggiornamenti del software.

Sono concetti generalissimi che valgono per qualsiasi sistema o servizio, ma sono concetti importanti: ogni volta che viene fatto un investimento bisogna capire quanto costa all'inizio e quanto costa mantenerlo. Soprattutto quello che in alcuni casi viene sottostimato è l'OPEX.

Anche per quanto riguarda la sicurezza è importante il CAPEX ma è importante soprattutto l'OPEX. Se metto in piedi un sistema di sicurezza e poi non lo aggiorno più, il sistema di sicurezza diventerà obsoleto e sarà come non averlo, o anzi offrirà delle vulnerabilità note a potenziali attaccanti, proprio perché non mi sono preoccupato di andare ad aggiornarlo quando venivano rilasciati gli aggiornamenti che andavano a riparare le falte su vulnerabilità note che precedentemente non erano ancora conosciute.

## Autenticazione e controllo degli accessi

I servizi di sicurezza di autenticazione e controllo degli accessi spesso vanno insieme.

L'autenticazione permette di accedere a un servizio solo dopo che ci conosce l'identità di chi vuole accedere. Permette quindi la comunicazione solo quando l'identità è nota, dove l'identità può essere sia di un utente umano, sia di una macchina. Questo implica che i due comunicanti devono utilizzare un protocollo comune per comunicare.

Il servizio di autenticazione tipicamente serve a evitare il problema del man in the middle, oppure dell'intervento di una terza parte. Quindi se ho un servizio di autenticazione, che mi permette di capire qual è l'identità del mio client, dovrei riuscire a prevenire gli attacchi di mascheramento.

**Il servizio di autenticazione** si occupa di garantire che una comunicazione sia autentica.

- Autenticazione di mittente/entità: garantire l'identità dei pari in una comunicazione (i peer utilizzano lo stesso protocollo per comunicare)
- Nessuna interferenza da parte di terzi mascherati

**Il controllo degli accessi** è la capacità di limitare e controllare l'accesso ai sistemi host e alle applicazioni tramite la rete. Potrebbe richiedere un'autenticazione preventiva.

Il controllo di accesso serve a limitare i diritti di accesso a un determinato sistema o servizio.

Di solito va con l'autenticazione, ma non è detto.

Bisogna discriminare tra un controllo dell'acceso interno o esterno. Se controllo l'accesso della rete dall'esterno verso l'interno utilizzerò tecniche di autenticazione e poi tecniche di controllo di accesso. Quindi alcuni utenti avranno accesso a determinati servizi, altri utenti avranno accesso ad altri servizi (diversi dai precedenti), oppure posso utilizzare delle tecniche che non prevedono il controllo di accesso, ad esempio le VLAN. In questo caso, se sono all'interno della mia rete posso ad esempio impedire che i dipendenti dell'area amministrativa accedono ai server dove vengono caricati i progetti in via di sviluppo. Quindi effettuo il controllo dell'accesso mettendo delle limitazioni all'interno della rete, quindi "segregando" il traffico.

## Confidenzialità dei dati

La confidenzialità dei dati è la protezione dei dati trasmessi da attacchi passivi (cioè l'ascolto per decodificare i contenuti).

Sono disponibili diversi livelli di protezione:

- Tutti i dati entrano/escono da un host
- Protezione della singola sessione
- Protezione del messaggio singolo

È differente a seconda dei protocolli che vengono utilizzati. Se utilizzo protocolli orientati alla sessione (tcp) posso proteggere il socket aperto dal tcp a livello di sessione, oppure a livello di messaggio (questo posso farlo anche con udp, ma inserendo la confidenzialità dei dati dentro l'applicazione), oppure posso proteggere tutti i dati che entrano ed escono da un sistema.

Protezione dall'analisi del traffico: sposto la cifratura, quindi mi proteggo tramite l'offuscamento. Devo cifrare il più possibile all'interno del traffico. Se riesco a cifrare non solo lo strato applicativo, ma anche lo strato 3 e 4, un utente malintenzionato non sarà in grado di osservare gli indirizzi ip mittente e destinazione, le porte, e quindi complico parecchio il lavoro dell'attaccante.

Per fare questo devo fare la cifratura a strato 2, cosa che ad esempio viene sempre fatta nelle reti wireless, che però hanno altri problemi.

Un'altra vulnerabilità tipica del traffico che viene esposta attraverso le tecniche di analisi del traffico è quella di andare a stimare la dimensione dei messaggi e la loro frequenza: potrei fare anche questo andando ad alterare la dimensione dei messaggi. Questo di solito si fa con un meccanismo di sicurezza chiamato padding, che inserisce byte inutili alla fine del messaggio che servono a variare in maniera artificiale il numero dei byte all'interno dei pacchetti. Oppure potrei anche inviare dei pacchetti vuoti, solo per mascherare l'attività dei client.

Se non riesco a risalire all'identità di chi sta scambiando le informazioni perché sto cifrando anche lo strato 3 e lo strato 4, o con tecniche di cifratura allo strato 2 per le reti wireless, o con tecniche di vpn allo strato 3, e poi vado ad alterare la dimensione dei pacchetti o inserisco pacchetti artificiali che non hanno informazione, allora vado a complicare in maniera significativa il compito di chi va ad effettuare traffic analysis, andando ad offuscare le statistiche del traffico.

Questo non crea problemi a chi effettua monitoring legittimo del traffico, che ha accesso al pacchetto originale e quindi sarà in grado di discriminare qual è il traffico corretto e quale invece serve solo come offuscamento.

## **Integrità dei dati**

Si utilizzano tipicamente meccanismi a chiave asimmetrica (chiave pubblica e chiave privata). Esistono anche meccanismi chiamati keyless, cioè senza chiave, basati sull'utilizzo di numeri pseudo casuali, ma sono meno utilizzati.

Nel caso di protocolli orientati alla sessione riesco a fornire protezione non solo sulla modifica dei dati ma anche ad esempio sulla loro cancellazione o sulla loro duplicazione, proprio perché utilizzo i numeri di sequenza e quindi mi accorgo se un dato mancava all'interno della sessione oppure se un dato è stato duplicato.

Se invece ho un meccanismo non orientato alla sessione, quindi orientato esclusivamente al datagramma, posso solo andare a stimare se il messaggio che ho ricevuto è integro oppure no, senza nessun legame con i messaggi precedenti.

L'integrità può applicarsi a un flusso di messaggi, a un singolo messaggio, a campi selezionati all'interno di un messaggio.

Un servizio di integrità orientato alla connessione garantisce che i messaggi vengano ricevuti così come inviati senza duplicazioni, inserimenti, modifiche, riordini o ripetizioni.

Questo servizio copre anche la distruzione dei dati.

Il servizio di integrità orientato alla connessione affronta sia la modifica del flusso di messaggi che la negazione del servizio.

Un servizio di integrità senza connessione generalmente fornisce protezione solo contro la modifica dei messaggi.

Tutti i servizi di integrità possono prevedere il recupero oppure no.

## **Non ripudiabilità e disponibilità**

La **non ripudiabilità** impedisce al mittente o al destinatario di negare un messaggio trasmesso o ricevuto.

Come si attua questo servizio? Con il meccanismo della notarizzazione, quindi con una terza parte trusted, fidata, a cui sia il mittente che il destinatario si rivolgono per essere sicuri che il destinatario non dica di non aver ricevuto il messaggio e il mittente non dica di non averlo spedito.

Il notaio, che in genere fa capo alla presenza di un'autorità di certificazione, fa sì che se mi è arrivato un messaggio firmato con il certificato del client, il client non può dire che non l'ha inviato, e così per il server.

La **disponibilità** è la proprietà di un sistema di essere accessibile e utilizzabile su richiesta da un'entità di sistema autorizzata.

Come si realizza questo servizio? Con l'autenticazione e controllo di accesso. Se limito l'accesso a un determinato numero di utenti, tipicamente rendo un servizio disponibile.

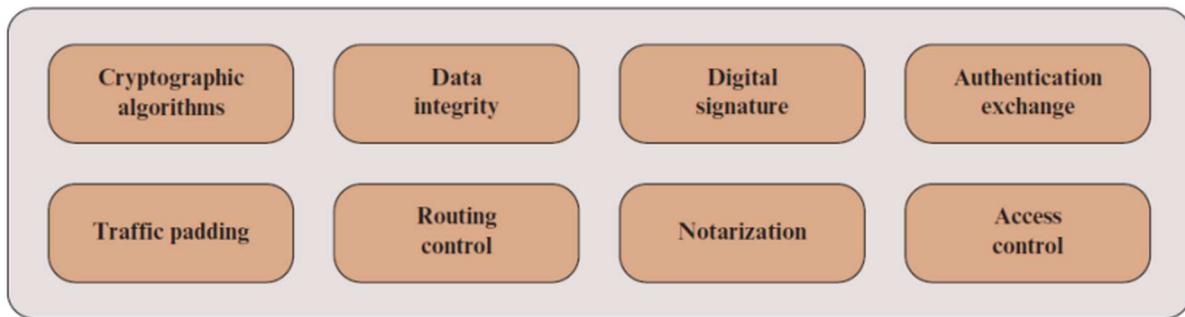
Posso farlo sia per gestire sovraccarichi del traffico, sia per gestire attacchi informatici.

Problema tipico che si verifica in alcuni eventi imprevisti: ad esempio quando c'è stato l'attentato alle torri gemelle sono andati giù tutti i server DNS che stavano nei seminterrati e quindi il traffico internet è stato rallentato perché tutto il traffico gestito da quei server è stato dirottato su altri server, quindi, c'è stato un calo delle prestazioni. In quel caso c'è stato un problema di indisponibilità.

Quindi in alcuni casi la potenza di disponibilità può dipendere da una cattiva progettazione, in altri da un evento inaspettato e in altri da attacchi di cybersecurity.

Altri richiedono un'azione fisica per prevenire/ripristinare la perdita di disponibilità di elementi di un sistema distribuito.

## Mecanismi di sicurezza



I meccanismi crittografici sono alla base del servizio di confidenzialità.

I meccanismi che assicurano l'integrità dei dati sono alla base del servizio di data integrity, insieme a quelli di firma digitale.

I protocolli di autenticazione che sono alla base del servizio di autenticazione.

Il padding che è alla base del servizio di prevenzione e analisi del traffico.

I meccanismi di routing control e di access control che possono permettere di mitigare problemi di disponibilità. Ad esempio, se ho una piattaforma che sta andando in sovraccarico mettere in piedi un meccanismo che permetta di replicare il servizio da un'altra parte e di dirottare efficacemente il servizio da un'altra parte è un meccanismo efficace per mitigare problemi di disponibilità.

La notarizzazione è il meccanismo abilitante dei servizi di non ripudiabilità.

## Tipi di sicurezza di rete

La cybersecurity la possiamo classificare sotto due grandi topologie: la sicurezza dell'informazione e la sicurezza della rete.

A sua volta la sicurezza della rete possiamo classificarla come la sicurezza della comunicazione, che tipicamente implemento attraverso la combinazione di protocolli e tecniche crittografiche, e la sicurezza dei dispositivi. Sono tutte tecniche che ricadono nella protezione dei dispositivi, o dei sistemi terminali, o delle reti che permettono a quei sistemi di accedere a Internet.

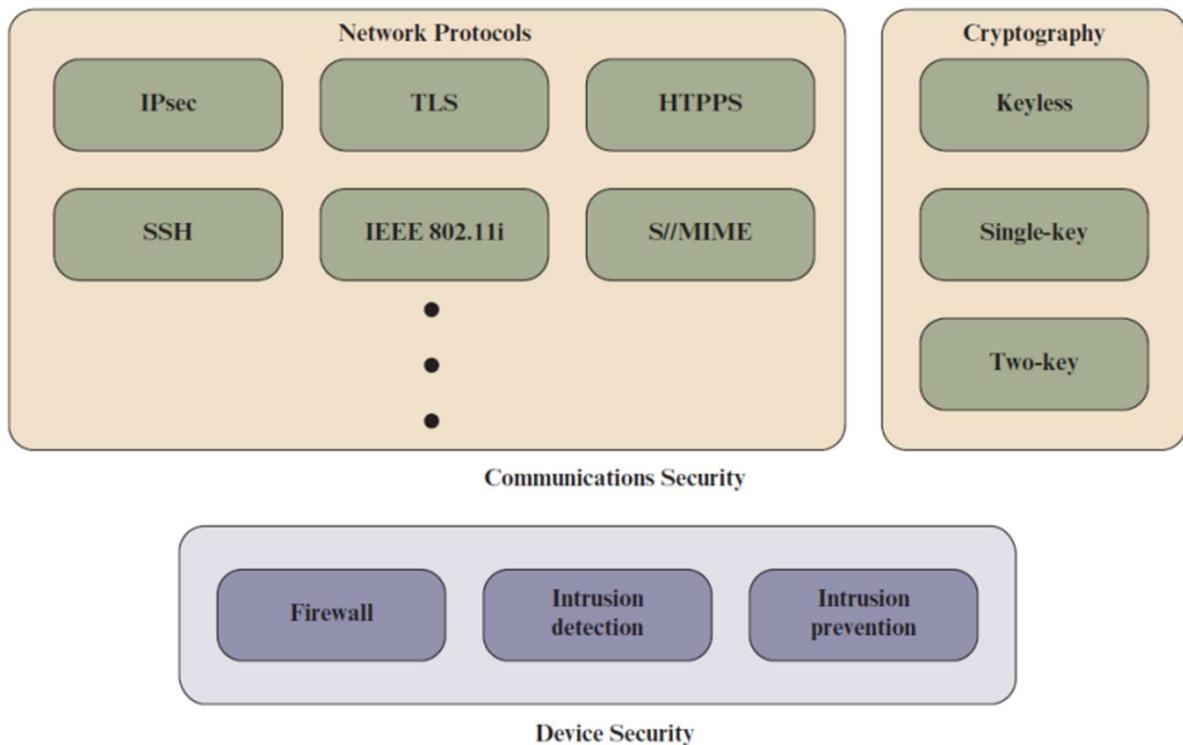
- Sicurezza delle comunicazioni: viene implementata con l'uso combinato di un certo numero di protocolli, a vari livelli protocollari, dipende dal tipo di sicurezza che voglio implementare, che andranno combinati con tecniche di sicurezza, che possono essere sicurezza simmetrica, asimmetrica o key-less.
- Sicurezza dei dispositivi (o di insiemi di dispositivi): protezione dei dispositivi di rete e dei sistemi terminali collegati alla rete. Avremo meccanismi che

regolano l'accesso (firewall), meccanismi che rilevano le intrusioni e meccanismi che prevengono le intrusioni.

Un meccanismo di prevenzione delle intrusioni è configurato per rilevare e bloccare un certo numero di intrusioni note.

Un meccanismo per la rilevazione delle intrusioni è un meccanismo che può essere configurato per rilevare intrusioni note o anche per rilevare attacchi o-day.

Un IPS tipicamente non può essere utilizzato per gli attacchi o-day.



## Sicurezza nello stack del protocollo IP

HTTP	FTP	SMTP
TCP		
IP/IPSec		

Network level

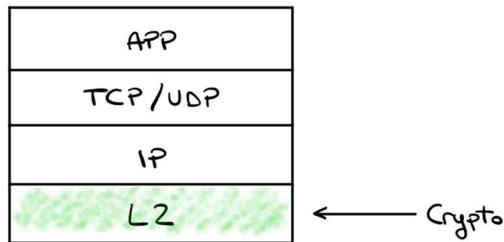
HTTP	FTP	SMTP
SSL or TLS		
TCP		
IP		

Transport level

	S/MIME	
Kerberos	SMTP	HTTP
UDP		TCP
IP		

Application level

I protocolli li posso applicare allo strato 3: IPSec, tra lo strato 4 e lo strato applicativo, quindi a livello di socket, oppure allo strato applicativo: riesco a proteggere anche il protocollo UDP, oppure a livello di strato 2.



Se li applico allo strato 2 hanno effetto non da estremo a estremo, no su grandi porzioni di rete, ma solo sul collegamento di questa tecnologia specifica. Quando cambio collegamento devo ripetere tutta l'operazione.

Le tecniche a strato 3 (tecniche tipicamente di VPN) avranno effetto solo su porzioni della rete, quelle coperte dal servizio VPN, quindi punto di ingresso e punto di uscita della VPN. Sono tecniche sopra TCP e quindi avranno effetto da estremo a estremo, quindi da client a server o da client a proxy (se ho una middlebox).

Application level sono veramente da estremo a estremo.

Nel mondo cloud esistono poi altri protocolli che servono per interconnettere i sistemi terminali che emulano alcune di queste tecnologie.

## Tipologie di minacce a livello applicativo

	<b>Threats</b>	<b>Consequences</b>	<b>Countermeasures</b>
<b>Integrity</b>	<ul style="list-style-type: none"> <li>• Modification of user data</li> <li>• Trojan horse browser</li> <li>• Modification of memory</li> <li>• Modification of message traffic in transit</li> </ul>	<ul style="list-style-type: none"> <li>• Loss of information</li> <li>• Compromise of machine</li> <li>• Vulnerability to all other threats</li> </ul>	Cryptographic checksums
<b>Confidentiality</b>	<ul style="list-style-type: none"> <li>• Eavesdropping on the net</li> <li>• Theft of info from server</li> <li>• Theft of data from client</li> <li>• Info about network configuration</li> <li>• Info about which client talks to server</li> </ul>	<ul style="list-style-type: none"> <li>• Loss of information</li> <li>• Loss of privacy</li> </ul>	Encryption, Web proxies
<b>Denial of Service</b>	<ul style="list-style-type: none"> <li>• Killing of user threads</li> <li>• Flooding machine with bogus requests</li> <li>• Filling up disk or memory</li> <li>• Isolating machine by DNS attacks</li> </ul>	<ul style="list-style-type: none"> <li>• Disruptive</li> <li>• Annoying</li> <li>• Prevent user from getting work done</li> </ul>	Difficult to prevent
<b>Authentication</b>	<ul style="list-style-type: none"> <li>• Impersonation of legitimate users</li> <li>• Data forgery</li> </ul>	<ul style="list-style-type: none"> <li>• Misrepresentation of user</li> <li>• Belief that false information is valid</li> </ul>	Cryptographic techniques

#### NUOVA SLIDE: 44 (NON ANCORA CARICATA)

È il riassunto dell'implementazione delle politiche di sicurezza.

Nel caso della sicurezza delle reti abbiamo la sicurezza della comunicazione, che la realizzo tramite l'uso congiunto di protocolli e tecniche crittografiche, nel caso della sicurezza dei dispositivi o di insiemi di dispositivi avrò dispositivi di controllo e dispositivi di rilevazione e prevenzione delle minacce.

## Firewall e IDS

24/10

### Firewall

Il firewall è un dispositivo inteso a proteggere una rete o una porzione di rete, che fornisce uno strato di difesa, isolando i sistemi interni (intera rete o una porzione di una rete) dalle reti esterne (internet) o da altre parti della rete interna (esempio: se ho una rete aziendale).

In genere, viene inserito un firewall tra la rete della sede e Internet per stabilire un collegamento controllato ed erigere un muro o perimetro di sicurezza esterno. Per proteggere la rete interna dagli attacchi basati su Internet per fornire un unico punto di strozzatura in cui imporre sicurezza e controllo.

I firewall vengono implementati anche all'interno della rete aziendale per separare parti della rete.

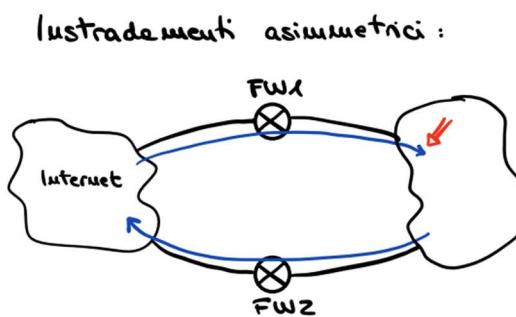
Un firewall serve a proteggere da un certo numero di attacchi, tipicamente esterni ma anche potenzialmente interni, intesi come collaborazione con altri attacchi.

Il firewall è inteso come un singolo punto di controllo, dove posso implementare politiche di sicurezza e di accountability, tracciamento.

La presenza del firewall mi permette di garantire fino a un certo livello servizi di sicurezza e nel momento in cui quei servizi di sicurezza sono bypassati, mi permette di garantire un certo livello di tracciamento delle operazioni, che mi permetterà di andare a capire dove è il problema e come può essere risolto per evitare intrusioni future.

Obiettivi di progetto del firewall:

1. Tutto il traffico dall'interno verso l'esterno e viceversa deve passare attraverso il firewall, tipicamente evitando instradamenti asimmetrici.



Supponiamo che la nostra rete abbia due router che agiscano anche da firewall e che garantiscono l'accesso a internet. Se ho delle richieste che entrano dal FW1 e escono dal FW2 mettono in crisi i firewall, perché non riusciranno a

mappare le richieste con le risposte, e potrebbero identificare del traffico legittimo come traffico malevolo, in particolare quello della freccia rossa. Perché che arrivi al fw una risposta senza una precedente richiesta potrebbe dar luogo all'implementazione di una politica di sicurezza che blocca tutto il traffico di risposta per cui non ho osservato una richiesta.

È fondamentale il fatto che il fw sia un unico punto di controllo (dal punto di vista logico); se ci sono porzioni di rete diverse devono essere servite da fw diversi. È importante che quando una richiesta passa attraverso un'istanza del fw anche la risposta passi attraverso la stessa istanza del fw.

Se violiamo questa regola il firewall diventa inefficace.

Ciò si ottiene bloccando fisicamente (dal punto di vista telematico) tutti gli accessi alla rete locale che non passino tramite firewall.

2. Sarà consentito il transito solo al traffico autorizzato, come definito dalla politica di sicurezza locale.

Vengono utilizzati vari tipi di firewall, che implementano vari tipi di politiche di sicurezza.

3. Il firewall stesso dovrebbe essere immune alla penetrazione.

Per questo dovrebbe essere realizzato da un sistema “sicuro” e aggiornato costantemente; visto che il fw è il punto di accesso primario dei sistemi della rete, non deve essere il fw il punto di debolezza della rete stessa.

Ciò implica l'uso di un sistema rafforzato con un sistema operativo protetto.

I sistemi informatici affidabili sono adatti per ospitare un firewall e sono spesso richiesti nelle applicazioni governative.

## Tecniche firewall

Il modo in cui realizzo gli obiettivi di sicurezza è attraverso le politiche di sicurezza, che sono implementate tramite i servizi di sicurezza, e il fw è uno di questi meccanismi abilitanti delle politiche di sicurezza.

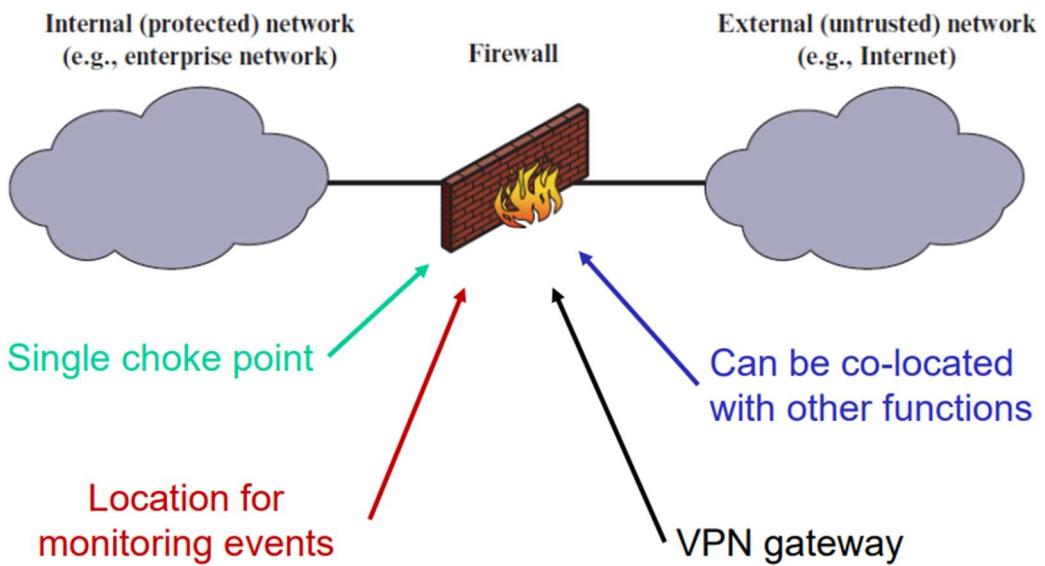
È possibile utilizzare quattro tecniche per controllare l'accesso e applicare la politica di sicurezza del sito.

- **Controllo dei servizi:** è possibile accedere ai tipi di servizi Internet, in entrata e in uscita; quali servizi possono essere acceduti.
- **Controllo della direzione di accesso:** direzione in cui può essere avviata una determinata richiesta di servizio. Posso accedere a quali servizi di internet? Utenti di internet posso accedere a servizi interni? Si, no, combinazioni varie.

- **Controllo utente:** quali utenti possono accedere a quali risorse. Spesso applicato agli utenti interni, per gli utenti esterni è richiesta l'autenticazione.
- **Controllo del comportamento:** come viene utilizzato un servizio. Ad esempio, all'interno di un servizio abilitato a determinati utenti potrei decidere di restringere l'accesso ad alcuni contenuti di un servizio abilitato oppure potrei limitare la rate di accesso o la quantità di dati che possono essere scambiati. Posso forzare delle politiche che modellino il comportamento degli utenti. Antispam o accesso limitato ad alcuni contenuti.

È fondamentale nei fw l'accountability, la possibilità del tracciamento. Abbiamo visto sulla prima parte del corso che uno dei meccanismi principali per effettuare il monitoraggio degli utenti che accedono a un servizio è l'uso dei log del servizio. Il fw tipicamente genererà dei log e il salvataggio dei log nella modalità che l'amministratore di rete ritiene più opportuna e l'accesso ai log e la possibilità di analisi del log sarà uno strumento di fondamentale importanza per realizzare alcune delle politiche di sicurezza. Ad esempio per controllare se le politiche di sicurezza messe in atto funzionano bene oppure se alcune di queste politiche non sono pienamente realizzate.

## Modello generale di FW



Il modello del fw è quello di un unico punto di controllo (single choke point) tra una rete e il resto del mondo, che può essere la porzione più esterna della rete che si vuole proteggere o l'intera internet, dal punto di vista concettuale non cambia niente.

Il fw è anche il luogo ideale per monitorare gli eventi, quali l'attività dei propri utenti, sia per monitorare altri tipi di eventi, ad esempio il consumo di risorse del fw stesso.

Proprio perché il fw è il singolo punto di passaggio, bisogna evitare che il fw diventi, dal punto di vista prestazionale, il collo di bottiglia del sistema.

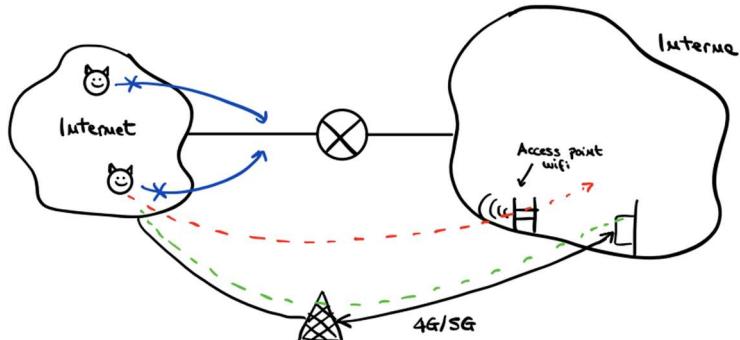
Il fw è tipicamente il punto della rete ideale per realizzare un gateway VPN (Virtual Private Network), permettendo così a utenti esterni di accedere alla rete come se fossero all'interno della rete. Gli utenti esterni possono accedere in qualità di utenti esterni a servizi che sono messi in opera sulla rete interna, quini attraverso il fw, oppure possono accedere come se fossero interni attraverso la VPN. La differenza è sostanziale perché per accedere tramite una vpn gli utenti sono identificati e tracciati in maniera più fine rispetto a un generico utente che accede via internet.

Il fw può essere il punto in cui vengono messe in opera altre funzioni di rete, e questo ha di nuovo a che fare con il problema del dimensionamento delle prestazioni. Se la rete utilizza sia indirizzi pubblici che indirizzi privati ad esempio il fw è il luogo ideale per realizzare servizi di natting. Se la rete utilizza indirizzi dinamici il fw potrebbe essere il punto ideale dove mettere in opera servizi di DHCP, e così via.

- ➔ Ci sono delle funzioni che potrebbero essere collocate nel fw e queste potrebbero, in alcuni casi, appesantirne il funzionamento e quindi limitare un po' le prestazioni.

## Limitazioni del firewall

1. Il firewall non può proteggere dagli attacchi che bypassano per il firewall, quindi da accessi che passano per canali secondari e che non passano per il fw.



Consideriamo questo schema. Se il fw è realizzato bene e aggiornato, i tentativi di accesso da parte degli attaccanti internet vengono bloccati (frecce blu).

Se all'interno della rete ho un access point wifi in cui la chiave è semplice (tipo "prova") è ragionevole che uno di questi attaccanti riesca ad entrare nella rete attraverso una connessione wifi mal protetta (freccia rossa).

Se invece abbiamo un utente che deve accedere ad un determinato servizio e il fw è configurato per impedire l'accesso a quel servizio, l'utente invece che

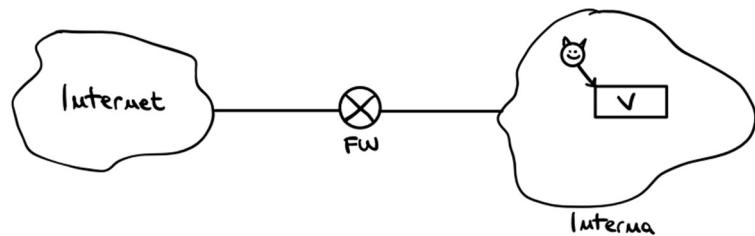
effettuare una richiesta per farsi aprire la porta del fw accende il cellulare e passa tramite una connessione 4G/5G e accede a quel servizio, creando un accesso secondario alla rete.

La differenza rispetto a prima è che un access point mal configurato è facile da rilevare perché è un dispositivo fisico, ma un dispositivo ad uso intermittente come un cellulare crea un punto di accesso secondario potenzialmente a intrusi che possono usare questo accesso per fare danni o per utilizzare questa rete per partecipare ad un attacco (spesso succede).

I sistemi interni possono avere funzionalità di chiamata in uscita per connettersi a un ISP.

È possibile che dall'esterno dell'organizzazione sia possibile accedere a una LAN wireless protetta in modo improprio.

2. Il firewall potrebbe non proteggere completamente dalle minacce interne, che siano volontarie o involontarie.



Se la vittima (V) e l'attaccante stanno all'interno della rete, l'attaccante attacca la vittima e questa non può essere protetta dal fw perché sono entrambi dietro il fw.

Questo attacco può essere consapevole o inconsapevole.

È inconsapevole quando si utilizza la pratica del BYOD (Bring Your Own Device), molto diffusa negli ultimi anni: sull'attività lavorativa si utilizzano i propri dispositivi.

Un dispositivo può essere infettato all'esterno della rete aziendale e quindi connesso e utilizzato internamente. È anche difficile poi tracciare come si è diffuso il contagio all'interno dell'azienda.

Questo può essere prevenuto dalle aziende impedendo l'uso di dispositivi non autorizzati all'interno dell'azienda.

L'attacco è consapevole quando qualcuno volontariamente infetta un dispositivo interno per propagare un malware o per realizzare un attacco.

## **FW basato su regole o basato su policy**

Possiamo avere fw con comportamento statico o dinamico. I fw con il comportamento statico sono più semplici e più diffusi e sono i fw basati su delle regole. I fw basati sulle policy sono più dinamici.

**Firewall basati su regole:** un fw permette tutto quello che le regole gli permettono di fare; quindi, permette l'accesso ai servizi esplicitamente permessi dalle regole e blocca tutti quelli bloccati dalle regole.

I sistemi firewall basati su regole utilizzano regole per controllare la comunicazione tra host all'interno e all'esterno del firewall.

I sistemi basati su regole sono statici: non possono fare nulla per cui non sono stati espressamente configurati. Deve esserci una riga in uno dei loro file di configurazione da qualche parte che dice loro esattamente cosa fare con ogni pacchetto che scorre attraverso il dispositivo.

(iptables è il classico esempio di firewall rule-based: è organizzato in tabelle, ogni tabella ha una catena e ogni catena ha una regola.)

La configurazione statica può essere aggiornata ma non si aggiorna da sola.

Ciò rende il sistema più semplice da configurare, ma meno flessibile e meno adattivo alle mutevoli circostanze.

**Firewall basati sulle policy:** sono più dinamici perché le policy (che possono essere un concetto di livello più elevato, più astratto) definiscono delle condizioni più generali sotto cui può essere permesso l'accesso ai servizi.

I sistemi basati sulle policy sono più flessibili.

Permettono all'amministratore di definire le condizioni alle quali sono consentiti tipi generali di comunicazione, di specificare quali funzioni e servizi verranno eseguiti per fornire tale comunicazione.

Un sistema basato su policy può impostare dinamicamente la comunicazione consentita verso indirizzi IP casuali.

Un esempio di fw basato sulle policy è che tutti gli utenti che sono stati autenticati accedono a internet o che tutti gli utenti che arrivano tramite VPN possono accedere ai servizi.

In generale tutto quello che riguarda politiche basate su VPN e autenticazione possono essere realizzate tramite fw basati sulle policy.

Qualsiasi sistema che supporti l'intestazione di autenticazione IPsec e l'incapsulamento del payload di sicurezza è considerato un sistema basato su policy.

→ Nella realtà abbiamo sempre un mix dei due fw, perché tipicamente abbiamo che anche i fw basati sulle policy, una volta che la policy è stata definita, utilizzano poi delle regole.

Esempio: tutti gli utenti autenticati nel gruppo a hanno accesso a questi servizi tramite queste regole.

Quindi possiamo dire che il livello di astrazione più elevato è realizzato con le policy e il livello di astrazione meno elevato e più stabile è realizzato con le regole.

FINE 24/10

**Recap:** Abbiamo visto che i firewall Rule-based sono di tipo statico mentre quelli Policy-based sono tipicamente dinamici, perché utilizzano le policy che passano per un processo di autenticazione, quindi, permettono una maggiore dinamicità nell'esecuzione nel ruolo di difesa.

## Tipi di Firewall

Un'altra classificazione dei Firewall è quella più classica in assoluto, ovvero, il firewall può essere una difesa che agisce come un filtro positivo o come un filtro negativo. Quindi, si va a definire qual è la regola di default e poi i filtri su cui si applica questa regola, ovvero le **eccezioni**.

- **Firewall Positivo:** un firewall che agisce come un filtro positivo ha una regola di default che blocca tutto e impedisce a tutto il traffico di passare. Quindi le regole o le policy stabilite dal firewall rappresentano le eccezioni rispetto al comportamento di default, ovvero specificano il tipo di traffico che può passare o il tipo di utente che può accedere.
- **Firewall Negativo:** un firewall che agisce come un filtro negativo ha un comportamento opposto a quello positivo e quindi la policy o regola di default è di accettare tutto il traffico. Si va a specificare solo il traffico che non può passare.

La prima soluzione è più restrittiva perché chiude tutto e apro solo al bisogno, mentre l'altro caso è più permissivo (la prima soluzione è più utilizzata).

## Cosa può ispezionare un firewall?

Un firewall può ispezionare le intestazioni dei pacchetti allo strato due, tre, quattro o applicativo a seconda del vantage point in cui è posizionato. Un'altra cosa che si può ispezionare sono i pattern, cioè l'insieme di pacchetti all'interno di un flusso. Quindi potrebbero esserci delle configurazioni all'interno di un flusso di pacchetti che possono rappresentare un tipico segnale di malfunzionamento, anomalia o attacco informatico.

Ci sono quattro tipi di firewall diversi:

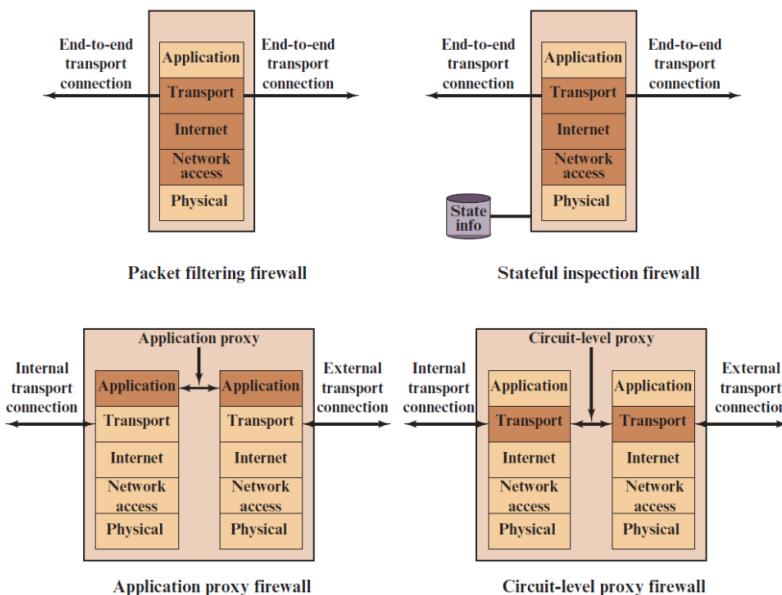
- Il primo tipo è il firewall che effettua il **Packet Filtering**. In laboratorio abbiamo utilizzato netfilter configurato con iptables per fare delle regole piuttosto semplici e abbiamo visto che in quel caso noi abbiamo specificato delle regole che riguardano dei campi dei pacchetti. Quello è un esempio classico di firewall basato sul filtraggio dei pacchetti;
- Il secondo tipo sono gli **Stateful Inspection Firewall**. In questo caso non si guardano i singoli pacchetti ma le connessioni (caso TCP) o i flussi (caso UDP);
- Il terzo e quarto tipo sono i **gateway a livello applicativo** e i **gateway a livello di circuito** che sono molto simili.

## Types of firewalls

- A firewall can act as filter:
  - Positive: FW specifies pkts allowed to pass
  - Negative: FW specifies pkts denied to pass
- It can inspect
  - Different protocol header
  - Pkt payload
  - Pattern generated by a pkt stream
- Main categories
  - **Packet filtering firewall**
  - **Stateful inspection firewall**
  - **Application-level gateway**
  - **Circuit level gateway**

9

## Types of firewalls



10

## Packet Filtering Firewall

Si applica a tutti i pacchetti che attraversano la nostra macchina e permette di modificare i pacchetti o di impedirne il passaggio. Agisce a livello protocollare “3/4”. In realtà agisce al livello tre, quindi allo strato di IP, tenendo però in considerazione anche informazioni di strato quattro. In alcuni casi questi firewall possono agire a livello due, o meglio essere posizionati al livello due, per non essere visibili né dagli utenti della rete né da utenti esterni alla rete. La differenza è che se operano su un nodo che agisce da router, chiaramente sia le porte d'ingresso che di uscita avranno

degli indirizzi IP e il pacchetto sarà modificato in ogni caso all'attraversamento del firewall, se poi il firewall agisce anche da NAT allora ci sarà anche il cambiamento delle porte. Se invece il firewall agisce su un dispositivo di strato due, il suo comportamento è trasparente e il traffico passerà indisturbato, mentre, nel caso in cui il traffico soddisfi una delle regole che richiedono delle modifiche, il traffico sarà modificato o gli sarà impedito di attraversare il nodo. Tipicamente le informazioni utilizzate sono quelle dello strato tre e quattro. In alcuni casi, potremmo avere che l'operazione di natting potrebbe essere effettuata per mascherare l'identità degli utenti che escono dalla rete (in quel caso l'operazione di natting è un'operazione di mascheramento).

## Packet filtering firewall

- It applies a set of rules to in/out pkts to
  - Forward
  - Deny
  - Modify
- Typical information used
  - Source/destination IP
  - Source/destination port
  - IP protocol field
  - Network interface
- Define a default policy (**forward/discard**)

## Stateful Inspection Firewall

È “un’evoluzione” del Packet Filtering Firewall, ovvero si basa sul concetto di sessione. La sessione viene creata nel momento in cui viene riaccettato il primo pacchetto della sessione stessa. Quindi, nel caso di connessione TCP, vanno intercettati i pacchetti SYN, quelli che aprono il tre way-handshake del protocollo, mentre, nel caso di pacchetti UDP deve essere intercettato il primo pacchetto che viene aperto da un certo indirizzo IP mittente verso un certo indirizzo IP destinazione con una certa porta mittente verso una certa porta destinazione.

Una volta identificata la connessione, si può definire il comportamento. In particolare, la cosa fondamentale è la direzione, ovvero questi firewall permettono un controllo della direzione di accesso, mentre i Packet Filtering Firewal non lo permettono (non hanno la capacità di capire se la comunicazione è iniziata da un utente interno verso un utente esterno e viceversa). Nel caso dei firewall stateful è possibile perché si va a

intercettare il primo pacchetto e quindi si vede se è originato dall'interno verso l'esterno o viceversa e la decisione presa sul primo pacchetto sarà mantenuta anche per pacchetti successivi appartenenti a quella connessione a quel flusso. Per fare questo viene conservata l'informazione di stato, cioè un database interno che permette l'esecuzione di questo tipo di politiche (stiamo parlando sempre di firewall rule-based ma la regola/policy è applicata al flusso e non al singolo pacchetto).

## Stateful inspection firewall

- It uses the **context** to take decision
  - Creates a state for each **TCP connection**
    - It can decide on the direction of the connection
      - ESTABLISHED, RELATED state in iptables
    - It can also track sequence numbers to block session hijacking
    - It can also inspect of data of control protocols to identify associated sessions
      - SIP and RTP traffic
  - May create also a state for **UDP flows**

## Tipi di regole del Packet Filtering Firewall

Nel packet filtering firewall possiamo avere diversi tipi di regole.

**Esempio A:** in questo schema, la prima regola ci dice di bloccare tutti i tentativi di accesso all'host destinazione chiamato **SPIGOT** e quindi ci sarà una maschera di rete associata a una determinata rete, ad esempio, 141.250.43.0/24. Non ci sono vincoli sull'host di ingresso o sulle porte del mittente e del destinatario (ci sono degli \*), mentre è ammesso il traffico in cui il mittente è il nostro gateway e la porta mittente è la 25. Questo permette al server SMTP di uscire verso qualsiasi altra destinazione ed è un esempio di policy classica perché tipicamente il nostro server di posta elettronica, ammesso che sia interno come in questo esempio, deve poter inviare e scambiare la posta con gli altri server che stanno in rete e quindi non si possono mettere restrizioni sugli host di destinazione. Tuttavia, in alcuni casi esistono delle liste di server SMTP, che sono compromesse e in quel caso posso applicare delle policy al mio firewall oppure posso configurare un firewall che impedisce al server stesso di provare a colloquiare con questi server.

In alcuni casi il ruolo del firewall può essere complementato dalla configurazione dei nostri client o dei nostri server. Questo significa che non bisogna per forza installare un firewall sui nostri client o server ma basta impostare delle regole a livello applicativo.

**Esempio B:** qui la regola è molto semplice perché ci dice che la regola di default è quella di bloccare tutto il traffico e quindi qualsiasi altra regola sarà una eccezione a questa (esempio di firewall positivo). Ovviamente una regola di questo tipo deve essere posta in fondo all'elenco e nel momento in cui ho un certo numero di regole e nessuna di queste viene eseguita allora andrà ad eseguire quella di default. In alcuni casi, più sofisticati, quello che succede è che dopo l'esecuzione di una regola posso passare all'esecuzione di un'altra regola.

## Packet filtering firewall

Rule Set A

action	Ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

Rule Set B

action	Ourhost	port	theirhost	port	comment
block	*	*	*	*	default

Rule Set C

action	Ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

Rule Set D

action	Src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

Rule Set E

action	Src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

12

## Possibili Attacchi

- ❖ **Vulnerabilità a livello applicativo:** Non si può fare nulla se si ha un packet filtering firewall a livello applicativo perché normalmente viene analizzato lo strato tre e quattro, quindi un programma a livello applicativo, bypassa completamente il firewall. Con problemi a livello applicativo ho bisogno di un firewall che agisce su questo livello e quindi con diverso vantage point;
- ❖ **Mancanza di autenticazione:** Nel caso in cui si ha la possibilità di fare autenticare gli utenti, è sempre bene farlo, perché questo garantisce un livello superiore di sicurezza e la possibilità di differenziare le regole tra gli utenti autenticati e quelli non autenticati;
- ❖ **Configurazione sbagliata:** Si ha una configurazione sbagliata quando qualcuno riesce a fare qualcosa che non era possibile fare.

## Attacchi Noti

- **IP Spoofing:** consiste nell'impersonare con la propria macchina che ha un determinato indirizzo IP, un'altra macchina che ha un indirizzo IP diverso. Questo può essere fatto per partecipare ad un attacco o per effettuare un attacco. La differenza è che nel primo caso se io partecipo ad un attacco, il mio firewall mi può impedire di partecipare ma la vittima non è all'interno della mia rete, invece, se la vittima è all'interno della mia rete significa che qualcun altro al di fuori della mia rete sta effettuando un attacco di impersonificazione o di mascheramento. Quindi questa tipologia di attacco prevede l'utilizzo di un indirizzo IP fasullo ed è uno tra le cose più semplici che si possono fare;
  - **Attacchi di source routing:** consistono in quei tentativi in cui all'interno del pacchetto c'è scritta tutta la strada o parti importanti della strada che il pacchetto deve seguire per arrivare a destinazione (questo si specifica facilmente in IPv6). Quindi quando ho un pacchetto che sa qual è il prossimo router da raggiungere per arrivare a destinazione l'instradamento viene fatto verso il prossimo salto e non verso la destinazione reale ed una volta arrivati a quel salto viene tolto quel pezzo dell'intestazione e viene instradato il pacchetto verso il prossimo salto e così via. In questo modo è possibile personalizzare l'instradamento del traffico e spesso viene utilizzato per bypassare alcune sezioni della rete dove ci sono firewall molto potenti o dispositivi che possono intercettare i tentativi di attacco. Una delle classiche regole è quella di droppare tutto il traffico che ha opzioni di source routing;
  - **Attacchi a piccoli frammenti:** considerazioni (potrei avere delle regole che impediscono ad una determinata combinazione, indirizzo ip mittente-destinatario e porta mittente-destinatario, di attraversare il mio firewall. Un attacco che in alcuni casi ha successo è un attacco che si basa su una estremizzazione dell'opzione di frammentazione dei pacchetti. Quando si effettua la frammentazione a livello IP, si spezza tutto il payload IP in più parti e replica solo l'intestazione IP attivando l'opzione more-fragment e l'offset, che ci dice quale parte del pacchetto sto trattando. Se ho un problema nell'attraversare il firewall dovuto alla combinazione degli indirizzi mittente e destinazione dallo strato tre allo strato quattro, questo frammento non passerà mai. Molti firewall però se questo frammento lo fanno passare allora faranno passare tutti i frammenti di quel pacchetto).
- L'attacco consiste nel fare frammentare il pacchetto in piccolissime parti e se il primo frammento passa, allora passeranno anche tutti gli altri frammenti e così facendo riesco a bypassare il firewall. La contromisura a questa tipologia di attacco è che la dimensione del pacchetto IP non può scendere sotto un certo valore e in questo modo se arriva un frammento troppo piccolo lo andrò a scartare e scarterò anche gli altri.

## Possible attacks

- In general, all those exploiting
  - Vulnerability at application layer
  - Improper configurations
  - Lack of authentication
- Known attacks:
  - IP address spoofing
    - Countermeasure: check the interface
  - Source routing attacks
    - Countermeasure: check the relevant option
  - Tiny fragment attacks
    - Countermeasure: check the minimum size of an IP pkt

13

### **P.S Parte della spiegazione in cui il prof fa una confusione assoluta. Ho cercato di capirci qualcosa.**

Le cose appena dette, si applicano anche ai firewall basati sul concetto di connessione o di flusso. Nel concetto di connessione o di flusso si utilizzano i termini: New, Related e Established. Quindi, le decisioni vengono effettuate solo sui flussi nuovi, successivamente tutti gli altri pacchetti relativi a quel flusso saranno etichettati come established o related. I pacchetti etichettati come established sono quelli per cui è stata effettuata una decisione nel momento in cui la connessione/flusso è stato creato/a e quindi posso avere un comportamento diverso per i pacchetti established rispetto a quelli new.

Ad esempio, se sono in una situazione in cui ho un volume di traffico elevatissimo, posso decidere, in base alla capacità della mia rete o dei miei sistemi terminali, di limitare un po' di questo traffico. Ovviamente non posso andare a impattare le connessioni TCP che già sono state create, perché altrimenti creerei solo un disservizio. Quindi posso avere delle differenze di trattamento per le nuove connessioni, rispetto al traffico delle vecchie connessioni e in questi casi nei protocolli che prevedono il numero di sequenza (TCP) è possibile effettuare dei controlli su di esso per evitare che pacchetti che sembrano appartenere a una connessione, ma in realtà non ci appartengono, attraversino il firewall.

Supponiamo di avere un firewall, un client e un server e voglio utilizzare una tecnologia di tipo Packet Filtering. Se ho un Packet Filtering Firewall, posso dire ad esempio che il client accede al mio server sulla porta ( $x > 1024$ ) e porta destinazione 443. Questo permette solo l'attraversamento da sinistra verso destra delle richieste. Per ricevere anche le risposte devo creare una regola che me lo permette. Supponiamo

che il flusso termini e che un attaccante invii del traffico, il Packet filtering firewall farebbe passare il pacchetto che ovviamente non è relativo a nessuna sessione e che ragionevolmente è un tentativo d'attacco, infatti, questa è una delle principali vulnerabilità dei firewall basati sui semplici filtraggi dei pacchetti, perché quando utilizzo il semplice filtraggio dei pacchetti effettuo il criterio a datagramma, cioè ogni pacchetto a sé senza guardare il flusso dei pacchetti.

Nel caso di uno Staetuf Inspection firewall, invece, quello che cambia è che i pacchetti non passerebbero e verrebbero scartati perché vengono identificati come appartenenti allo stesso flusso e questo tipo di firewall capirebbe che non c'è compatibilità tra l'intestazione dei pacchetti del flusso e quelli dell'attaccante. (Core della spiegazione in cui non si capiva nulla XD).

Possono esserci anche altri tipi di attacchi che dipendono da quanto è SMART l'attaccante, come identificare una sessione. A quel punto, sarà l'applicazione che dovrà essere abbastanza robusta da gestire pacchetti che non hanno particolare significato.

## Stateful inspection firewall

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

## Application-level gateway

L' Application-level gateway è un meccanismo che agisce a livello applicativo e di trasporto, quindi di fatto, bypassa lo strato IP e di solito è un tipo di firewall che richiede l'autenticazione. Generalmente viene applicato fra una parte esterna e una parte interna della rete, dove la parte interna della rete è quella che voglio proteggere e la parte esterna è Internet o una parte "meno protetta".

Il suo funzionamento non vede il firewall come il solito intermediario. In questo caso effettua anche un'operazione chiamata TCP Split. Essa prende questo nome perché termina la sessione tcp tra il nodo richiedente e sé stesso e la riapre tra sé stesso e il

nodo di destinazione. Quindi avrò: il mio client, il mio application proxy firewall e il mio server. Il client si connette al firewall (con interfaccia web come https o tramite ssh), che dal punto di vista protocollare non è nient'altro che un'applicazione, effettua la fase di autenticazione e indica l'identità del nodo di destinazione. A questo punto il firewall può decidere di creare una sessione tra lui e il nodo di destinazione e rilanciare i pacchetti che vengono scambiati in questa sessione sulla sessione tra il client e il firewall stesso. Di fatto non si avrà accesso diretto end-to-end al server, ma si avrà un accesso in cui all'interno del Firewall c'è la possibilità di avere uno Stateful inspection dei pacchetti, ovvero il firewall può analizzare il payload dei pacchetti che scambio con la mia applicazione. Questo avviene perché anche se mi sono autenticato sono comunque un utente esterno da tenere sotto osservazione.

Questo è un processo che può limitare in maniera significativa la velocità del firewall, perché ispezionare il payload significa ispezionare una grande quantità di traffico.

## Application-level gateway

- It acts as a **relay of application-level traffic**
  - Aka application proxy
- Typical behaviour
  - The user connects with a protocol, such as Telnet
  - The GW asks for the remote host to connect to
  - The user authenticates
  - The GW connects the user application with the remote host and relay TCP pkts
- More efficient logging support than pkt filters
  - TCP connection split + traffic analysis
- Drawbacks:
  - Only supported applications can be relayed
  - Additional processing load

16

## Circuit-level gateway

Il Circuit Level Proxy Firewall è una versione depotenziata dell'application proxy firewall perché viene sempre spartita la connessione TCP ma senza salire allo strato applicativo. Quindi quello che si fa è permettere ad un utente esterno di accedere ad un server sull'utente interno mascherandolo, così da far dimostrare all'utente che conosce l'indirizzo del server.

La differenza fondamentale è che nel secondo caso (quello del **circuit**) non si fa la stateful inspection, quindi non vado a effettuare nessun tipo di ispezione. Si utilizza questa versione quando la mia connessione parte dal server e non da un client esterno.

Quindi di solito si dice che le connessioni inbound (da fuori verso dentro) vengono gestite con un application proxy firewall, mentre le connessioni outbound (da dentro verso fuori) vengono gestite con circuit proxy firewall.

## Circuit-level gateway

- It splits the TCP connection towards a remote host
  - One with internal host
  - One with external host
  - **No pkts inspection**
- It is a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications
  - Aka circuit-level proxy
  - Administrator **trusts internal users**
  - Can be combined with **application-level gw for inbound traffic**

17

## DMZ (DeMilitarized Zone)

In alcune reti, esiste una particolare area in cui vengono messi a disposizione dei servizi. Quest'area prende il nome di **zona demilitarizzata o DMZ**. Un classico schema operativo è quello che vediamo sulla figura di sotto in cui abbiamo un router esterno, un firewall perimetrale (che dal punto di vista logico è separato dal router ma dal punto di vista fisico potrebbe essere collocato con il router) e una prima zona a cui accedo direttamente dopo aver superato firewall perimetrale (esterno) che prende il nome di DMZ in cui mettiamo tutti i servizi che vogliamo siano accessibili agli utenti esterni. Andiamo a separare questi servizi dalla parte interna della rete perché tipicamente la parte interna è la più importante ed è quella che sono interessato a proteggere maggiormente. Infatti, tra la DMZ e la rete interna adibisco un ulteriore firewall (interno) che può essere collocato con router o con uno o più switch.

La differenza sostanziale è che i servizi all'interno della DMZ sono servizi aperti al pubblico, ad esempio un server web o server DNS che, come tali, sono ospitati su un server. Ovviamente questi server devono poter accettare richieste di servizio che vengono dall'esterno e il livello di protezione da dare a queste macchine deve essere necessariamente un po' più blando perché non possiamo mettere delle regole sul firewall della DMZ che bloccino il traffico esterno. Tipicamente un firewall che protegge la DMZ va a limitare i tentativi di accesso che non sono compatibili con i

servizi offerti in questa zona. Gli stessi tentativi di accesso ma su delle workstation che stanno nella parte interna della rete saranno bloccate dal firewall interno.

Noi potremmo mettere delle regole per cui il firewall esterno fa passare quei tentativi di accesso solo verso i server e non verso tutti gli altri indirizzi IP ma in questo modo elimineremo la barriera del firewall interno tra la parte interna della rete e la DMZ. Questo significa che se per caso, attraverso una richiesta di servizio, uno dei server della zona demilitarizzata viene compromesso, automaticamente viene compromessa l'intera rete. Per questo con una segmentazione della rete in più zone, si può fare in modo che il firewall interno protegga la rete anche nel caso ben vengano compromessi i server della DMZ.

In generale, è possibile utilizzare un certo numero di firewall a seconda delle caratteristiche interne della rete, quindi oltre al firewall perimetrale posso avere anche più firewall adibiti a proteggere in maniera diversa altre sezioni della rete. In particolare, si tende a rendere più restrittivo l'accesso tramite le connessioni wireless, quindi è pratica abbastanza comune che quello che si può fare attraverso il Wi-Fi è più limitato rispetto a quello che si può fare con altri tipi di accesso (cablato o accesso tramite sicurezza aziendale).

Una cosa che solitamente si fa è che per isolare la rete interna rispetto alla DMZ è quello di far accedere ai server della DMZ su una porta diversa del firewall.

## Firewall and DMZ

### □ External FW:

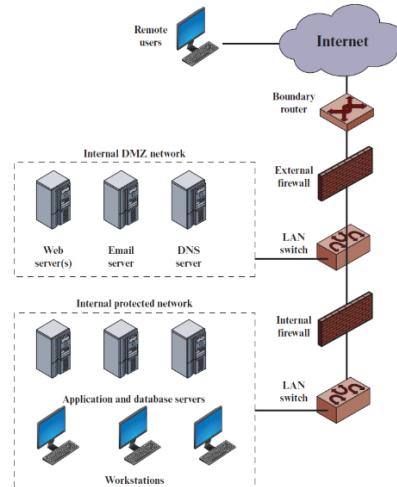
- Protects the Internet access

### □ Internal FW:

- Protects the core of enterprise network

### □ Demilitarized zone (DMZ)

- System accessible from the Internet
- Needing some protection



## Firewall and DMZ

### □ Main tasks for internal FW:

- More stringent filtering capability to protect enterprise servers and workstations
- Two-way protection wrt the DMZ
  - It protects the remainder of the network from attacks launched from DMZ systems.
    - Such attacks might originate from worms, rootkits, bots, or other malware lodged in a DMZ system.
  - It can protect the DMZ systems from attack from the internal protected network
- Multiple internal firewalls can be used to protect portions of the internal network from each other
  - A common practice is to place the DMZ on a different network interface on the external firewall

19

## Intrusion Detection System

- **Intrusione:** è una violazione delle politiche di sicurezza, tipicamente caratterizzato da un tentativo di compromettere uno dei 5 obiettivi che abbiamo visto in precedenza: confidenzialità, integrità, disponibilità, accountability e autenticità. Queste violazioni possono avvenire: da attaccanti che accedono al nostro sistema dall'esterno, da utenti autorizzati che cercano di bypassare le restrizioni per il proprio livello di sicurezza (utente guest che tenta di acquisire i permessi di root) oppure da un utente guest che non tenta

di acquisire i permessi di root ma permette di fare operazioni che non sono concesse a lui ma sono concesse solo all'utente amministratore;

- **La rilevazione dell'intrusione:** è il processo di raccolta e analisi delle informazioni per rilevare segni d'intrusione;
- **IDS:** l'intrusion detection system è una soluzione hw/sw il cui scopo è quello di rilevare le intrusioni e di eseguire le operazioni di intrusion detection;
- **Tipi di IDS:** gli **host-based** e i **network-based**. Gli IDS basati sugli host guardano solo il traffico in ingresso e in uscita dall'host. Esso ha un compito più semplice del network-based IDS perché tipicamente l'host-based viene messo sull'host stesso e quindi il suo vantage point è perfetto perché sta esattamente sul luogo che bisogna osservare. Invece, quando noi abbiamo un sistema di rilevamento delle intrusioni che deve rilevare le intrusioni sull'intera rete, abbiamo più punti di vantaggio e quindi potrebbero servire più IDS. L'IDS tipicamente non si occupa solo di guardare al traffico in rete, ma spesso va a combinare l'osservazione del traffico con l'osservazione dei log cercando di rilevare le intrusioni.
- **Sensori:** responsabili di collezionare i dati e le informazioni. Alcuni tipi di input possono essere pacchetti, file di log o call traces di sistemi. I sensori inviano i dati raccolti all' analyzer, ovviamente più informazioni avrò e più il carico computazionale dell'IDS aumenta;
- **Analyzers:** una stazione di analisi che mette insieme la parte di memorizzazione delle informazioni (grezze o pre-elaborate) e le eventuali analisi che posso fare su di esse, ovvero un sistema che raccoglie, memorizza e elabora i dati;
- **Interfaccia utente:** permette all'utente di visualizzare l'output degli analyzer o di controllare il comportamento del sistema. In alcuni casi può anche rappresentare una console per gestirne il comportamento.

# Intrusion Detection System

- Types of IDS:
  - Host-based IDS:
    - Violations are easy to verify (vantage point)
  - **Network-based IDS**
- An IDS includes 3 logical components:
  - **Sensors**
    - Sensors are responsible for collecting data. The input for a sensor may be any part of a system that could contain evidence of an intrusion
    - Types of input to a sensor include network packets, log files, and system call traces
    - Sensors collect and forward this information to the analyzer
  - **Analyzers**
    - Analyzers receive input from one or more sensors or from other analyzers. The analyzer is responsible for determining if an intrusion has occurred
    - The output of this component is an indication that an intrusion has occurred.
      - The output may include evidence supporting the conclusion that an intrusion occurred
    - The analyzer may provide guidance about what actions to take as a result of the intrusion
  - **User interface**
    - The user interface to an IDS enables a user to view output from the system or control the behavior of the system
    - In some systems, the user interface may be a manager, director, or console component

# Intrusion Detection System

- **Analyzers**
  - Analyzers receive input from one or more sensors or from other analyzers. The analyzer is responsible for determining if an intrusion has occurred
  - The output of this component is an indication that an intrusion has occurred.
    - The output may include evidence supporting the conclusion that an intrusion occurred
  - The analyzer may provide guidance about what actions to take as a result of the intrusion
- **User interface**
  - The user interface to an IDS enables a user to view output from the system or control the behavior of the system
  - In some systems, the user interface may be a manager, director, or console component

# Intrusion Detection System

- Naming relevant to IDS:
  - **Intrusion:** Violations of security policy, usually characterized as attempts to affect the CIA of a system
    - These violations can come from
      - attackers accessing systems from the Internet
      - from authorized users of the systems who attempt to overstep their legitimate authorization levels
      - who use their legitimate access to the system to conduct unauthorized activity
  - **Intrusion detection:** The process of collecting information about “network” events and analyzing them for signs of intrusions
  - **IDS:** Hardware or software products that exec intrusion detection process to raise suitable warnings

Tipicamente la linea di difesa di base di una rete è costituita da un sistema di identificazione in accoppiata con il controllo di accesso e il firewall. Un IDS che funziona bene permette di notificare un tentativo di accesso malevolo appena si verifica e quindi di limitarne gli effetti dannosi.

In alcuni casi la presenza di un buon IDS può essere un deterrente. Un possibile IPS (Instruction Prevention System) potrebbe essere costruito come la combinazione di un IDS e di un firewall configurato da una terza parte e che riceve in input gli **alert** dell'IDS e sulla base di questi va configurare di conseguenza il firewall.

**Osservazione:** In alcuni casi la presenza di un IDS può essere vista come una sfida perché se esso non è particolarmente performante può rappresentare un'attrazione per un certo numero di soggetti che vogliono sperimentare le proprie abilità in termini di cyber security. Quindi se voglio utilizzare un IDS devo assicurarmi che sia performante.

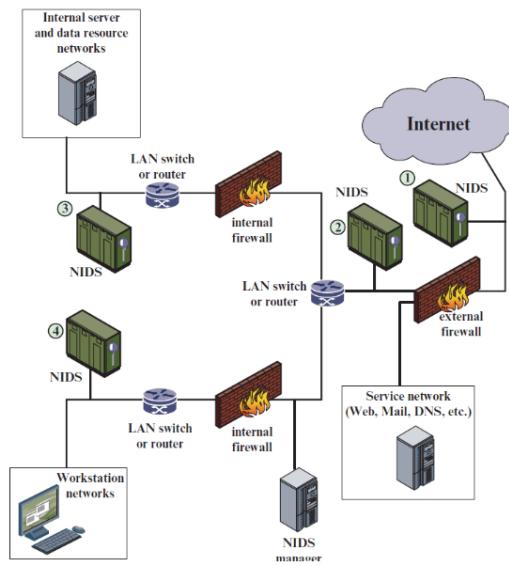
## Why using an IDS

- Baseline defense against intrusions:
  - Authentication, access control, FW
- IDS is a plus:
  - If intrusion is detected early, the intruder can make no or little damage to the system
  - The presence of an IDS can be a deterrent
  - It enables collection of info about intrusion strategies useful to strengthen IPS

In questo schema possiamo vedere come gli IDS siano posizionati lateralmente rispetto al traffico e rispetto al collegamento che unisce le workstation o i server con i firewall. In base a dove vado a posizionare il mio IDS posso andare a osservare una certa tipologia di traffico (ad esempio prima o dopo un firewall).

## Where to place an IDS

- ❑ ID means looking at the pkts as they pass by some sensor
- ❑ Interesting pkts match a signature
- ❑ Three primary types:
  - string signatures
  - port signatures
  - header condition signatures



## Recap:

Eraamo arrivati a discutere dei possibili piazzamenti delle sonde che agiscono come sistema di rilevamento delle intrusioni. Ci sono 3 principali sistemi IDS:

- quelli che guardano al contenuto del traffico, quindi del payload dei pacchetti e vanno a cercare quelle che prendono il nome di **string signatures**. Ad esempio, se abbiamo accesso al treno dei pacchetti HTTP, eventuali contenuti caricati tramite il messaggio di post.
- possiamo avere delle signature basate sulle port (**port signatures**). Ad esempio, una particolare combinazione di indirizzi IP, mittente e destinazione, porta di ingresso e di destinazione, e magari anche il volume dei tentativi di accesso ad un determinato servizio.
- Possiamo avere delle signature basate su delle combinazioni particolari nelle intestazioni dei pacchetti (**header condition signatures**). È sicuramente non un tentativo di accesso normale un pacchetto TCP che porta contemporaneamente il flag SYN e il flag FYN nella stessa intestazione. Il pacchetto serve ad aprire e contemporaneamente a chiudere una connessione, quindi è un pacchetto senza senso, e o è generato da un'applicazione che ha un bug, oppure può essere il sintomo di un tentativo di attacco che magari va a utilizzare una vulnerabilità.

## Approcci per il rilevamento delle intrusioni

I sistemi di rilevamenti delle intrusioni hanno come obiettivo quello di rilevare i tentativi di intrusione, non di bloccarli.

Possiamo dire che esistono due macro approcci per la rivelazione delle intrusioni.

Nessuno dei due è strettamente migliore dell'altro, la linea che separa i due comportamenti è spesso labile. Hanno usi diversi e spesso sono utilizzati in maniera combinata.

L'assunzione fondamentale è che il comportamento di un utente legittimo è tipicamente diverso dal comportamento di un utente che tenta di fare un attacco o tenta di intrufolarsi in un sistema nel quale non ha il diritto di entrare.

Nel caso in cui questa assunzione venga rispettata, è di fatto impossibile discriminare un utente malevolo da un utente benigno. Quindi se un utente malevolo è un utente autorizzato che è armato solo di cattive intenzioni, si vedrà quello che succede. Quindi, ad esempio, utilizzando solo una signature basata sulle porte (indirizzo mittente e destinazione, porta mittente e destinazione) o basata sulle intestazioni non saremmo mai in grado di rilevarlo. Poi dipenderà anche dai comandi che andrà a dare dal sistema, sempre che sia in grado di dare dei comandi.

Un'altra cosa fondamentale è che i sistemi di rete, sia i servizi sia le reti che connettono questi servizi a internet sono in generale continuamente sotto attacco, solo che la

stragrande maggioranza di questi attacchi non rappresentano minacce, e quindi non hanno un uso pericoloso.

Questo rende il sistema di rilevamento delle intrusioni, un sistema particolarmente critico, perché genererà le contromisure agli attacchi. Alcuni di questi allarmi avranno una rilevanza diversa dagli altri, e le capacità di chi analizza questi allarmi, sia esso un sistema informatico, un utente umano, un router, è proprio andare a determinare tra gli attacchi che non sono chissà quanto pericolosi da quelli che non sono pericolosi affatto.

È fondamentale che il comportamento di chi effettua un attacco sia diverso da quello di un utente legittimo.

I due macro approcci sono i seguenti: nomi simili ma sono il contrario uno dell'altro.

- **Rilevamento basato sull'uso anomalo: misuse detection**

Utilizziamo un database di signature quindi di impronte di attacchi noti, e andiamo a classificare il traffico sulla base della differenza del traffico che abbiamo ad osservare con queste impronte. Se c'è una corrispondenza, non solo noi vediamo un attacco, ma sappiamo anche che attacco è. E quindi possiamo prendere la contromisura necessaria, sempre se l'attacco è rilevante e necessita di una contromisura.

Lo scopo della misuse detection è quello di rilevare attacchi noti, facendo uso di database di impronte di attacchi.

- Utilizza algoritmi di tipo pattern-matching: ho un certo tipo di pattern, tutti i tipi di traffico che matchano con quel pattern saranno classificati come quel tipo di attacco. Il traffico che non matcha nessun tipo di attacco, di conseguenza è il traffico legittimo.

- Migliori prestazioni in termini di falsi positivi. I falsi positivi è il traffico legittimo che viene classificato come attacco.

Proprio perché io ho delle impronte ben precise che sono stati rilevate da attacchi passati e quindi, di solito, questi database vengono aggiornati con regolarità e sono disponibili anche diversi tipi di sistemi IDS.

È difficile che il traffico legittimo possa essere classificato come un attacco, e quindi la % di falsi positivi è molto bassa.

- L'altra cosa fondamentale è che proprio perché ho un database di impronte di attacchi, di solito questi software riescono a sfruttare in maniera molto efficiente le capacità di calcolo parallelo e quindi sono efficienti in generale dal punto di vista computazionale, sono veloci.

Esempio classico di IDS che utilizza il misuse detection è suricata. È un software open source che fa affidamento su un database che qualcuno ha costantemente aggiornato di impronte di attacchi.

Non riesce a rilevare tutti gli attacchi che non sono nel database delle impronte (gli attacchi nuovi o semplicemente attacchi al di fuori del database, e che siano

significativamente diversi dagli attacchi presenti nel database), per questo sono stati definiti gli IDS basati sull'anomaly detection.

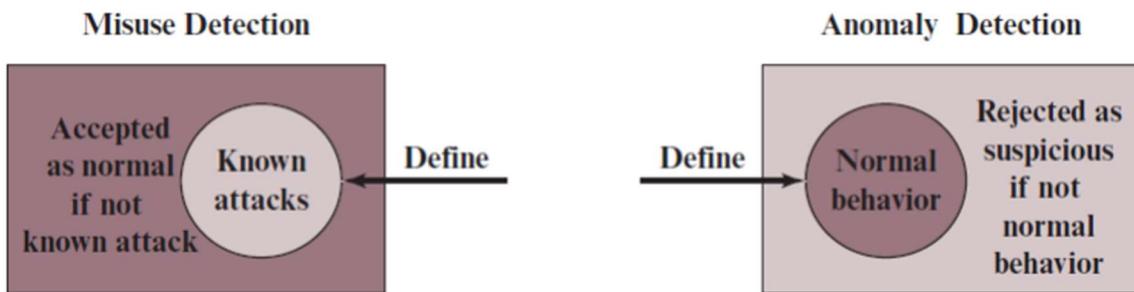
- **Rilevamento della anomalia: anomaly detection**

- Basato sulla ricerca di attività anormale; mentre il misuse detection è allenato sulle impronte di attacchi, quindi sul traffico al normale, questo invece è allenato sul traffico anormale. E quindi tutto ciò che devia da questa pseudonormalità, viene classificato come un attacco.
- Prestazioni abbastanza bilanciate di falsi positivi – falsi negativi.  
I falsi negativi sono il grosso problema del misuse detection. Un falso negativo è un attacco che erroneamente viene classificato come traffico legittimo. Questo succede nel misuse detection quando ho un attacco che non ho mai visto, cioè non è tra quelli riportati nel database.  
Nell'anomaly detection posso avere sia i falsi positivi che i falsi negativi.  
Ho i falsi positivi perché analizzo il traffico legittimo, però ci possono essere utenti che hanno tutto il diritto di scambiare un certo tipo di traffico che per la loro particolare attività potrebbero essere erroneamente classificati come attaccanti; il loro traffico potrebbe essere classificato come un attacco.
- Utile per rilevare attacchi o-day, gli attacchi nuovi, quelli mai visti.  
Quindi se ho un meccanismo basato su anomaly detection ci potrebbe essere una buona probabilità che riesco a identificare un tentativo di attacco.

**Differenza** enorme tra i due approcci: nel primo caso rilevo gli attacchi e riesco anche a identificare il tipo di attacco, nel secondo caso posso fare una discriminazione solo binaria, quindi traffico legittimo o anomalia=attacco. Nel primo caso, se un attacco matcha più profili di allarme ci sarà uno score e l'impronta che avrà uno score più elevato sarà quella identificata come il mio attacco.

Se conosco l'attacco sono anche in grado di mettere in piedi una contromisura, se non conosco l'attacco posso solo generare un allarme, poi sarà l'amministratore di sistema ad agire di conseguenza e a cercare di bloccare qualcosa o mettere temporaneamente offline il servizio o la rete, a seconda di quanto è critica l'infrastruttura di cui stiamo parlando.

Dal punto di vista grafico possiamo vederli in questo modo:



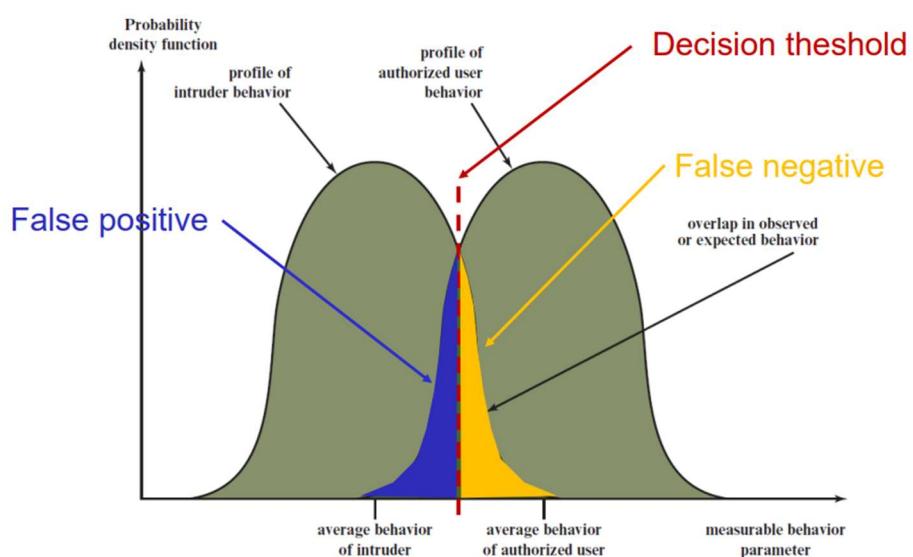
Io mi alleno sul cerchietto, il colore più scuro è il traffico normale, quello più chiaro sono gli attacchi.

Nel caso di misuse detection io conosco gli attacchi, sono quelli che vado a definire e sui cui mi alleno; tutto quello che non è attacco è traffico legittimo e passa. Quindi vado a cercare gli attacchi.

Nel caso di anomaly detection definisco il traffico normale su cui mi alleno, tutto ciò che non è normale lo definisco come attacco, quindi o viene bloccato o genera un allarme. Ricordiamoci che sono schemi IDS, quindi non sono sistemi di prevenzione.

Poiché i bordi non sono mai netti la decisione è critica (le cose non sono sempre facili!) e quindi entra in campo una branca dell'intelligenza artificiale che è il machine learning. È necessario un approccio probabilistico -> il machine learning è uno strumento adatto

I bordi non sono mai netti significa che in generale noi non abbiamo mai una distinzione netta tra quello che è un attacco e quello che è traffico legittimo. Perché ci sono dei casi in cui del traffico legittimo può sembrare un attacco, ma rimane traffico legittimo e invece un attacco può sfuggire all'IDS perché sembra a tutti gli effetti del traffico legittimo. Quindi c'è sempre una certa sovrapposizione tra il comportamento degli intrusi e il comportamento degli utenti legittimi.



In questo grafico in ascissa consideriamo un parametro misurabile che va a caratterizzare il comportamento degli utenti, quello nel gergo del ML prende il nome di feature, quindi questa sarà una delle features. Supponiamo per semplicità grafica che il nostro sistema possa essere descritto da una feature sola.

In ordinata ho una pdf (probability density function), quindi la densità di probabilità.

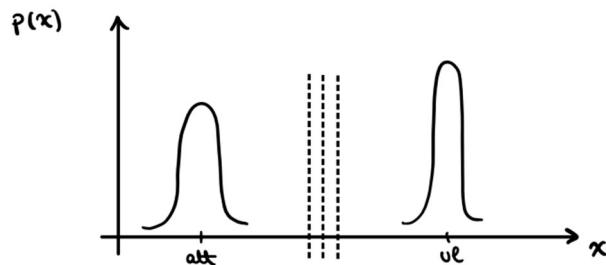
Queste due campane rappresentano la probabilità condizionata.

La campana di destra rappresenta la probabilità di questo parametro condizionata al fatto che questo parametro è generato dal traffico di un utente legittimo. La feature sarà ad esempio uno dei metadati che vado a catturare con netflow, o una statistica che vado a catturare attraverso una sonda che misura la latenza, o il numero di byte scambiati su una connessione.

La campana di sinistra invece rappresenta la pdf condizionata al fatto che i valori del parametro misurato sono stati generati dal traffico di un utente che tenta di entrare in maniera illegittima sul nostro sistema.

Le cose sarebbero semplici se le due pdf fossero completamente separate. Questo perché la sovrapposizione rappresenterà proprio la causa del nostro errore. Errore: ci sono i falsi positivi e i falsi negativi. Nessun sistema decisionale è in generale esente dagli errori. Qui possiamo avere due tipi di errori.

Se le campane fossero separate per discriminare tra utente legittimo e attacco mi basterebbe piazzarmi in un punto qualsiasi e tirare una linea che rappresenterebbe la mia soglia decisionale.



Invece noi abbiamo questa zona di sovrapposizione. Piazziamo una soglia sull'intersezione delle due campane. Tutta la porzione di traffico legittimo che avrà valori al di sotto della soglia (quella colorata in blu) saranno falsi positivi dal punto di vista della rilevazione dell'attacco, cioè se non ho un attacco non lo sono, sono la coda inferiore della campana di destra. Mentre tutti quei casi in cui il nostro parametro ha un valore più alto della soglia e quindi ricade per definizione nel traffico legittimo ma rappresenta invece la coda superiore della campana di destra parleremo di falsi negativi.

È sempre giusto piazzare la soglia sulla prima intersezione? Dipende dall'importanza relativa del falso positivo e del falso negativo. In genere si fanno sempre esempi di natura

medica: tumore. Se ho un falso positivo quello che faccio tipicamente è effettuare nuove analisi per essere sicuro che io ho il tumore, mentre se avessi un falso negativo, io ho il tumore e al primo screening sfugge, poi il tumore cresce e potrebbe portare alla morte.

Quindi nei casi in cui i falsi positivi sono più importanti dei falsi negativi o viceversa, la soglia andrà spostata di conseguenza. Ci saranno delle funzioni di perdita che dovremo utilizzare e saranno quelle che determinano la posizione della soglia.

### Matrice di confusione

Real behaviour	Attack X=1	No attack X=0
Test result		
Anomaly detected $\hat{X}=1$	<i>True positive</i>	<i>False positive</i>
Normal behaviour $\hat{X}=0$	<i>False negative</i>	<i>True negative</i>

Di solito andiamo a caratterizzare questi algoritmi sulla base della matrice di confusione. In questa matrice sulle colonne ho un comportamento reale e sulle righe il comportamento stimato.

La prima colonna rappresenta l'evento di attacco e la seconda colonna rappresenta l'evento di traffico legittimo.

La prima riga rappresenta l'evento di anomalia rilevata e la seconda riga rappresenta l'evento di anomalia non rilevata.

Positive è sempre relativo al concetto di rilevazione dell'attacco.

True positive: ho un attacco e l'ho rilevato come tale.

True negative: ho traffico legittimo e l'ho classificato come tale.

Gli errori sono sulla diagonale secondaria.

False positive: non ho un attacco e lo classifico come tale.

False negative: ho un attacco ma non lo classifico come tale.

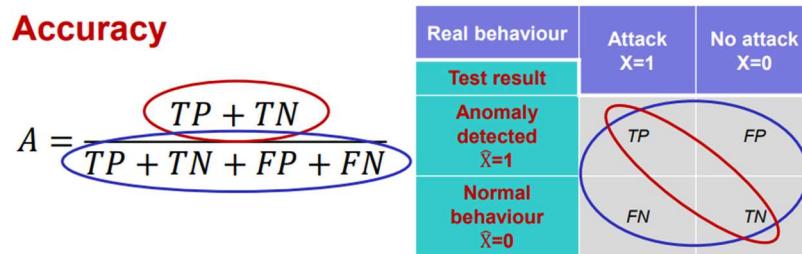
Gli IDS sono algoritmi di classificazione, in grado di classificare o il tipo di attacco o se c'è o meno un attacco; utilizzati sempre e solo per classificare.

## Metriche delle prestazioni

**Accuratezza:** misura statistica della capacità di un test di classificazione binaria di identificare o escludere correttamente una condizione; prossimità dei risultati della misurazione al valore reale.

Metrica principale ma non sempre utilizzabile. Ci dice quanto le nostre decisioni siano corrette rispetto all'insieme degli eventi. Ci dice di fatto quanti sono i veri negativi e i veri positivi rispetto a tutte le nostre decisioni. L'ottimo è quando ho la diagonale secondaria pari a 0.

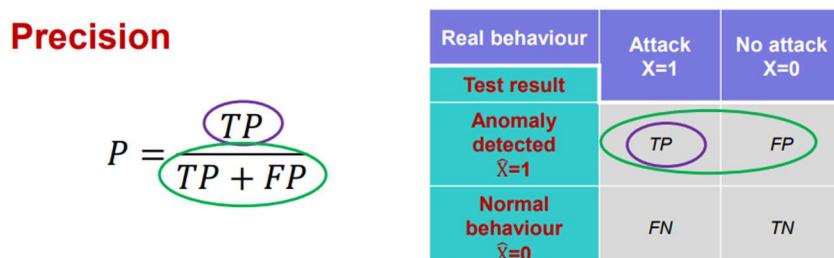
Non si utilizza sempre perché non la uso in quei casi in cui ho dataset particolarmente sbilanciati. Soprattutto perché nel caso delle minacce, che noi sappiamo essere molto poche rispetto all'insieme degli attacchi, quello che può succedere è che su migliaia di attacchi le minacce sono qualche decina o qualche unità. E quindi, essere in grado di andare a rilevare le minacce può essere un task particolarmente difficile, anche perché sono eventi rari. Quando sono eventi rari, il fatto di sempre che i TP siano tanti meno dei TN può fare in modo tale che io ho un'accuratezza modellata anche se i TP sono pari a zero.



Per questo si usano queste metriche secondarie, ma non di secondaria importanza, che sono la precisione e il richiamo.

**Precisione:** noto come valore predittivo positivo; riproducibilità della misura.

Ci dice quanto è buona la nostra stima dei positivi.



**Richiamo:** noto come sensibilità nella classificazione binaria diagnostica.

Ci dice quanti degli eventi di interesse (dei positivi) sono in grado di identificare. Vado quindi ad effettuare la valutazione per colonna: prendo come riferimento non la stima come prima, ma l'evento reale, cioè la colonna attacco.

## Recall

$$R = \frac{TP}{TP + FN}$$

Real behaviour	Attack X=1	No attack X=0
Test result		
Anomaly detected $\hat{x}=1$	TP	FP
Normal behaviour $\hat{x}=0$	FN	TN

**F-score:** la media armonica (ponderata) della precisione e del richiamo.

L'F-score, in particolare questo è l'F<sub>1</sub>-score combina precisione e richiamo e penalizza molto gli estremi. È una versione bilanciata tra il richiamo e la precisione.

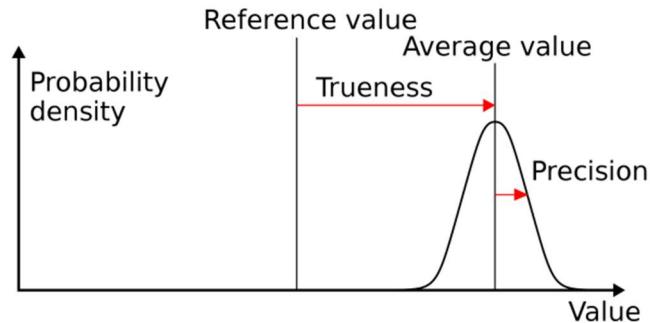
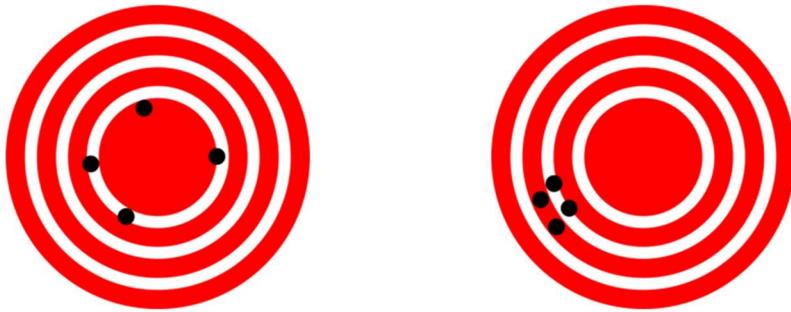
## F-score

$$P = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

$$= \frac{2TP}{2TP + FP + FN}$$

Real behaviour	Attack X=1	No attack X=0
Test result		
Anomaly detected $\hat{x}=1$	TP	FP
Normal behaviour $\hat{x}=0$	FN	TN

## Una vecchia storia: accuratezza vs precisione



Dal punto di vista pratico si usa spesso il formalismo del bersaglio.

Diremo che un meccanismo è tanto più preciso quanto i campioni tendono a essere raggruppati. In questo caso ho una buona precisione ma una scarsa accuratezza perché il valore medio è distante dal valore reale.

L'accuratezza ci dice quanto il valor medio della nostra stima è vicino al valore reale. In questo caso ho dei dati che sono distanti tra loro in cui il valor medio è tanto vicino al centro: ho una buona accuratezza e una scarsa precisione.

- ➔ Nell'ambito della sicurezza informatica, o in generale anche del network management, diremo che la cosa che più ci interessa è minimizzare i falsi negativi: voglio che tutte le anomalie siano identificate. Dobbiamo però stare attenti che questo non causi un aumento indiscriminato dei falsi positivi, cioè i falsi allarmi. Il falso allarme ha l'effetto di desensibilizzare il sistema perché se ho un numero esagerato di falsi allarmi la tendenza alla fine è quella di non considerare più nessuno. Questo è il modo migliore per tralasciare i veri attacchi. Avere un sistema tarato per identificare tutti i falsi negativi ma che genera un numero enorme di falsi allarmi è quindi un sistema che non funziona.

## Metriche delle prestazioni

Altre metriche: AUC e ROC.

La ROC rappresenta un modo per identificare la sensibilità del nostro criterio decisionale sulla scelta della soglia. È una curva che ha per ascissa l'FPR o P(FP) (probabilità di falsi positivi).

FPR è definita come falsi allarmi / (falsi allarmi + correct rejections). Ci dice quanti errori vado a considerare e quanti errori commetto nell'identificare il traffico legittimo.

Al numeratore ho il recall, quindi quanti degli attacchi reali riesco ad identificare con il meccanismo di predizione.

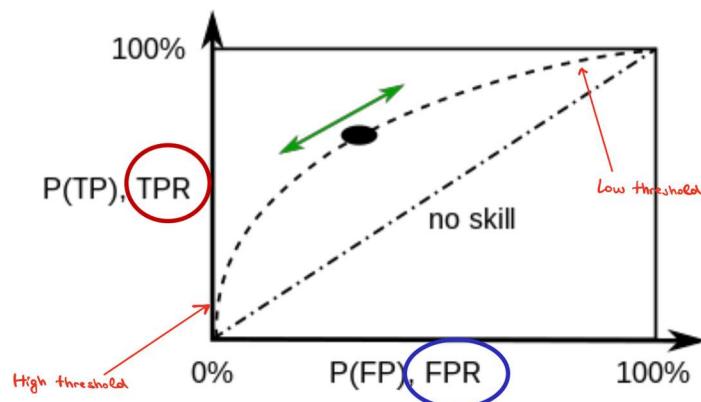
Prendiamo quindi la tabella per colonna: la blu è la FPR e la rossa è la TPR.

		recall	
		Attack X=1	No attack X=0
Real behaviour	Attack X=1		
	No attack X=0		
Test result	Attack X=1		
	No attack X=0		
Anomaly detected $\hat{X}=1$	Attack X=1	TP	FP
	No attack X=0	FN	TN
Normal behaviour $\hat{X}=0$	Attack X=1		
	No attack X=0		

La curva può essere interpretata in questo modo: se abbasso molto la mia soglia quello che ho è che cresceranno molto i falsi positivi, invece se metto una soglia troppo alta vado a minimizzare il numero di falsi positivi ma minimizzo anche il numero di attacchi veri che sono in grado di identificare.

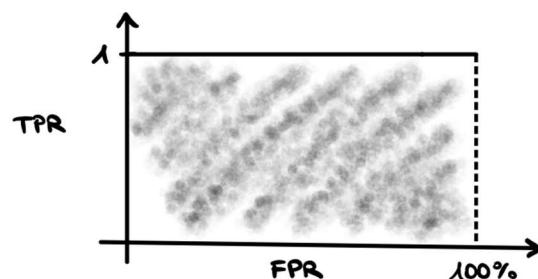
Questa curva mi dice, a seconda della mia soglia, qual è la performance del mio algoritmo di classificazione.

Esiste un criterio chiamato AUC, area sotto la curva ROC, che è di fatto una quantificazione invariante rispetto alla scelta della soglia. Può essere definita come la probabilità che un campione positivo scelto a caso abbia uno score inferiore (cioè viene classificato erroneamente) rispetto a un campione negativo scelto a caso.



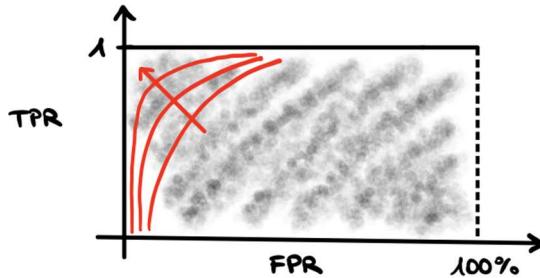
La cosa peggiore è quindi quando l'area è il triangolo: quindi la probabilità di falsi positivi è uguale alla probabilità di falsi negativi (come a dire che scelgo lanciando una monetina).

Il comportamento ideale sarebbe questo:



In questo caso avrò 0% di falsi positivi e 100% di veri positivi, quindi minimizzo gli errori.

Più la mia curva si sposta nella direzione indicata dalla freccia rossa e migliore è la prestazione dell'algoritmo, indipendentemente da come scelgo la soglia.



Il problema di queste metriche è che sono interessanti perché prescindono da alcuni aspetti pratici della configurazione ma tendono, soprattutto l'AUC, a non distinguere più tra falsi positivi e falsi negativi. Quindi se è importante andare a discriminare tra i due allora non vanno bene queste metriche.

Metrica che si sta diffondendo negli ultimi anni:

**Spiegabilità:** il motivo di un risultato di un algoritmo di ML può essere più importante del risultato stesso. Questo perché spesso gli algoritmi di ml sono utilizzati come black box e questo è un trend che bisognerebbe evitare il più possibile. È importante capire perché un algoritmo funziona e perché in alcuni casi funziona male. Non sempre questo è possibile, soprattutto con le reti neurali, che sono sistemi molto molto complessi dal punto di vista matematico, fortemente non lineari e quindi è spesso difficile avere una vera spiegabilità dei risultati che forniscono. Tuttavia, soprattutto nei tool più semplici, è importante capire a priori perché un certo tool si presta meglio a determinate situazioni rispetto ad un altro.

## Rilevamento delle anomalie: supervised ML

Il rilevamento delle anomalie può essere implementato con il **supervised learning** (SL) quando:

- ho un gran numero di campioni sia di traffico legittimo che di tipi di attacchi con cui addestrare il sistema -> avrò una classificazione multiclasse
- SL sarebbe particolarmente adatto per problemi in cui prevediamo che gli attacchi futuri siano simili agli attacchi passati, presenti nel training set

Può essere difficile trovare un pool rappresentativo di campioni positivi per addestrare bene l'algoritmo.

Le violazioni dei server sono talvolta causate da attacchi zero-day o da vulnerabilità appena rilasciate nel software: sfuggono a questo rilevamento. Però avere la possibilità di un set di attacchi noti è un vantaggio importante.

Per definizione, il metodo di intrusione non può essere previsto.

Spesso quando si mette in piedi un sistema di tipo IDS, soprattutto in ambito enterprise, si accoppiano sistemi basati sul misuso a sistemi basati sull'anomalia: uno cerca di classificare gli attacchi come attacchi noti e l'altro invece cerca solo di capire se c'è un attacco nuovo oppure no.

L'ordine in cui faccio girare gli algoritmi può dare vantaggi o svantaggi.

Ad esempio potrei utilizzare prima un algoritmo di tipo misuse detection e poi un algoritmo di tipo anomaly detection.

Nel primo caso il primo tipo di IDS andrà a classificare tutti gli attacchi noti e farà passare il traffico legittimo e gli attacchi che sfuggono alla classificazione di attacchi noti. Se poi applico a questo presunto traffico legittimo l'anomaly detection è come se l'applicassi a quello che reputo solo traffico legittimo. Se ancora rilevo delle anomalie, questo traffico anomalo sarà potenzialmente traffico dovuto ad attacchi che sfuggono alle impronte presenti del database dell'IDS che funziona con il misuse detection. Questo però non implica che in alcuni casi potrei avere del traffico classificato come attacco ma sbagliando il tipo di classificazione, è comunque un problema.

Se inverto la classificazione, quindi faccio prima anomaly detection e poi misuse detection, assumo che nell'anomaly detection raccolgo tutti gli attacchi e una volta radunato tutti gli attacchi tutto il traffico legittimo passa e tutto il traffico definito come attacco passa in un IDS che lavora con misuse detection. Tutto quello che il misuse detection classifica come attacco è un attacco e quello che classifica come traffico legittimo è ragionevolmente un attacco nuovo che non riesce ad identificare.

Di solito le prestazioni nei due casi sono simili. Altrimenti posso metterli in parallelo e andare a confrontare i risultati.

Problema principale: attacchi nuovi. Utilizzo il SL se è molto importante classificare gli attacchi noti per poi bloccarli il prima possibile e se insieme a questo utilizzo un meccanismo basato su anomaly detection per classificare i nuovi attacchi o se assumo che ci sia una comunità molto attiva che aggiorni in continuazione il database delle impronte degli attacchi e che quindi anche gli attacchi o-day possano essere classificati il prima possibile. Spero di non essere tra le prime vittime dei nuovi attacchi e che non appena un attacco viene rilevato viene classificato e inserito nel db.

Eventi relativamente rari => classe di problemi di squilibrio non adatta a SL.

## Rilevamento delle anomalie: euristiche

Le euristiche sono tecniche basate su soglie:

- Come impostare la soglia? La soglia deve essere impostata in modo da minimizzare i nostri errori (falso positivo e falso negativo).
- È uguale per tutti gli utenti? Una cosa fondamentale è che quando uso le euristiche non sempre faccio della analisi di tipo statistiche dei dati ma è importante essere più generali possibili, ad esempio andando a identificare dei casi notevoli. Se ho utenti che hanno permessi diversi io dovrei andare a impostare soglie diverse per utenti diversi.
- Con quale frequenza aggiornare la soglia? Avrò degli algoritmi pseudo deterministici che determinano il valore della soglia. Un aggiornamento continuo può portare instabilità e c'è sempre il problema di fare in modo che il sistema non diventi instabile.
- I metodi basati sulle soglie sono sempre efficaci? Dipende, di solito sono i metodi meno robusti. Anche gli attacchi con un numero basso di tentativi possono avere successo.

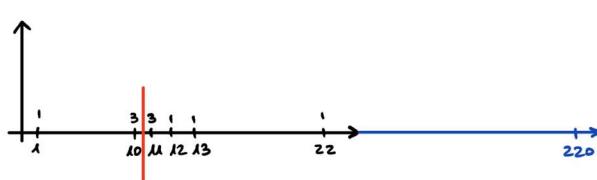
La soglia dovrebbe essere robusta:

- Soglia generata dinamicamente attraverso l'analisi dei dati
- Soglia diversa per utenti diversi
- Utilizzare le proprietà statistiche del dataset, non solo le medie: più robusti per gli outliers sono la mediana o i quartili. La media è quanto di più sensibile c'è agli outliers, cioè ai campioni diversi dalla maggior parte dei campioni.

Esempio numerico:

$$\mathbf{x} = \{1, 10, 11, 12, 11, 10, 13, 11, 10, 22\}$$

$$\bar{x} = 11,1$$



La linea rossa è la mediana. Ora se al posto del campione 22 avessimo 220 la media cambierà in maniera significativa mentre la mediana non sarà minimamente interessata dal cambiamento. Lo stesso con i quartili (disegno).

- Anomalie innescate da molteplici violazioni della soglia

## Rilevamento delle anomalie: data driven

Tutte le volte che effettuo scelte che sono data driven, e quindi che utilizzo meccanismi di machine learning e non di semplici euristiche dovrei cercare di soddisfare questi obiettivi:

- Mantenere gli errori, FP e FN, il più bassi possibile
  - Per ottimizzare le prestazioni del sistema
- I parametri del sistema IDS (diversi dalle features; i parametri sono le soglie, i quartili, ecc) dovrebbero essere facili da configurare, ottimizzare e mantenere
  - Se l'amministratore di sistema non capisce bene come utilizzare lo strumento lascia le impostazioni sempre ai valori di default e come va va
- I meccanismi di IDS si adattano ai trend dei dati
  - Trend dovuti a periodicità o stagionalità (esempio picchi di traffico che possono essere rilevati come anomali o meno a seconda di quando si verificano)
- Gli approcci utilizzati per IDS funzionano bene con dataset di diversa natura, dovrebbero essere il più generale possibili e non fare assunzioni matematiche che non sempre sono realistiche.
  - Classica assunzione: campioni distribuiti come una gaussiana singola per il singolo parametro o multi-variabile per più parametri. Questo non necessariamente è vero e avere un approccio che prescinda da questa assunzione è in generale robusto.
- Sistema efficiente in termini di risorse, adatto per il tempo reale
  - Computazionalmente efficiente
- Meccanismo spiegabile
  - Perché quell'allarme è stato lanciato o no!

Ingegneria delle features:

Quando utilizzo tecniche di machine learning o di intelligenza artificiale è fondamentale la scelta delle features da utilizzare perché ci sono alcune features che sono combinate con le altre e spesso alcune delle feature sono combinazioni lineari di altre features. Il valore aggiunto di queste feature è pari a 0 però peggiorano molto l'efficienza computazionale.

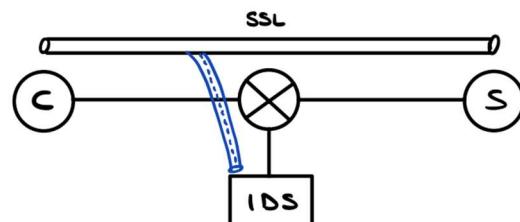
Altra cosa fondamentale: i sensori che acquisiscono dati di traffico tipicamente generano dati come se fossero serie temporali. Queste serie temporali sono "immutabili", ovvero non vengono mai aggiornate perché i dati di traffico sono quelli, non è che cambiano nel tempo. Quindi un record di una funzione di una sessione molto lunga una volta scritto rimane tale. La dimensione temporale di questi dati in alcuni casi può essere non correttamente presa in considerazione dagli algoritmi di intelligenza artificiale e questo è un problema. Questo accade perché non tutti gli algoritmi di intelligenza artificiale utilizzano la dimensione temporale all'interno dei

propri meccanismi. Soluzione: invece di dare in pasto al motore di machine learning tutti i dati insieme, posso utilizzare il concetto di località temporale, ovvero utilizzo delle finestre temporali e tengo conto del risultato finestra per finestra.

Diamo ai motori di machine learning:

- Metadati del traffico di rete: quelli che ad esempio estrae netflow o altre features che derivano dalla misura delle prestazioni (numero di pacchetti persi, latenza nel servire una richiesta di servizio). Utilizzare stateful inspection quando possibile.
- In alcuni casi, soprattutto quando ho a che fare non con il traffico vero e proprio, ma con i log di un server che eroga un servizio potrei avere a disposizione o i log o la possibilità di andare ad analizzare il traffico. Se lo faccio sul server che eroga un servizio non mi crea problemi, ma se lo voglio fare su un nodo a monte ho bisogno di tecniche che prendono il nome di deep packet inspections e che hanno bisogni di decifrare il traffico.

Questo implica che se ho una sessione SSL questa assicura sicurezza da estremo a estremo, cioè tra il client e il server. Per analizzare il payload dei dati nell'IDS come faccio? Posso analizzare solo quello che è esterno a SSL, quindi intestazione di strato 3 e 4 ma non posso osservare niente. Devo quindi fare in modo che quando diretto il traffico sia in grado di terminare la sessione SSL in modo tale da estrarre il traffico e analizzarlo. Questo può però rappresentare una vulnerabilità stessa del sistema.



(Quando abbiamo fatto i fw non avevamo visto i proxy level fw in cui la sessione tcp veniva terminata sul proxy e veniva effettuata l'ispezione dei pacchetti, in quel caso dovrebbe scrivere le funzioni di IDS insieme alle funzioni di fw perché ho il pacchetto in chiaro visto che è terminata la sessione tcp e anche la sessione SSL sul fw e quindi in quel caso è possibile ispezionare il payload del pacchetto. Altrimenti devo far girare l'IDS sul server o sul nodo terzo che però può comportare un pericolo di compromissione del servizio stesso.)

Tipi di attacchi più noti:

- Denial-of-service (dos)

- Network probe (probe): ad esempio quello effettuato da nmap per capire quali porte del sistema sono aperte o sui cui determinati host rispondono
- Remote-to-local (r2l): un utente remoto tenta di entrare dentro il sistema
- User-to-root (u2r): una volta che l'utente è entrato con credenziali di utente semplice tenta di elevare i suoi permessi a livello di root

Attacchi che combinati insieme prendono il nome di escalation: Probe => r2l => u2r: è un pattern noto. È una delle caratteristiche fondamentali di un dataset storico che viene utilizzato per insegnare la sicurezza informatica nelle reti e prende il nome di ?kdb? (lo utilizzeremo nelle prossime esercitazioni).

Features ben note:

- durata della sessione (diversa dalla dimensione temporale perché la dimensione temporale ha a che fare con l'istante di inizio della sessione più che con la durata della sessione; la dimensione temporale non sempre viene utilizzata ma è di particolare importanza)
- tipo di protocollo (tcp, udp, altri protocolli di strato 4)
- il servizio (il campo type of service del protocollo ip o la porta di destinazione dello strato 4, 80 per http, 443 per https, 53 per dns e così via)
- flag (normale o errore)
- src/dst bytes (quantità di byte scambiati in un verso o nell'altro, quindi dal client al server e viceversa)
- land (pacchetti marchiati come oggetti nell'intestazione del tcp)
- frammento sbagliato (numero di frammenti che arrivano sbagliati)
- pacchetto urgente

Features aggiuntive, relative all'accesso di servizio:

- tassi di errore relativi al servizio: ad esempio il numero di sessioni che non riesco a servire da parte del mio server o dei miei server
- % di errori di pacchetti SYN rispetto ai pacchetti REJ (rigettati): quando non riesco a stabilire la sessione tcp
- % di connessioni agli stessi servizi o a diversi servizi

Riduzione delle feature per la pulizia iniziale del dataset:

- Features engineering: rimuovere le feature strettamente legate l'una all'altra in modo da non inondare l'algoritmo con dati utili che non portano però informazione

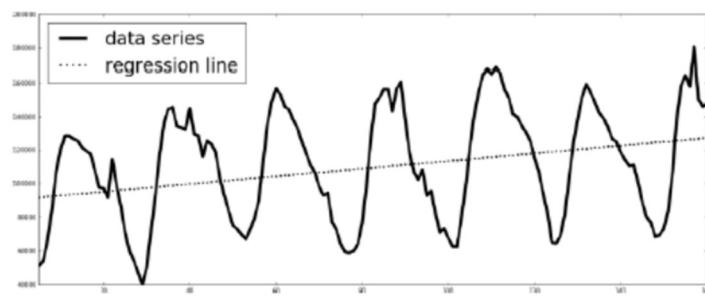
- Rimuovere le ridondanze: potrei acquisire informazioni da più sensori; se due sensori diversi mi danno la stessa informazione quell'informazione la porto una volta sola, non n volte.

La riduzione dovrebbe comportare una perdita di prestazioni minima o nulla, non devo perdere informazioni perché sennò perdo anche nelle prestazioni del sistema.

Uno degli approcci che possono essere utilizzati in particolare per andare a identificare gli attacchi DDoS o DoS sono le tecniche che vanno sotto il nome di **forecasting**, previsioni, che tipicamente sono gestite con tecniche di supervised learning (SL).

Tipicamente sono serie temporali di una o più features in cui la cosa fondamentale è l'istante temporale in cui la feature viene raccolta. Devono permettere di raccogliere i trends (sul lungo periodo), le stagioni (ripetizioni periodiche, stagionalità) e altri cicli (periodicità secondarie diverse dalla stagionalità).

Ad esempio la stagionalità può essere quella di periodo di lavoro/vacanze e all'interno di questa ci sono dei cicli tipo giorno/notte e giorni lavorativi/finesettimana.



Modelli che posso utilizzare:

- Modelli autoregressivi (ad esempio ARIMA)  
ARIMA sono modelli semplici in cui:

$$y_t = f(y_{t-k}, x) \quad \text{ARIMA}$$

$$ay_u = b_1 y_{u-1} + b_2 y_{u-2} + kx$$

y è in genere funzione lineare, x è l'ingresso.

Si parla di modello ARIMA, modello auto regressivo integrale a media nobile, quando la funzione di trasferimento ottenuta risulta avere dei poli con delle radici pari a 1 o maggiori di 1. Tipicamente quindi non è un modello stazionario proprio perché deve inserire fenomeni di tipo temporale.

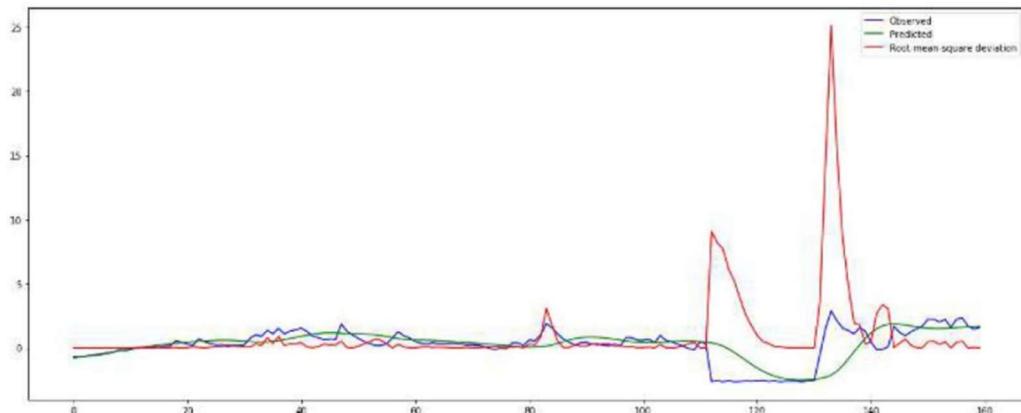
- Reti neurali, in particolare le reti neurali ricorrenti, quelle che tengono memoria (LSTM, Long Short Term Memory, sono le più utilizzate).

Proprio perché se includo la dimensione temporale al dataset e so che nel profilo di grafico di richieste di servizio ho dei pattern che tendono a ripetersi, tra cui possono emergere anche dei trend, allora ho bisogno di strumenti che siano caratterizzati dalla possibilità di tenere memoria.

LSTM hanno una memoria sia a breve che a lungo termine e sono in grado di ricordarsi profili diversi.

Quando si utilizza la previsione:

- Metrica a valore singolo nel tempo: quando ho un solo parametro (ad esempio il numero di richieste o il tempo di servizio) questi modelli permettono di stimare il valore futuro del parametro nel tempo e quindi di prevenire situazioni di anomalie. Quando ho più di un parametro potrei avere problemi a trovare un criterio decisionale efficace.
- Non adatto per il rilevamento di valori anomali. È utile per stimare le anomalie solo se il set di training NON contiene valori anomali, altrimenti, si adatterà agli inlier e agli outliers. A quel punto se ho un comportamento diverso dal trend e/o dalla stagionalità o dai cicli noti sarò in grado di rilevare le anomalie. Se invece il dataset ha le anomalie questi modelli integreranno le anomalie all'interno del modello stesso e quindi potenziali attacchi potrebbero essere scambiati per fenomeni attesi, non riuscendo a identificarli.
- Adatto solo quando emergono trend osservabili, con fluttuazione limitata. Anomalie osservate con deviazione RMS (deviazione standard: radice della media dei quadrati)



In questo esempio si vede qual è il valore osservato, quello blu, piuttosto stazionario, e qual è il valore predetto, quello verde. Nei punti in cui ho valori che sono significativamente diversi da quelli che ho normalmente la deviazione standard ha un balzo, che è uno scostamento elevato e quindi riesco a rilevare un'anomalia.

Questo funziona bene solo se ho un pattern omogeneo tipo questo (curva blu e verde molto simili), se ho un pattern con grosse fluttuazioni il profilo predetto non riesce a stare ai dati osservati e quindi ho fluttuazioni continue anche della deviazione standard tra quello predetto e quello osservato e non riesco a rilevare le anomalie.

## Anomaly Detection: Data Driven

Con l'ultima lezione eravamo arrivati a trattare le problematiche di forecasting e avevamo visto che queste possono rappresentare un valido strumento per la rilevazione delle anomalie.

Queste tecniche vanno ad identificare un certo numero di pattern e stagionalità e quindi a valutare come anomalie dei valori, in determinate istanze temporali, che si discostano in maniera significativa dalle previsioni. Ovviamente questi strumenti sono tanto più efficaci, quanto più è semplice identificare questi pattern. D'altro canto una delle principali limitazioni è che tipicamente, queste tecniche, lavorano per un solo parametro e quindi funzionano bene se si può limitare l'osservazione ad un'unica feature (ES. latenza del servizio, percentuale di risorse di calcolo effettivamente utilizzate).

Nel caso di sistemi con più parametri le tecniche appena descritte non funzionano bene e tipicamente per andare a rilevare a un'anomalia, la cosa più semplice è andare a calcolare la deviazione standard istantanea tra valori reali (osservati) e valori predetti, nel caso di scostamenti significativi si è probabilmente in presenza di una anomalia. Si utilizza la deviazione standard in quanto lo scostamento reale e la differenza tra valori osservati e predetti potrebbero non essere elevatissimi, mentre la deviazione standard può crescere in maniera significativa, in questo modo si dispone di un criterio più efficace nella identificazione delle anomalie.

Altri strumenti che possono essere più o meno efficaci, sono quelli che prendono i nomi di "strumenti statistici" come ad esempio, la media mobile dei dati e confrontata con una specifica soglia. Questi strumenti statistici sono particolarmente facili da utilizzare quando le distribuzioni dei dati sono note a priori.

Volendo fare un esempio: preso un dataset in cui i dati sono distribuiti secondo una distribuzione gaussiana e andando a normalizzare, è possibile identificare come "in-layer" (punti normali e accettabili) tutti quelli che rientrano nell'intervallo che va da -2 a 2.

### ESEMPIO

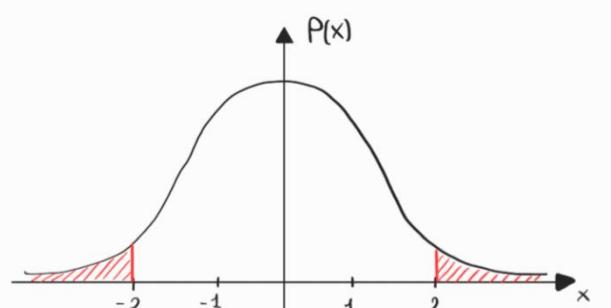
Dataset  $X = \{x_i \in R^n, i = 1, \dots, k\}$

N = Dimensionalità delle features

D = Numero di campioni

X è vettoriale.

NOTA -> Per semplicità grafica passiamo al caso monodimensionale



Se il nostro dataset può essere modellato come una gaussiana, allora possiamo "normalizzare", ovvero riportare a un dataset normale, standardizzato, con

$$y = \frac{x - \mu}{\sigma} \quad \text{con} \quad \mu = E[x], \quad \sigma = \sqrt{\text{Var}(x)}$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}} \quad p(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$$

In questo modo è possibile identificare le percentuali di punti all'interno dell'intervallo [-2;2]. Determinato questo criterio si può dire di essere abbastanza confidenti che probabilmente tutti i punti, di questa rappresentazione normalizzata, cadono tra -2 e 2 e risultano essere dei punti inlier.

Si può affermare ciò in quanto la probabilità di una distribuzione gaussiana è di avere x compreso tra -2 e 2 è all'incirca del 95%. Mentre i valori al di fuori dell'intervallo hanno solo il 5%.

In questa situazione quindi, è possibile sacrificare alcuni punti inlier, e quindi considerare alcuni punti come falsi positivi, permettere in questo modo una rilevazione più efficace degli outlier. Facendo ciò, anche se qualche anomalia comincia a manifestarsi in questi punti estremati viene comunque controllato.

Nel campo della rilevazione delle anomalie è accettabile la presenza di falsi positivi in quanto, dietro un sistema automatizzazione di rilevazione, la correzione delle anomalie avviene per mano di utente umani, il quale, prima di agire, va a controllare se l'anomalia è reale o presunta. In entrambi i casi, sia nel caso di anomalia reale, che presunta, è sempre molto importante cercare di capire perché il sistema ha lanciato l'allarme e generato un segnale di anomalia.

#### NOTA

I metodi visti suppongono l'assenza di etichette, in quanto in presenza di etichette si parla di Misused Detection e non di Anomaly Detection.

*I problemi visti sono tutti di classificazione?*

Praticamente tutti i problemi sono di classificazione, ad eccezione del forecasting, che è un mix tra predizione e classificazione, nello specifico sfrutta la deviazione dalla predizione per fare la classificazione. Utilizza entrambi gli strumenti: effettua una predizione e poi la classifica in base alla differenza tra i valori osservati e i valori predetti.

I casi in cui lo avviso funziona sono solamente due:

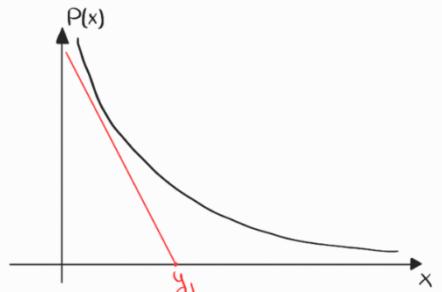
1. Il dataset utilizzato è molto pulito, quindi non sono presenti outlier;
2. Il dataset è veramente gaussiano.

Nel caso di dataset non gaussiano è ovvio che non può funzionare, in quanto si avrà una distribuzione di proporzionalità e una formula della PDF completamente diverse.

Prendendo in esempio la distribuzione in figura, in questo caso è possibile calcolare il valore medio, la deviazione standard e tutti i dati già visti. Ma in ogni caso il valore di  $\lambda$  ottenuto avrà poco senso e sarà inutile al fine prefissato.

Riepilogando, questo approccio funziona se il dataset è Gaussiano o in generale se riesco a riconoscere la mia distribuzione e se il dataset è pulito

Nell'esempio in matlab si andrà ad analizzare un dataset con distribuzione gaussiana di 1000 punti contaminato con alcuni outlier (fino ad un massimo di 4). Come già accennato, al posto della stima della media e della varianza si potra utilizzare la mediana perché di solito è più robusta.



*Come si fa ad utilizzare la mediana in questo contesto?*

La mediana si può usare come sostituto per il valore atteso, ma dato che il criterio si basa su quante volte è lontano dal valore atteso, e poiché questo dato è espresso in termini di numero di deviazioni standard, si ha la necessità di qualcosa che vada a sostituire la deviazione standard. Questo strumento deve essere robusto alla presenza di outlier e funzionante anche in cui non si abbia a disposizione una distribuzione gaussiana, si ha quindi bisogno di un tool un pochino più generale, che sarà sempre possibile tarare attraverso un certo numero di tentativi; questo sostituto della deviazione standard si chiama MAD, ed è la mediana del valore assoluto della deviazione della mediana.

$$MAD = \text{median}(|x - \text{median}(x)|)$$

con  $x$  = feature

La mediana è meno suscettibile della media e la MAD è meno suscettibile della deviazione standard agli outlier.

#### [ESEMPIO MATLAB]

I vantaggi di questo approccio sono la semplicità e soprattutto le possibilità di guardare ad una variabile per volta; è quindi un metodo alternativo, per i casi in cui non si riesce a fare delle previsioni temporali, o comunque non si non riesco a identificare pattern o ciclicità o stagionalità significative. Allora, in quel caso, se tutto risulta rumoroso e piatto, invece di andare a fare una stima del rumore è possibile usare questi altri tools. Siamo sempre parlando della singola feature, quando cioè si va a misurare un solo parametro alla volta, ad esempio il numero di connessioni al minuto, la latenza del servizio, e così via. Queste tool sono spiegabili, riproducibili e facili da configurare e mantenere nel tempo, e questi sono tutti vantaggi tutt'altro che banali, in particolare la spiegabilità permette di capire non solo quando le cose non funzionano ma anche perché ciò avviene. Alcuni casi infatti, è più importante capire come mai un problema si è verificato piuttosto che la rilevazione del problema stesso.

Come già accennato quando si è parlato del modello gaussiano, in generale, non esiste solo il modello gaussiano, ma ne esistono tante distribuzioni di probabilità note che possono essere utilizzate per andare a calcolare le code di probabilità, ovvero le code della distribuzione che possono presentare i gli outlier.

In generale, dato un dataset che non è possibile modellare a priori perché sia dispone semplicemente di un insieme numerico unidimensionale, esistono tutta una serie di test che servono a identificare qual è la distribuzione di probabilità che meglio si adatta al dataset.

Queste tecniche prendono il nome di Goodness-of-Fit First, quindi vanno a misurare la bontà del fitting, aiutano quindi a identificare un buon modello teorico, utilizzabile per andare poi a identificare le nostre anomalie. Questo permette di diminuire i tempi di lavoro in quanto si può andare ad utilizzare una funzione matematica conosciuta può decidere se un campione è normale o è una anomalia. Nel mondo reale, in alcuni casi questo è fattibile, però non sempre è facilmente identificabile una distribuzione e soprattutto questi metodi tipicamente lavorano sempre e soltanto (tranne pochissime eccezioni), in una dimensione.

*Quindi come si fa a decidere di applicare la tecnica gaussiana?*

Si applica uno di questi test al dataset per vedere se il dataset può essere ben modellato da una distribuzione gaussiana. Una volta stimata la media e la varianza, di fatto si ha il modello ed è possibile applicare questo approccio qui, oppure l'approccio basato sulla MAD, che appunto non ha bisogno del modello.

#### **NOTA**

Il modello aiuta a capire come è fatta la pdf.

*E se abbiamo più dimensioni (che è il caso più classico)?*

Se abbiamo più dimensioni le cose si complicano, nel caso in cui è ancora valido il modello gaussino ma la dimensione è superiore a 1, si può utilizzare una tecnica chiamata Elliptic Envelop Fitting (Fitting dell'inviluppo ellittico). Funziona soltanto se le feature sono gaussiane, però possono essere anche più di una, dal punto di vista della validazione grafica posso arrivare al massimo fino a 3 feature, dopodiché si devono utilizzare solo i numeri, non si può guardare quello che esce fuori, perché non c'è modo di visualizzare i dati.

*Come funziona il Fitting dell'Inviluppo Ellittico?*

È basato sul fatto che se è disponibile una stima di quanto il dataset è sporco, in quanto, questa tecnica funziona assumendo che il dataset non sia completamente pulito, è quindi presente qualche outlier. Se non si riesce ad avere una buona stima della percentuale di punti che rappresentano anomalie, allora si può provare a filtrare queste anomalie attraverso una distribuzione gaussiana multivariata.

La cosa che non è tenuta in conto da nessuna di queste tecniche è sempre la dimensione temporale. Per quanto riguarda le tecniche di prima, quelle con una sola feature come si è già detto, se la dimensione temporale è valida, allora si devono utilizzare altri approcci, tipo i modelli auto-regressivi o le reti neurali. In quanto, quelli sono in grado invece di identificare pattern periodicità stagionali e così via.

Ma nel multidimensionale la maggior parte dei tool non riescono a gestire questa situazione, perché ad esempio una configurazione nel pieno della notte per un servizio, può essere anomala, ma nel pieno della mattina può risultare normale. E quindi è sempre importante adattare i modelli ai periodi del giorno in cui questi modelli vengono applicati.

*Come funziona l'inviluppo ellittico?*

$$\rho(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_d \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_{11} & \dots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \dots & \sigma_{dd} \end{bmatrix}$$

Il vettore X rappresenta le features e che si assumono tutte descrivibili da distribuzioni gaussiane, non necessariamente incorrelate.

Quella che si ottiene è una generalizzazione della gaussiana, in cui al posto della variabile X, sarà presente x vettore (stesso ragionamento vale per  $\mu$ ).

Al posto della deviazione standard ci sarà la matrice di covarianza  $\Sigma$ , composta come segue

$$\widehat{\sigma}_{ik} = \frac{1}{n-1} \sum_{k=1}^n (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j) \quad (\text{varianza stimata o computazionale})$$

con n = numero di campioni

Quindi x è il vettore delle features, ma il numero dei campioni è n, si avrà quindi un dataset e una matrice di dimensioni differenti. Questo implica che gli elementi sulla diagonale siano le varianze delle singole features, ad esempio l'elemento  $\sigma_{11}$ , sarebbe la varianza della feature n° 1

La covarianza di due variabili X e Y è data da

$$\text{Cov}(x, y) = E[(x - \bar{x})(y - \bar{y})] \quad \text{con } y \in \mathbb{R}^d, y^T \Sigma y > 0 \quad \det(\Sigma) = |\Sigma| > 0$$

Da ciò è chiaro che questa matrice è simmetrica. In cui la diagonale rappresenta le varianze tra le singole features ed ovvio che  $\Sigma_{12} = \Sigma_{21}$ , perché invertendo solo l'ordine dei fattori nel prodotto visto prima. Questa è anche una matrice che si dice "definita positiva".

### NOTA

Una matrice si dice "definita positiva" se è una matrice se per ogni vettore  $y \in \mathbb{R}^D, y^T \Sigma y > 0$

Il caso in cui è uguale a 0 è un caso poco interessante perché se nella matrice di covarianza, si ha un caso 0, significa che è presente una dipendenza lineare nella matrice.

Se per qualche valore fosse uguale a zero, significherebbe che anche il determinante della matrice sarebbe pari a 0 e quindi la matrice non sarebbe composta da righe o da colonne linearmente indipendenti, e si cadrebbe nel caso di una dipendenza lineare, in cui almeno una feature sarebbe data dalla combinazione delle altre, rendendo ovvia l'eliminazione di quella feature.

Assumendo che il caso appena descritto non si verifichi, si può arrivare a dire che la matrice è definita positiva, e che quindi il suo determinante  $> 0$  (strettamente positivo).

Se valgono queste proprietà, è possibile generalizzare la distribuzione di probabilità di  $X$  come segue

$$P(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}[(x-\mu)^T \Sigma^{-1} (x-\mu)]}$$

con  $\Sigma$  INVERTIBILE poiché  $|\Sigma| > 0$

È chiaro che  $\Sigma^{-1}$  esiste avendo il determinante  $> 0$  ed è anche invertibile.

Per quanto riguarda la matrice  $\Sigma$  e il concetto di autovalori e autovettori, si può dire che

$$\sum v = \lambda v \quad \Phi = [v_1, v_2, \dots, v_d] \quad \text{AUTOVETTORI di } \Sigma$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_d \end{bmatrix}$$

Con  $\lambda$  autovalore e  $v$  (vettore colonna con la stessa dimensione di  $x$ ) autovettore.

Questa relazione qui si può generalizzare in forma matriciale utilizzando la matrice  $\Phi$ , che ha per colonne gli autovettori della matrice  $\Sigma$  e la matrice  $\Lambda$  diagonale, che ha sulla diagonale i valori degli autovalori. Allora, fatte queste premesse Risulta che

$$\sum \Phi = \Phi \Lambda \Rightarrow \sum = \Phi \Lambda \Phi^{-1}$$

$\Downarrow$

$$\Phi = \sum^{-1} \Phi \Lambda \quad \text{NOTA} \quad \Phi^{-1} = \Phi^T$$

$\downarrow$

$$\Phi \Lambda^{-1} = \sum^{-1} \Phi$$

$\downarrow$

$$\Phi \Lambda^{-1} \Phi^T = \sum^{-1} \Rightarrow \sum^{-1} = \Phi \Lambda^{-1} \Phi^T$$

$\Lambda^{-1}$  è una matrice diagonale invertibile poiché il determinante è  $> 0$ , in quanto tutti gli autovalori sono  $> 0$  e non c'è nessun autovalore nullo. Ogni elemento di  $\Lambda^{-1}$  sarà l'inverso del corrispondente elemento di  $\Lambda$ . Questo serve in quanto:

$$\sum^{-1} = (\underbrace{\Phi \Lambda^{-\frac{1}{2}}}_{M})(\underbrace{\Lambda^{-\frac{1}{2}} \Phi^T}_{M^T}) = M M^T \quad \text{dove} \quad M = \Phi \Lambda^{-\frac{1}{2}}$$

Lo posso esprimere come prodotto di una matrice per le sue trasposte

$\Downarrow$  esponente di  $P(x)$  con  $x \in \mathcal{V}$  ( $= \mathcal{X} \times \mathcal{Y}$ )

$$(x - \mu)^T M M^T (x - \mu)$$

$$= [M^T (x - \mu)]^T [M^T (x - \mu)] =$$

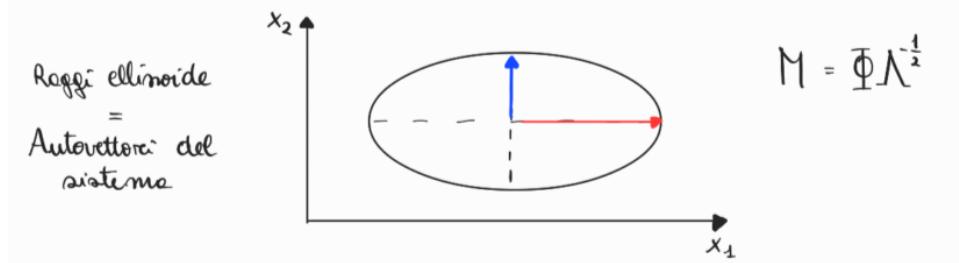
$$= |M^T (x - \mu)|^2 \rightarrow \text{nello spazio } d\text{-dimensionale}$$

$$|\dots|^2 = \text{costante} \text{ è un ellisoido}$$

I due vettori  $((x - \mu)$  e  $(x - \mu)^T)$  sono di fatto l'uno la versione trasposta dell'altro

Questa rappresentazione è particolarmente interessante perché ha a che fare con il concetto di inviluppo. Ponendo infatti il termine  $|M^T(x - \mu)|^2 = \text{costante}$ , questo rappresenta un ellissoide nello spazio bidimensionale. Per questo motivo si parla di fitting dell'Inviluppo Ellittico.

Si utilizza l'ellissoide in quanto i raggi dell'ellissoide corrispondono agli autovettori del sistema



$M$  è quindi strettamente legato alle auto autovettori del sistema.

Riepilogando, assunto un dataset gaussiano ciò che si ottiene è che la rappresentazione di questo dataset nello spazio, può essere vista come racchiusa all'interno di un ellissoide.

#### *Come mai ciò è importante l'Anomaly Detection?*

Assumendo di avere un ellissoide ben dimensionato, le anomalie saranno rappresentate dai punti che rimarranno fuori dall'ellissoide in quanto tutti i punti della distribuzione si troveranno all'interno. Di conseguenza il target è andare a definire bene i confini dell'ellissoide.

#### *Come si definisce l'ellissoide?*

Come già visto, i raggi dell'ellissoide sono tanti quanti sono le proprie dimensioni, e sono dati dagli autovettori del sistema. Ciò non basta a definire l'ellissoide in quanto c'è bisogno degli autovalori, per questo all'interno di  $M$  è presente anche un legame con gli autovalori. La funzione principale degli autovalori è di fatto quella di strecciare (*to stretch*) gli autovettori. Ripensando alla relazione fondamentale tra una matrice e il suo autovalore, ovvero che una matrice moltiplicata per l'autovettore è uguale a  $\lambda$  per l'autovettore stesso; quindi di fatto l'effetto che ha passare l'autovettore in una matrice è equivalente a strecciarlo (*to stretch*) di una quantità pari al suo autovalore corrispondente.

E infatti si ha bisogno, per andare a dimensionare l'ellissoide, sia degli autovettori che degli autovalori, perché gli autovettori danno la forma dell'ellissoide, ma gli autovalori dicono di quanto bisogna scalare ogni ramo; ovvero quanto l'ellissoide deve essere grande.

Tra l'altro, il determinante della nostra matrice di covarianza è un indicatore del volume dello spazio multidimensionale che racchiude i punti della distribuzione. Quindi ciò che bisogna fare è andare a selezionare la matrice di covarianza a determinante minimo che riesce a racchiudere tutti i punti di interesse del dataset. Per capire quali sono i punti di interesse c'è bisogno di avere un'idea, almeno a livello teorico, di quale percentuale del dataset rappresenta degli outlier.

Si va allora a calcolare la matrice di covarianza su varie configurazioni del dataset, in base a questo si va a scegliere la matrice di covarianza e a generare l'ellissoide, che contiene tutti i punti di interesse, ad esempio il 99 o 99,5% dei punti, ma con il determinante minimo, perché se il determinante è molto grande, significa che l'ellissoide potrà essere anche ben fatto ma così grande andrà a prendere tutti i punti, mentre l'ellissoide deve essere aderente alla nuvola di punti.

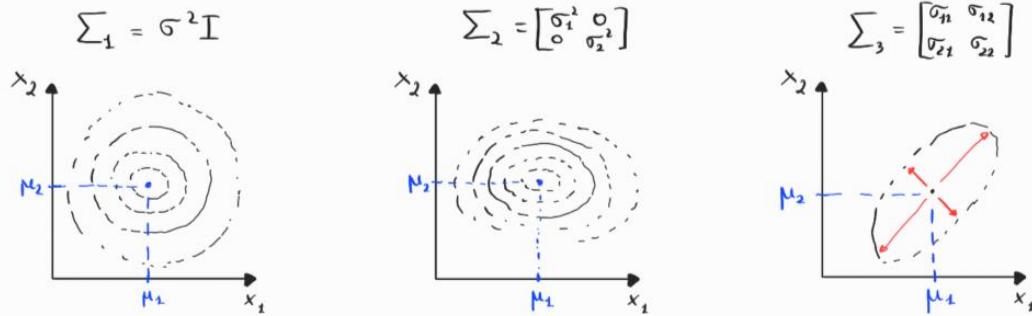
Questo approccio, ovviamente, funziona solo se le features sono gaussiane (possono essere sia gaussiane indipendenti sia gaussiane correlate). Nel caso di features non solo indipendenti l'una dalle altre, ma anche distribuiti allo stesso modo, cioè con stessa media e stessa varianza (oppure la media possono anche non avercela, importa la varianza).

Quindi, ad esempio, avendo

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \underline{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

Come è fatto l'ellissoide?



### CASO 1

Siccome la varianza è stessa, l'ellissoide collassa e diventa un cerchio, e quindi si devono allargare queste circonferenze fino a quando non si raggiungono tutti i punti. Questo ovviamente dipende dal determinante della matrice di covarianza, più si spancia più si raccoglie, che non sempre è una cosa positiva.

### CASO 2

Avendo la varianza diversa, il grafico sarà un ellisse.

### CASO 3

Matrice di covarianza generica in cui, in teoria, i quattro elementi sono diversi da zero. Sapendo che se il coefficiente di covarianza è positivo, significa che le feature sono correlate positivamente, cioè se una aumenta, aumenta anche l'altra. Se invece il coefficiente di covarianza è negativo, significa che aumentano in maniera inversamente proporzionale, quando una aumenta l'altra diminuisce.

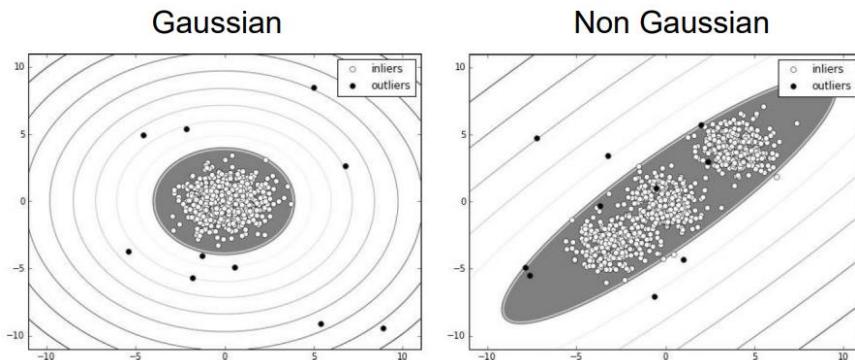
Supponendo che siano correlate positivamente, l'ellissoide sarà inclinato nella direzione della bisettrice, del primo e del terzo quadrante è dovuto principalmente a correlazione positiva, se invece nell'altro verso (secondo e quarto quadrante) si tratta di correlazione negativa. Quindi, se le due variabili sono gaussiane, allora si riesce effettivamente a fare una rilevazione.

L'ellisse è in generale definita non dalla distanza euclidea, che equivale vedere l'ellisse come la combinazione lineare della distanza rispetto ai due fuochi, in alternativa alla distanza euclidea si può utilizzare la distanza di Malanobis, ovvero la distanza che definisce il luogo dei punti che formano l'ellisse.

DISTANZA di MALANOBIS

$$(\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu})$$

Nell'esempio di destra (figura sotto) evidentemente non funziona perché si vedono all'interno dell'ellisse delle regioni non omogenee ma piuttosto dei cluster di punti. Risulta abbastanza immediato che questa rappresentazione, nonostante inglobi la percentuale desiderata dei punti, non rappresenta un buon fit per il dataset analizzato. Si vede infatti che parecchi degli outlier sono caduti all'interno della zona di normalità e allo stesso tempo degli inlier sono stati classificati come outlier.



Si può quindi concludere che questo approccio è sicuramente funzionante, a patto che le features possano essere approssimate con delle gaussiane.

La cosa interessante è che, oltre alla gaussianità, non c'è bisogno di altre informazioni. In altre parole, non c'è la necessità né di dover gestire una feature singolarmente, né di aver bisogno di un dataset pulito. Si potrà, quindi, lavorare anche con un dataset contaminato e non etichettato, ricadendo quindi nelle classiche condizioni dell'Anomaly Detection. L'unica cosa di cui bisogna essere sicuri è che le features possano essere modellate complessivamente come una distribuzione gaussiana multivariata, altrimenti i risultati ottenuti potrebbero non essere soddisfacenti.

Un ulteriore cosa fondamentale è data dal fatto che questo approccio lavora bene su features che assumono valori tipicamente continui, mentre si presta male a tutti quei casi in cui le features assumono valori binari o categorici. Un esempio di valore categorico potrebbe essere il valore campo "Next Protocol" del protocollo IP che indica quale è il protocollo trasportato dal payload, i cui valori possono essere, ad esempio TCP, UDP, ICMP oppure, nel caso di tunnel, GRE. Un altro esempio potrebbe essere dato dal campo che indica la versione del protocollo IP. Quindi, se una features assume un numero limitato di valori si dice che è una features di tipo categorico. Non è importante come rappresento queste variabili in quanto sicuramente non assumono valori continui, e mal si presterebbero ad essere modellata con una variabile gaussiana.

Quindi, in tutti quei casi, in cui le features assumono valori da un insieme finito, le approssimazioni gaussiane o in generale le modellizzazioni che fanno uso di PDF di tipo continuo non si adattano bene e quindi nasce la necessità di utilizzare degli approcci più generali.

## Metodi basati sul concetto di Densità

Anche in questo caso si parla di metodi UL (Unsupervised Learning) e in letteratura, tipicamente, sono indicati il K-Means e il KNN.

Tipicamente i metodi supervisionati servono a trovare delle similitudini all'interno dei dataset e a raggruppare i punti che condividono queste similitudini, cioè a creare dei cluster.

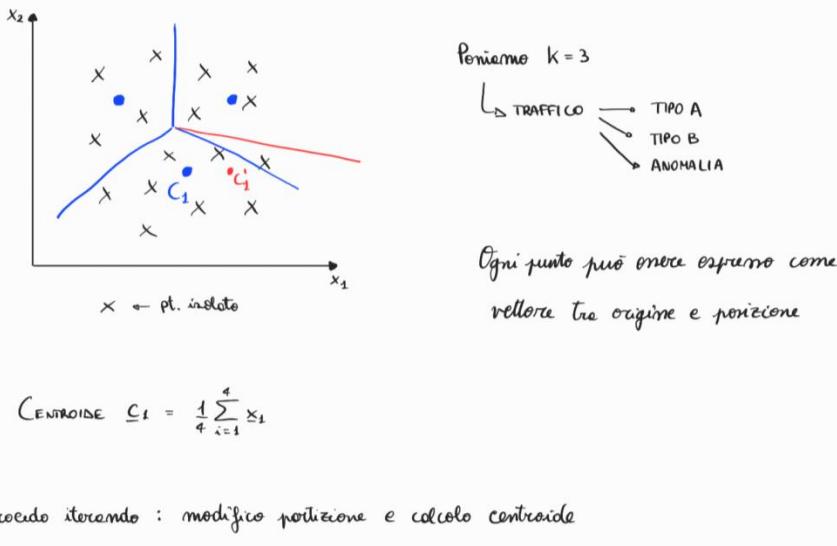
Come posso creare questi cluster?

### K-Means

L'approccio K-Means si basa sul fatto che si definisce a priori il numero di cluster, che è proprio pari a K.

Quindi prendendo il dataset e assumendo che le sue features siano definite su un insieme bidimensionale, supponendo K = 3 si ha

K-MEANS definisce il numero di cluster (=k)



Procedo iterando : modifco posizione e calcolo centroide

Ci si aspetteranno quindi 3 cluster:  $C_1, C_2, C_3$ .

La prima cosa da fare è partizionare le features in 3 sotto insiemi, che per definizione di partizione popolano lo spazio e hanno intersezione nulla.

Se si è in possesso di qualche indicazione per scegliere la partizione iniziale tutto funziona meglio e ragionevolmente, il risultato finale sarà migliore, in alternativa la partizione può essere completamente casuale.

Questi dati si possono vedere come elementi di uno spazio vettoriale, in questo caso bidimensionale, e ogni punto può essere rappresentato come un vettore.

All'interno di questi 3 cluster iniziali, si può definire il cosiddetto "centroide", quindi se ne cluster  $C_1$  ci sono 4 punti, si ha che il centroide  $c_1$  è pari a

$$c_1 = \frac{1}{4} \sum_{i=1}^4 x_i$$

Se fino ad ora si è proceduto completamente a caso, adesso, per ogni punto dello spazio misuro la distanza tra ciascun centroide e sulla base di quello più vicino si riassegna il punto a un cluster potenzialmente diverso.

Si va avanti ricalcolando il centroide e aggiornando il cluster fino a quando non ci sono più variazioni o le variazioni sono al di sotto della soglia di tolleranza. Tipicamente l'algoritmo si ferma quando per una iterazione non ci sono più modifiche, converge sempre ma non sempre necessariamente all'ottimo.

Nel caso in cui si posseggano delle indicazioni per la creazione dei cluster, che prendono il nome di Euristica, è possibile scegliere in maniera opportuna i cosiddetti pseudo-cluster iniziali, permettendo al sistema di funzionare bene.

All'interno di questo meccanismo, che serve ad identificare i cluster e non le anomalie, quello che si vede è che ci sono tutti i punti che sono abbastanza vicini ad un punto, che è il centroide del cluster.

*Come potrei classificare l'anomalia in questo modo?*

L'anomalia potrebbe rappresentare un cluster vero e proprio, che però è un caso poco realistico, oppure ci sono dei punti che risultano appartenere a un cluster, ma essere molto lontani dal proprio centroide. Per questo motivo in particolare, questi metodi prendono il nome di metodi basati sulla densità.

Ovviamente anche i punti isolati vengono mappati per forza su un cluster anche se effettivamente molto lontani.

## KNN

È un approccio che di fatto è senza training, quindi per ogni punto si decide l'etichetta sulla base dell'etichetta che si è decisa per i punti circostanti, diciamo che vado a maggioranza. Prendo i K punti più vicini e sulla base delle loro etichette decido l'etichetta e così vado avanti.

Anche in questo caso il concetto da utilizzare è il concetto di densità.

Che cosa significa densità? Quanto è distante un punto dai suoi K vicini? Qual è la distanza media di un punto dai suoi 7 vicini? Qual è la distanza media di ciascuno dei 7 vicini dai propri 7 vicini?

Tutte queste considerazioni sono state formalizzate in un approccio che riprende un po' sia il KNN sia il K-Means ma in una maniera leggermente diversa che prende il nome di LOF o Local Outlier Factor.

**RECAP:** l'altra volta abbiamo visto la teoria degli approcci basati sul clustering o in generale gli approcci non supervisionati. È stata data una brevissima panoramica su come funziona k-NN, quindi come si costruisce un cluster.

## Anomaly detection: data driven

- Density-based methods:
  - UL clustering methods, such as k-means
    - Density-based methods are well suited for **high-dimensional datasets**
    - Difficult to deal with using the other classes of anomaly detection methods
  - The main idea behind all of them is to form a cluster representation of the training data
    - Under the hypothesis that **outliers/anomalies be in low-density regions** of this cluster representation
    - Robust to outliers in the training data
      - Such instances will likely also be found in low-density regions
  - k-NN can be used for this purpose as well
    - Points with high distance to the k-th nearest neighbours
  - LOF is a nice approach as well

## Rilevamento delle anomalie: Algoritmo LOF

(**Premessa:** Come abbiamo detto, le anomalie sono punti abbastanza diversi dalle istanze normali o benigne. Questo porta a pensare che in una rappresentazione nell'iperspazio, un campione che rappresenta un'anomalia sia in una regione a bassa densità, ovvero circondato da pochi altri campioni e abbastanza lontano da tutto il resto. Però, non esiste una definizione univoca per definire una regione a bassa densità.)

La principale forza (e anche debolezza) dell'algoritmo LOF (Local-outlier factor) è proprio quella di essere un algoritmo non parametrico, dove non si vanno a definire delle soglie per identificare la densità e si va a decidere sulla base delle informazioni locali se un determinato campione può essere considerato un'anomalia. Questo suo punto di forza è anche la sua principale debolezza perché anomalie locali in punti particolari dello spazio spesso sono semplicemente punti a bassa densità, che rilevate come anomalie comporterebbe a dei falsi positivi.

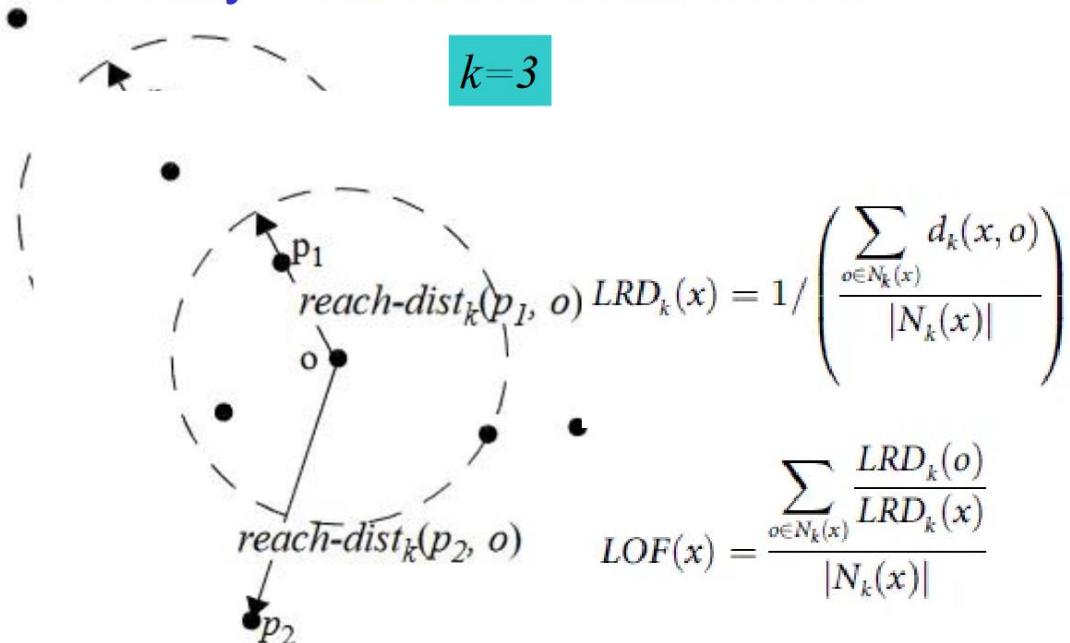
L'algoritmo LOF agisce come segue. Per ogni campione o test point (che chiameremo A) fissiamo un parametro k (numero di vicini) e andiamo a valutare quella che prende il nome di **distanza di raggiungibilità o reachability-distance** calcolata come il

massimo tra la distanza normale (da A a B) e la distanza del terzo vicino (terzo vicino con  $k=3$  ad esempio) del destinatario (B). A questo punto si definisce la densità di raggiungibilità locale, andando a calcolare il valore medio su tutti i  $k$  vicini della distanza di raggiungibilità, ovvero facendola media della distanza e poi l'inverso. A questo punto dopo aver calcolato la densità locale, per capire se il mio punto è un'anomalia o no, vado a confrontare la densità locale di A con quella dei  $k$  vicini, in modo da verificare se sono in un'area confrontabile con la loro o meno. Quindi, vado a calcolarle il rapporto tra la mia densità normale e la densità locale di tutti i miei  $k$  vicini (i vicini devono essere almeno pari a  $k$ ), faccio la media di questo rapporto e se la media viene nettamente inferiore a uno è un punto normale, se è più o meno intorno a uno possiamo considerarlo ancora nel bordo e se invece è nettamente maggiore di 1 si classifica come **anomalia (outlier)**.

## Anomaly detection: data driven

- Local-outlier factor (LOF), for each test point A:
  - Evaluate the reachability-distance from the  $k$ -neighbour
    - $d_k(A,B) = \max [ d(A,B), k\text{-dist}(B) ]$
    - Just a variation of the typical distance between 2 points to increase smoothness
  - Evaluate the local reachability density (lrd) of A towards its  $k$ -neighbours
    - $\text{lrd}(A) = 1 / \text{average} ( d_k(A) )$
  - Evaluate the ratio
    - $\text{LOF} = \text{average} ( \text{lrd}(A) / \text{lrd}(k\text{-neighbours of } A) )$ 
      - If ratio  $< 1 \rightarrow$  A inlier
      - If ratio  $\approx 1$  likely A is an inlier
      - If ratio  $>> 1 \rightarrow$  A is an outlier

## Anomaly detection: data driven



Si possono utilizzare delle **strategie di insieme o di ensamble** che consistono nell'applicare un criterio a maggioranza, ovvero applico il mio LOF con il campo K=10 e vedo se un è un'anomalia o meno, sucessivamente, vado ad applicare un LOF con K=11,12, 13 fino a 15 e in base alla maggioranza decido se quel punto è un'anomalia o meno (Ad esempio con K=10 e 11 il punto è un 'anomalia ma con K=12,13 e 14 no, allora vince la maggioranza e si prende il NO). Un difetto di questo approccio è che essendo basato sul concetto di distanza, assume implicitamente che i cluster abbiano una forma sferica nell'iperspazio e se il cluster ha delle forme piuttosto diverse dalle sfere l'approccio non funziona benissimo. Un altro difetto importante è che i bordi di cluster che sono vicini ma diversi tendono a creare delle difficoltà perché a volte un punto che è membro di un altro cluster, risulta un'anomalia per il cluster vicino (il numero di falsi positivi aumenta).

Ancora una volta i dati devono essere di tipo numerico (non categorie) e devono essere abbastanza continui, quindi nel caso dei sistemi IT, i dati che possono essere trattati facilmente con un approccio di questo tipo sono: le latenze, le percentuali di pacchetti o chiamate perse, durata della sessione, numero di pacchetti della sessione.

Invece altri parametri, come ad esempio: protocollo di strato di trasporto oppure flag del TCP molto spesso utilizzati, essendo non continui non vanno bene per questo approccio.

## Anomaly detection: data driven

- LOF only relies on its direct neighborhood and the score is a ratio mainly based on the k neighbors only
- In respect to parameter selection, **k-value** is crucial
  - It is possible to use an ensemble strategy for computing LOF score with different k-values
  - The min value of k can be 10 and max can be dependent on the data 'clusters'
- I may generate a lot of false positives:
  - It detects both local and global anomalies, but only the second ones are interesting
- Borders of close clusters (assumed **spherical**) difficult to detect
- I requires data to be numerical and vary continuously

50

LE SLIDE DA 51 a 56 LE FA DOPO GLI ALBERI DEL PACCO MACHINE LEARNING

## ALGORITMI AGGIUNTIVI DI MACHINE LEARNING

In generale ma soprattutto in questo campo, l'uso dell'intelligenza artificiale è fondamentale visto l'enorme quantitativo di dati da gestire. Questi dati tipicamente vengono dalle consultazioni dei database (ad esempio risultati delle attività degli utenti su Internet) o da altri campi (come: biologia, medicina, ingegneria, etc.).

### Perché se abbiamo tanti dati c'è bisogno di tecniche di machine learning?

Perché abbiamo bisogno di applicazioni che non riusciamo a programmare in maniera esplicita.

### Why do we need Artificial Intelligence (AI)?

- **Database mining** – large datasets from growth of automation/web:
  - Web click data
  - Medical records
  - Biology
  - Engineering
    - Predictive maintenance, [IT Operations](#), etc
  - Finance
- Processing these datasets = **build applications that cannot be explicitly programmed**

### Tipi di Problemi

I tipi di problemi che potremmo affrontare possono essere:

- **Supervised learning;**
- **Unsupervised learning;**
- **Altri schemi.**

La cosa fondamentale è ricordarsi che quando parliamo di problemi supervisionati o non supervisionati, la classificazione non dipende tanto dall'approccio quanto dall'informazione che abbiamo a disposizione. Noi possiamo fare una classificazione o una predizione tramite un approccio supervisionato, solo se qualcuno ci ha dato delle etichette che ci serviranno per l'addestramento dei nostri algoritmi. Senza addestramento, devo cercare di muovermi nella confusione dei dati, quindi andare a cercare le strutture ricorrenti (unsupervised learning).

Altrimenti esistono altri approcci come l'apprendimento con rinforzo (**Reinforcement learning**). Ad esempio, questi algoritmi sono molti utilizzati nei sistemi di controllo ma in senso lato, per esempio, se consideriamo gli algoritmi di instradamento sui sistemi molto grandi, posso utilizzare tecniche di reinforcement learning per decidere dove far passare i miei flussi e nel caso abbia una ricompensa positiva (ad esempio latenza bassa del flusso) tenderò a farcelo passare nuovamente, altrimenti se la latenza è elevata non lo farò passare più.

## Types of Problems

### Supervised learning:

- The correct answers are given in advance for training
  - Classification or prediction

### Unsupervised learning:

- Find structures in the dataset → data clustering

### Other schemes:

- Semi-supervised learning
- Reinforcement learning
- Recommender systems

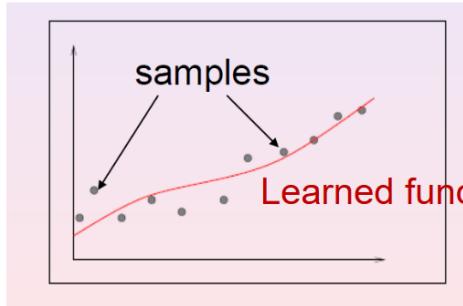
## Due tipi di apprendimento supervisionato

Per quanto riguardo il supervised learning, esistono 2 macro-approcci:

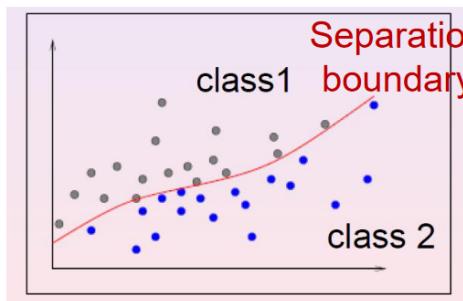
- ❖ **Classificazione:** che cerca di creare una superficie di separazione tra classi diverse. La classificazione può essere binaria o n-aria (in cui ci saranno più superfici che separano) e l'obiettivo dell'apprendimento è identificare questa linea di demarcazione;
- ❖ **Regressione:** l'obiettivo è quello di trovare un modello che ci permetta di predire il comportamento futuro del nostro sistema. Sulla base di un certo x (input) predire l'output, quindi creare un modello di input-output.

In questo corso quelle che ci interessano maggiormente sono le tecniche di **classificazione**, sia per quanto riguarda gli approcci supervisionati che per i non supervisionati. Noi utilizzeremo gli approcci non supervisionati per fare **Anomaly detection**.

## Two kinds of Supervised Learning



- Regression: Learn a **continuous input-output mapping** from a limited number of examples



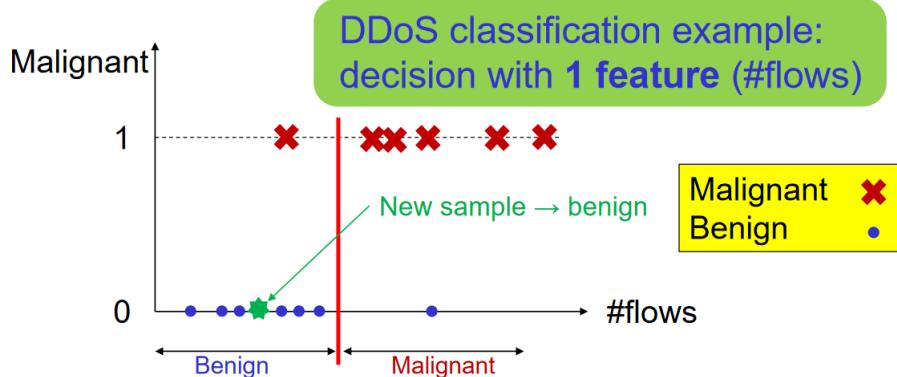
- Classification: outputs are discrete variables (category labels)
- Learn a **decision boundary** that separates one class from the other

## Supervised learning

Questo è un esempio molto semplice in cui abbiamo un solo feature (#flows) su cui prendere la nostra decisione e supponiamo che il nostro target è quello di definire se è in corso un attacco di DOS e la nostra feature è il numero di flussi che arrivano nell'unità di tempo. A seconda del numero di flussi, io andrò a dire se c'è o meno l'attacco. L'algoritmo servirà per discriminare una regione in cui decido che non c'è l'attacco e l'altra regione dove c'è.

## Supervised learning

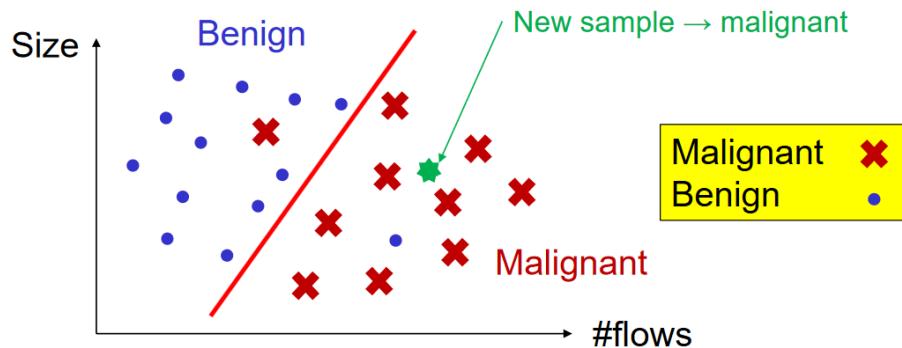
- Given a set of training inputs and corresponding outputs (**correct answers**), produce the “**correct**” outputs for the **new inputs**
- Possible applications: **classification** or **regression**



Lo stesso esempio lo posso avere quando il numero di features su cui prendere la decisione cresce. In questo esempio non ci sono solo il numero di flussi (#flows) ma anche la dimensione di ciascun flusso (size) e una possibile superficie di separazione potrebbe essere questa retta rossa.

## Supervised learning

DDoS classification example:  
decision with **2 features** (size and #flows)

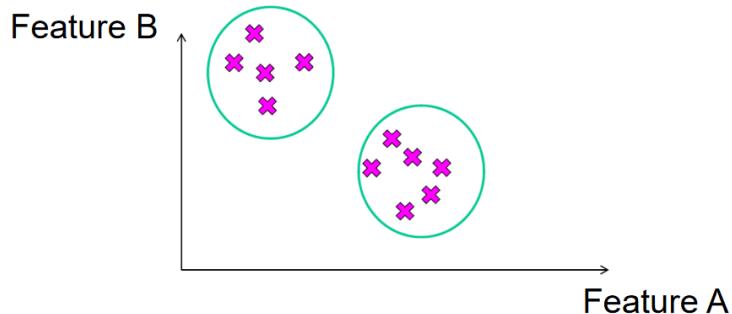


## Unsupervised learning

Per quanto riguarda l'apprendimento non supervisionato mancano le etichette. Posso utilizzare questo approccio per scoprire tre tipi di informazioni:

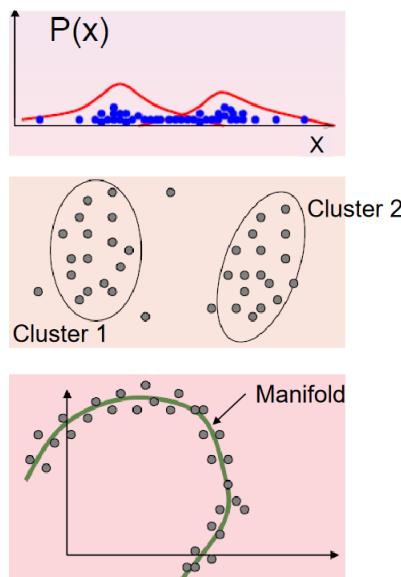
1. posso cercare di clusterizzare (raggruppare i dati) in modo tale applicare un'etichetta ad ogni cluster;
2. scoprire delle irregolarità all'interno dei dati;
3. andare a definire in maniera empirica le densità dei dati stessi.

# Unsupervised Learning



- Given only inputs as training (**without correct answers!**), find structures in the world:
  - Discover clusters
  - Discover manifolds
  - Characterize the areas of the space to which the observed inputs belong

# Unsupervised Learning



- Density Estimation: find a function  $f(x)$  approximating the probability density of  $x$ ,  $P(x)$ , as well as possible
- Clustering: discover “clumps” of points
- Embedding: discover low dimensional **manifold** or **surface** near which the data lives

**Approfondimento:** Una tra le prime applicazioni in assoluto del clustering è stata quella di John Snow (medico londinese non del Trono di Spade), che si accorse dell'esistenza molto forte della correlazione tra i pozzetti inquinati e i bozzetti inquinati e i cluster di colera.

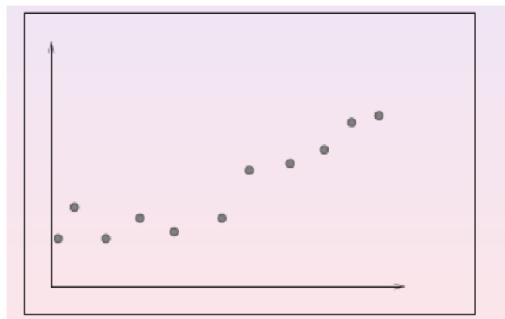
## Perché è difficile apprendere?

È difficile apprendere perché ci sono tanti possibili risultati (in teoria infiniti) che possono soddisfare la nostra relazione, ma non necessariamente sono tutti giusti perché noi facciamo l'apprendimento su un certo numero di punti, ma il nostro risultato dovrebbe soddisfare anche i punti che non vediamo.

Allora in questo diciamo in questo caso la soluzione più semplice è quella del filosofo medievale occam che dice: **quando ci sono tante soluzioni che sembrano funzionare, non bisogna aggiungere ciò che non è richiesto.**

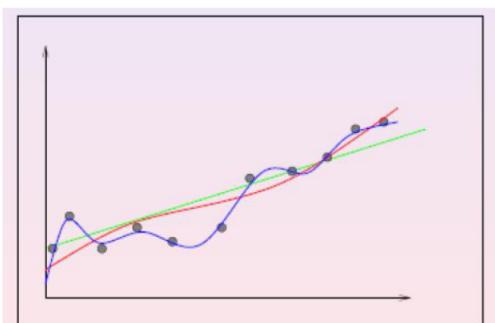
## Why Learning is difficult?

- Given a finite amount of training data, you have to derive a relation for an infinite domain
  - In fact, there is an infinite number of such relations



## Why Learning is difficult?

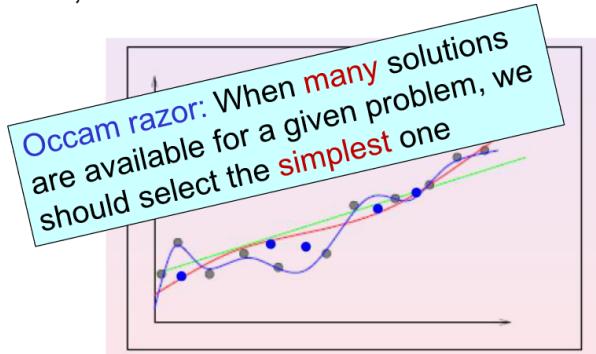
- Given a finite amount of training data, you have to derive a relation for an infinite domain
  - In fact, there is an infinite number of such relations



- Which relation is more appropriate?

## Why Learning is difficult?

- Given a finite amount of training data, you have to derive a relation for an infinite domain
  - In fact, there is an infinite number of such relations



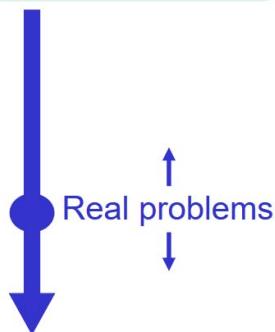
- ... the hidden test points...

## Principio Generale di Classificazione

In generale il principio guida del machine learning è che noi dobbiamo spostare una manopola verso sopra o verso sotto a seconda della nostra situazione reale. Maggiori sono le informazioni che conosciamo, più è semplice risolvere il problema, meno conosciamo e più è difficile risolvere il problema.

## Classification general principle

A lot is known → “easier”



Little is known → “harder”

Quindi possiamo avere più situazioni:

- **Teoria bayesiana della decisione:** in cui conosciamo già le curve di probabilità e dobbiamo solo guardare il nuovo campione e decidere come classificarlo;
- **L'uso del machine learning per la stima bayesiana dei parametri:** conosciamo la forma delle nostre densità di probabilità, ma non conosciamo i

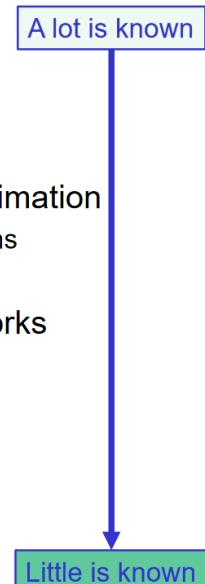
parametri e quindi utilizziamo la teoria della stima per andare a stimare i parametri e poi applichiamo comunque dei criteri teorici dei modelli;

- **Reti neurali o funzioni con curve di separazione di tipo lineare:** in questo caso sappiamo come è fatta la funzione di separazione e dobbiamo semplicemente calcolarne i parametri. Quindi dobbiamo andare a stimare i parametri dalla superficie di separazione e non più dal modello che descriviamo.
- **Metodi non parametrici:** senza nessuna distribuzione di probabilità ma con le etichette dei dati;
- **Unsupervised Learning and Clustering:** nessuna distribuzione di probabilità e nessun'etichetta dei dati.

Vediamo come scendendo da sopra a sotto le cose diventano sempre più difficili. Le cose che vediamo in questo corso, si classificano sulla parte medio-bassa.

## Classification overview

1. Bayesian Decision theory (rare case)
  - Know probability distribution of the categories
  - Do not even need training data
  - Can design optimal classifier
2. Machine Learning and Bayesian parameter estimation
  - Need to estimate Parameters of probability distributions
  - Need training data
3. Linear discriminant functions and Neural Networks
  - The shape of discriminant functions is known
  - Need to estimate parameters of discriminant functions
4. Non-Parametric Methods
  - No probability distribution, labeled data
5. Unsupervised Learning and Clustering
  - No probability distribution and unlabeled data



## Alberi Decisionali

Gli alberi decisionali sono degli algoritmi supervisionati (ma che hanno bisogno delle etichette) che permettono di classificare i dati attraverso una sequenza di domande e sulla base delle risposte fornisce una classificazione del campione. Le domande non sono tutte preordinate, ovvero, data una domanda, la prossima dipende dal risultato della domanda precedente (struttura ad albero con delle biforcazioni). Questi algoritmi sono particolarmente utili perché possono essere utilizzati per i dati **non metrici (nonmetric data)** che sono una tipologia di dato molto diffusa nel campo dei sistemi e servizi e delle reti. Al contempo, questi algoritmi non funzionano bene con

attributi di tipo continuo perché ogni volta che io faccio una domanda devo avere un numero finito di risposte (ogni risposta corrisponde ad un ramo).

Posso utilizzare gli alberi se ho dei dati continui solo se viene fatta un'operazione di pre-elaborazione in cui definiamo dei range di validità. Se ad esempio ho un attribuito che varia in maniera continua tra 1 a 10 posso andare a definire un numero di intervalli ognuno con la sua etichetta. La dimensione degli intervalli non deve essere necessariamente la stessa e questa dimensione viene scelta in base all'osservazione dei dati (**Competenza di dominio**).

**IMPORTANTE!** Uno dei vantaggi fondamentali degli alberi è la loro forte spiegabilità che permettono all'utente di capire l'algoritmo navigando a vista l'albero stesso.

## Decision trees

- Intuitive to classify a pattern **through a sequence of questions**
  - Next question depends on the answer to the current question
- Particularly useful for **nonmetric data**
  - The answers could be
    - yes/no,
    - true/false,
    - property  $\in$  set\_of\_values
- A quite common choice for **network scenarios**, where discrete values are more common

**Esempi di come potremmo scegliere la classificazione:** Un esempio d'interesse per questo corso potrebbe essere come classifichiamo il numero di porta di destinazione di un pacchetto, utilizzando il numero di porta destinazione come una feature. Alcune scelte potrebbero essere:

1. Potremmo fare una classificazione di tipo binario, solo che noi sappiamo che tutte le porte sotto la 1024 (sono well-known) sono assegnate per servizi noti.
2. Potremmo fare una classificazione in due sottoinsiemi del tipo: sotto la 1024 e sopra la 1024 però avremo insiemi chiaramente sbilanciati, perché nel primo caso ho circa 1000 elementi e nell'altro 64.000.
3. Potremmo avere una classificazione più articolata dove andiamo raggruppare un certo numero di porte che chiameremo web (80,443) o un certo numero di porte utilizzate per il management oppure le porte relative ai protocolli di posta elettronica e così via.

**Chi ci dà la risposta dell'albero sull'identità della nostra istanza?**

L'identità dell'istanza viene data dal nodo **foglia**. Quindi noi dobbiamo navigare da un nodo radice che è comune per tutti, fino ad un nodo foglia che avrà l'etichetta che ci classificherà il campione.

La cosa importante è che non necessariamente devo attraversare tanti nodi quante sono le features per essere classificato. Un'altra cosa importante è costruire bene un albero e per farlo bisogna andare ad eliminare i dati inutili, ovvero che non portano informazioni utili nel processo decisionale per la classificazione.

## Decision tree representation

- Decision trees classify **instances** by sorting them down the tree from the **root node** to some **leaf node**
  - each node in the tree specifies a **test** of some **attribute** of the instance
    - each branch descending from that node corresponds to one of the possible **values** for this attribute
  - **the leaf provides the classification of the instance**

Da adesso in poi faremo riferimento a questo dataset piccolissimo con 14 istanze e etichetta sulla colonna finale. Dato un flusso di traffico caratterizzato da quattro attributi: protocollo di trasporto (protocol), quantità dati scambiati sul flusso (data size), porta di destinazione (dest port) e indirizzo IP di destinazione (dest ip). L'obiettivo è quello di sapere se il traffico che sto osservando è traffico di gestione o traffico utente. Questo ci viene detto dall'etichetta Management Traffic.

## Training set: example



Event	Protocol	Data size	Dest Port	Dest IP	Mgmt traffic
E1	TCP	Large	Well-known	Internal	No
E2	TCP	Large	Well-known	External	No
E3	ICMP	Large	Well-known	Internal	Yes
E4	UDP	Mild	Well-known	Internal	Yes
E5	UDP	Low	Other	Internal	Yes
E6	UDP	Low	Other	External	No
E7	ICMP	Low	Other	External	Yes
E8	TCP	Mild	Well-known	Internal	No
E9	TCP	Low	Other	Internal	Yes
E10	UDP	Mild	Other	Internal	Yes
E11	TCP	Mild	Other	External	Yes
E12	ICMP	Mild	Well-known	External	Yes
E13	ICMP	Large	Other	Internal	Yes
E14	UDP	Mild	Well-known	External	No

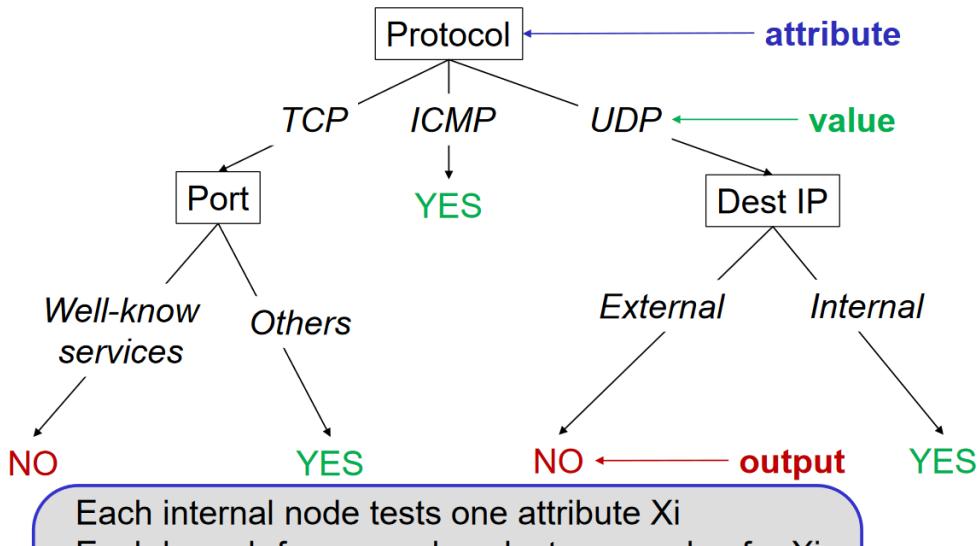
21

Adesso andiamo a costruire il nostro albero sulla base del dataset in figura sopra. Ci aspettiamo di costruire un albero di questo tipo (figura sotto).

In cima all'albero sceglierò un attributo che sarà associato al nodo radice, quindi, ogni nodo testa un attributo e sulla base del numero di valori che l'attributo può prendere, avrà un certo numero di rami in uscita. Quindi ogni nodo agisce da biforcazione.

L'unico nodo che non agisce da biforcazione è la foglia che ci fornisce il risultato della nostra classificazione.

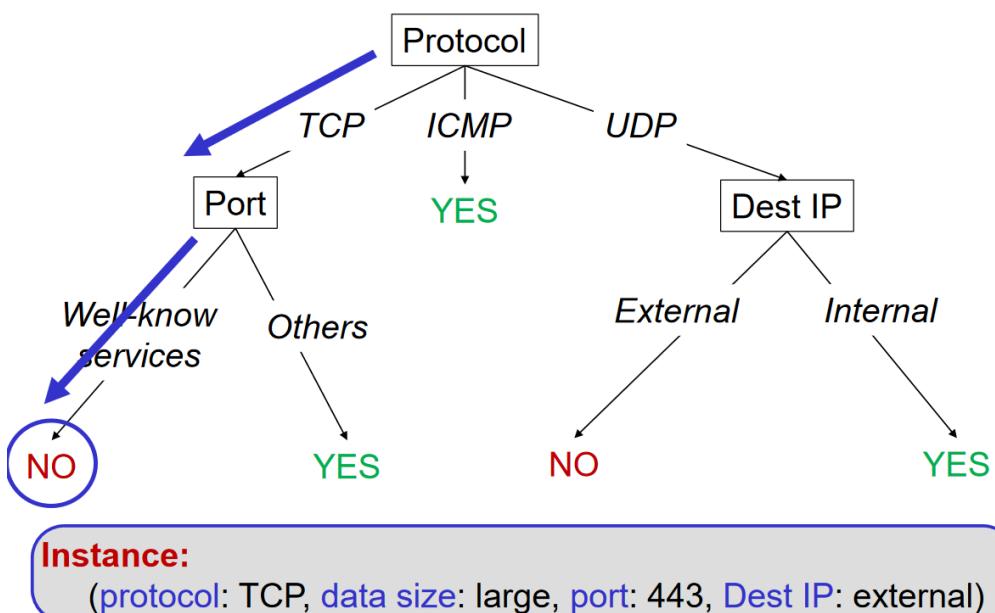
## Decision tree representation: example



22

Ad esempio: se noi abbiamo un nuovo campione che ci dice che il nostro protocollo è TCP, ci sposteremo dal nodo radice Protocollo sul ramo TCP di sinistra. Successivamente visto che abbiamo selezionato il ramo TCP, la prossima domanda riguarda la porta di destinazione (se avessimo selezionato UDP la domanda successiva sarebbe stata diversa) con valori possibili well-know o others. Nella mia istanza la porta destinazione è 443, quindi sarà well-know e quindi la risposta sarà NO, non è un'istanza di management.

## Decision tree representation: example



## **IMPORTANTE!**

Gli attributi di tipo continuo devono essere raggruppati in un numero definito di categorie potenzialmente non troppo elevato. Inoltre, se noi abbiamo delle categorie che assumono più di due valori, ci aspettiamo di avere degli alberi che non siano binari ma di tipo n-ario, in cui ogni biforcazione può avere da due a N valori assumibili. Questo però è il modo in cui un umano andrebbe a costruire l'albero. I calcolatori fanno solo alberi binari perché esiste un algoritmo di trasformazione bidirezionale, tra un albero binario e un albero n-ario qualsiasi. Solo alberi binari che partono da alberi n-ari sono difficili da leggere perché richiedono più rami per costruire lo stesso albero. Per costruire un albero il numero di classi deve essere noto a priori e il numero di campione deve essere nettamente superiore al numero di valori in cui posso classificare.

## Decision tree characteristics

- Attribute-value description:
  - object or case must be expressible in terms of a fixed collection of properties or attributes
    - This means that we need to discretize continuous attributes
    - Or this must have been provided in the algorithm
- Predefined classes (target attribute values):
  - The categories must have been established beforehand (**supervised data**)
- Discrete classes:
  - a case does or does not belong to a particular class
  - there must be more cases than classes
- Sufficient data:
  - Usually, hundreds or even thousands of training cases

## PSEUDO-CODICE

- **Inizializzazione:** guardo tutti gli attributi e scelgo l'attributo che meglio si adatta al ruolo di radice (**best root**);
- **Main loop:** una volta che ho scelto il nodo radice, effettuo la scelta degli attributi successivi. Assegno il nodo scelto come attributo decisionale e per ogni valore assunto da quell'attributo creo un nuovo discendente, quindi, una biforcazione ed ho un nuovo nodo fino a raggiungere la foglia e continuo fino a classificare tutti gli esempi che devono essere utilizzati per il training.

Non tutte le parti dell'albero hanno la stessa profondità. Più è bassa la profondità, prima si arriva alla decisione (più semplice). La profondità è legata dal fatto che ci sono dati difficili da classificare, mentre quelli facili da classificare hanno bassa profondità. Inoltre, i dati caratterizzati da un'elevata profondità sono tipicamente dati che tendono a clusterizzarsi. Questo perché molti dati sono simili ad altri e quindi ho bisogno di un sacco di domande prima di arrivare alla foglia.

## Building a decision tree: pseudo-code

### 1. Initialization:

1. Test all attributes and select the one performing as BEST ROOT

### 2. Main loop:

1.  $A \leftarrow$  the **best** decision attribute for next node
2. Assign A as decision attribute for node
3.  $\forall A_{value}$  create a new descendant for node
4. Sort training example to leaf node
5. Continue until training examples are classified

Greedy search for an acceptable decision tree

25

## Selezione degli Attributi

Abbiamo detto che bisogna scegliere la root e anche i nodi successivi che mi permettono di costruire l'albero e la sequenza delle domande (dei nodi).

### Ad esempio, se ho due possibilità (A<sub>1</sub> e A<sub>2</sub>) devo scegliere A<sub>1</sub> o A<sub>2</sub>?

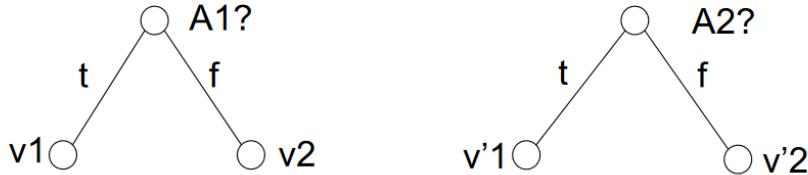
Il criterio che viene adottato è che noi cerchiamo un attributo che renda i dati raggiungibili attraverso il discendente, i più **puri possibili**.

### Che significa puri?

Significa omogeneo, quindi vogliamo utilizzare come attributo (quindi come domanda) quella la cui risposta ci aiuti a discriminare il meglio possibile i nostri dati. Bisogna cercare domande che danno maggiori informazioni.

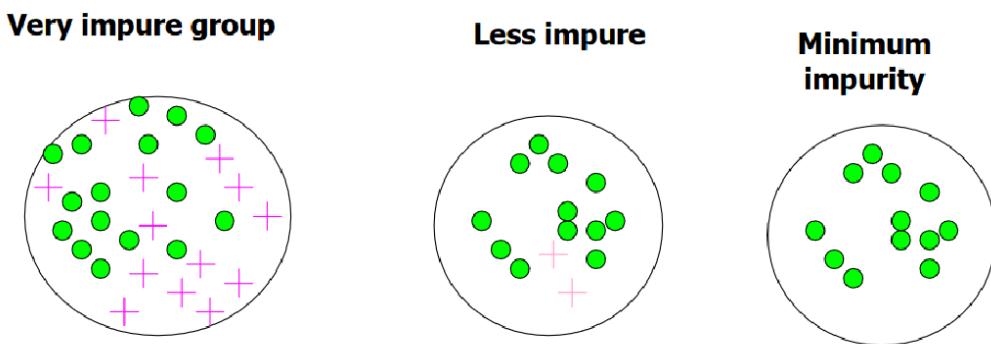
## Attribute selection

- Which attribute should be taken next, A1 or A2?



- We seek an attribute that makes the data reaching the immediate descendent nodes as **pure** as possible
  - More convenient to define the **impurity** of a node
  - We want to measure the level of impurity in a group of examples

## Group impurity



## Entropia e Impurità

Più che cercare la purezza, facciamo il processo opposto e andiamo a cercare l'impurità. Andiamo a misurare il livello di impurità di un gruppo di risposte e cerchiamo la risposta che ci va ad abbassare maggiormente il livello di impurità. Inizialmente avremo un livello di partenza e un livello di destinazione, quella che ci fornirà la maggiore distanza, cioè quella che ci riduce maggiormente l'impurità, sarà il nostro obiettivo.

Come indicatore dell'impurità utilizziamo una nota grandezza della teoria dell'informazioni che si chiama **entropia**. Il concetto è sempre lo stesso, maggiore è il caos, maggiore è l'impurità e potenzialmente maggiore è l'informazione. **Dov'è che ha**

**una maggiore informazione?** Dove ho più incertezza perché se io non ho incertezza sono arrivato alla foglia. Quindi devo guardare quelle configurazioni che mi garantiscono la maggiore incertezza, perché proprio lavorando su quelle potrò guadagnare step by step, sempre di più e arrivare rapidamente ad una condizione di purezza. L'obiettivo è: parto dalla maggiore incertezza e utilizzo gli attributi che mi permettono di decrementarla sempre di più.

## Entropy

Entropy comes from information theory



**The higher the entropy the more the information content**

Per fare questo si utilizza, una denominazione dell'entropia che si chiama entropia impurità.  $P$  rappresenta la probabilità e  $W_j$  rappresenta i valori che possono essere assunti dal nodo  $N$ . Ma ricordiamoci che nel nostro albero, un nodo è solo una rappresentazione astratta di un attributo. Quindi, la sommatoria tra le  $J$  che va da uno a  $N$  dipende dai tipi di valori che può assumere il nostro nodo.

Questa probabilità rappresenta la frazione di percorsi al nodo  $N$  che sono nella categoria in questione. Successivamente, moltiplichiamo il logaritmo della probabilità per la probabilità stessa e poi li andiamo a sommare per tutti i possibili valori del nostro attributo. Non avrò mai un valore di  $W$  per cui la sua probabilità è uguale a zero, quindi non lo avrà. Il meno davanti alla formula serve per avere un numero positivo.

## Node impurity

- Let  $i(N)$  define the **impurity** of a node N:
  - in all classes we want  $i(N) = 0$  if **all the patterns that reach the node belong to the same category**
  - $i(N)$  to be large if all the categories are equally presented

- The most popular measure is Entropy Impurity

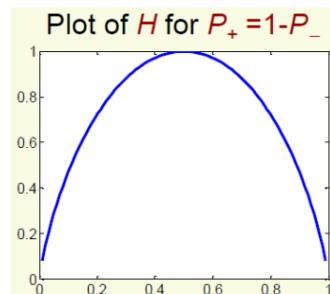
$$i(N) = - \sum_{j=1}^n P(w_j) \log_2 P(w_j)$$

Fraction of patterns at node **N** that are in category **w<sub>j</sub>**

## Entropy

- The entropy  $H(X)$  of a discrete random variable X is

$$H(X) = - \sum_{i=1}^n P(X = i) \log_2 P(X = i)$$



- Example of two classes: X takes only w<sub>+</sub> and w<sub>-</sub> values
  - $H(X) = -P(w_+) \log_2 P(w_+) - P(w_-) \log_2 P(w_-)$
  - $P(w_+) = 0, P(w_-) = 1 \rightarrow H(X) = 0$
  - $P(w_+) = 0.5, P(w_-) = 0.5 \rightarrow H(X) = 1$

29

La forma a campana ci dice che quando le probabilità (le frequenze) di presenza del nostro attributo sono comparabili, ho un elevato grado di impurità. Il nostro obiettivo è proprio arrivare ad avere entropia uguale a zero.

### ***Roba Aggiuntiva che non c'è sulle slide:***

Tipicamente non viene utilizzata l'entropia come metodo per definire l'impurità, ma un'altra impurità che prende il nome di Gini. Gini è un altro modello, di tipo polinomiale, che si dimostra essere più efficiente a livello computazionale (mentre a livello teorico è più efficiente l'entropia) perché l'entropia comporta l'uso del

logaritmo che è una funzione più costosa dal punto di vista di esecuzione, mentre la Gini utilizza il prodotto di polinomi.

## Recap

Abbiamo introdotto l'entropia, cioè disordine e pertanto incertezza, impurità di informazione nel processo di costruzione degli alberi decisionali. Il target è quello di identificare gli attributi che presentano una maggiore impurità e andarli a ridurre in modo tale che l'impurità venga ridotta al minimo, a zero. Non avere impurità equivale a dire avere indice pari a 0 e questo accade nel caso binario quando la tipologia dell'attributo definisce in modo completo la decisione dell'albero. Ovvero, supponiamo che un attributo sia binario: se quell'attributo, arrivato a un certo nodo dell'albero avrà dimensione 0 o 1, e da lì in poi le opzioni sono tutte 0 è chiaro che avrò entropia pari a 0, ma anche se sono tutte 1 avrò entropia uguale a 0. Avrò quindi grado di impurità pari a 0, quindi grado massimo di purezza e sarò arrivato a determinare una delle foglie dell'albero. Sarò arrivato al punto in cui grazie a quell'attributo sono in grado di prendere una decisione finale.

Più alta è l'entropia maggiore sarà il contenuto informativo. Il nostro compito è quello di andare a rimuovere l'impurità per arrivare alla certezza, che è poi l'output della nostra classificazione. Per far questo definiamo una grandezza derivata dall'impurità o entropia (li uso come sinonimi, anche se posso utilizzare l'impurità in modo diverso dall'entropia): il guadagno di informazioni.

## Guadagno di informazioni

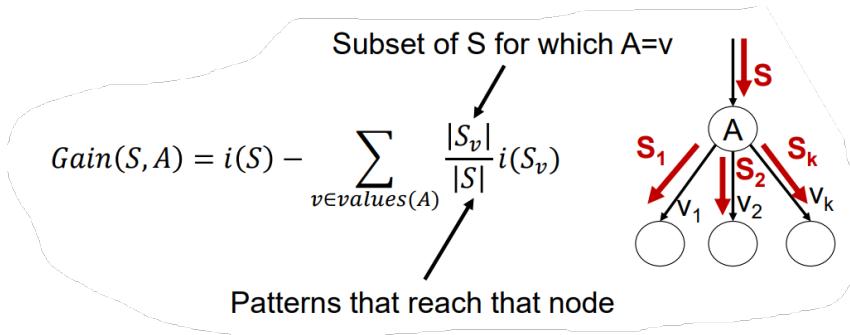
Serve a identificare le scelte che mi abbassano in maniera più drastica il grado di impurità selezionando un certo attributo rispetto ad un altro.

L'algoritmo di costruzione di un albero funziona nel modo seguente: ad ogni nodo seleziono un attributo ed effettuo le diramazioni sulla base dei possibili valori che può assumere l'attributo e poi vado a ripetere questo processo sui rami appena creati andando a selezionare un altro attributo fino a quando il risultato di un attributo è o tutto vero o tutto falso, assumendo di andare a fare una classificazione di un albero che ha solo 2 classi.

- ➔ Il guadagno ci permette di scegliere ad ogni nodo l'attributo che fornisce maggior guadagno. Il maggior guadagno è quello che massimizza la riduzione dell'impurità.

Scegli l'attributo che riduce maggiormente l'impurità del nodo.

Il guadagno di informazioni ha misurato la riduzione prevista dell'entropia dovuta all'ordinamento sull'attributo A.



Il guadagno è definito come l'impurità del nodo da cui si parte ( $i(S)$ ) – la somma pesata dell'impurità dei nodi che andiamo a generare attraverso il branching.

Assumiamo di arrivare attraverso  $S$  in ingresso al nodo, sceglio l'attributo  $A$ , che può assumere 3 valori  $v_1$ ,  $v_2$  e  $v_k$ .  $S_1$ ,  $S_2$  e  $S_k$  rappresentano il numero di entry rimaste nel dataset, quindi non ancora utilizzate, tale per cui l'attributo  $A$  assume valore  $v_1$ ,  $v_2$  o  $v_3$  (con  $k=3$  nell'esempio).  $S_1 + S_2 + S_3$  deve essere uguale a  $S$ , cioè al numero totale di entry non ancora utilizzate.

Ogni volta che utilizzo un attributo effettuo una partizione nella parte di dataset ancora non esplorata. Dopo che ho effettuato l'esplorazione il dataset verrà partizionato in dei sotto blocchi e ripete in maniera ricorsiva questo approccio.

Quindi quello che adesso è  $S$  sarà  $S_1$  e quando vado a biforcire avrò un numero di entry più piccolo da andare ad analizzare, perché ho già preso una decisione sull'attributo  $A$ , che perciò è fisso e non lo posso più considerare.

### Training set: esempio

Questo è l'esempio che utilizzeremo per andare a costruire l'albero.

Event	Protocol	Data size	Dest Port	Dest IP	Mgmt Traffic
E1	TCP	Large	Well-known	Internal	No
E2	TCP	Large	Well-known	External	No
E3	ICMP	Large	Well-known	Internal	Yes
E4	UDP	Mild	Well-known	Internal	Yes
E5	UDP	Low	Other	Internal	Yes
E6	UDP	Low	Other	External	No
E7	ICMP	Low	Other	External	Yes
E8	TCP	Mild	Well-known	Internal	No
E9	TCP	Low	Other	Internal	Yes
E10	UDP	Mild	Other	Internal	Yes
E11	TCP	Mild	Other	External	Yes
E12	ICMP	Mild	Well-known	External	Yes
E13	ICMP	Large	Other	Internal	Yes
E14	UDP	Mild	Well-known	External	No
E15	TCP	Large	Other	External	No

In questo esempio le etichette sono l'ultima colonna. Avremo 4 attributi (protocollo, dimensione dei dati, porta di destinazione e indirizzo di destinazione) e serviranno per classificare un flusso come flusso di management o flusso di dati, non di management.

### Esempio

In tutti i casi in cui ho solo due classi per classificare (management si e management no) un modo conveniente, che permette di guardare meno volte al dataset, per svolgere i conti è andare a identificare l'entropia attraverso il simbolo  $E([X+,Y-])$ .

X+: numero di istanze che portano a una classificazione positiva (management si)

Y-: numero di istanze che portano a una classificazione negativa (management no)

Ora dobbiamo decidere qual è l'attributo da mettere in testa, cioè da utilizzare nella root. Non possiamo utilizzare il guadagno perché quello prevede che ho qualcosa che viene da sopra, quindi in realtà usiamo il guadagno ma non abbiamo il ramo precedente e quindi assumiamo che il valore entrante S sia l'entropia di tutto il dataset. Non avendo un nodo entrante, un predecessore, assumo che l'entropia sia quella del dataset, non del nodo precedente.

Pertanto, l'entropia per i dati di addestramento,  $i(S)$ , può essere rappresentata come  $E([9+,5-])$ , a causa dei 14 esempi di formazione (14 righe del dataset), 9 sono sì e 5 no.

Cominciamo calcolando l'entropia del training set:

$$i(S) = E([9+, 5-]) = (-9/14 \log_2 9/14) + (-5/14 \log_2 5/14) = 0.94$$

Per definizione di entropia, utilizzo al posto della probabilità l'interpretazione frequentista della probabilità, quindi  $P+$  e  $P-$

$$P_+ = \frac{9}{14} \quad P_- = \frac{5}{14}$$

$$i(S) = -P_+ \log_2(P_+) - P_- \log_2(P_-)$$

$$\log_k x = \frac{\ln x}{\ln k}$$

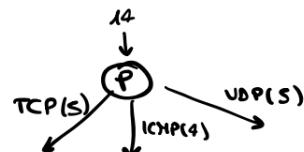
$$i(S) = 0,94$$

Il massimo dell'impurità è 1, quindi il nostro dataset è piuttosto impuro.

Successivamente calcola il guadagno di informazioni  $G(S, A)$  che ci fornisce la scelta di uno dei 4 attributi come radice. Calcoliamo il guadagno per ciascun attributo A dove A è preso dall'insieme {protocollo, dimensione dei dati, porta di destinazione, IP di destinazione}.

Il guadagno di informazioni per il protocollo è:

$$G(S, \text{protocol}) = i(S) - [5/14 * i(\text{protocol}=TCP) + 4/14 * i(\text{protocol}=ICMP) + 5/14 * i(\text{protocol}=UDP)] = E([9+, 5-]) - [5/14 * E([2+, 3-]) + 4/14 * E([4+, 0-]) + 5/14 * E([3+, 2-])] = 0.94 - [5/14 * 0.971 + 4/14 * 0.0 + 5/14 * 0.971] = 0.246$$



$$\begin{aligned}
 i(S, \text{prot}) &= i(S) - \sum_{j=1}^3 P_j \cdot i(P_{\text{prot}, j}) = \\
 &= 0,94 - \frac{5}{14} \in ([2+, 3-]) - \frac{4}{14} \in ([4+, 0-]) - \frac{5}{14} \in ([3+, 2-]) = \\
 &\quad - \frac{2}{5} \log_2 \left( \frac{2}{5} \right) - \frac{3}{5} \log_2 \left( \frac{3}{5} \right)
 \end{aligned}$$

La funzione di entropia è simmetrica, quindi  $E([2+, 3-])$  è uguale a  $E([3+, 2-])$ .

Il guadagno di informazioni per la dimensione dei dati è:

$$\begin{aligned} G(S, \text{Data size}) &= 0.94 - [4/14 * i(\text{Data size}=large) + 6/14 * i(\text{Data size}=mild) + 4/14 * i(\text{Data size}=low)] = 0.94 - [4/14 * E([2+,2-]) + 6/14 * E([4+,2-]) + 4/14 * E([3+,1-])] = \\ &= 0.94 - [4/14 + 6/14 * 0.918 + 4/14 * 0.811] = \mathbf{0.029} \end{aligned}$$

➔ Il guadagno è circa 10 volte più piccolo rispetto al guadagno del protocollo, quindi Data size non è un buon candidato.

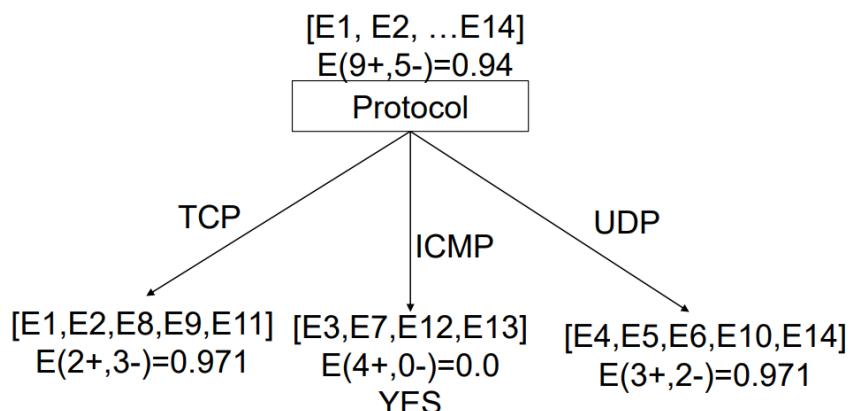
Il guadagno di informazioni per la porta di destinazione è:

$$\begin{aligned} G(S, \text{Dest port}) &= 0.94 - [7/14 * i(\text{Dest port}=well-known) + 7/14 * i(\text{Dest port}=other)] = \\ &= 0.94 - [7/14 * E([3+,4-]) + 7/14 * E([6+,1-])] = 0.94 - [7/14 * 0.985 + 7/14 * 0.592] = \mathbf{0.1515} \end{aligned}$$

Il guadagno di informazioni per l'IP di destinazione è:

$$G(S, \text{Dest IP}) = 0.94 - [8/14 * 0.811 + 6/14 * 1.00] = \mathbf{0.048}$$

Il protocollo è il nostro vincitore! Impostiamo la radice con protocollo e impostiamo quindi il primo set di nodi che derivano dalla radice:



Sulla radice abbiamo tutto il dataset, in questo caso le righe  $[E_1, E_2, \dots, E_{14}]$ . Nel momento in cui andiamo a selezionare uno dei valori del primo attributo, di fatto andiamo a ridurre il dataset (avremo 5 righe per TCP, 4 righe per ICMP e 5 righe per UDP) effettuando una partizione dell'insieme originale.

A questo punto dobbiamo ripetere il processo su un sottoinsieme del dataset.

Dobbiamo prendere la nuova configurazione e andare a calcolare l'entropia di partenza, che è l'entropia dell'insieme  $S$ , cioè del sottoinsieme del dataset selezionato attraverso la scelta dell'attributo precedente.

Consideriamo adesso ICMP: calcoliamo prima l'entropia di ICMP e poi valutiamo con quale degli attributi rimasti (Data size, dest port e dest IP) abbiamo la massima riduzione. Se l'entropia è già 0 abbiamo finito e abbiamo trovato una foglia.

Con ICMP (abbiamo 4 positivi e 0 negativi) l'entropia è pari a 0 e quindi non dobbiamo biforcicare ulteriormente, ma assegniamo l'etichetta YES, perché tutti i campioni dicono più.

Cosa diversa per TCP e UDP. Una volta calcolata l'entropia (0.971) dobbiamo andare ad analizzare le sotto biforcas considerando gli altri 3 attributi candidati.

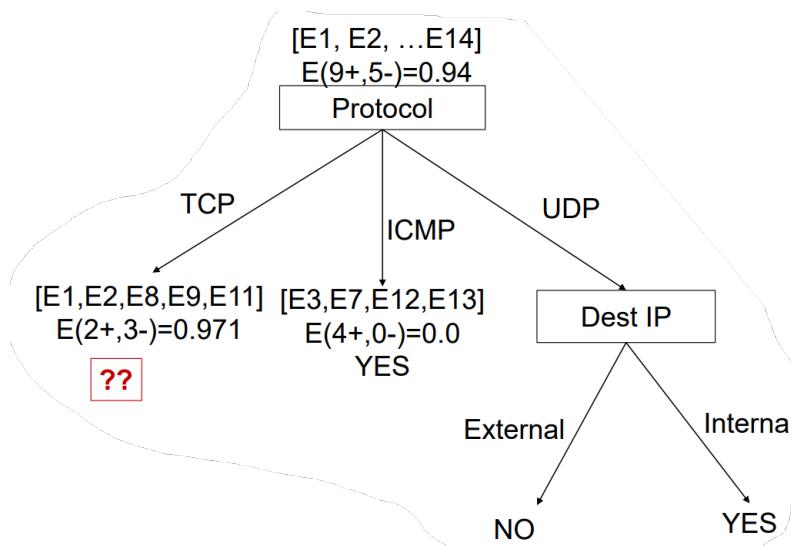
Avendo la radice, dobbiamo ora trovare ricorsivamente i nodi che dovrebbero andare sotto TCP, ICMP e UDP. Una volta fatta la partizione dobbiamo ripetere il processo su un sottoinsieme del dataset.

$$G(\text{Protocol}=UDP, \text{Dest port}) = 0.971 - [2/5 * i(\text{Protocol}=UDP \& \text{Dest port}=\text{well-known}) + 3/5 * i(\text{Protocol}=UDP \& \text{Dest port}=\text{other})] = 0.02$$

$$\begin{aligned} G(\text{Protocol}=UDP, \text{Dest IP}) &= 0.971 - [3/5 * i(\text{Protocol}=UDP \& \text{Dest IP}=\text{Internal}) + 2/5 * i(\text{Protocol}=UDP \& \text{Dest IP}=\text{External})] = 0.971 - [3/5 * E(3+,0-) + 2/5 * E(0+,2-)] = \\ &= 0.971 - [3/5 * 0 + 2/5 * 0] = 0.971 \end{aligned}$$

- ➔ Il guadagno è totale: è pari all'entropia di partenza. Utilizzando questo attributo azzerò l'entropia generando due foglie (perché ho due valori e per ciascuno ho una decisione univoca e opposta). Se avessi avuto per entrambi sempre si o sempre no sarebbe stato come il caso di ICMP e quindi avrei avuto già da prima di considerare gli attributi entropia residua pari a 0.

$$\begin{aligned} G(\text{Protocol}=UDP, \text{Data size}) &= 0.971 - [2/5 * E(1+,1-) + 3/5 * E(2+,1-) + 0] = \\ &= 0.971 - [2/5 + 3/5 * 0.918] = 0.02 \end{aligned}$$



Quale attributo dovrebbe essere testato sul ramo sinistro?

$$S_{TCP} = [E_1, E_2, E_8, E_9, E_{11}]$$

$$G(TCP, \text{Dest port}) = 0.97 - (3/5) * 0.0 - (2/5) * 0.0 = 0.97$$

$$G(TCP, \text{Data size}) = 0.97 - (2/5) * 0.0 - (2/5) * 1.0 = 0.57$$

$$G(TCP, \text{Dest IP}) = 0.97 - (2/5) * 1.0 - (3/5) * 0.918 = 0.019$$

- Prendo l'attributo Dest port che ha il guadagno maggiore, che è pari all'entropia di partenza. Avremo due foglie perché l'attributo assume due valori.

### Terminazione dell'albero decisionale

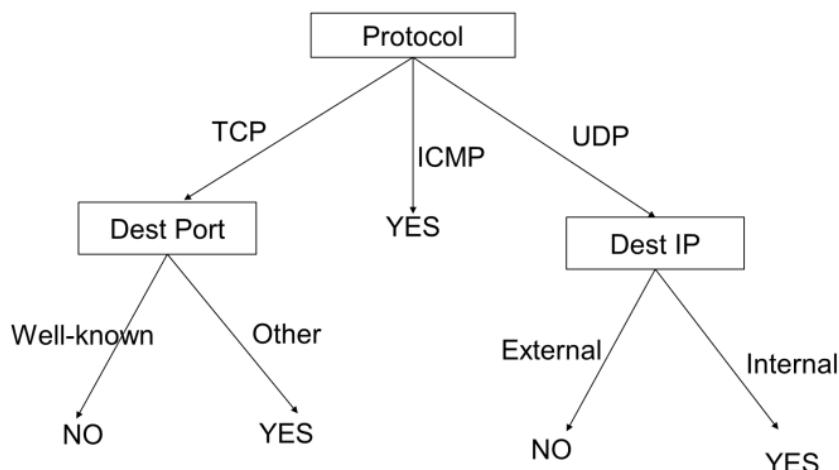
Questo processo continua per ogni nuovo nodo foglia finché non viene soddisfatta una delle due condizioni:

1. ogni attributo è già stato incluso lungo questo percorso attraverso l'albero
2. gli esempi di training associati a questo nodo foglia hanno tutti lo stesso valore dell'attributo target (ovvero, la loro entropia è zero)

Ci fermiamo ogni volta che o abbiamo finito gli attributi o abbiamo generato un set di foglie (tutte le nostre biforazioni sono terminate nella foglie anche se alcuni attributi non sono stati utilizzati, come nell'esempio).

Algoritmo particolarmente semplice da spiegare perché porta a un mapping diretto tra la struttura dell'albero e una sequenza di regole decisionali.

### Esempio



Questo è il nostro albero finale. Siamo arrivati alla fine dell'albero perché abbiamo una serie di nodi tutti con entropia pari a 0, e questo non avendo considerato un singolo attributo (Data size), che risulta quindi essere superfluo.

## **Quale albero dovremmo produrre?**

ID<sub>3</sub> esegue una ricerca euristica attraverso lo spazio degli alberi decisionali dal più semplice al sempre più complesso, guidata dal guadagno di informazioni. Si ferma all'albero più piccolo accettabile.

Il rasoio di Occam: preferire l'ipotesi più semplice che si adatti ai dati.

Avremo un albero che utilizza in testa i criteri più ovvi e via via usa i criteri più particolari.

## **Rasoio Occam**

1. Il numero di ipotesi semplici che potrebbero accidentalmente adattarsi ai dati è piccolo, quindi le probabilità che semplici ipotesi scoprano alcune conoscenze interessanti sui dati sono maggiori.

Le soluzioni più semplici saranno poche ed essendo poche è ragionevole che ciascuna di esse fornisca una quantità di conoscenza significativa rispetto al problema. Quindi con poche scelte semplici tipicamente riesco ad ottenere la maggior parte della conoscenza che sto cercando.

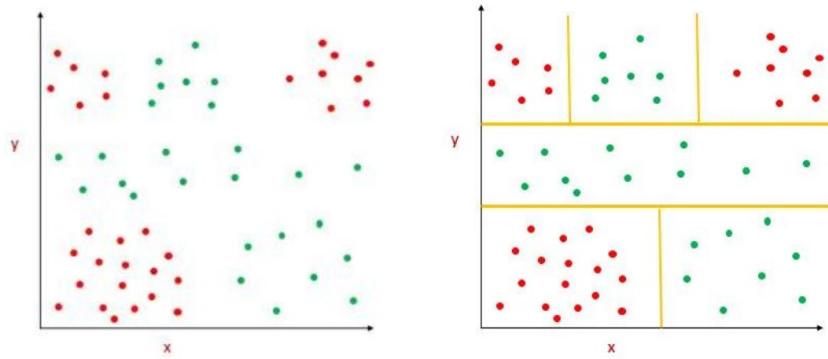
2. Gli alberi più semplici non suddividono lo spazio delle funzionalità in troppe piccole scatole, quindi, può generalizzare meglio.

Il risultato è che quando ho un certo numero di queste scelte semplici io finisco per costruire un albero semplice, cioè piccolo, poco profondo. Un albero poco profondo è in grado di generalizzare molto bene perché riesce a separare il nostro spazio in un certo numero di scatolette n dimensionali che contengono i nostri dati.

3. Gli alberi complessi possono avere una casella separata per ciascun campione di dati di addestramento, portando così ad un overfitting.

Se abbiamo alberi complessi perché non riesco a trovare un criterio guida ragionevole che mi permette di generare l'albero, ma devo affidarmi ai conti e all'entropia ma senza avere un'idea del perché certi numeri vengano fuori da questi conti vado a finire in albero complesso, cioè profondo. Un albero profondo fa del suo meglio per adattarsi ai dati, quindi per andare a generare delle foglie che abbiano entropia 0. La conseguenza è che tipicamente questo albero è troppo basato sui valori del dataset e quindi è andato in overfitting rispetto al dataset. Questo significa che se poi gli andremo a presentare nuovi campioni non offrirà prestazioni abbastanza soddisfacenti.

## **Partizione dati rettangolare**



Classificare significa fare una partizione dello spazio n-dimensionale del nostro dataset. L'approccio che usa decision tree dal punto di vista della partizione dello spazio è un approccio che spezzetta lo spazio in rettangoli n-dimensional. Più i rettangoli sono piccoli, quasi intorno al singolo campione, più significa che l'albero è molto profondo. Invece, se i rettangoli sono belli grandi il nostro albero è poco profondo e riesce a generalizzare bene. Quindi maggiore è il numero dei campioni di ciascun rettangolo e meglio è, e soprattutto il numero di foglie, se ho poche foglie generalizzo bene, se ho tante foglie anche con tanti campioni dentro generalizzo male. Il fatto che ho tanti campioni dentro magari significa che ho un dataset con parecchi elementi.

Nota: la cosa interessante da vedere è che, rispetto all'esempio visto ieri, in cui ho una feature e una linea, due features e una linea, tre features e un piano, quattro features e una linea e un iperpiano e così via, questo perché eravamo ricaduti nel caso numero 3 (slide 17). Il dataset non riesco a modellarlo ma riesco a stabilire un modello teorico per la funzione che agisce da discriminante sul dataset. In questo caso devo risolvere un problema di ottimizzazione che determina i parametri della superficie. Per quanto riguarda gli alberi siamo nel caso numero 4, metodi non parametrici. Dati rispettati ma non c'è nessuna distribuzione di probabilità, quindi vado a segmentare lo spazio secondo forme arbitrarie e nel caso degli alberi queste forme sono sempre rettangolari.

Qui abbiamo una visione parziale perché abbiamo solo due features: x e y. Tipicamente un dataset ha n features e se le vado a graficare posso fare solo la visualizzazione di due features, tanto tipicamente si graficano le features con maggiore importanza e questo si applica anche prima della visualizzazione delle tecniche di riduzione della dimensionalità (le vedremo nel corso di deep learning).

### **Overfitting nell'albero decisionale**

Se abbiamo un dataset grande che porta a crescere il nostro albero troppo in profondità significa che siamo in overfitting. In realtà il problema è molto più strutturale, non dipende solo dal singolo dataset ma è proprio una tipica debolezza dell'albero. Se al

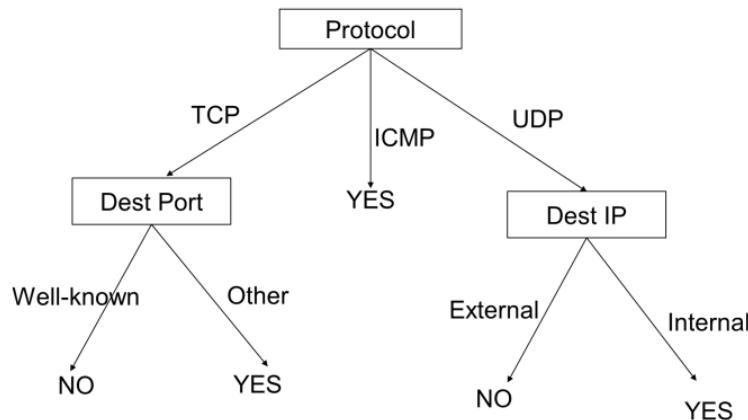
dataset utilizzato per costruire l'albero aggiungiamo o togliamo pochi campioni (poche righe dal dataset) tipicamente la struttura dell'albero resta uguale.

Provate a ricostruire a casa l'albero utilizzando il dataset di prima più la riga in verde, avrete quindi 15 righe e vedrete che l'albero cambierà in maniera significativa.

Considera l'idea di aggiungere un esempio rumoroso:

*TCP, Data size large, Dest port other, Dest IP external, Mgmt traffic=No*

Come influisce sull'albero precedente?



Il rumore nei dati o il numero limitato di esempi di addestramento portano a un overfitting.

In generale basta aggiungere o togliere pochi campioni per avere delle forme stravolte. Questo significa che gli alberi sono tecniche troppo sensibili alla struttura del dataset: grossa debolezza. Come facciamo a renderli più robusti? Dobbiamo farli meno profondi.

### Training set: esempio

Event	Protocol	Data size	Dest Port	Dest IP	Mgmt Traffic
E1	TCP	Large	Well-known	Internal	No
E2	TCP	Large	Well-known	External	No
E3	ICMP	Large	Well-known	Internal	Yes
E4	UDP	Mild	Well-known	Internal	Yes
E5	UDP	Low	Other	Internal	Yes
E6	UDP	Low	Other	External	No
E7	ICMP	Low	Other	External	Yes
E8	TCP	Mild	Well-known	Internal	No
E9	TCP	Low	Other	Internal	Yes
E10	UDP	Mild	Other	Internal	Yes
E11	TCP	Mild	Other	External	Yes
E12	ICMP	Mild	Well-known	External	Yes
E13	ICMP	Large	Other	Internal	Yes
E14	UDP	Mild	Well-known	External	No
E15	TCP	Large	Other	External	No

## **Evitare un overfitting**

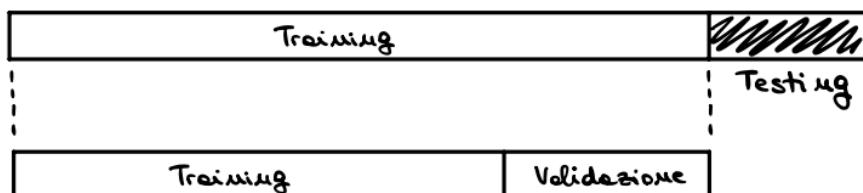
Come evitare l'overfitting? Fare l'albero più corto, in due modi:

- Smette di crescere prima, durante la costruzione dell'albero, prima di classificare perfettamente i dati di addestramento.  
La crescita posso fermarla sia sulla base della profondità sia sui livelli di entropia raggiunti.
- Fai crescere l'intero albero, quindi pota l'albero: vado in post-processing (soluzione tipicamente utilizzata).  
Per fare questo devo fare una procedura a ritroso. Una volta che ho l'albero devo andare ad effettuare un algoritmo di ottimizzazione che va minimizzare sia la dimensione dell'albero sia il numero degli errori.

Come selezionare l'albero “migliore” (come fare la potatura):

- Misurare le prestazioni su set di dati di convalida separati
- Scambia la complessità con l'accuratezza: riduci al minimo [dimensione(albero) + dimensione(classificazioni errate(albero))]

Vado a riservare una porzione del dataset per la validazione della potatura per vedere se la potatura è potenzialmente efficace.



Divido la parte di training in una parte che è ancora di training e in una parte di validazione. Perché l'albero generato tramite il training è potenzialmente instabile, allora uso i dati di validazione per valutare quanto la potatura è potenzialmente efficace.

## **Potatura decisionale degli alberi**

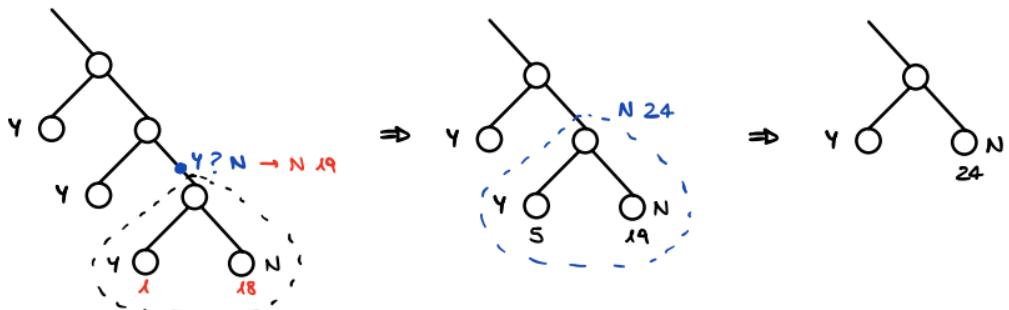
Un modo per affrontare il problema dell'adattamento eccessivo negli alberi decisionali è sfoltire l'albero. Ciò significa rimpicciolirlo eliminando i nodi che non risultano utili.

Usa un'avidità ricerca in salita per la potatura:

- per ogni nodo interno (non foglia) n
  - rimuovere il sottoalbero al n
  - sostituisco con un nodo foglia contrassegnato con + o -, a seconda della maggior parte degli esempi di addestramento che rientrano in quel sottoalbero
  - valutare le prestazioni del nuovo albero decisionale sul set di validazione

- conservare l'albero potato con le migliori prestazioni
- ripetere fino a quando non si ottiene alcun miglioramento rimuovendo qualsiasi nodo interno
- se un albero tagliato ha le stesse prestazioni dell'albero attuale, allora usa l'albero tagliato

Vediamo un tipico algoritmo di potatura. Un algoritmo di potatura va a ritroso e di solito si fa la cosa seguente:



Supponiamo che l'albero fornisca una classificazione binaria: yes o no.

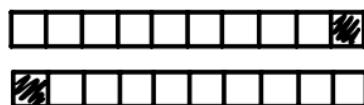
La potatura consiste ad esempio nell'identificare a ritroso un possibile ramo da tagliare (quello tratteggiato) e da sostituire con una foglia equivalente. La foglia è yes o è no? L'approccio più semplice è contare le istanze in fase di training che permettono di classificare si e no che sono interne a questa foglia candidata. Avendo 1 istanza per il si e 18 per il no è ovvio che vado a classificare con il no e prendo tutte e 19 le istanze.

Ora posso valutare questo albero con la porzione di validation, perché quei dati non li ho utilizzati per fare training e quindi sono un buon candidato per capire se l'operazione ha introdotto un errore tollerabile (un errore è normale che ci sarà) oppure no. Ci sarà un errore nella fase di training ma non è detto che ci sarà anche della fase di validazione, anzi.

Posso classificare la bontà dell'albero attraverso il numero degli errori di classificazione che vengono fatti nel dataset di validazione.

Quindi per valutare se la potatura è efficace o troppo aggressiva utilizzo questa porzione del dataset. Nel momento in cui ho scelto la configurazione finale vado a calcolare la vera prestazione della configurazione finale, ottenuta attraverso il processo di training e validazione del training, attraverso la porzione di testing che ho lasciato nel dataset.

In alcuni casi posso anche effettuare delle prove diverse e ripetute: lascio porzioni diverse di validazione.

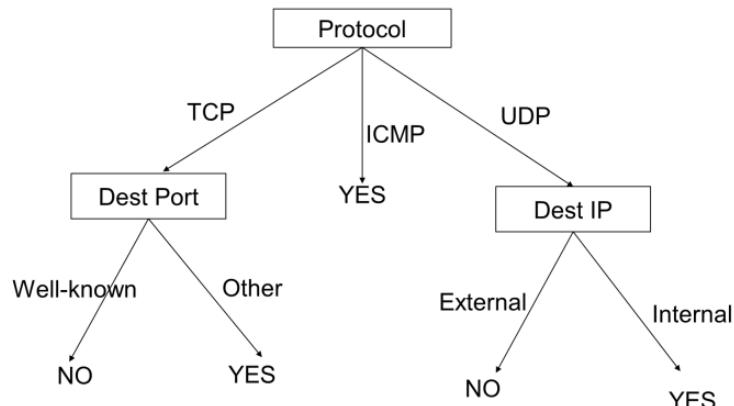


Tipicamente se andiamo a considerare le prestazioni solo sulla fase di training l'albero potato fa sempre un po' peggio, sulla fase di validazione potrebbe anche avere prestazioni migliori o comparabili. Se sono comparabili l'albero più semplice è sempre quello che classifica meglio perché dipende meno dagli specifici dati usati per addestrarlo.

### Regola post-potatura

- Fai crescere l'albero fino a quando i dati di training non saranno nel miglior modo possibile
- Converti l'albero appreso in un insieme equivalente di regole, creando una regola per ciascun percorso
- Eliminare gli antecedenti delle regole non necessari per semplificare le regole
  - Le regole con un solo antecedente non possono essere ulteriormente semplificate, quindi consideriamo solo quelle con due o più
  - Per semplificare una regola, eliminare gli antecedenti che non hanno alcun effetto sulla conclusione raggiunta dalla regola
- Ordina le regole finali nella sequenza desiderata per l'uso

### Conversione di alberi in regole



Sequenza di regole decisionali (esempio):

IF (Protocol=TCP) AND (Dest port=well-known) THEN Mgmt traffic=No

IF (Protocol=TCP) AND (Dest port =other) THEN Mgmt traffic =Yes

Potevamo anche specificare tutti gli yes e poi dire che tutto il resto è no, o viceversa. Soluzione più semplice: diciamo solo quei due no (che sono in minoranza) e diciamo che tutto il resto è yes.

Nota: se utilizzo tutti gli attributi l'albero si sviluppa in lunghezza, e questa è una caratteristica indesiderata perché l'albero che si sviluppa in lunghezza overfitta. E come

tutti gli algoritmi che overfittano significa che dipende troppo dal dataset su cui è stato allenato e quindi non sarà in grado di generalizzare bene rispetto a dati che non ha mai visto.

### Punti di forza e di debolezza

I vantaggi dell'albero decisionale sono:

- Gli alberi decisionali sono in grado di generare regole comprensibili, facilmente spiegabili.
- Gli alberi decisionali eseguono la classificazione senza richiedere molti calcoli (non richiedono troppa computazione): in fase di testing, di operation, una volta che ho addestrato l'albero per generare la decisione devo semplicemente prendere un'istanza dei dati e farla navigare nell'albero fino a incontrare una foglia. Tipicamente questo è estremamente veloce.
- Gli alberi decisionali sono in grado di gestire in maniera efficiente sia variabili continue che categoriche, a patto che le variabili continue siano facilmente confinabili in un numero non troppo grande di sottoinsiemi, trattandole così di fatto come variabili discrete.
- Gli alberi decisionali forniscono una chiara indicazione di quali features sono più importanti per la previsione o la classificazione, quindi per prendere le decisioni. Sono le features che stanno in cima all'albero, dalla radice in poi: questo rappresenta l'ordine di importanza.

Gli alberi sono spesso utilizzati per effettuare classificazioni sia di malfunzionamento, sia di anomalie legate alla sicurezza o di altro tipo nel campo delle reti e dei servizi delle comunicazioni proprio perché sono pieni di variabili binarie, booleane, categoriche e alcune più o meno continue o facilmente racchiudibili in poche categorie.

Sono quindi una tra le strategie maggiormente utilizzate. Poi sono utilizzate in tanti altri campi in cui non dominano le variabili continue.

I punti deboli dei metodi dell'albero decisionale sono:

- Gli alberi decisionali sono meno appropriati per attività di stima in cui l'obiettivo è prevedere il valore di un attributo continuo.
- Gli alberi decisionali sono soggetti a errori nei problemi di classificazione con molte classi e un numero relativamente piccolo di esempi di training.
- Il training dell'albero decisionale può essere computazionalmente costoso, soprattutto per dataset molto grandi. La cosa importante è che siano molto veloci dal punto di vista dell'esecuzione, una volta che il modello è stato allenato perché li pone come buoni candidati per essere utilizzati in approcci real time come ad esempio la rilevazione delle anomalie (passare flussi di traffico tipo di netflow su un albero è immediato, molto al di sotto del secondo).
  - Il processo di crescita di un albero decisionale è computazionalmente costoso:

- In ciascun nodo, ciascun campo di suddivisione candidato deve essere ordinato prima che sia possibile trovare la suddivisione migliore
- In alcuni algoritmi vengono utilizzate combinazioni di campi ed è necessario effettuare una ricerca per i pesi di combinazione ottimali
  - Gli algoritmi di potatura possono anche essere costosi poiché è necessario formare e confrontare molti sottoalberi candidati.
- Gli alberi decisionali non trattano bene le regioni non rettangolari:
  - La maggior parte degli algoritmi dell'albero decisionale esamina solo un singolo campo alla volta
  - Ciò porta a caselle di classificazione rettangolari che potrebbero non corrispondere bene alla distribuzione effettiva dei record nello spazio decisionale

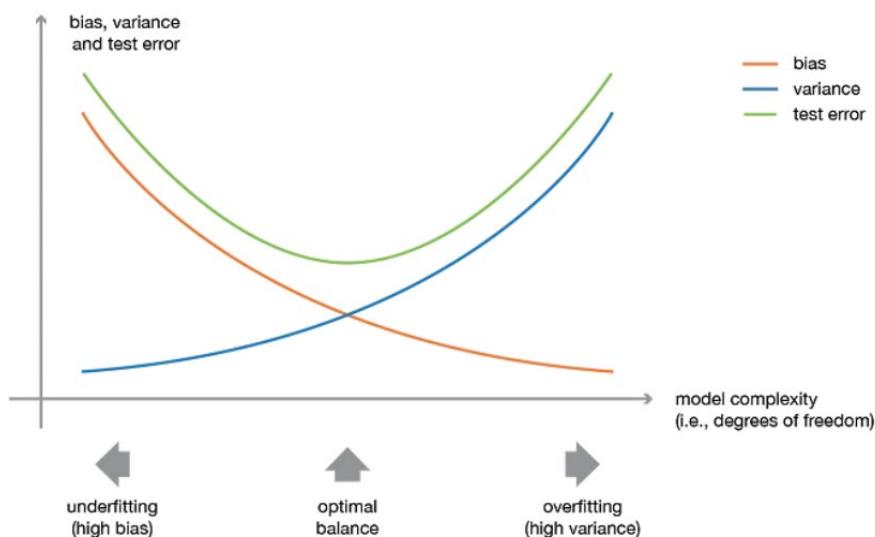
## RANDOM FORESTS

Le foreste sono un insieme di alberi.

I principali svantaggi degli alberi sono:

- **Overfitting**: poiché la precisione migliora con ciascun nodo interno, l'allenamento tenderà a far crescere un albero al massimo per migliorare la metrica delle prestazioni; ciò deteriora la capacità di generalizzazione e l'utilità dell'albero sui dati invisibili poiché inizierà a modellare il rumore.
- **Instabilità** dell'albero: anche piccole modifiche ai dati di training, come l'esclusione di alcune istanze, possono dare come risultato un albero completamente diverso; ciò ci lascia con la domanda su quale albero fidarsi.

## Bias-variance trade-off



In generale, mano mano che aumento con l'overfitting, quindi con il bias, aumento la complessità del modello. Man mano che overfittato mi aspetto che in fase di testing l'errore cresca.

Tuttavia mano mano che overfittato ottengo che cresce anche la varianza, e questa non è una proprietà gradita agli algoritmi. Questo comporta che le prestazioni in fase di testing tendono ad avere un minimo in cui decrescono all'aumentare della complessità dell'algoritmo, ma poi se la complessità è eccessiva e tendono ad overfittare soffrono del fatto che i campioni del testing possono essere anche abbastanza diversi da quelli del training e se c'è stato overfitting l'albero è cresciuto troppo simile ai campioni del training e le prestazioni in fase di testing saranno scarse.

Le foreste casuali affrontano entrambi i problemi, overfitting e instabilità.

L'idea è quella di costruire più alberi utilizzando diversi sottoinsiemi di dati di training e di features per ciascun albero della foresta. Quindi, aggregiamo le loro previsioni fornendo il voto della maggioranza o il valore medio.

Utilizzo quindi più alberi contemporaneamente, e questi alberi li voglio più semplici di quello di partenza, quindi più corti e anche con una visione parziale dei dati. Costruisco di fatti i based-learners o weak-learners, ovvero coloro che apprendono in modo basilare o debole. Ciascuno degli alberi della foresta avrà prestazioni nettamente inferiori dell'albero addestrato su tutto il dataset, ma nel momento in cui vado ad utilizzare una strategia di insieme allora questi weak-learners funzionano molto meglio di un singolo grande albero globale. Se voglio utilizzare l'albero per fare predizioni farò la media dei risultati, se voglio fare classificazione dovrò votare a maggioranza.

Ci sono due istanze considerate dalle foreste: visione parziale dei dati, per ridurre l'overfitting, e visione parziale delle features, per ridurre i problemi di instabilità e per fare in modo che alcuni attributi che vengono poco selezionati ma che potenzialmente potrebbero essere di interesse abbiano più chance di salire nelle parti alte degli alberi.

Fondamento logico: un insieme di modelli è probabilmente più accurato di un singolo albero. Anche se un albero si adatta eccessivamente al suo sottoinsieme, ci aspettiamo che altri alberi nella foresta lo compensino. Utilizzare sottoinsiemi diversi per ciascun albero, per costringerli ad affrontare il problema da diverse angolazioni.

## Training in Random Forests

Di fatto si vanno ad utilizzare 2 procedure: bagging e selezione casuale delle features.

Complessità:

- Addestrare una foresta di  $B$  alberi richiederà  $B$  volte più tempo che costruire un singolo albero
- Poiché montiamo tutti gli alberi in modo indipendente, possiamo costruirli contemporaneamente
- Ciò accelererà il processo di formazione ma richiede un'infrastruttura adatta al calcolo parallelo!!

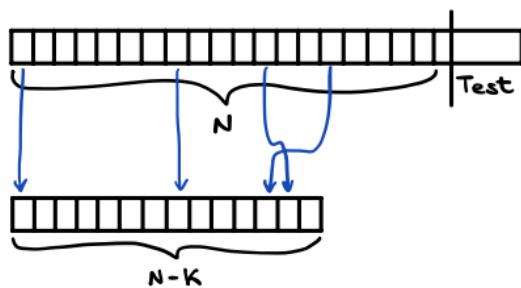
Configurazione:

- Oltre agli iperparametri di un albero, dobbiamo considerare gli alberi  $B$  nella nostra foresta
- Di conseguenza, aumenta il numero di combinazioni di iperparametri da controllare durante la convalida incrociata

Idea: cresco una foresta di  $B$  alberi, che avrà complessità più grande di quella di costruire il singolo albero. Quanto è grande il dataset su cui cresce gli alberi della foresta? Posso avere due strategie.

- 1) Utilizzo un dataset che è un sottoinsieme del mio dataset
- 2) Utilizzo un dataset che ha la stessa dimensione, ma non è uguale, al dataset di partenza

Esempio:



Potrei avere un dataset in cui prima divido i dati di test; per quanto riguarda il training lo divido in un certo numero di parti che rappresentano una partizione, e poi assegno ad ogni albero della foresta  $N - K$  di questi blocchetti (di solito  $K=1$ ). In questo modo la dimensione del dataset su cui vengono allenati i singoli alberi è comunque significativa ed è vicina a quella originale, ma i dati sono leggermente diversi. È bene che i dati siano leggermente diversi così ogni albero overfitterà su una porzione diversa e quindi tenderanno ad essere alberi simili ma leggermente diversi. Se sono alberi simili ma leggermente diversi quando si vota a maggioranza ci si riesce a svincolare in maniera significativa dal processo di overfitting del singolo albero. Stiamo parlando di qualche centinaio di alberi e quindi dal punto di vista delle complessità computazionale quanto cambia? Dipende dalle capacità di calcolo della macchina che si utilizza, se è una macchina con molti core non cambierà tantissimo perché nel caso del singolo albero,

che è un processo mono thread, la parallelizzazione può essere un valido aiuto anche se non toglie che le risorse consumate saranno sempre le stesse.

In questo caso avrò un dataset che è leggermente più corto, di dimensione  $N - K$ .

In percentuale:  $\frac{N-K}{N}$

Se  $K$  è piccolo i dataset sono simili ma non identici. Siccome sappiamo che gli alberi sono molto sensibili al dataset di partenza, significa che questi alberi saranno abbastanza diversi, e questo è sicuramente un bene.

### Interpretabilità nelle foreste casuali

Le foreste casuali contengono più alberi, quindi anche se uno si adatta eccessivamente ai dati, probabilmente non sarà il caso degli altri.

Ci aspettiamo che un insieme sia più accurato di un singolo albero e, quindi, abbia maggiori probabilità di fornire la classificazione corretta.

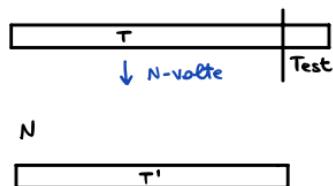
Tuttavia, le foreste perdono l'interpretabilità che ha un albero.

Problema principale della foresta: perdiamo la spiegabilità dell'albero. Nel momento in cui togliamo la visibilità su tutti i dati e quindi ogni albero ha un sottoinsieme pseudocasuale dei dati e vado a maggioranza, quindi non c'è un albero che pesa di più rispetto a un altro ma pesano tutti allo stesso modo, è ovvio che avrò una decisione che non sono più in grado di spiegare in maniera semplice guardando la struttura di un albero.

Una volta addestrato è sempre molto veloce: è tanto più lento quanto sono gli alberi. Se gli alberi sono  $B$  o sarà  $B$  volte più lento su una macchina non parallela o un po' più lento su una macchina parallela. Questo nel caso in cui utilizzo come strategia un dataset leggermente più piccolo.

Esiste anche un'altra opzione, il bagging, che tipicamente si utilizza di più. La dimensione del dataset è esattamente pari a quella del dataset di partenza. Supponiamo che il dataset di partenza abbia  $N$  elementi.

Bagging :



Ad ogni iterazione, quindi per  $N$  iterazioni, vado a pescare in maniera casuale uno dei campioni che stanno nel dataset di training.  $T'$  può avere dei doppioni andando a

pescare a caso. Siccome ogni estrazione è casuale, la probabilità di estrarre un campione è pari a  $\frac{1}{N}$ , dove N è la cardinalità del dataset.

La probabilità di non estrarre un campione è pari a  $1 - \frac{1}{N}$ .

La probabilità di non estrarre mai un campione, cioè un campione che esiste in T ma non esiste in T' è  $(1 - \frac{1}{N})^N$ . Dopo N tentativi, se ancora non l'ho pescato, avrò un T' che è sicuramente diverso da T. Questo non ci dice quante ripetizioni ci stanno di ogni campione presente in T', ci dice soltanto qual è la probabilità che un campione non venga selezionato mai da T in T'.

$$\left(1 - \frac{1}{N}\right)^N \xrightarrow{N \rightarrow +\infty} \frac{1}{e} = 0,32 = P_{MS} \text{ (Probabilità di non venire selezionato)}$$

Quindi quanti campioni non verranno selezionati dentro al mio dataset ( $D_{ns}$ )? È pari al 68%

$$D_{ns} = N \cdot P_{MS} \quad 68\%$$

Con il 32% genero automaticamente un validation dataset.

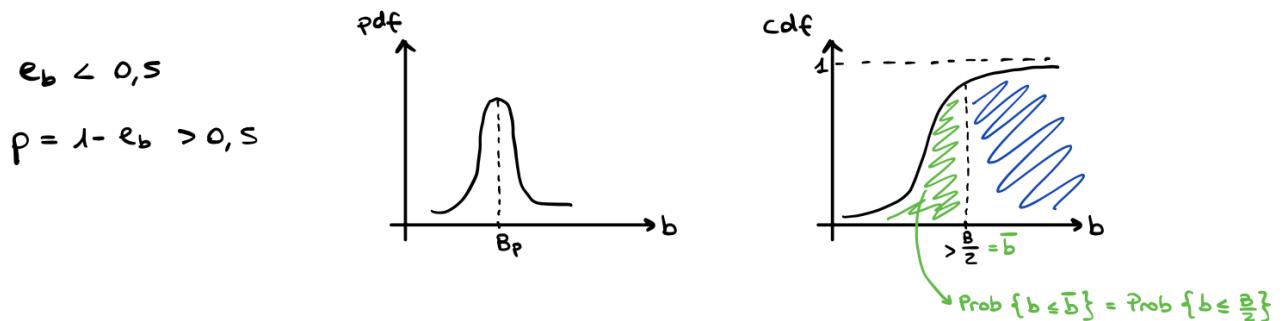
Questo inoltre genera alberi con un sacco di ripetizioni, che non andrò a togliere perché le ho messe apposta. Queste ripetizioni genereranno un dataset a minore varianza (aumentando le ripetizioni la varianza si riduce) e quindi avrò degli alberi meno profondi perché vedono meno dati (i dati visti uguali aumentano la rapidità con cui scendo verso le foglie). Avrò poi un po' di polarizzazione perché sono presenti le ripetizioni. Quando avrò molte istanze di questi alberi che lavorano su dataset simili ma in realtà anche diversi, andando a fare il processo di media o di votazione a maggioranza andrò a ridurre il Bias. Avrò alberi decisionali più scarsi e più semplici, quindi generalizzano meglio. Le proprie deficienze vengono in parte sopperite dal fatto che ho un processo di insieme, e che nel caso della classificazione un processo che va a maggioranza.

Facciamo adesso un caso matematico molto semplice: andiamo a vedere se questi weak-learner rappresentano una buona soluzione per la nostra decisione. Per fare questo facciamo una prima assunzione: è forte e non è neanche vera ma ci da il caso limite. I nostri weak-learner, quindi i nostri alberi dentro la foresta cresciuti su dataset fatti in questo modo, si comportino come variabili iid, variabili casuali indipendenti e identicamente distribuite. Ci può stare che siano identicamente distribuite perché sono cresciuti su dataset molto molto simili, ma che siano indipendenti ovviamente no perché partono da una base molto comune.

Se non c'è dipendenza, assumendo di avere un numero abbastanza grande di alberi, so che se fossero indipendenti, assumendo che ogni albero abbia un tasso di errore (capacità di sbagliare la classificazione) pari a  $e_b$ , con  $e_b < 0.5$ , allora possiamo pensarlo

come un meccanismo associato all'estrazione ripetuta di una monetina. Quindi la probabilità di successo è pari a  $1 - e_b > 0,5$ .

Quindi la decisione fornita da ogni albero possiamo vederla come la probabilità di una distribuzione binomiale.



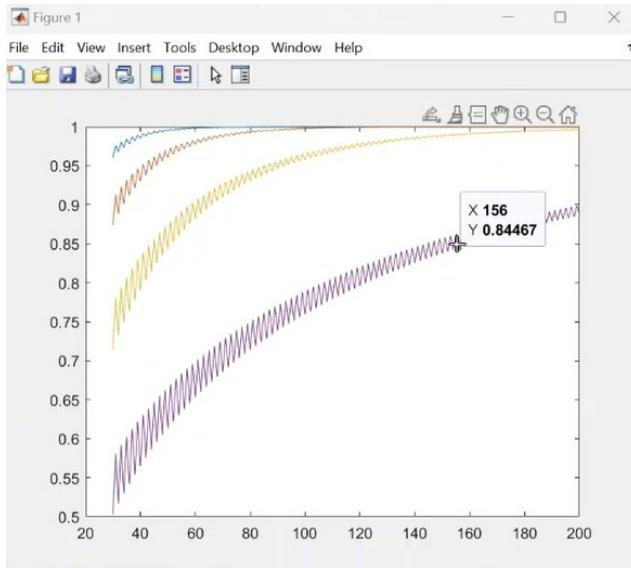
$b$  rappresenta il numero di alberi che sono d'accordo.

Supponiamo che la decisione sia binaria (si o no). Il sistema deciderà per il si o per il no quando  $b \geq \frac{B}{2}$ .

Assumiamo di avere  $B$  alberi e che ogni albero si comporti come una variabile indipendente binaria. Ci interessa di andare a vedere la probabilità complessiva di tutta la foresta di avere ragione, ovvero quando il numero di alberi che danno lo stesso verdetto è o tutto positivo o tutto negativo.

Supponiamo che per una specifica istanza il risultato che ci aspettiamo sia quello di essere positivo. Quindi vogliamo che la probabilità di avere più di  $\frac{B}{2}$  alberi che diano risultato positivo sia il più grande possibile. Vediamo quanto effettivamente può essere grande.

FILE MATLAB: considero una probabilità di errore che va dal 30 al 45% e considero una foresta molto piccola che va da 30 a 200 alberi. Vado a calcolare la probabilità che il numero di alberi concordi sia maggiore della metà, e poi lo vado a graficare. Quello che si vede è che per la probabilità di errore pari a 0,3 (altissima) già con 60 alberi ho una probabilità di successo pari al 100%. Per una probabilità di errore di 0,35 ci arrivo con 100 alberi al 100%, se ho il 40% di errore ci arrivo con 200 alberi. Le prestazioni sono più modeste con il 45% di errore di partenza ma con 200 alberi mi avvicino al 90% di successo. Il 45% di tasso di errore equivale dire che sono quasi alla monetina, quindi è abbastanza irragionevole.



Questo ci dice che se gli alberi fossero indipendenti basterebbe questo meccanismo di addestrarli su dataset simili ma non identici, e facendogli vedere un po' meno dati per garantirci una decisione molto molto affidabile. Il problema è che questa è una situazione ideale e noi non siamo nella condizione di totale indipendenza.

Se non abbiamo la condizione di indipendenza la varianza della stima è pari a

$$\rho \sigma^2 + \frac{(1-\rho)}{\beta} \sigma^2$$

dove  $\sigma^2$  è la varianza del singolo albero e  $\rho$  è il coefficiente di correlazione. Se non sono indipendenti avrò un coefficiente di correlazione positivo e quindi aumentando gli alberi posso andare ad abbassare drasticamente il secondo termine (x rossa), ma ho comunque  $\rho\sigma^2$ .

Allora entra il gioco il secondo meccanismo della foresta: selezione casuale delle features.

### Foreste casuali: ultime parole

Le foreste casuali risolvono i problemi tipici dei DT: Overfitting + Instabilità.

Le foreste randomiche utilizzano due approcci: il primo approccio è quello di randomizzare il dataset, o attraverso il bagging, che è la strategia predefinita, o attraverso la strategia vista prima, cioè ridurre il dataset in maniera deterministica.

L'approccio è la combinazione di due metodi:

- Bagging (aggregazione bootstrap): dati di campionamento con sostituzione. Riduce la varianza dei campioni, ma introduce gli weak learners. Il bias introdotto viene filtrato tramite l'insieme.

- **Selezione casuale di attributi:** cerca di rimuovere attraverso decisioni pseudocasuali la correlazione tra i vari alberi. Gli alberi sono per forza correlati perché sono cresciuti sullo stesso dataset. Come si rimuove la correlazione?

Per la costruzione dell'albero ad ogni nodo ho un certo numero di attributi ed effettuo il calcolo rispetto a tutti gli attributi per determinare quello vincente per effettuare il prossimo split. Vogliamo fornire la possibilità di selezione anche per attributi non forti. Ciò viene fatto considerando un sottoinsieme di attributi ( $m$ ) rispetto all'insieme completo ( $p$ ) nel punto di scissione:

$$m = \sqrt{p} \text{ per la classificazione}$$

$$m = \frac{p}{3} \text{ per la predizione}$$

Quindi di fatto limito in maniera casuale enormemente il numero di attributi e se ogni albero, non solo cresce su un dataset leggermente diverso che ne limita la varianza e in generale l'overfitting, ma è anche significativamente decorrelato rispetto all'altro perché non vede tutti gli attributi, quindi effettua lo splitting solo su un sottoinsieme molto limitato di attributi, allora effettivamente questi alberi tendono a essere decorrelati in maniera significativa.

Doppio vantaggio:

- **Evitare l'instabilità:** c'è meno overfitting perché ho un certo numero di alberi praticamente indipendenti, cresciuti su dataset simili ma andando ad utilizzare per la costruzione dell'albero in realtà un numero molto inferiore dei dati presenti all'interno del dataset.
- **Inspiegabilità:** siccome questo viene fatto in maniera randomica, l'effetto netto è che gli alberi sono molto robusti rispetto ai DT e praticamente inspiegabili, perché risultano dal doppio processo di bagging e selezione casuale delle features. Quando il numero degli alberi è 100 o 200 non riusciamo più ad avere idea del perché esca fuori quella decisione. Non solo, se io utilizzo l'approccio random forests 1, 2, 3 o 10 volte di fila sullo stesso dataset, proprio perché la componente randomica sia nel bagging sia nella selezione degli attributi è molto pesante, posso avere risultati leggermente diversi. In realtà il framework è molto robusto e quindi la differenza sarà sempre molto limitata, però ci possono essere delle piccole differenze.

Le librerie di machine learning fanno tutto in automatico, ma bisogna capire quali sono gli algoritmi più adeguati e perché.

Le random forests sono adeguate in tutti quei casi in cui sono adeguate gli alberi decisionali, e anzi sono di fatto il tool più utilizzato al posto degli alberi decisionali.

L'albero decisionale si usa quando voglio avere un'idea di quello che sta succedendo. Ma se voglio un criterio robusto non posso affidarmi a un singolo albero decisionale, devo usare le foreste che sono meno dipendenti dal dataset e quindi molto più robuste,

anche se non permettono di spiegare in modo dettagliato quello che succede all'interno del processo di selezione degli attributi di split.

Le random forests sono tra gli approcci più utilizzati che non siano vere e proprie reti neurali. Sono robuste e hanno tempi di esecuzione molto competitivi. Le reti neurali hanno caratteristiche tipicamente superiori ma possono avere tempi di apprendimento dell'ordine delle ore o anche dei giorni mentre per le random forests spesso bastano qualche decina di secondi anche per dataset molto grandi.

Per dataset per cui gli alberi decisionali non sono adeguati, non sono adeguate neanche le random forests (le foreste sono fatte da alberi).

FINE 9.11

N.B MANCA SOLO SBOBINA DEL 20/11 DOPO DI QUESTA.

## Esperienze di laboratorio con il dataset KDD'99

Abbiamo analizzato il dataset KDD'99, che viene dall'aggregazione di sorgenti di natura differente.

Alcuni dati sono infatti relativi ad un'analisi che avviene per flusso di una traccia di traffico. Ad esempio, viene identificato il protocollo di strato applicativo (attraverso il numero di porta), viene identificato il protocollo di strato 4, vengono conteggiati i byte in una direzione e nella direzione opposta.

Queste sono tutte informazioni che possono essere recuperate attraverso un'analisi effettuata con Netflow o sFlow.

Altre informazioni, che sono le features del dataset, vengono da una sorgente completamente diversa, in quanto sono relative ad attività di login effettuate all'interno di una macchina remota. Ragionevolmente le features derivano quindi dall'analisi di log o di sistema o di server applicativi.

È quindi un dataset complesso, rappresentativo di un dataset reale.

Abbiamo visto quali sono i tipi di traffico che vogliamo classificare (traffico normale o 4 tipi di attacchi). Molti attacchi presenti nel dataset, raggruppati in:

- traffico benigno
- 4 diverse categorie di attacco (dos, probe, r2l, u2r)

La classificazione diretta delle 5 classi fornisce scarsi risultati. Il problema principale è non tanto distinguere gli attacchi dal traffico normale, ma è la classificazione errata degli attacchi tra loro, e questo è comune a quasi tutti gli approcci.

Approcci testati:

- Decision tree
- KNN
- SVM (sia lineare sia con una funzione kernel, in particolare RBF, Radial Basis Function)
- Random forests

A che cosa può essere dovuta questa prestazione poco incoraggiante? È sicuramente dovuta a un forte sbilanciamento del dataset.

Infatti, abbiamo un certo numero di categorie di traffico (traffico normale e attacchi dos) che di fatto dominano il dataset.

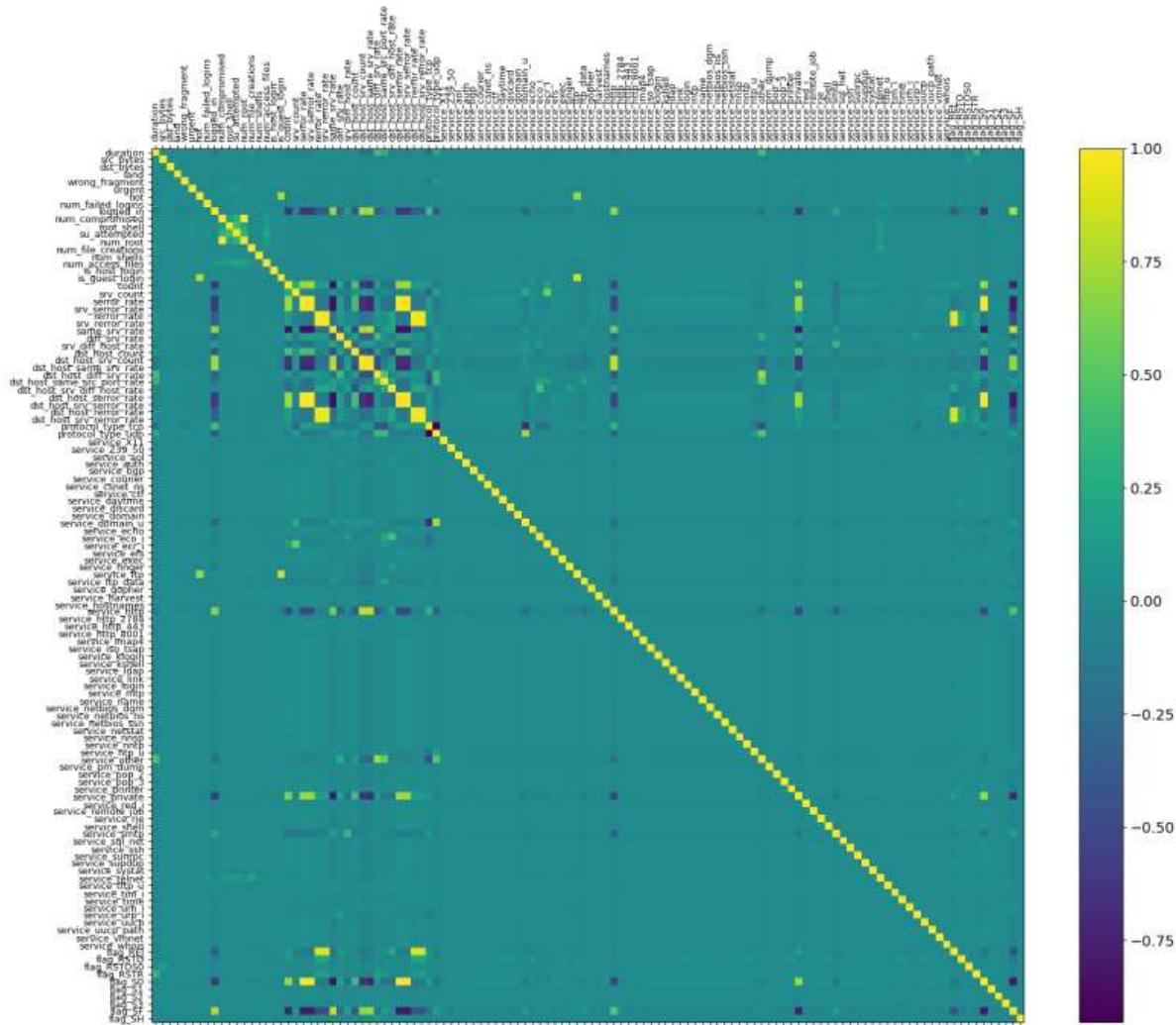
Ci sono altre tipologie di attacchi, come probe, ma soprattutto r2l e u2r (che contano solo qualche centinaio di istanze), che sono nettamente inferiori a dos. Abbiamo quindi un dataset molto sbilanciato, con delle classi fortemente sottorappresentate.

Le tecniche che possono essere utilizzate per migliorare questa soluzione sono quelle basate sul sottocampionamento: creiamo un dataset più piccolo in cui teniamo i campioni già esistenti delle classi minoritarie e andiamo a campionare degli elementi delle classi maggioritarie, in modo da avere un dataset abbastanza bilanciato.

Sul Chio c'è un'analisi di questo tipo e si vede che i risultati sono abbastanza modesti perché uno dei tipi di attacco è fortemente sottorappresentato, e se andiamo a riscalare tutto il dataset verso una tipologia di attacco fortemente sottorappresentata, otterremo un dataset troppo piccolo.

Abbiamo provato sia ad utilizzare l'analisi diretta sia a utilizzare il preprocessing.

Qui vediamo l'analisi della correlazione:



I puntini gialli rappresentano una forte correlazione, i puntini blu scuri invece rappresentano una correlazione fortemente negativa. Tutto ciò che sta nel range del verde (verde chiaro e verde scuro) significa che le variabili sono scorrelate.

Tranne il quadrato in alto a sinistra, in cui sono presenti dei quadrettoni gialli, non c'è una grande possibilità di andare a ridurre il dataset e, se proviamo ad effettuare operazioni di riduzione, otteniamo che possiamo diminuire il dataset da 118 features a 110, perché poi bisogna stare attenti a cosa cancelliamo.

In particolare, una volta che abbiamo applicato la trasformazione con one note encoding succede che il dataset è stato riorganizzato nel modo seguente.

Nella parte iniziale del dataset ci sono le variabili numeriche e le variabili che partivano già come binarie.

Tutte le variabili che erano di tipo categorico sono state trasformate in variabili binarie, con la regola 1 variabile categorica -> n-1 variabili binarie, con n numero di valori unici che la variabile assume. Queste variabili sono state spostate tutte in fondo.

Bisogna stare attenti a non andare a cancellare le variabili codificate in maniera one note perché se le cancelliamo andiamo a togliere un pezzo di informazione al dataset. Queste variabili servono tutte insieme perché il complemento a 1 delle variabili fornisce informazioni sulla variabile "mancante".

L'unica cosa che si potrebbe fare è quindi togliere qualche variabile in alto, dove abbiamo fortissima correlazione o fortissima anti-correlazione, perché di fatto sono equivalenti, l'informazione è la stessa.

L'altra cosa che possiamo fare è effettuare la pulizia dei dati, con operazioni preprocessing che possono essere di:

- **Standardizzazione** (StandardScaler): andiamo ad assumere che la variabile è gaussiana e riportiamo quei valori ad una variabile normalizzata, a media nulla e varianza unitaria.

Partendo da una feature di tipo X otteniamo una feature di tipo Y ( $\mu$  è il valore medio di X e  $\sigma$  è la varianza di X).

$$Y = \frac{X - \mu}{\sigma}$$

La standardizzazione funziona bene solo se le variabili possono essere assunte come gaussiane.

Se anche le features possono essere modellate come variabili gaussiane, se queste sono contaminate da outliers, abbiamo visto in un'esercitazione passata che questo comporta una forte distorsione della stima del valore medio e soprattutto della varianza.

Quindi, dovremmo scalare la feature rispetto alla media e alla varianza che non sono affette da outliers; siccome questo non lo possiamo fare "a priori", è necessario adottare degli approcci alternativi.

- **Normalizzazione** (MinMaxScaler): riporta ogni variabile di tipo numerica o con range generico nel range 0-1.

$$Y = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Questo tipo di preprocessing è obbligatorio in tutti quei tipi di applicazioni in cui bisogna andare a valutare la grandezza dell'errore, perché l'errore percentualmente piccolo su una feature potrebbe dominare o essere fortemente dominato da un errore percentualmente piccolo su un'altra feature.

Nel momento in cui tutte le variabili sono comprese tra 0 e 1, con 0 valore minimo e 1 valore massimo, hanno tutte dei range confrontabili e quindi questo problema non esiste.

- **Scaling robusto**: si parte da una feature di tipo X e si ottiene una feature di tipo Y, scalando la mediana di X (che è piuttosto robusta alla presenza di outliers) e dividendo per il range interquartile IQR (tra il 75° e il 25° percentile).

$$Y = \frac{X - median(X)}{IQR}$$

Tipicamente, per le variabili contaminate da outlier o che in generale non fittano bene nel paradigma gaussiano, una buona soluzione è il RobustScaling.

Una volta applicate queste tre tecniche al dataset, notiamo che la standardizzazione non fornisce alcun beneficio, così come la normalizzazione. L'unico approccio che porta il dataset ad essere omogeneo è SVM, ma non ne beneficia come prestazioni, ne beneficia solo come tempo di convergenza, per la sua struttura, cioè per come è fatto SVM.

Tutti gli altri approcci hanno un minimo di benefici, e soprattutto KNN beneficia molto, con lo scaling robusto, in cui i dati vengono scalati in modo tale da non tener conto dell'effetto degli outliers. Decision tree e random forests hanno dei miglioramenti marginali applicando questo approccio, KNN ha miglioramenti significativi.

Se non è possibile affrontare in maniera diretta il problema della classificazione multipla, una buona idea potrebbe essere quella di effettuare una classificazione in due passaggi. Il primo classifica semplicemente attacchi/non attacchi (potrebbe anche basarsi su approcci di rilevamento delle anomalie) e il secondo discrimina tra attacchi noti (approccio della misuse detection).

La prima fase può essere implementata in modo efficace sia con DT che con RF con il robust scaling. Si hanno valori di precisione molto elevati, valori molto bassi di falsi positivi e falsi negativi (precisione e richiamo sono davvero soddisfacenti) e tempi di training ragionevoli e test rapidi. Il classificatore di secondo stadio è ancora un problema aperto...

Questo perché dall'analisi delle prestazioni abbiamo visto che nella matrice di confusione i TP (True Positive) sono correttamente identificati dalla maggior parte degli approcci. Gli attacchi erroneamente classificati come traffico benigno, i FN, sono un numero abbastanza limitato rispetto ai TP, e quindi abbiamo gli approcci con una buona precisione relativamente al traffico benigno.

Anche le istanze di traffico benigno classificate come uno dei possibili attacchi, i FP, sono numericamente dominate dai TP.

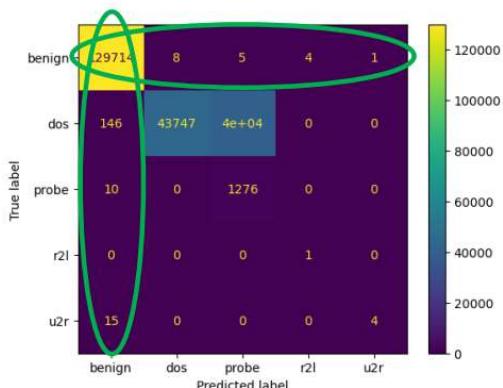
Rispetto al traffico benigno abbiamo quindi prestazioni di richiamo e di precisione molto buone.

Complessivamente l'accuratezza è intorno all'80% perché c'è una forte misclassificazione degli attacchi di dos come attacchi di probe: questo è il problema più significativo.

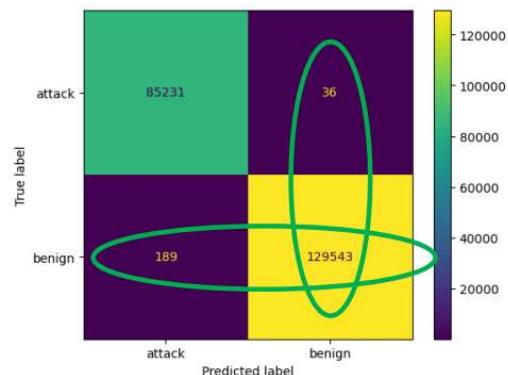
Visto che la distinzione attacco/non attacco viene fatta abbastanza bene affrontando il problema con un numero di classi superiore a 2, proviamo a vedere se aggregando tutti gli attacchi insieme (rendendo anche il dataset più bilanciato) le prestazioni migliorano.

Effettivamente le prestazioni migliorano, in particolare l'approccio random forests era quello migliore.

**Single layer: RF**



**First layer: RF (robust scaling)**

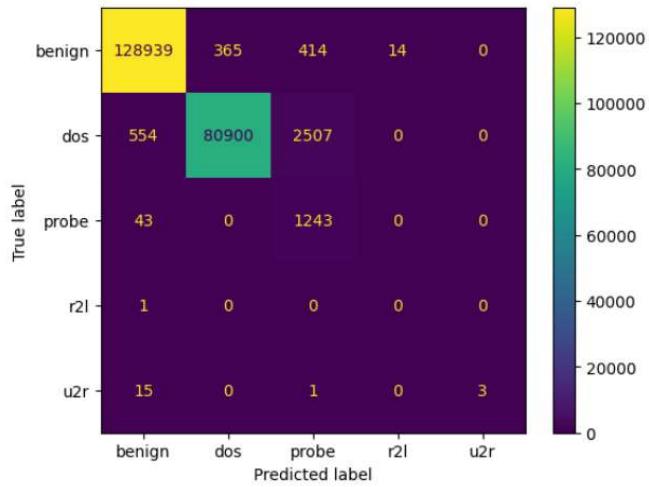


La matrice di confusione viene trasposta quando si va a fare la classificazione binaria.

In particolare, una prestazione molto importante è quella dei falsi negativi, mentre è un po' più deficitaria quella dei falsi positivi.

Se bisogna scegliere è sempre meglio penalizzare i falsi positivi rispetto ai falsi negativi.

Per quanto riguarda gli altri tipi di classificazione si è visto che la configurazione che funziona meglio è KNN con robust scaling e numero di vicini pari a 5.



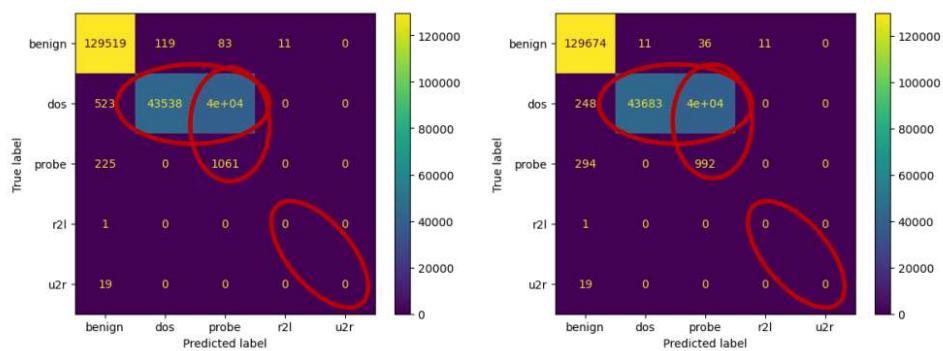
La prestazione è buona dal punto di vista del richiamo per quasi tutte le classi, anche se per r2l e u2r le prestazioni sono inaccettabili.

La precisione è ancora bassa. Molti attacchi dos sono classificati come attività di probing. Tuttavia, è la migliore rispetto a tutte le altre alternative.

Il tempo di prova non è fattibile: circa 40 minuti per l'elaborazione del dataset di test (215k righe) e circa 11 ms per singola istanza, troppo lenti per l'utilizzo in tempo reale.

Questa è la prestazione che si ottiene con SVM usando lo scaling MinMax.

**SVM: Linear kernel**      **SVM: RBF kernel**



I problemi sono quelli evidenziati dai cerchietti rossi, che quindi riguardano le misclassificazioni e la mancata corretta classificazione delle due classi minoritarie.

Utilizzando lo scaling robusto la prestazione è leggermente diversa, ma non di tanto.

L'unica cosa che cambia in maniera significativa è il tempo di elaborazione, infatti, nel caso di SVM sia lineare sia con la RBF, con gli altri tipi di scaling non c'è una convergenza, ovvero fornisce un messaggio di warning dicendo che non è riuscita a trovare l'ottimo globale.

Questo non succede se viene usata la tecnica MinMax, e anzi l'output viene fornito in un tempo ragionevole. Support vector machine (SVM) converge solo con la normalizzazione dei dati.

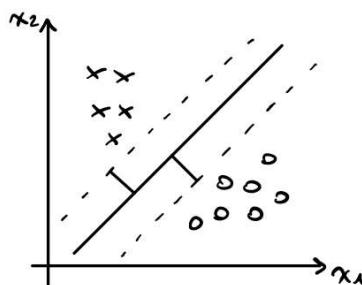
Tempo di esecuzione:

- Kernel lineare (LinearSVC in scikit-learn): 30 s per l'addestramento e 0,11 s per il test (0,5 us per istanza)
- Radial basis function (SVC in scikit-learn): 11 minuti per l'addestramento e 151 s per i test (0,7 ms per istanza)

Prestazioni ancora non soddisfacenti dal punto di vista di accuratezza, richiamo e precisione. Ci mette poco ma non serve a niente.

Le istanze u2r e r2l sono ancora una volta mal interpretate.

SVM va a cercare i vettori di supporto, ovvero i punti potenzialmente più vicini alla superficie di separazione (che è una retta nel caso bidimensionale).



*Come si fa a fare una classificazione multclasse con SVM?*

Nel caso lineare, se abbiamo n classi si effettuano n classificazioni del tipo:

$$n \text{ classi} \rightarrow \begin{aligned} & (\text{classe } 1, \text{ rest}) \\ & (\text{classe } 2, \text{ rest}) \\ & \vdots \end{aligned} \quad ] \quad n \text{ classificazioni}$$

Viene effettuata n volte una classificazione binaria. Di fatto, un'istanza dovrebbe essere classificata correttamente solo in una di queste n classificazioni, in quanto dovrebbe risultare sempre nel resto delle classi (rest) e solo una volta in una classe.

Quando utilizziamo il kernel trick, quindi portiamo i campioni in un mondo a più dimensioni e poi applichiamo l'iperpiano, viene fatta una classificazione 1 a 1. Ad esempio, viene fatta una classificazione benigno-dos, benigno-probe, benigno-r2l, benigno-u2r, e via dicendo per tutte le classi. Il numero di classificazioni da testare è:

$$\frac{n(n - 1)}{2}$$

Questo è il motivo della netta differenza temporale tra SVM lineare e non lineare.

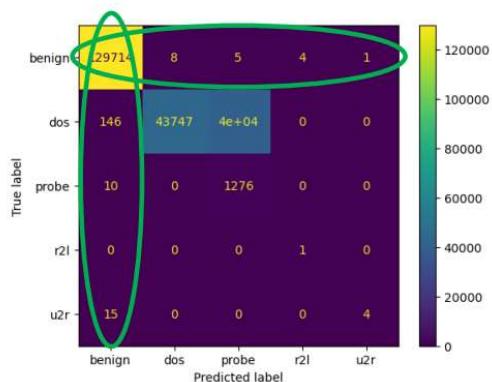
SVM soffre tantissimo due tipi di cose: dataset molto grandi, proprio perché richiede un tempo di elaborazione molto elevato, e dataset molto sbilanciati.

SVM non è quindi un buon candidato per questa classificazione.

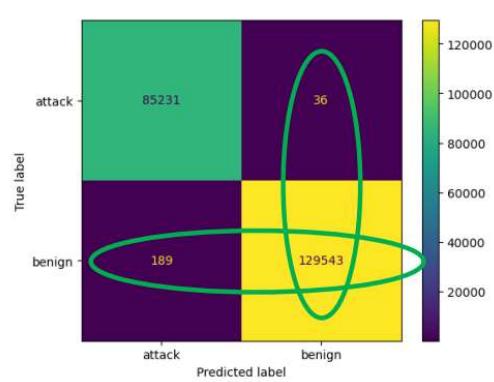
Per quanto riguarda Random Forests, è quello che si comporta meglio escluso KNN. KNN ha però il problema del tempo di elaborazione superiore ai 40 minuti. KNN si comporta bene soprattutto sulla discriminazione degli attacchi dos da quelli probe, e di fatto migliora le prestazioni di richiamo e di precisione per gli attacchi dos.

SVM lineare è molto più veloce di RF, ma classifica peggio, quindi preferiamo RF. SVM lineare inoltre non arriva neanche a convergenza. Stessa cosa per SVM non lineare, e in più le prestazioni sono veramente scarse.

**Single layer: RF**



**First layer: RF (robust scaling)**



Vediamo cosa riusciamo ad ottenere con l'approccio bidimensionale.

Classificatore IDS a singolo stadio: numero totale di righe: 1074992, training (859993) + test (214999).

Classificatore IDS supervisionato a due stadi: training per il primo livello: 859993, test per il primo livello: 214999; training per il secondo strato: 176911 (dataset di training con features etichettate solo come attacco), test per il secondo livello: 85397 (è il risultato del primo stadio, quasi 1/3 rispetto al singolo strato).

La cosa interessante da ricordare è che, quando andiamo ad utilizzare l'approccio con due passate, il secondo classificatore risulta essere allenato con un numero molto più basso di istanze. Il numero con cui viene allenato il secondo classificatore è il numero di attacchi etichettati come tali nel dataset di training.

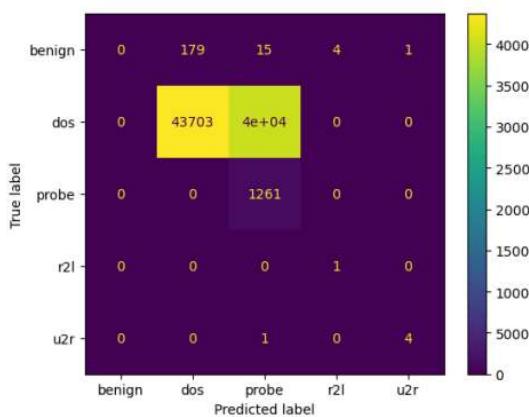
I tempi di training del secondo classificatore saranno perciò significativamente ridotti. Per KNN ci aspettiamo che anche i tempi di test siano molto ridotti.

Anche per il testing il carico di elaborazione sul secondo stadio sarà più basso, perché al secondo stadio arriveranno tutte quelle istanze classificate come attacchi, correttamente o erroneamente.

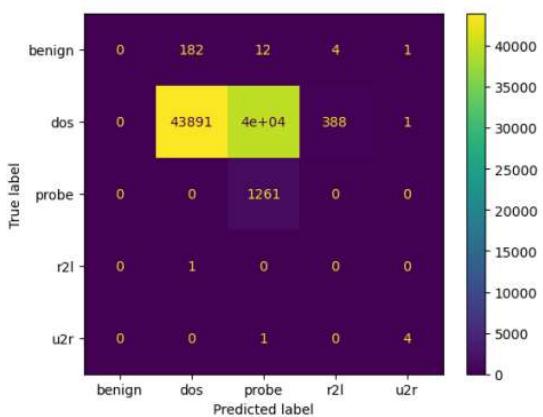
Il classificatore di secondo livello sarà allenato con 4 classi, quelle degli attacchi noti, ma si troverà di fronte traffico da 5 classi, perché ci sarà qualche istanza di traffico benigna che era stata erroneamente classificata come attacco. Proprio per questo la matrice di confusione sarà ancora 5x5.

Applichiamo l'approccio a due stadi ai classificatori visti finora.

**Second layer: RF**



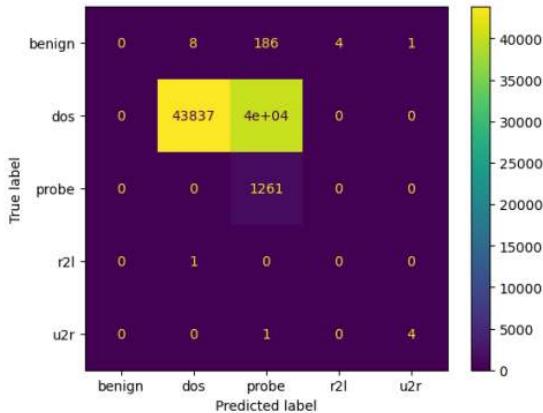
**Second layer: Linear SVM**



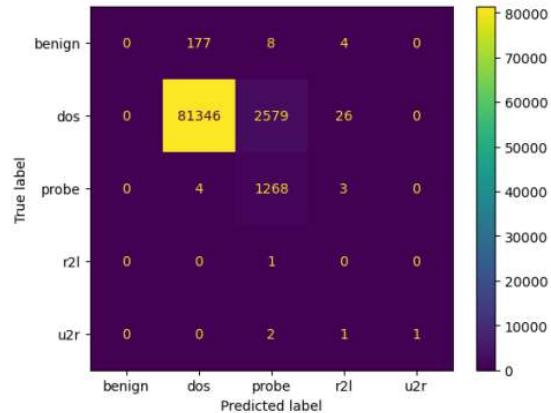
Con RF non abbiamo nessun miglioramento significativo, visto che il problema era quello della difficoltà di distinguere gli attacchi dos da quelli di probe. Riesce però ad avere delle prestazioni accettabili per quanto riguarda gli attacchi minoritari.

SVM lineare non funziona bene. Ha gli stessi difetti di RF, con la differenza che l'unica istanza di r2l la classifica come dos. È peggiore di RF.

## Second layer: RBF SVM



## Second layer: KNN (k=5)



SVM non lineare non ha prestazioni soddisfacenti e r2l è completamente misclassificata.

KNN funziona abbastanza bene dal punto di vista del richiamo, ma solo su dos e probe. La precisione su dos è eccellente. Prestazioni poco soddisfacenti anche sugli attacchi minoritari.

Con KNN all'inizio avevamo un tempo per istanza di circa 11 ms. In questo caso il tempo per istanza si è abbassato, è di 7,4 ms. È ovvio che ci mette meno tempo perché è stata ridotta la dimensionalità del dataset su cui opera KNN.

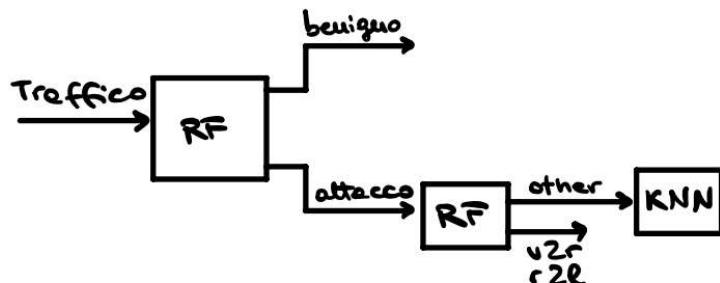
Prima quegli 11 ms era il tempo medio per classificare ogni tipo di istanza; adesso, abbiamo bisogno di classificare solo quelle istanze passate come potenziali attacchi dal classificatore di primo livello.

→ L'approccio migliore potrebbe essere quello in cui, sulla base del classificatore del risultato di primo livello, potremmo mettere più classificatori di secondo livello in parallelo. Si va poi a vedere il risultato della classificazione di secondo livello.

Se il risultato è r2l o u2r, dove sappiamo che RF funziona abbastanza bene, prendiamo la classificazione di RF.

Se invece il risultato è dos prendiamo la classificazione di KNN.

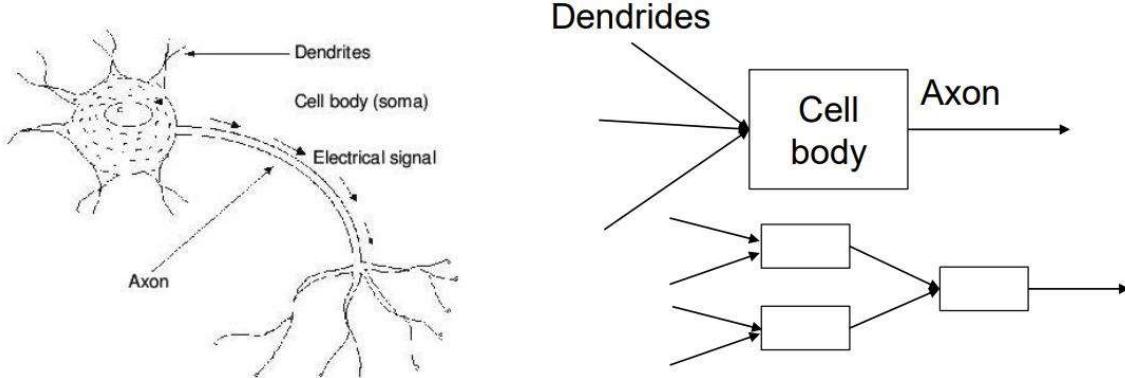
Se è probe sappiamo che la stima è poco precisa, è più affidabile KNN rispetto a RF, per quanto sia comunque poco affidabile.



## RETI NEURALI

### Cos'è una rete neurale

Vediamo un disegno schematico di un neurone.



Un neurone è composto da tre parti:

- Dendriti: rami di input. Possono avere migliaia di rami, generalmente corti.
- Corpo: la parte centrale, che prende il nome di soma. Corpo cellulare che contiene le informazioni sul DNA nel nucleo.
- Assone: ramo di output. Ha una struttura lunga, che all'estremità si divide in possibili migliaia di rami.

Questo è lo schema di un neurone singolo. Il neurone si attaccherà poi a tanti neuroni tramite le terminazioni finali dell'assone che si connettono con i dendriti di un altro neurone.

Le reti neurali sono unidirezionali: il segnale entra dai dendriti, viene processato all'interno del corpo della cellula neuronale e poi, se l'elaborazione è positiva, viene inviato il segnale lungo il ramo di uscita, che è uno e poi si biforca alla fine (dal punto di vista logico è come se ce ne fossero m).

Come si comporta un neurone:

- Input: il neurone raccoglie i segnali da altri neuroni attraverso i dendriti.
- Processore: i segnali vengono accumulati ed elaborati dal corpo cellulare.
- Output: se la forza dei segnali in entrata è sufficientemente grande, il corpo cellulare invia un segnale all'assone.

Il segnale sul ramo di uscita è un picco di attività elettrica.

Il neurone è quindi un elemento di elaborazione non lineare, con n rami in ingresso e m rami in uscita.

L'unica cosa da sapere è che la connessione che avviene tra le terminazioni finali dell'assone e i dendriti del prossimo neurone non è una connessione diretta. I neuroni, perciò, non si toccano l'uno con l'altro.

Il processo che si ha è quello che prende il nome di comunicazione sinaptica. La sinapsi è lo spazio tra il terminale dell'assone e il terminale del dendrite.

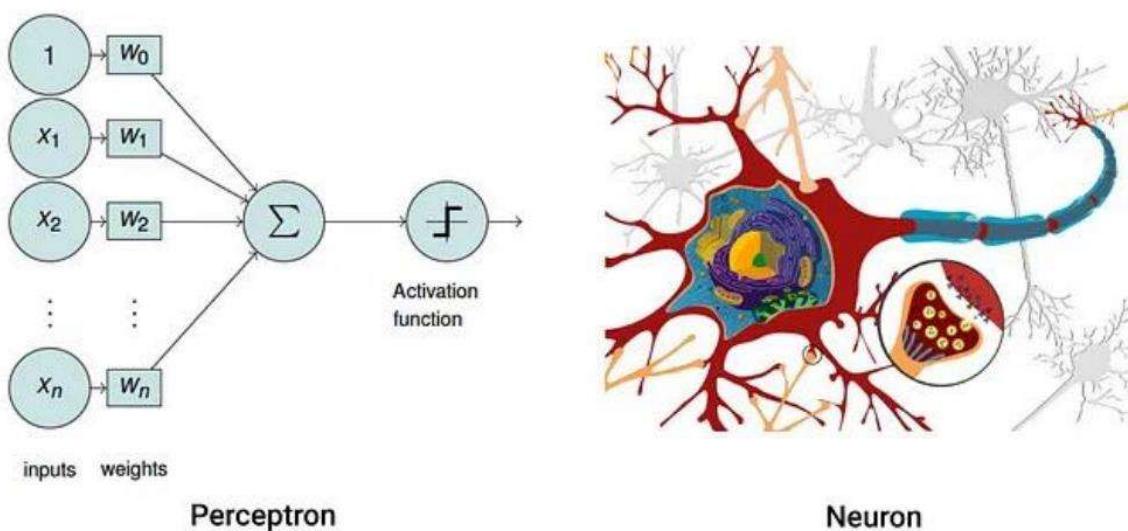
L'attività elettrica arriva fino in fondo sull'assone, stimola la produzione di ioni che vengono raccolti dentro delle vescicole sull'assone. Le vescicole arrivano sulla superficie dell'assone, si fondono e rilasciano sullo spazio sinaptico questi ioni.

Gli ioni, per diffusione, cioè per moto browniano, percorrono lo spazio e arrivano ai recettori del dendrite.

Il segnale elettrico diventa segnale chimico e, quando un numero sufficiente di ioni viene ricevuto dal dendrite, genera di nuovo una corrente che arriva al soma.

Comunicazione elettrica -> chimica -> elettrica.

Questo non ha impatto sul modello teorico del neurone per le reti neurali che è il percepitrone.



Abbiamo un certo numero di segnali di input, che sono quelli che vengono dai neuroni che stanno a monte. Questi segnali di input sono scalati sulla base di quello che attraversa lo spazio sinaptico (coefficiente  $w$ ), poi sono sommati tutti insieme, eventualmente anche scalati tramite un contributo di Bias, e poi vengono passati su una funzione di attivazione che è non lineare.

È importante che sia non lineare perché così è possibile mappare una relazione ingresso-uscita anche complessa, anche fortemente non lineare, qualsiasi.

Esiste un teorema che dice che una rete neurale con un numero molto elevato di neuroni in un unico livello nascosto può mappare qualunque relazione tra ingresso e uscita.

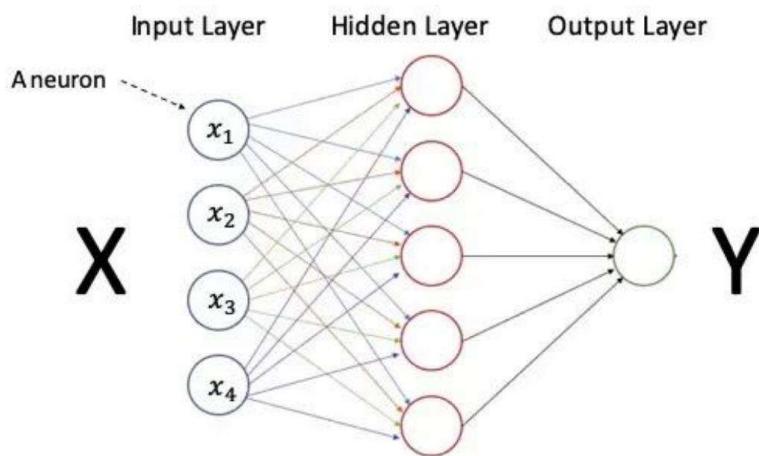
Con delle funzioni non lineari è possibile trovare questo mapping.

Le funzioni di attivazione, per essere matematicamente gestibili, devono essere derivabili. La funzione segno, che è nel disegno, è non derivabile.

Di solito, anche se il percepitrone nasce con la funzione segno, essendo questa non derivabile crea dei problemi nell'utilizzo in schemi più complessi e quindi non viene utilizzata.

## Rete neurale artificiale

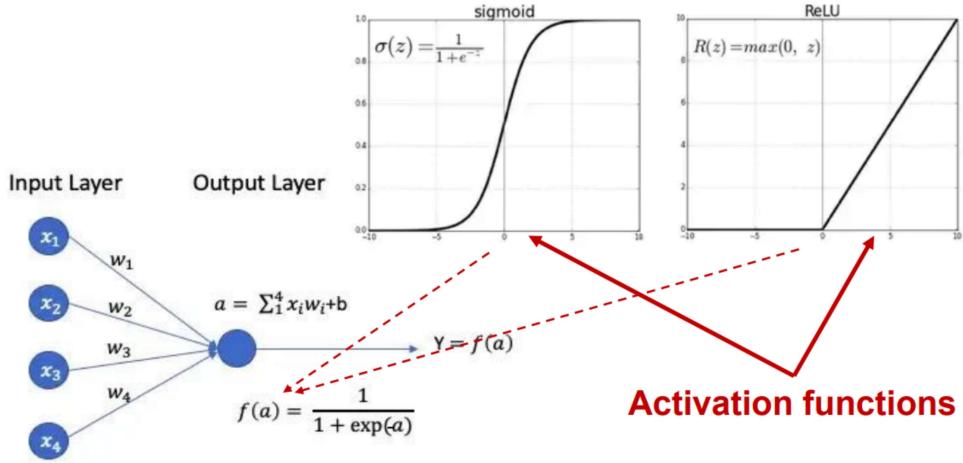
Questo schema, assumendo che ciascun elemento della rete neurale sia assimilabile a un percepitrone, può essere riassunto da questo diagramma:



Si parla di tre tipi di neuroni (i neuroni sono diversi a seconda di dove si trovano):

- Neuroni di ingresso: non applicano linearità, sono un artificio matematico per modellare una feature.
- Neuroni dello strato intermedio (livello nascosto): questo strato viene realizzato tramite un numero variabile di percetroni con funzione di attivazione non necessariamente uguale alla funzione segno, ma normalmente sempre con funzioni non lineari.
- Neuroni di uscita: sono veri e propri neuroni, tipicamente realizzati tramite percetroni. Raccolgono tutti gli input degli strati intermedi e li elaborano per effettuare una decisione.

Se abbiamo un neurone finale significa che la decisione è binaria; se invece c'è bisogno di effettuare una stima multiclasse, allora dovremo avere un numero di neuroni pari alle classi da stimare e viene utilizzata la funzione softmax.



Quello che arriva a un neurone viene pesato attraverso dei coefficienti. Si ottiene che in input, il nucleo del neurone (dove viene fatta l'elaborazione) ha la somma pesata degli input più un termine di Bias (a o z, a seconda dei formalismi). Il tutto viene poi passato attraverso una funzione di attivazione f, che darà come risultato Y.

Una tra le funzioni più utilizzate è la sigmoide, la cui espressione è:

$$\frac{1}{1 + e^{-a}}$$

dove a è il termine di articolazione dei segnali di ingresso. È ovviamente non lineare.

La sigmoide è un'approssimazione continua e derivabile della funzione segno del percettore originale.

L'altra funzione molto utilizzata è la funzione lineare rettificata, ReLU. È una funzione non lineare perché è composta da due spezzate. Tutti i valori di a minori di 0 vengono saturati a 0, mentre tutti i valori di a maggiori di 0 vengono lasciati passare senza modifiche. È continua e non derivabile, ma impostando artificialmente il valore della derivata in zero, essendo continua diventa anche derivabile.

La rete neurale collappa alla regressione logistica nel caso di un singolo neurone senza strati nascosti.

## Recap

Un autoencoder è una rete neurale (interstadio?) in cui la caratteristica principale è che il numero di neuroni dei livelli nascosti risulta avere un profilo con una strettoia centrale e un allargamento laterale.

La classica caratteristica dell'autoencoder è che il numero dei neuroni di input è uguale al numero di neuroni di output, proprio perché l'autoencoder mira a ricostruire quello che trova in input attraverso la codifica delle informazioni latenti nello stadio centrale.

È fondamentale la strettoia perché altrimenti non avrei un modello sintetico dell'informazione, ma avrei un modello ampio in cui passa tutto, sia l'informazione che tutto quello che è legato all'informazione, senza essere però fondamentale per ricostruire il segnale di input.

## Autoencoder per la rilevazione delle anomalie

Per quanto riguarda la rilevazione delle anomalie l'idea è la seguente: addestro la rete locale fatta in questo modo solo con campioni non anomali in modo tale che quello che vado a costruire nella fase di addestramento è un modello che rappresenti il funzionamento del sistema in condizioni normali.

L'obiettivo è avere che l'output, nonostante la strettoia, sia più fedele possibile all'input, quindi vado a modellare il traffico benigno (senza anomalie) e a ricostruirlo nel modo più fedele possibile.

Se addestro la rete a riconoscere e ricostruire il traffico benigno, nel momenti in cui ci passo del traffico "anomalo" la rete non funzionerà bene, proprio perché è stata addestrata a riconoscere solo traffico benigno.

La conseguenza sarà che i valori delle features in uscita saranno abbastanza differenti da quelli di ingresso, e quindi mi daranno un valore dell'MSE alto. Nel momento in cui ho un errore di costruzione significativo ho un campanello di allarme e quindi potenzialmente ho un'anomalia.

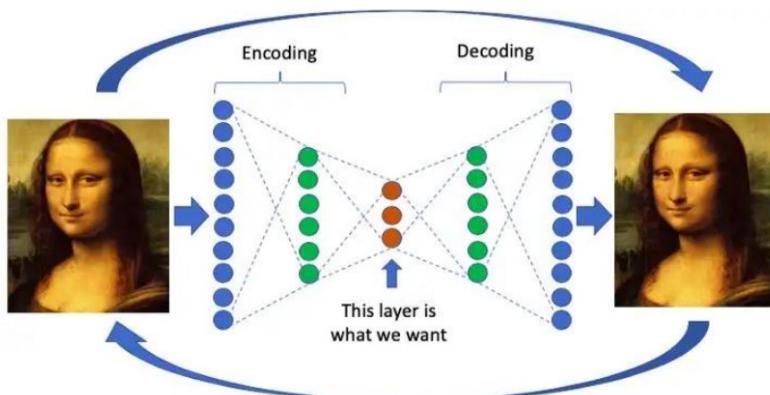
Diciamo che non è un approccio supervisionato perché ho le etichette solo di un tipo di traffico, che è come non averle.

Non è complicato avere campioni di traffico normale, sono le anomalie, alcune delle quali possono essere molto rare, che potremmo non avere a disposizione. Oppure potremmo non averle a disposizione perché ancora non sono state viste in nessun dataset.

Questo può essere un approccio per rilevare un attacco zero-day, che per quanto sia ben fatto sarà leggermente diverso dal traffico benigno.

- ➔ L'idea è addestrare la rete solo con traffico benigno e tutto quello che è diverso sarà un'anomalia: è un rilevatore generico di anomalie. Che tipo di anomalie ce lo dirà qualcun altro.

Concetti di base: rete neurale, rete feed forward (viene attraversata solo in una direzione, senza cicli e senza percorsi chiusi), si addestra con l'algoritmo della back propagation e deve avere un profilo decrescente fino alla strettoia, che è il cuore centrale, e poi crescente. Quanti livelli hidden ci metto? Dipende. Posso metterne 3, come nell'immagine, o se servono anche di più. I livelli che precedono la strettoia aiutano a generalizzare l'informazione che viene dalle features, poi il livello centrale, che sarà il più piccolo, costruirà il livello più astratto possibile e da lì in poi si va in costruzione, quindi a partire dal modello astratto cerchiamo di ricostruire quello che ci aspettiamo che era stato presentato in ingresso.



Lo strato nascosto è il nucleo dell'autoencoder. Quando il numero di neuroni negli strati nascosti è inferiore a quello degli strati di input, gli strati nascosti estrarranno le informazioni essenziali dai valori di input. Ciò costringe gli strati nascosti ad apprendere la maggior parte dei modelli di dati e a ignorare il rumore.

In un modello AE, i livelli nascosti devono avere dimensioni inferiori rispetto a quelle dei livelli di input o di output.

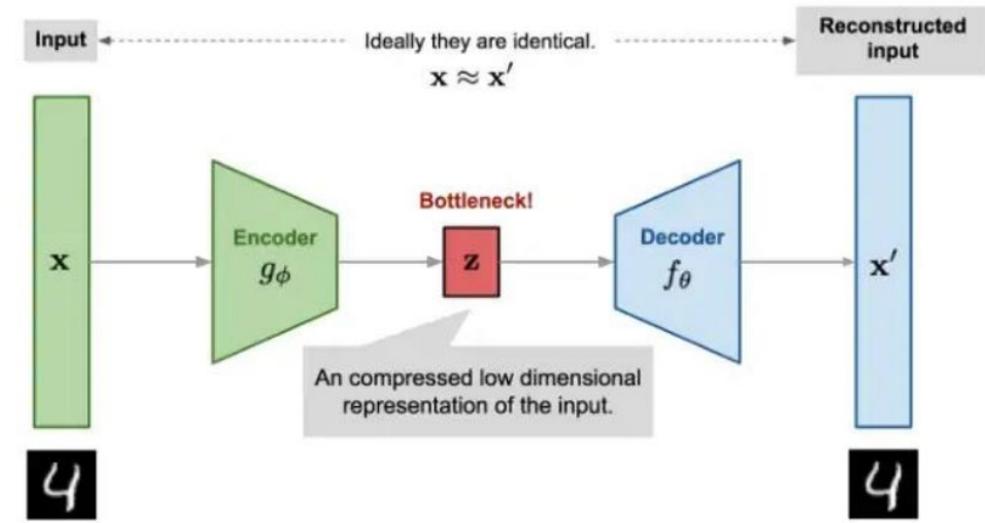
Il processo di codifica comprime i valori di input per arrivare allo strato principale.

Il processo di decodifica ricostruisce le informazioni per produrre il risultato.

La maggior parte dei professionisti adotta semplicemente questa simmetria.

AutoEncoder è un algoritmo generativo di deep learning non supervisionato utilizzato per ricostruire dati di input ad alta dimensione utilizzando una rete neurale con uno stretto strato di collo di bottiglia al centro contenente la rappresentazione latente dei dati di input

## Autoencoder



Schema di base: feature (neuroni in ingresso), encoder (funzione di encoding), collo di bottiglia, decoder (funzione di decoding, tipicamente simmetrica rispetto a quella di encoding) e ricostruzione.

$$MSE(x, x') = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2$$

$x$  è quello che sta in ingresso,  $x'$  quello che sta in uscita

È fondamentale che prima di effettuare l'addestramento utilizziamo uno scaling, che in questo caso quello raccomandato è min-max, tipicamente tra 0 e 1, perché devo evitare che errori diversi, potenzialmente poco significativi per una feature ma in magnitudo grandi rispetto a un'altra feature, possano andare a dominare altri fenomeni non rilevanti.

### Rilevamento anomalie tramite Autoencoder

Vediamo cosa fare dal punto di vista pratico. Se l'errore sarà superiore ad una certa soglia avrò un'anomalia. La scelta della soglia è uno dei problemi classici, qui abbiamo un approccio metodologico preciso per la scelta della soglia.

Scegliamo la soglia a valle del processo di addestramento (di training) in modo tale da rendere il livello centrale il più skilled possibile per ricostruire il nostro modello. Significa che avremo un MSE il più basso possibile.

- ➔ Anche se è un problema di classificazione, noi lo tratteremo come un problema di regressione. Siamo interessati a ricostruire un valore per ogni feature, e quindi abbiamo un output di cardinalità pari all'input.

Non andiamo a utilizzare, come nei classici problemi di classificazione, la cross entropy e le funzioni soft max, ma usiamo l'MSE, che ci garantisce la possibilità di andare a discriminare quanto l'input è diverso dall'output.

Per input significativamente diversi dall'output mi aspetto che l'input possa essere un'anomalia, proprio perché non fitta il modello che ho addestrato con del traffico di tipo trusted.

Passaggi per rilevare anomalie in un dataset.

Durante l'addestramento, inserire nell'Encoder solo traffico normale (nessuna anomalia!!)

Il livello del collo di bottiglia apprenderà la rappresentazione latente dei normali dati di input.

Anche se è classificato come unsupervised, non è problematico addestrarlo al traffico normale, ne abbiamo in abbondanza.

Il decoder utilizzerà l'output dei livelli del collo di bottiglia per ricostruire le normali transazioni dei dati di input originali.

Una transazione fraudolenta sarà diversa da una transazione normale. L'Autoencoder avrà difficoltà a ricostruire il traffico dell'attacco o le anomalie in generale e quindi l'errore di ricostruzione sarà elevato. L'input sarà significativamente diverso dall'output poiché si tratta di dati imprevisti.

Un evento di input viene contrassegnato come basato su un'anomalia se presenta un errore di ricostruzione > soglia.

La soglia può essere generata valutando il k-esimo percentile, ad esempio, di MSE.

Creazione delle statistiche di una metrica osservata: MSE viene generalmente utilizzato per problemi di regressione. Quantifica la distanza tra il campione originale  $x$  e quello ricostruito  $x'$ .

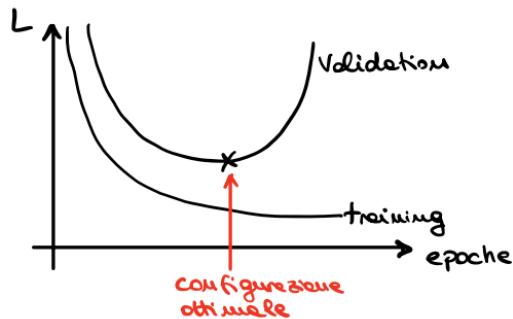
$$MSE = \frac{1}{N} \sum_{i=1}^N (x_i - x'_i)^2$$

*i-th feature of the dataset sample x*

Selezionare un livello accettabile di falsi positivi per determinare automaticamente una soglia T.

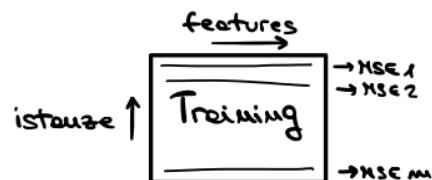
I campioni con  $MSE > T$  verranno classificati come ANOMALIE.

La loss sarà l'MSE, ma non posso usarla in maniera diretta, perché l'MSE non ci dice quando è alto o basso, non abbiamo un termine di paragone. Devo trovare quindi un modo per costruire la soglia.

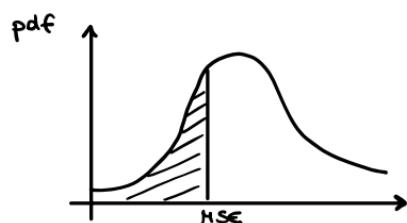


Fermo l'addestramento quando trovo il minimo della funzione di perdita della validation, perché da quel momento in poi, se continuo ad addestrare il modello andrò in overfitting.

Durante l'addestramento, e in particolare per la configurazione ottimale (quella con la loss più bassa per la validation), vado a registrare i valori dell'MSE.

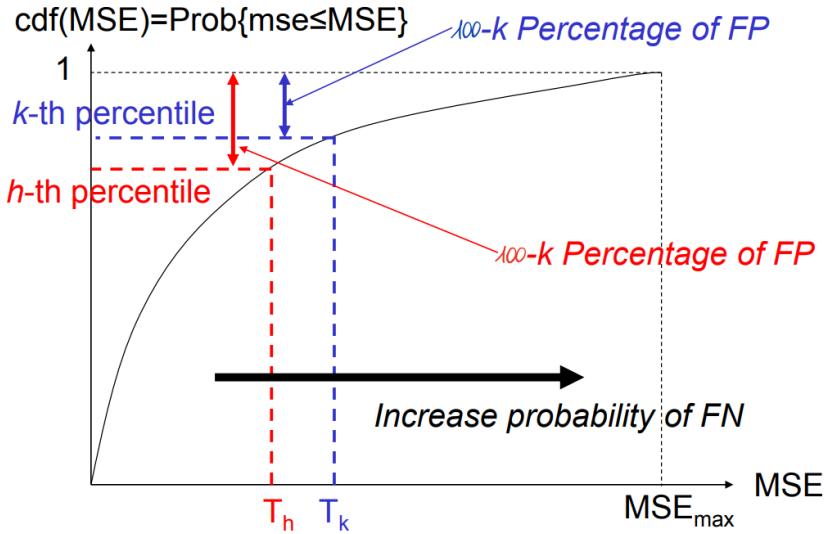


Per ogni istanza avrò il rispettivo MSE; alla fine della fase di training avrò m (m = numero delle features) campioni, quindi m valori dell'MSE. Con questi campioni vado a costruire una curva di distribuzione empirica dell'MSE, cioè la curva che approssima la distribuzione di probabilità dell'MSE (è una curva sempre crescente).



La cdf (distribuzione cumulativa di probabilità) per un determinato valore di MSE è la probabilità che l'MSE sia minore o uguale a quel valore. La cdf fino a un certo valore dell'MSE sarà pari all'area sotto la pdf (per questo la cdf sarà sempre una curva non decrescente!!).

Nel grafico in ascissa abbiamo l'MSE e la curva non è infinita, dipende dai campioni raccolti durante la fase di training e quindi in ascissa i valori dell'MSE vanno da 0 fino al valore massimo osservato. I campioni minori o uguali al massimo sono tutti, cioè 100%, cioè 1.



A questo punto prendiamo la nostra decisione, che sarà basata sul concetto di percentile. Ad esempio, voglio che la soglia corrisponda al 95° percentile della distribuzione dell'MSE. Segno quindi la percentuale di interesse, traccio una linea orizzontale, e poi tiro giù l'intercetta verticale. Quello che leggo sull'asse dell'ascisse in rosso,  $T_h$ , è la mia soglia: ci dice che la probabilità con cui nel dataset di training ho visto valori di MSE inferiori a quella soglia è pari a  $h$ . Se  $h = 95$  significa che la probabilità di avere dei valori di MSE inferiori a quella soglia è del 95%.

Se alzo la soglia (troppo bassa non posso metterla altrimenti non rileverò niente, devo stare abbastanza vicino a 1), ad esempio considero  $k = 99$  otteniamo che se alzo l'ordinata, visto che la funzione di distribuzione è crescente, si sposta verso destra l'ascissa. Quindi per raccogliere il 99° percentile dovrò avere una soglia più grande perché la curva cresce. Se voglio il 100° percentile la soglia sarà il valore massimo. Non posso prendere il massimo perché sappiamo che vicino al massimo posso cominciare ad avere qualche anomalia.

Esistono sempre i falsi positivi, in questo caso saranno quelli che hanno generato un errore di costruzione un po' più grande, vicino al massimo. Il punto è: quanti falsi positivi tollero? I falsi positivi stimati sui dati di training sono il complemento a cento della soglia di percentile. Se decido che calcolo la soglia sul 95° percentile significa che quelle istanze che avranno un errore di costruzione inferiore a quella soglia saranno classificate come traffico normale, quelle che avranno un errore di costruzione più grande della soglia saranno taggiate come potenziali anomalie. Ma fino adesso le anomalie non ci sono perché ho addestrato il sistema con traffico benigno, quindi mi lascio una percentuale nota a priori di falsi positivi, cioè di traffico benigno etichettato potenzialmente come anomalia, e ci convivo. Decido perciò a propri quale sarà la percentuale di falsi positivi.

È una percentuale realistica? Dipende dal fatto che il campione di traffico benigno che ho fornito al sistema sia rappresentativo di tutto il traffico benigno o solo di quello che avevo a disposizione. Se ho fatto un buon lavoro mi devo aspettare che la percentuale di falsi positivi non sarà casuale, ma sarà legata al valore di percentile che ho scelto per determinare la soglia.

Quindi se ho un valore di  $k$  pari a 99 ho solo l'1% di falsi positivi, ma più tiro su la soglia più tollero errori di costruzione grandi e quindi potrei avere delle anomalie con un errore di costruzione non troppo grande che potrebbero ricadere a sinistra della soglia (i falsi negativi). In tutti questi casi per tenere bassi i falsi positivi sto incamerando dei falsi negativi, che nel mondo della rilevazione delle anomalie sono più negativi.

! FINE TEORIA !