**Traffic monitoring**

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.

**Dati di rete**

* Un'analisi di rete efficace richiede la raccolta dati provenienti da fonti eterogenee
  + ognuno fornisce parte di un'immagine su un particolare evento che si svolge su una rete

Quando si parla di traffic 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 nell’analisi basata sui dati è: raccogliere dati sufficienti per ricostruire eventi rari senza raccogliere così tanti dati da rendere le query impraticabili

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.
  + Nella security, questo problema è complicato dalla rarità delle reali minacce alla sicurezza: 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.

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

- Fault

- Configuration

- Accounting

- Performance

- Security

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.

**Dati di rete e minacce**

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.

Gli attacchi ci sono, ma la maggior parte sono innocui (scanning, spam e così via).

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.

Gli attacchi sono comuni, le minacce sono rare. Solo un piccolo sottoinsieme rappresenta minacce reali, Più difficile da rilevare!

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 generale della struttura dell'analisi di rete**

**Immagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamenteImmagine che contiene testo, diagramma, Piano, linea

Descrizione generata automaticamente**

In questo schema abbiamo 5 componenti fondamentali:

* **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 dispostivi 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 o inviato 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 elaborali 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 streaming, 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 impattare troppo sul costo complessivo del servizio.

1. **Tipi di dati**

**Archivio**

Esistono delle macrocategorie di dati che posso memorizzare e sono:

* 1. *Dati relativi al traffico di rete*

Posso raccogliere statistiche di tipo SNMP, statistiche relative ai flussi (tramite soluzione NetFlow) 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 una query 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.

* 1. *Dati relativi alle 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.

* 1. *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 log. ERROR invece è il più limitato e scrive una riga nel file di log solo quando si verifica un errore.

1. **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 importante 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 fare transazioni di sola lettura o cancellazione, mentre solo le sonde (che possono essere log di sistema e servizi, catture di traffico e informazioni di stato) possono scrivere nuovi record. Quindi, una volta che transazione ha avuto luogo non è possibile modificare il record, il dato rimane immutato, al massimo può essere cancellato, ma mai alterato; 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 si scrive sul disco.

In che cosa consiste la transazione?

La transazione da parte dei sensori è data da operazioni di scrittura oppure di sola lettura da parte degli utenti, tramite il query processor. L’*isolamento* deve valere perché sensori diversi possono scrivere in parallelo e la *durabilità* deve valere perché, anche se il sistema è soggetto a una failure, una volta che una transazione è andata a buon fine il dato deve rimanere.

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 cancellati e sostituiti con le loro elaborazioni, ma cancellare un dato è diverso da modificarlo con una transazione.

**Annotazioni**

Le annotazioni sono un modo per aumentare i dati con una sorgente diversa, che non dipende dalle sonde che catturano traffico.

L’esempio classico è quello della geolocalizzazione oppure, per quanto riguarda la sicurezza, l’aggiunta di ulteriori basi di 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 scollega 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.

I dati di annotazione si riferiscono alle informazioni che utilizzi per integrare i dati del tuo archivio. Ad esempio, Intelligence sulle minacce, dati di geolocalizzazione, rete reputazione e altre forme di mappatura come il DNS repository.

I dati delle annotazioni differiscono dai dati degli eventi in quanto ha un orario valido:

* Il proprietario di un indirizzo IP può essere un'organizzazione dal 5 al 10 marzo e un altro l'11 in poi
* Il coordinamento di queste informazioni temporali è fondamentale quando considerando l'annotazione

I dati dell’archivio sono immutabili, mentre i dati delle notazioni non necessariamente sono immutabili (vedi l’indirizzo IP che può essere utilizzato da owner diversi).

Combinare con l'archivio o tenere separati? Dipende anche da come voglio fare le query.

**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 avere solo le tracce di traffico e nient’altro, oppure potrei avere le tracce di traffico e 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 l’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 utilizzate un processo funzionante o ancora più semplicemente sto facendo riferimento a dati troppo distanti nel tempo e non più presenti in memoria.

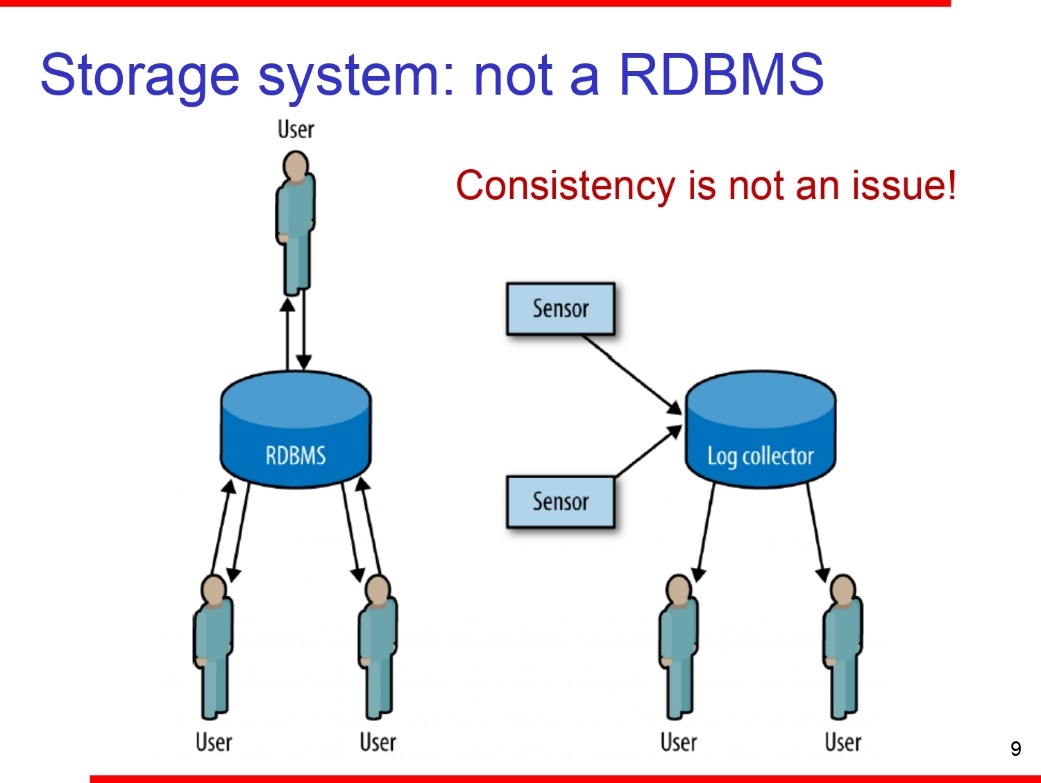
Nell’archivio quindi metto i dati di evento, dove con evento si intende qualcosa di estremamente generico come, per esempio, la generazione di un record SNMP, un pacchetto catturato, una statistica di traffico aggiornata, un antivirus che genera un report di una minaccia; un evento è quindi qualcosa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

KB è costituito da informazioni e giudizi che l'organizzazione si è sviluppata nel tempo; informazioni specificamente rilevanti per il target impresa (es: inventari cespiti, dati del personale, calendari interni).

I dati KB sono in genere non strutturati e di solito relativamente piccoli. Potrebbe essere in gran parte moderato dalle persone.

Fine 27/09

**Sistema di storage: non un RDBMS**

****

**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 sintetizzarlo.

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.

Immagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamenteInfatti, 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 rime è efficiente oppure produce dei dati con informazione troppo compressa e quindi perdita di informazione stessa.

Il pre-processing può semplicemente scrivete 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 di raccolta dei dati:

* relativi alla rete (network sensing):
  + sonde associate ai NIDS (Network Instrusion 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 Instrusion 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 stringe 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.

**Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente**

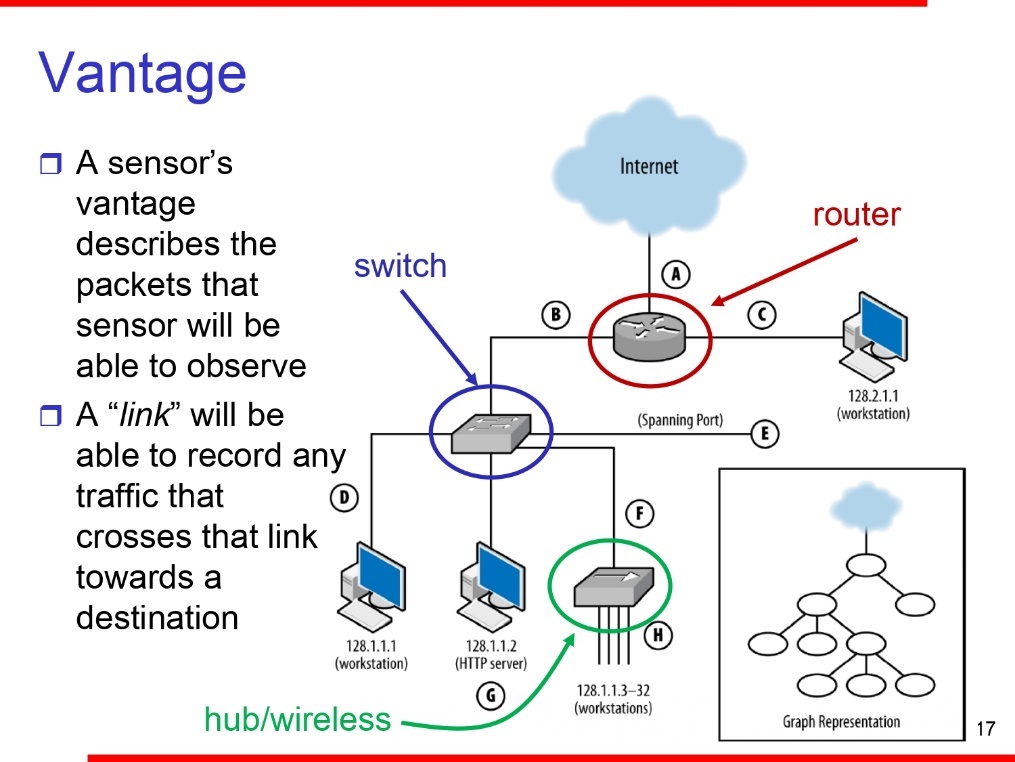
(+ esempio fatto in aula con disegno)

**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.

Un “link” sarà in grado di registrare tutto il traffico che attraversa quel collegamento verso una destinazione.

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.



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 vero 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 e 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.

04/10

**Osservazioni**

Catturare su un collegamento spesso non ci permette di avere una visione completa del contenuto di ciò che catturo. Potrò vedere al massimo le intestazioni di strato 2, 3 e 4 perché tipicamente non riesco a vedere il payload dello strato applicativo. Questo perché ad oggi la maggior parte dei protocolli applicativi utilizza la cifratura e quindi senza avere a disposizione la chiave di cifratura (spesso valida da sessione a sessione) non riuscirò ad avere accesso alle informazioni che stanno sul payload. Se tuttavia sono in possesso della chiave di sessione allora tramite una cattura dei pacchetti sarò in grado di decodificare anche il payload dello strato applicativo e quindi avrò accesso a tutti i dati.

In generale non catturo tutti i byte del pacchetto, ma solo quelli relativi all’intestazione perché è l’unica cosa che riesco a decodificare.

Un caso notevole è quello delle connessioni wireless. Se ho un accesso wireless normalmente riesco a vedere solo le informazioni di strato 2 perché la cifratura viene apposta tra lo strato 2 e lo strato 3 e quindi l’intestazione IP è già cifrata.

In alcuni casi, per quanto riguarda le reti in area locale (Wi-Fi) è possibile accedere alla chiave di cifratura, solo nei casi in cui ho un’infrastruttura di tipo personale. La chiave, PSK (pre-shared key), è configurata sia sull’access point sia sui vari dispositivi, cioè su tutti gli utenti che devono accedere. Se si ha a disposizione la PSK gli analizzatori del traffico (es: wireshark) vi permettono, andando a inserire la chiave in un campo specifico all’interno delle varie configurazioni, di decodificare il traffico, e quindi vedere non solo lo strato 2 ma anche lo strato 3 e 4.

Per quanto riguarda invece le reti wi-fi di tipo enterprise invece non utilizzano questa modalità, ma utilizzano un protocollo di autenticazione che permettono la negoziazione automatica della chiave volta per volta (802.1X e vrod). Di fatto, non permette questo meccanismo perché ogni utente avrà delle credenziali diverse associate al profilo e quindi per decodificare il traffico di utente si dovrebbero avere le credenziali dell’utente e intercettare la prima parte del traffico, i primi pacchetti dove viene negoziata la chiave di sessione, senza di quelli non si ha accesso a nulla.

Per le reti cellulari la cosa è ancora più complicata. Ogni utente ha la sua chiave che deriva dalla chiave segreta che sta nella sim e che tipicamente non è estraibile. In quel caso, a meno che non siate l’operatore, non è possibile decodificare il traffico.

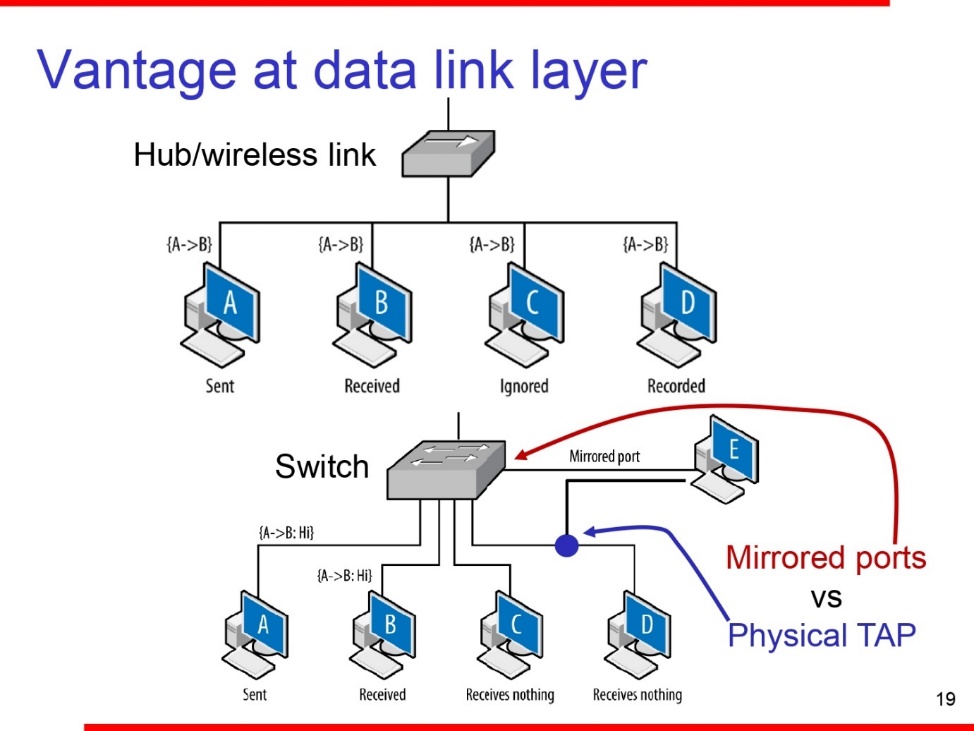
La cosa è diversa se invece di piazzare delle sonde dentro i dispositivi di rete, quelli che vengono attraversati dal traffico, vado a piazzare le sonde nei sistemi terminali (è il caso di G). C e D si riferiscono ai collegamenti tra le workstations e il dispositivo di rete a monte (router per C e switch per D). Per quanto riguarda la cattura in G invece, la cattura viene fatta sul server e non sul link e avrò accesso quindi ai pacchetti decodificati. Posso sia effettuare la cattura dei pacchetti in chiaro e sia aumentare l’informazione con i log del servizio.

**Come selezionare un punto di osservazione**

Quindi la procedura in generale per determinare il vantaggio è un processo in tre fasi:

1. Acquisizione di una mappa sia logica sia fisica della rete e di un elenco di potenziali punti di strumentazione
2. Determinazione dei potenziali punti di osservazione, quindi capire dove piazzare fisicamente le sonde (hardware o software)
   * implica l'identificazione di ogni posizione potenzialmente strumentabile sulla rete
   * determinare cosa può vedere quella posizione (L2/IP/TCP/app)
3. Determinazione della copertura ottimale (determinare il numero minimo di sonde che permette di catturare tutto ciò che mi interessa monitorare)
   * L'obiettivo è scegliere una serie di punti che forniscano il monitoraggio con una ridondanza minima.

A volte può essere limitato utilizzando regole di filtro.

**Vantaggio a strato di collegamento**

Se il terminale A invia del traffico al terminale B, il terminale C riceve comunque il traffico inviato da A a B ma lo ignorerà. Se piazzo una stazione di monitoraggio sul terminale D, o il terminale D è esclusivamente una stazione di monitoraggio, riceverà quello che viene inviato da A e da B. Questo vale al 100% nel caso dell’hub, ma non vale al 100% nel caso di una rete wireless. Se la stazione di monitoraggio D è lontana, siccome nella tecnologia moderna vengono utilizzate le antenne direttive, e quindi potrebbe non ricevere tutti i segnali. In questo caso la cattura potrebbe essere problematica.

Lo switch risolve il problema dell’hub e invia a tutti il traffico che dovrebbe essere confinato tra due porte. Lo switch tipicamente non ha la possibilità di installarci una sonda.

Gli switch sono di due tipi:

* managed: più sofisticati, hanno la possibilità di attivare SNMP in tutte le sue versioni oppure effettuare il mirroring delle porte
* unmanaged: poco costosi, funzionano da soli senza dover fare niente

Lo switch quindi tipicamente non riesce ad effettuare una cattura esso stesso, a meno che non ho uno switch virtuale (realizzato in software); in quel caso ho a disposizione tutto l’ambiente di calcolo per catturare ciò che voglio.

Per switch di tipo fisico ho due possibilità:

1. devo duplicare il traffico che voglio vedere su una **mirrored port** (o spanning port). Copio il traffico decidendo il verso dello stesso e così riesco ad utilizzare diverse modalità. Chiaramente, decidendo il verso mi perdo altro tipo di traffico.
2. Se lo switch ha già raggiunto il numero massimo di porte che possono mandare il mirroring verso una stazione di monitoraggio o se non supporta il mirroring, posso utilizzare dei **Physical TAP**. Questo “rubinetto” fa sgocciolare dal link tutti i pacchetti che ci passano. È un dispositivo fisico, tipicamente a più porte, che copia tutto il traffico che passa. Si utilizza inserendo su una porta il cavo che viene dallo switch, sun un’altra il cavo che arriva dal terminale D e si collega la stazione di monitoring alla terza porta del physical tap.

Per far questo devo avere accesso alla locazione fisica dello switch o del tap. Il physical tap viene piazzato in genere vicino allo switch oppure può essere messo vicino a D.

Tutto quello che riguarda l’accesso ai dati che fluiscono in rete è sotto il controllo dell’amministratore di rete, soprattutto se i dispositivi sono protetti, non dal punto di vista informatico ma dal punto di vista dell’accesso fisico (ad esempio sono dentro un armadio chiuso con un lucchetto).

* 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 ho bisogno di agganciarmi alla stazione radio base/access point oppure mettere in opera una stazione di cattura che funziona bene tutte le volte che è molto vicina alla stazione radio base/access point, riuscendo così a catturare una porzione molto significativa del traffico.

**Vantaggio a livello IP**

**Immagine che contiene testo, schermata, diagramma, design

Descrizione generata automaticamente**

Il primo problema che devo affrontare è capire se l’instradamento del traffico è di tipo simmetrico o asimmetrico.

Nel caso in cui l’instradamento è asimmetrico potrebbe non essere sufficiente piazzare una sonda su uno dei router del cammino perché il cammino di andata e quello di ritorno possono essere diversi. Quindi se sono interessato soltanto al traffico in un verso specifico non ho problemi, invece se sono interessato alla sessione (al traffico in un verso e alla risposta ricevuta nel verso opposto devo fare in modo di piazzare la sonda in una parte del percorso che è condiviso tra l’uplink e il downlink.

L’esempio in basso a sinistra è estremo: A e B sono entrambi connessi a due router e l’instradamento è perfettamente asimmetrico; infatti, il traffico da A a B passa dal router che sta sopra, mentre il traffico da B ad A passa attraverso il router che sta sotto. Se metto la stazione di monitoraggio in C vedrò solo il traffico da A a B, se la aggancio a D vedrò solo il traffico da B ad A.

L’altra informazione fondamentale è quella del TTL. Il TTL è il tempo di vita ed è un numero scelto dal mittente che viene scritto sul pacchetto IP e viene decrementato da ogni router che il pacchetto IP attraversa.

Nell’esempio, se il nodo A invia del traffico con un TTL troppo piccolo al nodo B, succede che il nodo B non riceverà mai nessun pacchetto. Questo potrebbe accadere per effettuare un’operazione di disturbo, per creare congestione sui nodi C e D. Quindi, se voglio creare un attacco che crea disservizio in una rete ma voglio evitare che i sistemi di monitoraggio messi in opera sul server se ne accorgano posso fare in modo, una volta stimato il TTL tra A e B, di inviare una significativa quantità di traffico da A verso B con un TTL leggermente più basso in modo da inondare di traffico i nodi C e D ma non riesco ad accorgermi di questo se ho in opera il sistema di monitoraggio del traffico sul server B. È necessario mettere in opera più punti di monitoraggio, non solo sui sistemi terminali ma anche sui sistemi di rete per avere un’idea di cosa sta succedendo. Utilizzando punti di monitoraggio con vantaggio diverso acquisisco informazioni diverse.

Andando, per esempio, a mettere una sonda su C mi rendo conto di un potenziale attacco di denial of network service (attacco che coinvolge solo i router e non arriva alle postazioni di servizio) che può essere potenzialmente più insidioso di un classico denial of service.

Per ripristinare il servizio in caso di un attacco di questo tipo si può andare a limitare il numero di nuove richieste, cioè nuove sessioni TCP; in questo modo le sessioni già stabilite non verranno toccate, mentre le nuove sessioni possono essere scartate. Questo meccanismo di rate limiting da solo non funziona perché andrò a limitare anche diverse richieste legittime; mi servirà quindi in maniera combinata un limite sia al numero massimo di richieste che può fare un terminale sia al numero massimo di richieste che possono essere sopportate complessivamente.

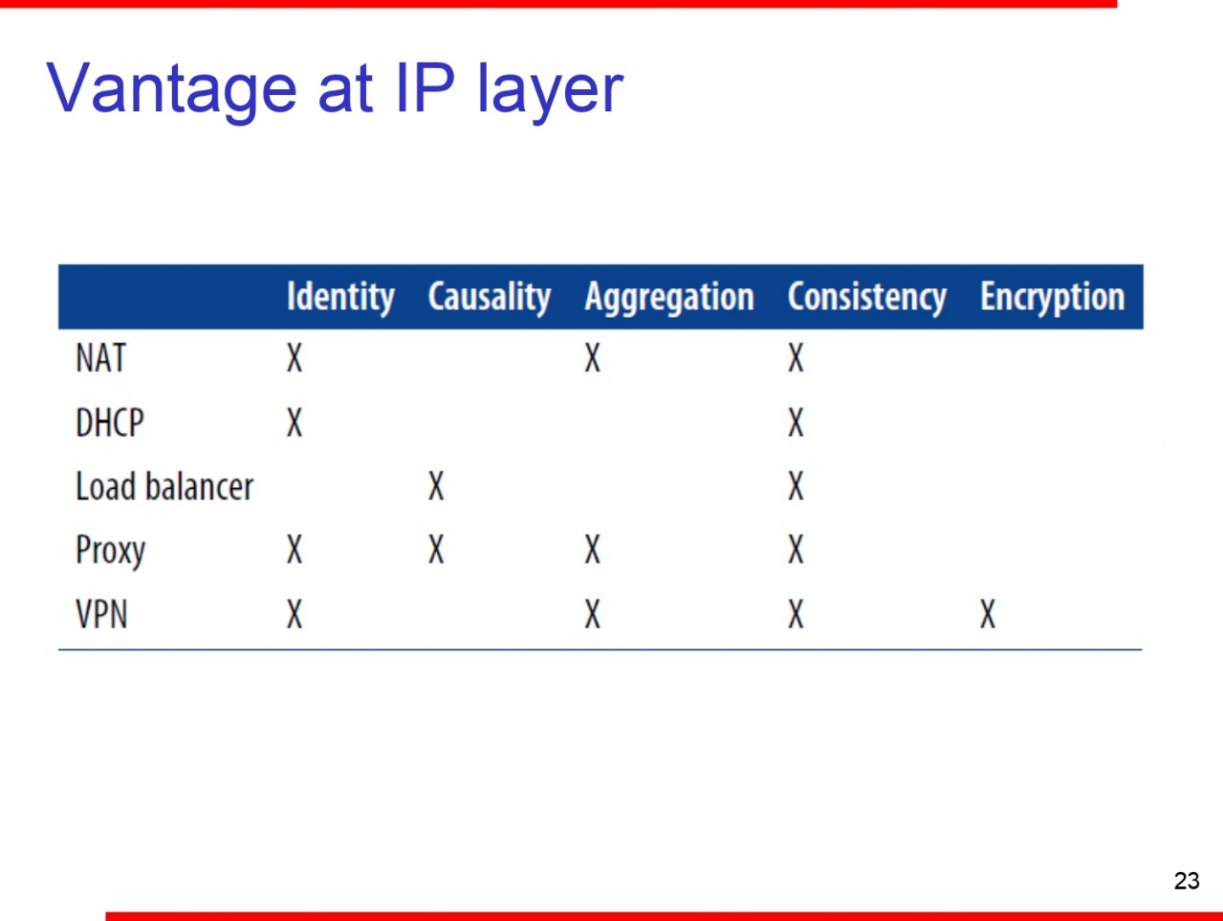
Vero problema del vantaggio dei punti di monitoraggio a livello IP: presenza delle middlebox.

Le middlebox sono “scatole che stanno nel mezzo” e fanno tutte quelle funzioni che tipicamente non dovrebbero essere fatte da un router. Un router ha tipicamente due funzioni: funzione decisionale (instradamento) e attuativa (sposta fisicamente il pacchetto dall’interfaccia di ingresso all’interfaccia di uscita). Siccome si è visto con il tempo che queste funzioni sono insufficienti, sono stati aggiunti ai router altre funzioni che permettono di erogare servizi più avanzati, ma complicano in maniera significativa il compito del monitoraggio.

In particolare, quando un router non si comporta come un dispositivo piatto di strato 3, cioè, instrada solamente i pacchetti, ma svolge altre funzioni possiamo diversi problemi. Le middlebox possono quindi sollevare una serie di problemi relativi alla validità dell'indirizzamento:

* **Identità**: in alcune situazioni, l’identità degli individui non è distinguibile a causa di una operazione di rimapping tra l’interfaccia di ingresso e l’interfaccia di uscita della middlebox
* **Causalità**: le informazioni a valle della middlebox non hanno lo stesso flusso delle interazioni a monte della middlebox; quindi quello che vedo osservando il traffico dopo la middlebox può essere diverso da quello che vedrei se osservassi il traffico a valle, e in particolare a monte potrei avere molto più traffico rispetto a quello che vedo a valle
* **Aggregazione**: la stessa identità può essere utilizzata per identificare più individui contemporaneamente
* **Consistenza**: la stessa identità può cambiare nel corso del tempo
* **Cifratura**: le middlebox sono collegate tra loro attraverso collegamenti cifrati che impediscono di vedere il traffico a livello IP

Le principali funzionalità delle middlebox sono le seguenti:



**NAT**: tipicamente si utilizza lo schema di NAT del molti a uno. Ho un router che fa anche da NAT a cui sono attaccati i diversi client. Tutti i terminali, una volta che il loro traffico ha attraversato il NAT, vedranno mappato l’indirizzo mittente dei propri pacchetti con l’indirizzo esterno del router che fa da NAT. Il nome esatto in questo caso è NAPT (Network Address Port Translation) perché viene mappata la combinazione indirizzo IP mittente-porta mittente con indirizzo IP mittente-porta mittente dell’interfaccia del router.

* Problema di aggregazione perché un unico indirizzo viene utilizzato per mascherarne N
* Problema di identità perché guardando il traffico in certi punti non sono in grado di sapere l’identità dei singoli individui
* Problema di consistenza perché nel tempo l’associazione tra indirizzo del client e indirizzo di NAT può variare; non varia se ho il mapping molti a uno, ma varia se ho a disposizione un pool di indirizzi da poter utilizzare per fare il NAT

**DHCP**: problema di identità e di consistenza allo stesso tempo perché lo stesso nodo, accedendo alla rete in momenti diversi, può avere un indirizzo IP diverso.

**Load balancer**: serve a smaltire delle richieste non verso un server, ma verso più server.

Kubernetes adotta un meccanismo nativo di load balancing (metalLB): quando si aumenta il numero di richieste inviate verso un servizio che sta in uno specifico container, si potrebbe superare il limite impostato per quel pod; se il limite del pod è del 50% della cpu assegnata a quel pod, e questo limite viene superato, automaticamente Kubernetes mette in campo un altro container. A quel punto si hanno due container gemelli che offrono lo stesso servizio. MetalLB decide di volta in volta se una nuova richiesta viene inviata all’uno o all’altro container. Questo è uno scenario molto comune in ambito cloud e non.

Ad esempio, abbiamo visto in laboratorio che ubuntu.com viene risolto su almeno 3 macchine diverse e abbiamo dovuto catturare il traffico all’inizio della sessione per capire da quale dei tre indirizzi stavamo scaricando di volta in volta l’immagine di ubuntu server.

Load balancing può avvenire a livello client (esempio ubuntu.com) ma una volta che arrivate allo specifico indirizzo magari questi server sono nattati e poi di volta in volta viene inviata una richiesta a un server o a un altro. Quando parliamo di richiesta intendiamo una sessione TCP, e sessioni diverse faranno potenzialmente strade diverse.

* problema di causalità perché a monte vedo tutte le richieste ma a valle no in quanto load balancer splitta le richieste su un numero potenzialmente anche variabile di istanze di servizio
* problema di consistenza perché nel tempo il load balancer varia e anche parecchio; in un momento di basso traffico potrei avere un’unica istanza di servizio ma in un momento di traffico di picco quello che vedo su uno specifico link a valle è una porzione bassa di quello che vedo su uno specifico link a monte. Se il traffico è troppo da elaborare, come fa load balancer a gestirlo? Load balancer non deve elaborare la richiesta come i server, ma guarda solo la richiesta e decide dove mandarla

**Proxy**: il proxy può girare sul router oppure il router può essere configurato per direzionare determinate richieste di tipo applicativo a un’altra macchina che fa da proxy. Quello che vedo in ingresso può essere diverso da quello che vedo in uscita perché il proxy termina la sessione tcp; quindi, si sostituisce al server remoto e cambia l’indirizzo di destinazione del pacchetto con il proprio.

A quel punto, supponendo che sia una richiesta web, se la pagina richiesta è già in memoria perché è stata richiesta precedentemente, serve la pagina senza dover attraversare internet e senza consumare la banda sul collegamento di accesso. Se invece la pagina non ce l’ha, sarà il proxy che farà da client nei confronti del server remoto, la recupera e se la mantiene in memoria, in modo che il prossimo che richiederà la pagina la riceverà più velocemente senza consumare la banda.

Questo può essere fatto per migliorare le prestazioni in presenza di link a banda limitata, oppure in presenza di un controllo strettissimo nel traffico di ingresso e di uscita in cui solo alcune pagine o alcuni server possono essere consultati e quindi il proxy sarà poi configurato a servire alcune pagine limitando l’accesso all’esterno (serve anche per monitorare cosa fanno i dipendenti di un’azienda).

Il proxy, quindi, può essere usato o come un sistema di stretto controllo o un sistema di ottimizzazione delle prestazioni o entrambi.

* problema di identità perché il proxy fa la richiesta al server remoto perché le richieste dei client terminano sul proxy e quindi chi fa la richiesta dopo la middlebox è diverso da chi fa la richiesta prima della middlebox
* problema di causalità: non tutte le richieste fatte prima corrispondono alle richieste fatte dopo perché, se il proxy ha il contenuto lo serve in maniera diretta e non genera traffico in uscita
* problema di aggregazione: n richieste possono dare origine a 1 richiesta in uscita perché, se tutti fanno la stessa richiesta il proxy all’inizio non ce l’ha e la recupera, ma poi la mette in memoria e la fornisce direttamente
* problema di consistenza: l’identità cambia nel tempo perché non sono sempre gli stessi utenti a triggerare le richieste

**VPN**: le VPN possono essere fatte in due modi diversi. Nella modalità road warrior, in cui è il client che crea una VPN verso un nodo remoto e accede alle risorse della rete remota (caso della VPN dell’unipg); in questa modalità ho solo il problema della cifratura perché è vero che ho un indirizzo virtuale però l’indirizzo con cui contatto il VPN gateway è il mio indirizzo reale.

Ho un problema più significativo quando la cifratura non la fa il client (modalità road warrior) ma la fanno i router (gateway + gateway). Questo si applica tutte le volte che ho un’infrastruttura distribuita e quindi creo un link cifrato tra le interfacce dei rispettivi router, che agiscono sia come router che come VPN gateway e quindi quello che viene cifrato è il collegamento esterno. Questo causa un problema di identità perché ho un rimapping, un problema di aggregazione perché n indirizzi IP vengono visti come un unico indirizzo IP, un problema di consistenza perché indirizzi diversi vengono mappati sullo stesso indirizzo ma poi ci possono essere VPN diverse e quindi posso avere in momenti diversi identificativi diversi, e poi un problema di cifratura, che di fatto è quello che crea tutti questi problemi.

Immagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamente

Con questi due schemi riesco a vedere come faccio a bypassare i problemi. La soluzione nel caso del NAT e del proxy è semplice: ho bisogno di osservare tutte e due i link, a monte e a valle. Devo quindi mettere un rubinetto a cui attacco la sonda sia prima che dopo; solo così riesco a capire se il meccanismo di NAT o di proxy stanno funzionando bene, perché posso essere interessato sia al traffico generato da X e Y sia a monitorare se il proxy o il NAT hanno funzionato correttamente (se riesco anche ad analizzare i log del NAT e del proxy monitoro ancora meglio).

Altrimenti, se osservo da una parte sola non avrò una visione chiara di quello che succede; solo se guardo in ingresso e in uscita riesco a capire se il mapping funziona e riesco a bypassare i vari problemi. Devo poter monitorare prima e dopo la middlebox.

Solo nel caso di DHCP server basta che monitoro i log del server DHCP, ma tranne quel caso, tipicamente per monitorare il comportamento di una middlebox ho la necessità di osservare quello che arriva prima e quello che arriva dopo. Questo comporta un onere maggiore dal punto di vista dell’elaborazione, archiviazione e trasmissione del traffico, soprattutto se queste operazioni non le faccio in locale ma le faccio su un nodo dislocato.

Fine 04/10

10/10 (deve caricarla Andrea)

**Azione**

L’azione di un sensore descrive come il sensore interagisce con i dati che raccoglie.

A seconda del dominio, le azioni possibili sono:

* Report: un sensore di report fornisce semplicemente informazioni su tutti i fenomeni osservati dal sensore
* Evento: un sensore di eventi differisce da un sensore di report in quanto utilizza più origini dati per produrre un evento che riepiloga alcuni sottoinsiemi di tali dati
* Controllo: un sensore di controllo modifica o blocca il traffico quando invia un evento

Sensore di report:

* I sensori dei report sono semplici e importanti per la definizione di base
* Sono utili anche per sviluppare firme e avvisi per fenomeni che i sensori di controllo non sono ancora stati configurati per riconoscere
* I sensori di report includono raccoglitori NetFlow, tcpdump e log del server

Sensore eventi:

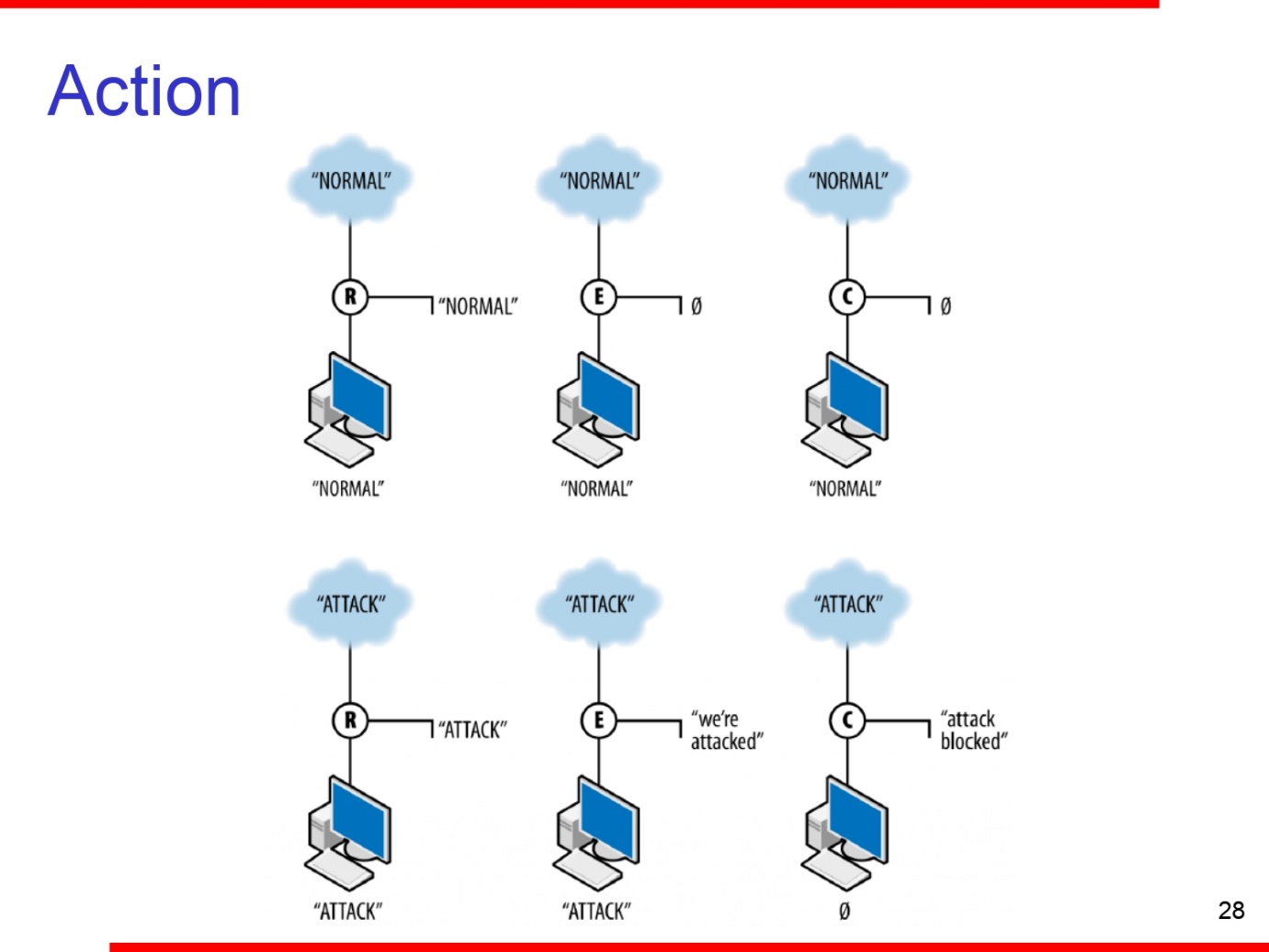
* Nella loro forma più estrema, i sensori di eventi sono scatole nere che producono eventi in risposta a processi interni sviluppati da esperti
* I sensori di eventi includono IDS (Intrusion Detection System) e antivirus (AV)

Sensore di controllo:

* I sensori di controllo non solo ma riescono anche a mettere insieme avrò azioni diverse e quindi per avere azioni in domini diversi potrei avere bisogno di sensori di controllo diversi. Azione di controllo strettamente legata al dominio
* I sensori di controllo includono intrusion prevention systems (IPSs), firewall, sistemi antispam e alcuni sistemi antivirus

I firewall operazione di restrizione dell’accesso. Sono dispositivi statici, configurati per riconoscere alcune situazioni e preconfigurati per bloccarle.

Esempio:



Nella parte alta della slide abbiamo una situazione di traffico normale, mentre nella parte bassa della slide abbiamo una situazione di traffico

(le stesse tecniche )

In alto a sinistra abbiamo una sonda R che analizza il traffico e genera periodicamente dei report, che una volta analizzati

Quel “normal” è asssociato a una post eelaborazione

Al centro abbiamo un sensore E che è configuarto per

È sempre un sensore passivo e il suo output è un insieme vuoto

La stessa cosa per il sensore C, sensore sia passivo che attivo; in questo caso è passivo perché

Nella riga sotto, in presenza di un attacco effettuato da quakcuno che sta in internet

Nel caso della sonda R avrò che le informazioni, una volta elvìborate, mi diranno che sto subendo un attacco. Il report rappresenta le staticsche del traffico cattyrato, saranno poi le azioni di post processing che andrannoa la presenza id un attacco

Nel caso della sonda E, il sensore genera uno specifico messaggio che dice che siamo attaccati. È sempre un sensore passivo ma genera un messaggio a un dispositivo configuari per ricervere le notiche

Nel terzo caso del sensore C, come nel secondo caso rileva l’attacco e ci possono essere due possibili eventi: rilevo l’attacco ma non sono in grado di bloccarlo, oppure rilevo l’attacco ma riesco a bloccarlo e blocca l’attacco tanto che il calcolatore sotto non vede più niente. Quindi il sensore riesce a bloccare il tentativo di attacco

* I sensori di controllo sono sensori ad eventi

**Formato dei dati trattati**

Formato proveniente da acquisizioni live/offline:

* File Pcap (tshark, tcpdump) con o senza filtri di cattura (i filtri di cattura riescono a limitare il traffico allo stesso tempo non permettono di catturare alcuni dati che con il senno di poi )

Di solito quando catturare payload cifrato è poco la maggior parte dell informazioni sta sull’instestazione

* Dati pre-processati (metriche/KPI)
  + tshark, strumenti attivi + linguaggi di elaborazione in linea (awk)

ca sono sempre report perché elaborano tutti i pacchetti e mostrano le statistiche di tutti i pacchetti fino a quando tutto

* Log (di un server web, di un proxy, o di un qualsiasi server)
  + Completi o preelaborati

Esistono server che hanno la capacità di log configurabili e quindi possono configuarre come infom, normal, warning o error

Nwl momwnto in cui lo congifuro come warning o error si comporta come dipende da come è configuario il processo che il log lo genera e dipende dalla maturarità del software ce sto utilizzando

* Informazioni organizzate per flusso
  + NetFlow

**NetFlow**

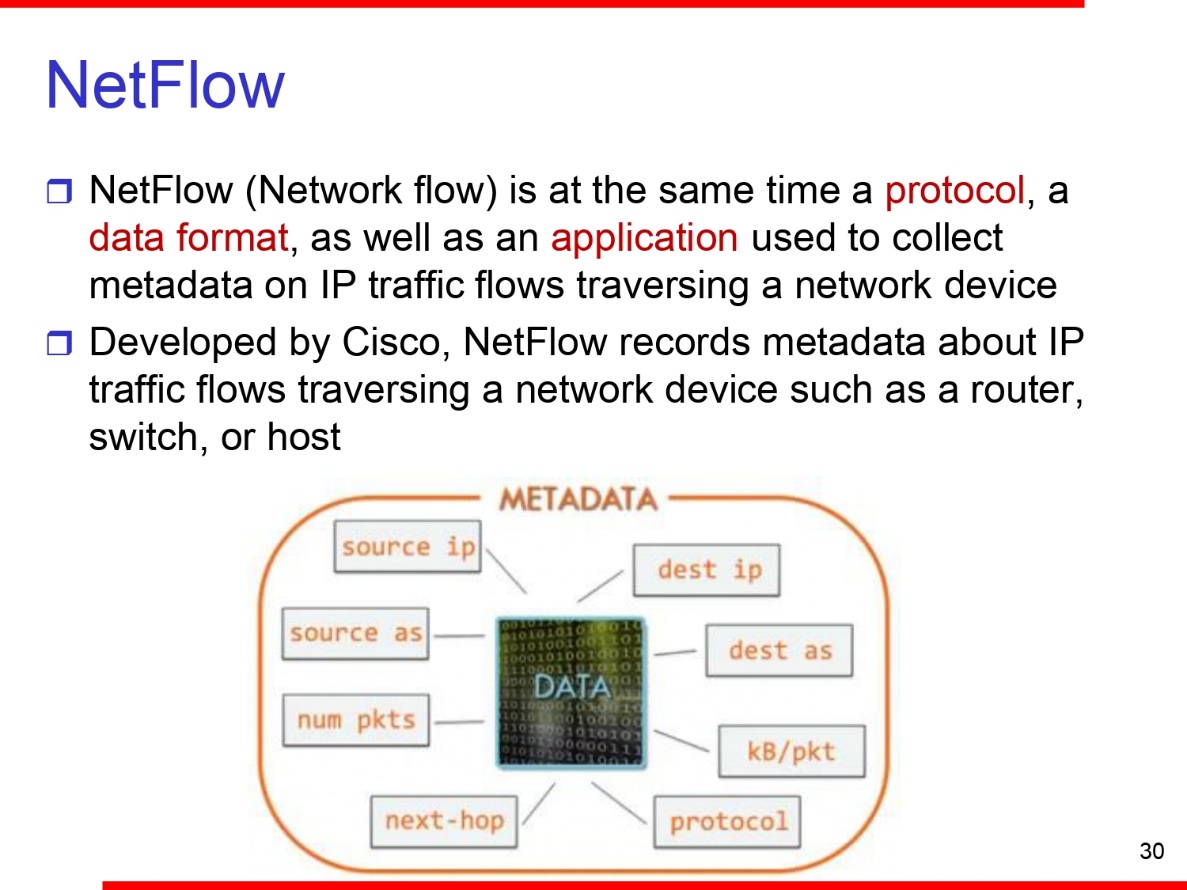
NetFlow (flusso di rete) è allo stesso tempo un protocollo, un formato dati e un'applicazione utilizzata per collezionare metadati sui flussi del traffico IP che attraversano un dispositivo di rete.

Sviluppato da Cisco, NetFlow registra metadati sui flussi di traffico IP che attraversano un dispositivo di rete come un router, uno switch o un host.

Infine standardizzato da

I metadati non sono i dati dei pacchetti ma sono dati estratti dai pacchetti e sono i seguenti:

* as (autonomus sistem)
* prossimo salto



Un dispositivo abilitato per NetFlow:

* genera metadati a livello di interfaccia
* invia informazioni sui dati di flusso a un raccoglitore di flusso
  + Il collezionista è l'archivio

Può vedere ..

Fornisce un riepilogo compatto delle sessioni di traffico di rete a cui è possibile accedere rapidamente. Contiene le informazioni di maggior valore in un formato relativamente compatto.

Un operatore di rete può utilizzare i dati NetFlow per determinare la portata della rete, la perdita di pacchetti, la congestione del traffico e i problemi di sicurezza a un livello di interfaccia specifico (ad esempio potrei accorgermi di ).

NetFlow è la tecnologia sviluppata da Cisco.

IPFIX è uno standard IETF molto simile nell'approccio e nella struttura all'ultima versione di NetFlow.

sFlow (sampling Flow) è stato introdotto da InMon Corp

* sFlow non campiona tutti i pacchetti come NetFlow, né contrassegna i flussi di traffico
* Si basa su un campionamento statistico accurato e affidabile
  + Riduce la quantità di informazioni da elaborare e analizzare

Una volta che il pacchetto viene campionato viene

Netlow orienattoa strato 3 e 4, sflow cattura tutto il pacchetto, ma decide qualce pacchetto analizzare e utlizza questi campioni per generrare statisctiche

Il problema è che catturo un paccheto ogni …

Dal punto di vita dei metadati generati sono abbastanza simili

Altri formati di record di flusso specifici del fornitore che sono di natura simile a queste 3 versioni:

* J-Flow di Juniper Networks
* NetStream di 3Com/Huawei

In generale sono tutti compatibili con netflow che è il livello minimo, e poi ci sono delle informazioni aggiuntive

Versione 10 e versione 5

**Definizione di flusso**

Il cuore di NetFlow è il concetto di flusso, che è un'approssimazione di una sessione TCP. Ricordiamo che le sessioni TCP vengono assemblate all'endpoint confrontando i numeri di sequenza. Gestire tutti i numeri di sequenza coinvolti in più sessioni TCP non è fattibile su un router, ma è possibile fare un'approssimazione ragionevole utilizzando i timeout.

Un flusso è una raccolta di pacchetti con indirizzo identico (L3/L4) strettamente raggruppati nel tempo

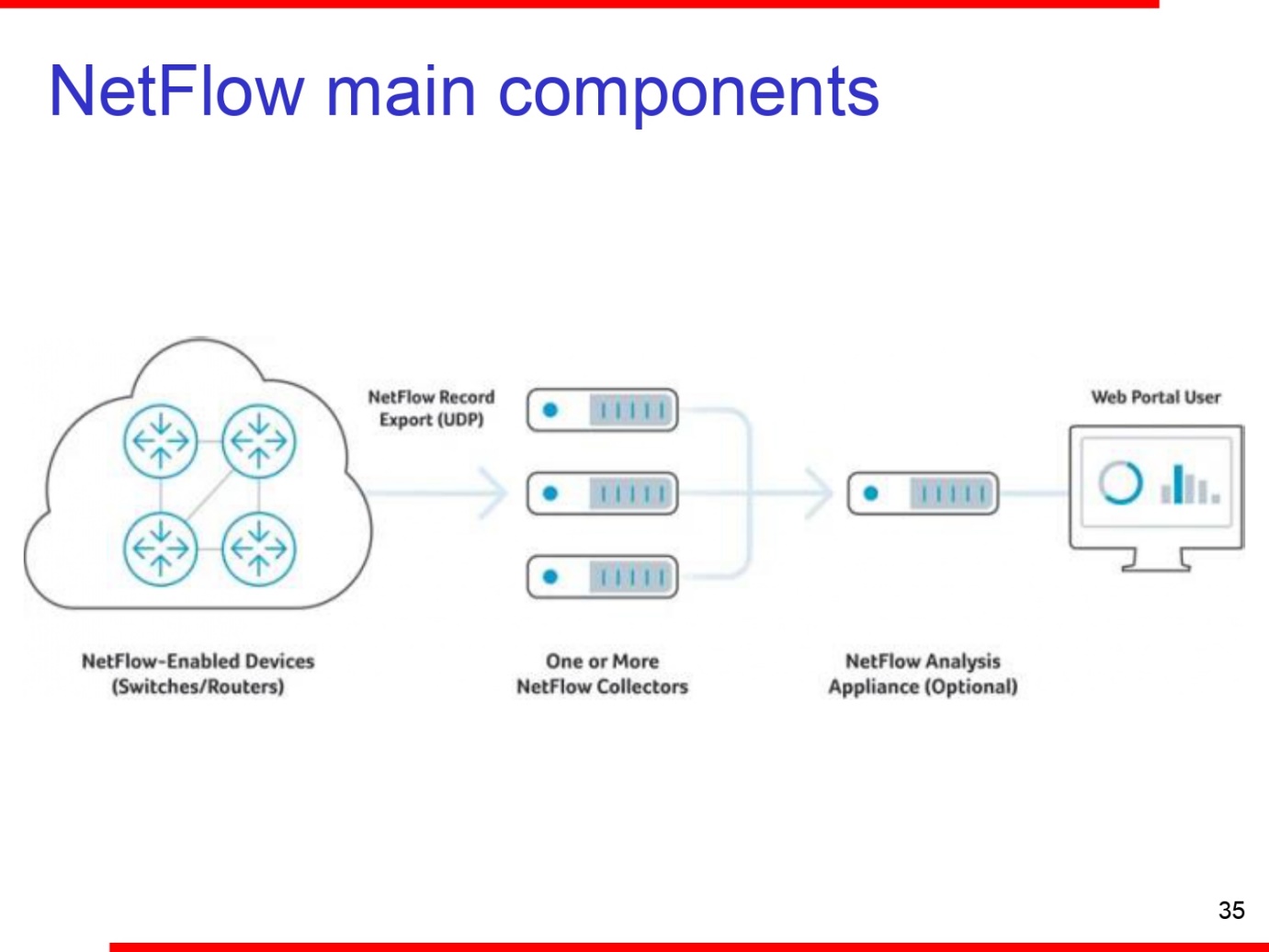
Stesso indirizzo di strato 3 strato 4

Il flusso di NetFlow è una sorta di approssimazione del flusso tcp

**Componenti principali di NetFlow**

Le soluzioni di monitoraggio NetFlow sono generalmente costituite da tre componenti principali:

* Esportatore di flussi: un dispositivo abilitato per NetFlow che genera record di flusso e li esporta periodicamente in un raccoglitore di flusso
* Collettore di flusso: un programma in esecuzione su un server responsabile della ricezione, archiviazione e pre-elaborazione dei record di flusso ricevuti dai dispositivi abilitati NetFlow
* Analizzatore di flusso: un'applicazione di analisi che elabora i record di flusso raccolti dal raccoglitore di flusso in report, avvisi e altri risultati interpretati



Il router o lo sitche attraversato dal traffico quest i metadati vengono poi esportati tramite un protocollo apllicaticp che utlizza UDP come protocollo di traspoeto verso altre unità chiamte collettori

Pochi collettori hanno le statistiche di un elevato numero di dispositivi NetFlow

I metadati

Le prime due componenti sono obbligatorie, il terzo componente è opzionale

Una volta che i metadati poi questi dati possono essere recuperati da una console di analisi che può estrarre delle statistiche aggregate

Schema slide 4 con appunti

**Dati NetFlow**

Dalla definizione viene fuori che i flussi sono unidirezionali

Un esportatore NetFlow identifica un flusso come un flusso unidirezionale di pacchetti aventi in comune (almeno) quanto segue:

* Porta di interfaccia di ingresso
* Indirizzo IP di origine e destinazione
* Numero di porta di origine e destinazione (L4)
* Campo del protocollo di livello 3
* Tipo di servizio

Questi attributi che compongono i metadati di base sono inclusi nel "record di flusso" di Netflow. Ogni volta che un nuovo flusso di traffico IP unidirezionale inizia a attraversare un dispositivo, viene creato e monitorato un nuovo record nella cache Netflow di bordo del dispositivo.

Un record di flusso è pronto per l'esportazione quando si verifica una delle seguenti condizioni per il flusso corrispondente:

1. Il flusso è inattivo (nessun nuovo pacchetto ricevuto) per una durata definita in un timer

* I timer sono configurabili ma in genere vengono utilizzati i valori predefiniti

1. Il flusso è di lunga durata (attivo) ma dura più a lungo del timer attivo (ad esempio, un lungo download FTP)
2. Un flag TCP (ad esempio FIN, RST) indica che il flusso è terminato

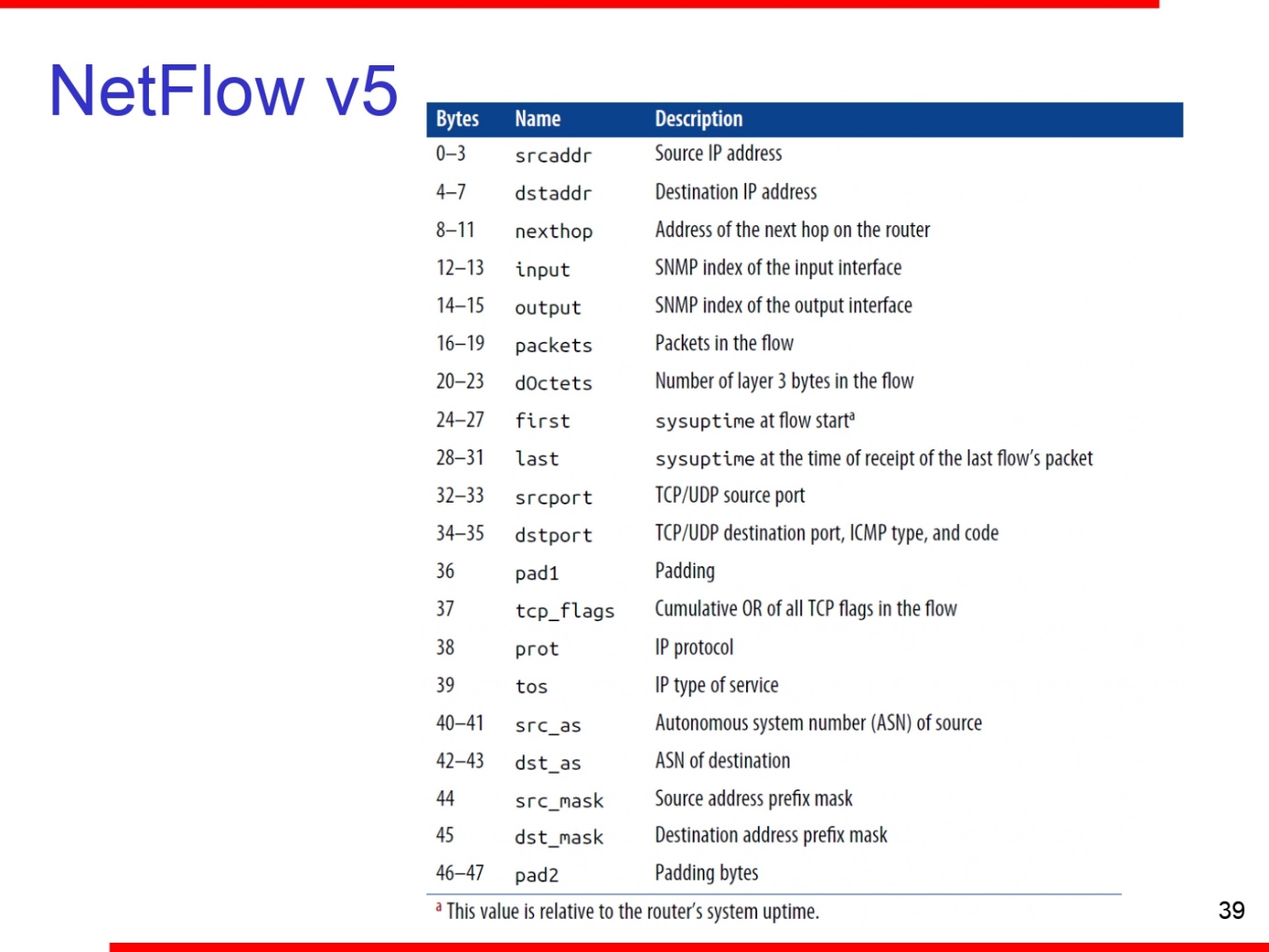
**NetFlow v5**

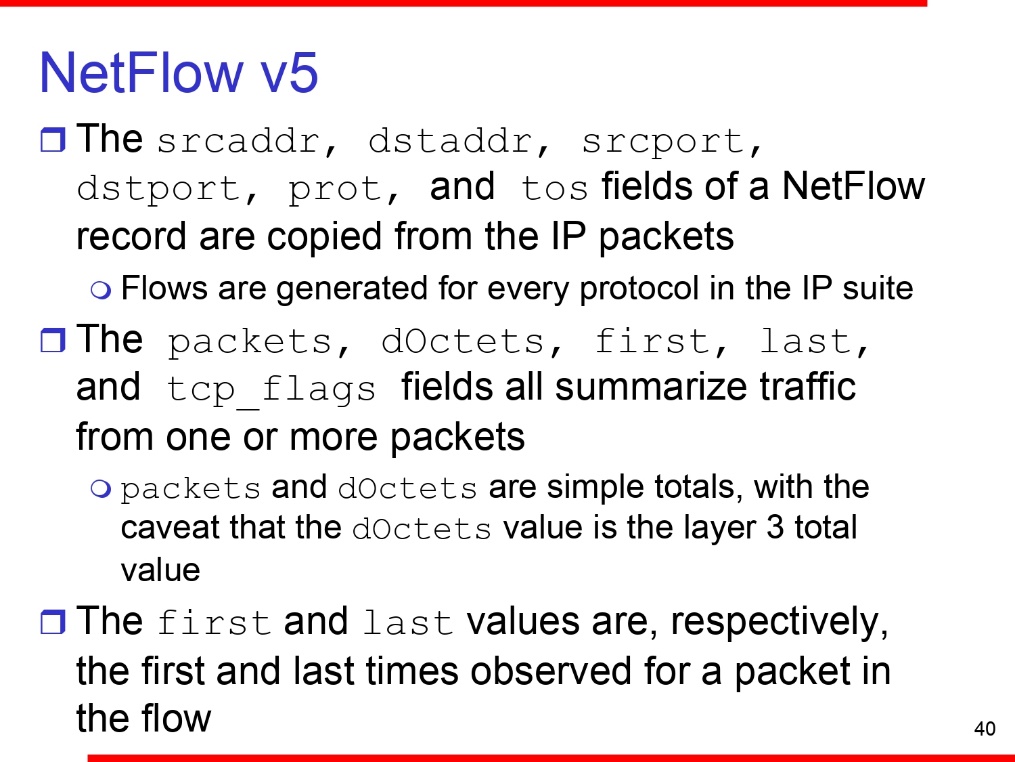
Il record di flusso viene incapsulato in un datagramma UDP e inviato a un raccoglitore NetFlow

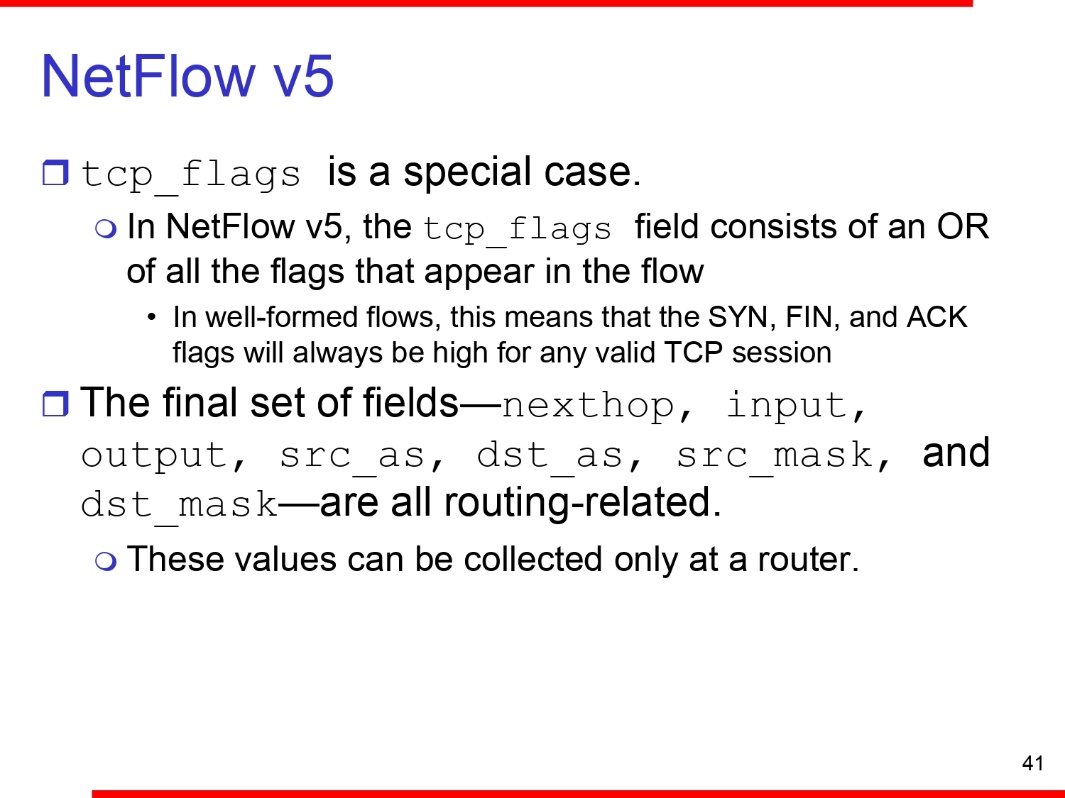
Il raccoglitore archivia i metadati del flusso in un record il cui formato è definito dal protocollo

I dati trovati in un record NetFlow v5 in genere includono:

* Metadati menzionati in precedenza (indirizzi L3/L4)
* Packet and byte counts
* Timestamp di inizio e fine
* Interfaccia di ingresso e uscita
* Flag TCP e protocollo incapsulato (TCP/UDP)
* Informazioni sull'instradamento BGP







**NetFlow v9 and IPFIX**

NetFlow v5 è uno standard limitato e obsoleto:

* incentrato su IPv4
* progettati prima che i flussi fossero comunemente utilizzati

La nuova versione importante è la v9 (di Cisco)

Uno standard di reporting del flusso basato su modello che consente agli amministratori del router di specificare quali campi includere nel flusso

IPFIX (noto anche come v10) è NetFlow basato su modelli standardizzato da IETF

* v10 supporta completamente IPv6
* Campi speciali per il monitoraggio NAT nei dispositivi di fascia alta

**NetFlow generation issues**

Vengono generati i record NetFlow:

* direttamente tramite dispositivi hardware di rete (router, switch)
* utilizzando un software per convertire i pacchetti in flussi (In esecuzione in nodi diversi dagli apparecchi)

La generazione basata su dispositivi implica l'utilizzo di qualsiasi funzionalità NetFlow offerta dal produttore dell'hardware.

* Molti fornitori con molte funzionalità
* Impossibile una vera configurazione standardizzata

La generazione di NetFlow può causare problemi di prestazioni sui router, in particolare sui modelli più vecchi

Possibili soluzioni alternative (regola pratica):

* Ridurre la priorità del processo
  + e lasciando cadere i record...
* Scaricamento dell'attività di generazione NetFlow su hardware opzionale
  + Soluzione costosa e proprietaria
* Utilizzare il campionamento per ridurre il carico delle prestazioni
  + Per l'analisi della sicurezza, è consigliabile utilizzare record non campionati
* Utilizza formati di aggregazione e reporting predefiniti
  + Raccogli NetFlow grezzo, non aggregazioni
  + Potrebbe verificarsi un'elevata perdita di informazioni nell'aggregazione a livello di flusso

L'alternativa è utilizzare un'applicazione che generi NetFlow dai dati pcap:

* Queste applicazioni accettano pcap come file o direttamente da un'interfaccia di rete e aggregano i pacchetti come flussi
* Questi sensori non hanno il vantaggio di un router, ma possono farlo:
  + dedicare più risorse di elaborazione all'analisi dei pacchetti
  + produrre output NetFlow più ricchi, incorporando funzionalità come l'ispezione approfondita dei pacchetti

**Aumentare i dati con i log**

Quando si utilizzano dati middlebox, è buona pratica archiviare i dati e quindi applicarli caso per caso.

L'approccio alternativo a questo è annotare al volo altri dati (come flusso o pcap) con le informazioni della casella centrale:

* Troppo complesso
* Difficile definire una procedura standardizzata:
  + I log VPN dovrebbero essere i primi da considerare
  + I log proxy potrebbero ignorare la necessità dei log di servizio

Fino a qui 10/10

11/10

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 estrare dai pacchetti troncati.

I dispositivi devono riuscire ad indentificare 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).

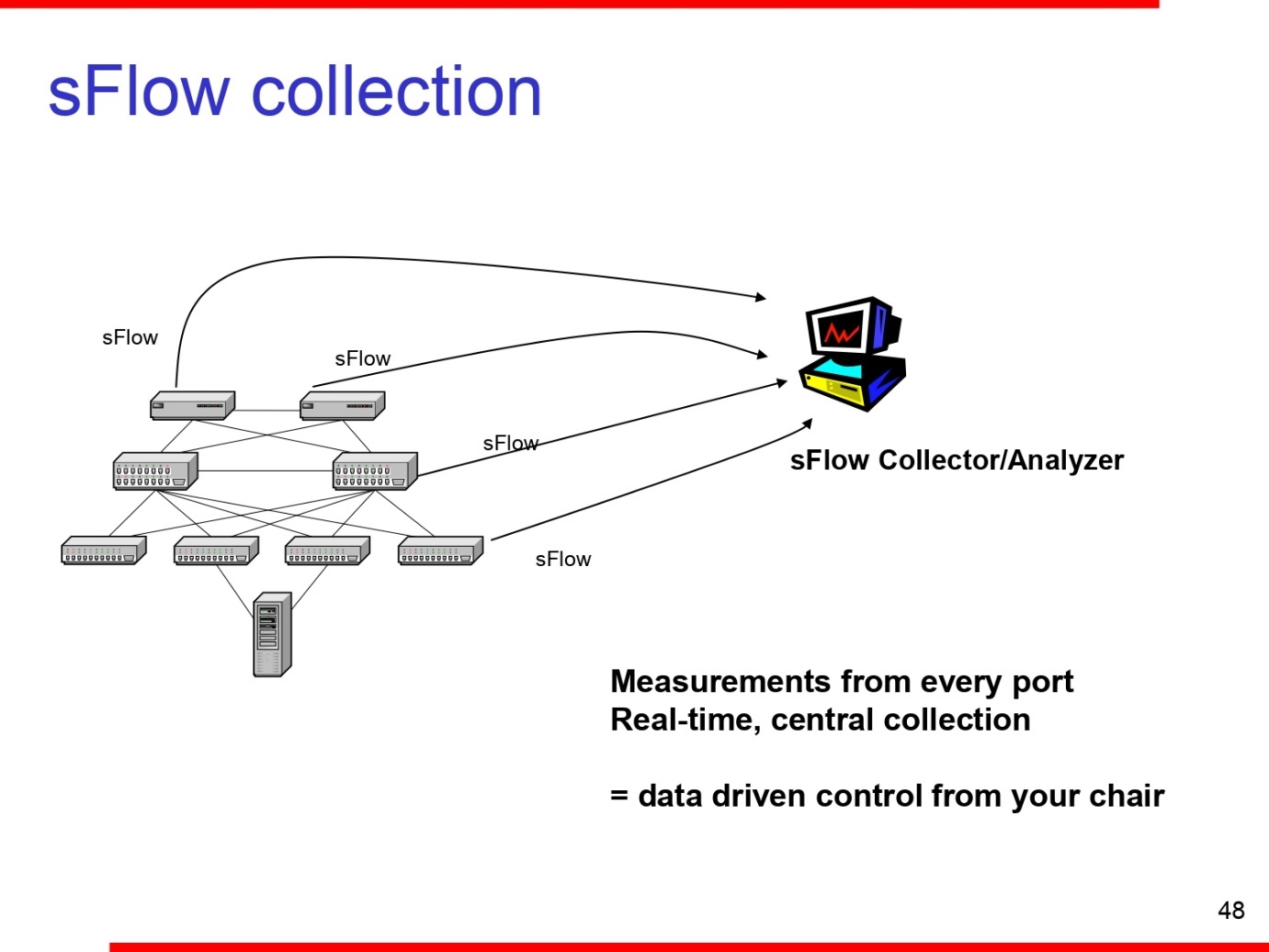
Vantaggio rispetto a NetFlow: riesco ad avere informazioni aggiuntive, che sono quelle di strato 2, e poi siccome posso definire in maniera arbitraria la rate di campionamento r (dati N pacchetti che passano, quanti ne vado ad osservare) riesco a stare dietro ai processi in maniera più affidabile

r = > 1 , con N numero di pacchetti totali e n 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.

**Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamente**

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.

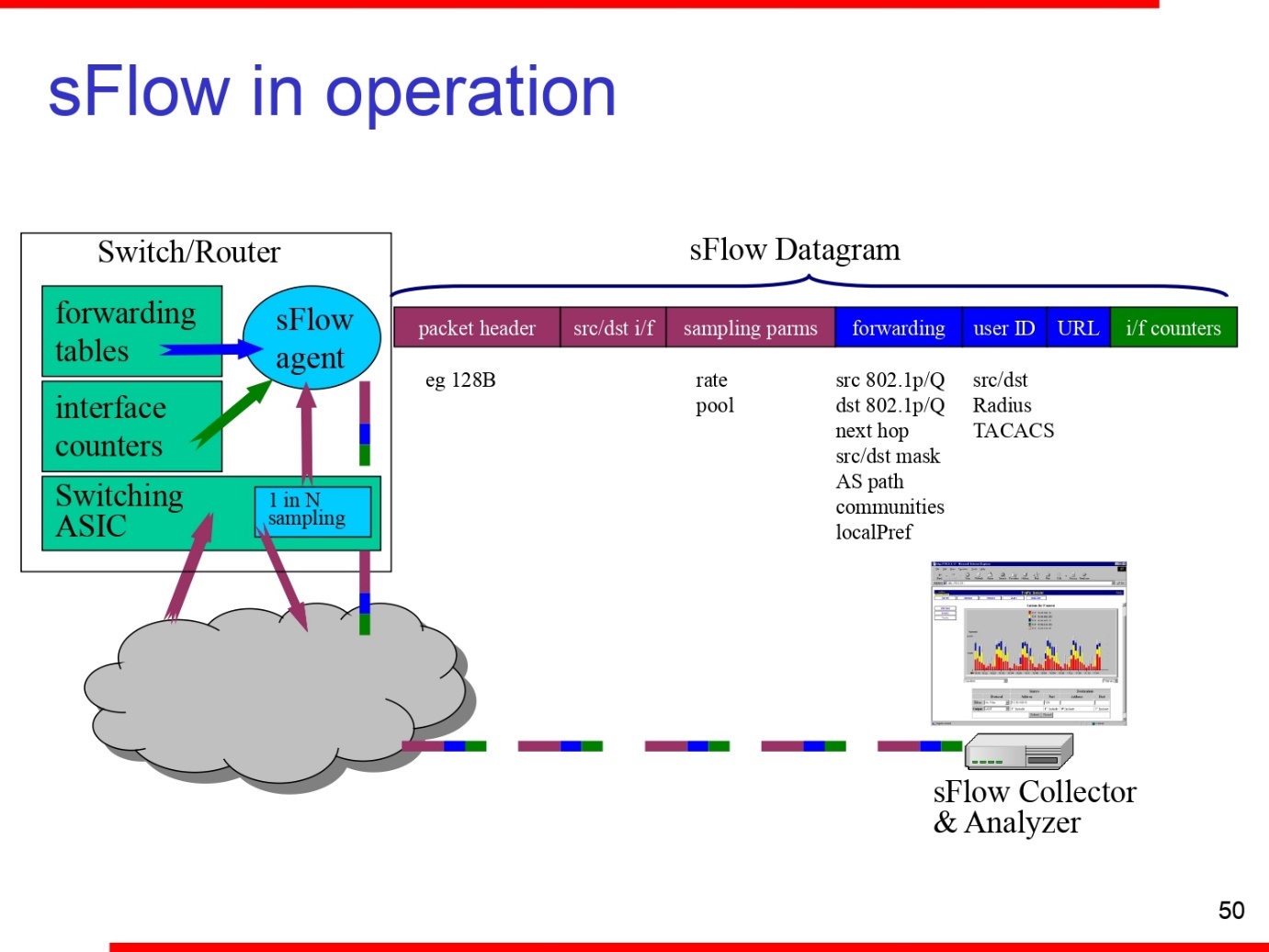
Questa cosa si ripete fino a quando un nuovo pacchetto sarà considerato per il campionamento ma nel momento in cui andrò a decrementare la variabile skip, questa passerà da 1 a 0. Entro quindi nel ramo di sinistra, reinizializzo skip a r e incremento il numero dei pacchetti campionati.

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 pesco 1.

****

A questo punto l’agente invia al collettore:

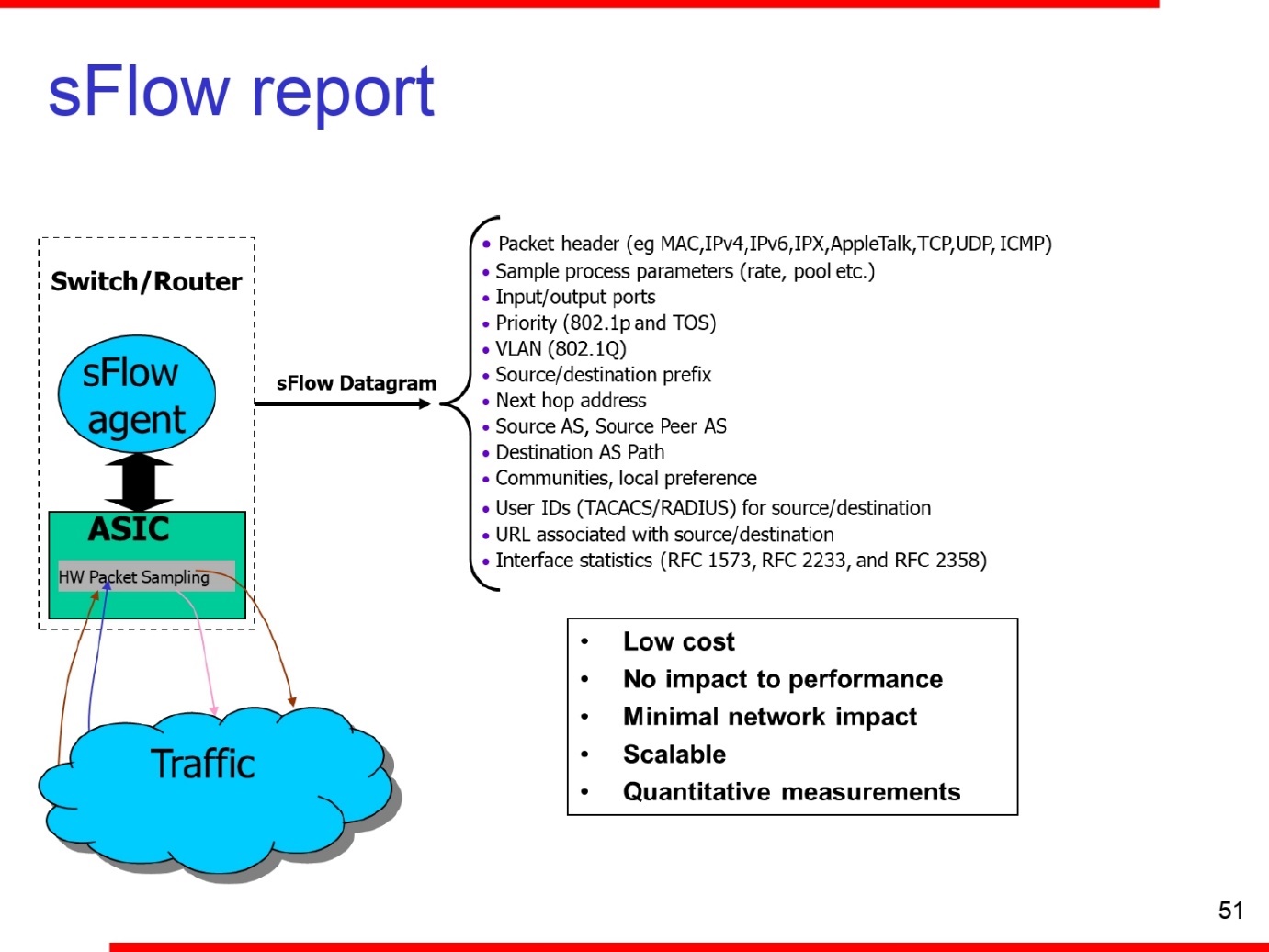
* il pacchetto troncato (l’header)
* le interfacce sorgente e destinazione del router
* i parametri di campionamento (ad esempio la rate, cioè ogni quanti pacchetti ne viene estratto uno, o il pool, cioè le regole che ci dicono quali pacchetti considero o meno per il campionamento)

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

Immagine che contiene linea

Descrizione generata automaticamente

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.

Fine 11/10

17/10

**Monitoraggio delle evoluzioni delle tecniche**

NetFlow e sFlow sono due metodi tradizionali di monitoraggio del flusso ampiamente utilizzati. NetFlow e sFlow condividono un certo numero di benefici, che sono quelli associati al fatto di riuscire ad identificare un flusso, a differenza dei tool basati sulla pura cattura che invece riescono solo a catturare pacchetti e poi mandano in post-processing il compito di identificare i flussi.

NetFlow acquisisce e aggrega informazioni sui flussi mentre passano attraverso gli switch. Vediamo gli svantaggi:

* NetFlow richiede risorse di calcolo addizionali e significative, siano esse all’interno del dispositivo dove gira NetFlow siano esse dislocate in un dispositivo esterno che effettua l’acquisizione di dati rilanciati in mirroring e che poi devono essere elaborati tramite una soluzione di tipo NetFlow
* Un sistema di monitoraggio riceve in automatico i dati dai dispositivi e questi dati rimangono sui collettori. Poi una stazione di analisi deve interrogare, tramite una funzione di calling, i collettori stessi per ottenere i dati analizzati sul flusso
* Altro problema classico di NetFlow è che non è un sistema real-time per due motivi:
  + L’esportazione avviene solo quando un flusso è giudicato terminato, e la terminazione di un flusso può avvenire attraverso intercettazione di messaggi di segnalazione espliciti (ad esempio i FYN del tcp) oppure dopo un determinato periodo di inattività (può avvenire con tcp e per l’udp è invece l’unica soluzione disponibile, in quanto non esiste il concetto di sessione). (Tempo di esportazione nell'ordine delle decine di secondi)
  + Nel caso di flussi che durano per molto tempo NetFlow utilizza una modalità per cui il monitoraggio del flusso viene interrotto, viene fatto l’export dei dati e poi nella finestra di misura successiva viene inizializzato il nuovo flusso e poi starà alla piattaforma di monitoraggio andare a combinare questi sotto flussi che però appartengono ad un unico flusso.

Vantaggi di NetFlow: soluzione standard utilizzabile nella maggior parte dei dispositivi cosiddetti gestiti, nonché nei dispositivi open source (tramite soluzioni come quella vista la volta scorsa in laboratorio).

Per quanto riguarda sFlow le considerazioni sono simili, la differenza principale è che le risorse di calcolo richieste da sFlow possono essere significamente minori; questo perché ogni N pacchetti vado a catturare n pacchetti. Quindi andando a aggiustare il rapporto tra il numero di pacchetti che passano e il numero di pacchetti che analizzo posso rendere la soluzione scalabile e quindi sostenibile dal punto di vista computazionale. Questo campionamento introduce però un’incertezza sulle grandezze gaussiane; quest’incertezza si quantifica andando a definire un intervallo di confidenza all’interno del quale ho una certa probabilità che ricada il valore reale che intendo stimare.

sFlow campiona i pacchetti dai flussi che passano attraverso gli switch. sFlow preleva un campione una volta ogni n pacchetti con una frequenza di campionamento configurabile 1/n. Quindi, il pacchetto campionato viene inviato immediatamente a un sistema di monitoraggio per ulteriori analisi.

Gli svantaggi di sFlow sono associati al campionamento (che è il suo vantaggio):

* Il campionamento permette di rendere la soluzione scalabile ma riduce l’accuratezza. Abbiamo infatti un enorme incertezza nell’identificare quando un flusso inizia.
* C’è la possibilità che flussi small (non con pochi byte ma con una durata piccola) siano completamente ignorati, perché nessun pacchetto associato a questi flussi potrebbe essere campionato, e quindi il flusso c’è ma non si vede. Si potrebbero quindi perdere eventi di rete come picchi e anomalie.

Come evolvono queste tecnologie di misura?

Abbiamo categorizzato due tipi di approcci alla misura del traffico: le tecniche passive (più diffuse e più affidabili, tra queste abbiamo sia quelle basate sulla rilevazione dei flussi come NetFlow e sFlow sia quelle basate sulla cattura dei pacchetti come tshark e tcpdump) e le tecniche attive (iniettano del traffico in rete e analizzano gli effetti che questo traffico induce, e sono ad esempio ping, traceroute e nmap).

Nelle tecnologie classiche queste misure vengono effettuate attraverso un terminale o un dispositivo di misura. Posso farle quindi attraverso le sonde, messe o sui sistemi terminali o sui dispositivi intermedi che supportano questo tipo di misure. Sono quindi misure che dipendono dal tipo di dispositivo.

Negli ultimi anni si è affermata una nuova tecnologia di rete: SDN (Software Defined Networking). SDN consiste nel centralizzare il piano di controllo dei dispositivi di rete.

SDN è vincolata perché si basa su switch SDN a funzione fissa.

Esempio: se ho 3 router nell’approccio classico i router si scambiano le informazioni tra di loro (con OSPF o IS-IS per i protocolli all’interno di un dominio, BGP per l’instradamento tra domini) e costruiscono in maniera distribuita alla costruzione delle tabelle di routing che siano il più possibile stabili. Nell’approccio SDN i 3 router non parlano più tra 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 (le tabelle di instradamento) all’interno dei nodi.

Immagine che contiene schizzo, clipart, disegno, Line art

Descrizione generata automaticamente

Si passa da un approccio distribuito, in cui l’intelligenza è distribuita sui dispositivi di rete, a un approccio centralizzato, in cui i dispositivi di rete diventano più veloci e più stupidi e l’intelligenza è spostata in un componente terzo.

Il vantaggio è che non devo andare ad aggiungere intelligenza a questi nodi, e quindi posso disaccoppiare la tecnologia di costruzione di nodi sempre più veloci dal piano di controllo. Il vantaggio vero è che se voglio inserire un nuovo protocollo in un approccio classico devo aspettare che il costruttore del dispositivo rilasci il firmware che includa quel protocollo, mentre se devo farlo con SDN devo sviluppare un software da caricare solo sul controller e poi sarà il controller che dirà ai nodi cosa fare.

Un controller può anche gestire dispositivi eterogenei quindi, ad esempio, anche dispositivi di costruttori diversi.

Nell’SDN è stata standardizzata l’interfaccia di comunicazione tra i nodi del data plane e il control plane.

Il protocollo definito per primo si chiama OpenFlow ed è stato implementato a Standford. È quello che tipicamente si usa in tutti i data center del mondo, a partire da Google.

Nel caso delle misure di rete, il vantaggio di avere un piano di controllo centralizzato è che questo permette di configurare i dispositivi stupidi in modo tale da abilitare nuove funzioni, senza che quelle funzioni debbono essere supportate dal costruttore del dispositivo.

Esempio: se non supporto il mirroring dei dispositivi ma sono compatibile con OSPN, basta che il mio controllore mi carichi una regola che mi dice che per alcuni tipi di pacchetti devo creare una copia e inviarla su una certa interfaccia. Così facendo ho costruito un mirroring custom, sicuramente più efficiente rispetto al mirroring di tutti i pacchetti di una certa porta e più adattabile alla topologia della rete e ai dispositivi che ho a disposizione.

Utilizzo quindi un framework dove è il software che fa networking a farne da padrone e permette di aumentare il numero di funzioni per il monitoraggio e per il controllo della rete.

Nel network management abbiamo sempre il classico triangolo: monitorare, analizzare e reagire.

Immagine che contiene schizzo, disegno, Line art, bianco

Descrizione generata automaticamente

Qui riesco a fare meglio rispetto al caso di base queste operazioni perché sarà il controllore che potrà creare delle regole custom per reagire a una situazione non consona.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

La programmazione avviene a livello control plane.

Il grosso svantaggio di questi sistemi è che alcune volte la comunicazione tra data plane e control plan può diventare un collo di bottiglia perché alcune funzioni, quelle più complesse, sono obbligato a realizzarle nel control plane (il data plane supporta solo alcune funzioni).

**In-band network telemetry**

Posso avere un’innovazione se invece di programmare in maniera custom il control plane arrivo a programmare in maniera custom il data plane. Significa che metto delle funzioni intelligenti, programmabili, all’interno di questi dispositivi: si chiama network programmability. Sviluppo un pezzo di codice e lo carico all’interno del dispositivo, che lo esegue a run time senza spegnerlo e senza cambiare la configurazione, all’interno del proprio kernel, siano essi dispositivi fisici o dispositivi virtuali.

Questa branca del network monitoring prende il nome di In-band Network Telemetry. In banda perché questa telemetria (processo di misura) viene fatta sui nodi mentre passa il traffico; quindi, aggiungo a dei nodi delle sonde molto a basso livello che permettono al nodo di effettuare operazioni anche sofisticate consumando poche risorse di calcolo e soprattutto andando alla cosiddetta wire rate.

Problema principale: costo. Questi dispositivi costano tantissimo perché sono diffusi ma non ancora lo standard di un dispositivo enterprise.

Si chiama telemetria attiva perché per fare queste operazioni vengono aggiunte delle intestazioni sui pacchetti. Non siamo più nel caso di monitoring passivo, ma è un monitoraggio di tipo attivo. In realtà è un monitoraggio attivo ma di cui l’utente non ha percezione perché vedremo che questi dispositivi, così come aggiungono intestazioni, le tolgono prima che il traffico arrivi ai nodi terminali e quindi il mittente e il destinatario non si accorgono che questi nodi hanno fatto questo processing. Il risultato dell’elaborazione viene inviato a un collettore che poi effettuerà le relative operazioni.

Siccome siamo tornati sul data plane siamo di nuovo a operazioni che possono permettere di implementare bene questo meccanismo ma anche in questo caso con certe limitazioni perché i programmini che posso caricare sui dispositivi hanno una dimensione limitate e quindi alcune funzionalità sono limitate. Sono comunque molto potenti e permettono di fare cose particolarmente avanzate.

Noi ci occuperemo principalmente di una soluzione che sfrutta questi principi di programmabilità per il monitoraggio.

Ma c’è bisogno di questa cosa?

NetFlow comincia a non essere più adeguato per reti ad altissima velocità perché richiede di elaborare troppi pacchetti e questo potrebbe avere un impatto negativo sulle prestazioni del dispositivo o generare risultati non affidabili.

sFlow, d’altra parte, effettua il campionamento e quindi alcuni dati possono essere persi, soprattutto i dati relativi ai flussi di piccola entità che, soprattutto all’interno dei data center, sono la stragrande maggioranza. Ci sono flussi di piccola entità che vengono scambiati tra i vari moduli che compongono i servizi. Questo perché tipicamente i servizi non vengono più costruiti in maniera monolitica all’interno di un calcolatore fisico ma viene utilizzato l’approccio ai microservizi, dove componenti diversi di un servizio che girano su macchine diverse o all’interno della stessa macchina ma su diverse macchine virtuali o su diversi container e quindi colloquiano in rete.

In generale NetFlow ci dice quando un flusso inizia e quando finisce, il numero di pacchetti che compongono il flusso (unidirezionale) e il numero di byte. Quindi ho una visione astratta del flusso, ma non dei singoli pacchetti.

In alcuni casi, per identificare una situazione anomala ho bisogno di una visione globale della rete. Piccoli flussi si intersecano tra loro, cioè attraversano gli stessi nodi, non dall’inizio alla fine ma solo per una o due tratte e si combinando con un numero di pacchetti molto elevati per tempi molto piccoli causando perdite per buffer overflow all’interno di questi nodi che poi causano ritrasmissioni tcp e quindi problemi di rate.

La bit rate del flusso è ben al di sotto della capacità del link che attraversa. Se però questi flussi, per progettazione sbagliata, o per semplice causalità, si combinano a inviare tutti i propri pacchetti in una finestra temporale molto breve e sovrapposta con altri flussi possono comunque causare buffer overflow. Questo causa malfunzionamenti nei servizi, principalmente aumenta a dismisura le latenze.

Quindi cosa devo fare?

Devo essere in grado di fare la telemetria del flusso, quindi di tracciare tutti i nodi che il flusso attraversa, quel è l’interfaccia di ingresso, di uscita, l’istante in cui arriva e quanti pacchetti trova in coda.

Solo così riesco a capire se c’è un collegamento che risulta sovraccarico. La soluzione: cambiare l’instradamento.

Questo problema, che è uno dei principali all’interno del data center, può essere debuggato solo con tecniche di telemetria attiva, ovvero attaccando delle etichette sui pacchetti e vedere dove questi pacchetti passano e, nodo per nodo, andare a vedere cosa trovano quando attraversano un nodo.

È stato proposto un piano dati programmabile che trasforma gli interruttori a funzione fissa in quelli programmabili.

Nelle reti grandi e ad alta velocità, i metodi di monitoraggio devono fornire informazioni di rete in tempo reale, granulari ed end-to-end.

A volte è necessario il monitoraggio a livello di pacchetto:

* debug in un ambiente di routing multipercorso
* debug di interfacce difettose che interessano solo un gruppo specifico di pacchetti con le stesse caratteristiche

Recentemente è stato proposto INT per risolvere questi problemi e ha subito attirato l'attenzione.

La telemetria di rete in banda combina l'inoltro di pacchetti di dati con la misurazione della rete.

La telemetria di rete in banda raccoglie lo stato della rete inserendo metadati nel pacchetto cambiando nodo.

I dati di telemetria della rete in banda e i dati utente in genere condividono lo stesso collegamento o addirittura lo stesso pacchetto.

Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamente

INT sta per In-band Network Telemetry.

Un sistema di gestione basato sulla telemetria attiva avrà un certo numero di dispositivi (quelli all’interno del riquadro tratteggiato) che possono essere configurati uno per uno e possono essere controllati da remoto tramite tecniche SDN. Questi dispositivi applicheranno delle etichette sui singoli pacchetti sulla base di regole definite attraverso il sistema di controllo. Quando un pacchetto con un’etichetta attraversa un nodo tipicamente scatena una serie di operazioni, che possono comportare la modifica, la rimozione e l’aggiunta di un’etichetta o l’invio di un messaggio di monitoraggio verso un collettore di dati. Il collettore è chiaramente fuori banda, quindi non segue il percorso dei dati. Il collettore (o i collettori) invieranno poi i dati sul database e la piattaforma di gestione potrà recuperare quei dati attraverso le query al database e potrà andare a capire che cosa succede.

Vediamo questo esempio che rende tutto più semplice:

Immagine che contiene testo, schermata, diagramma, Piano

Descrizione generata automaticamente

Supponiamo che il nodo a sinistra sia il mittente e il nodo tutto a destra sia il destinatario. Questi due si inviano un certo numero di pacchetti (di strato 3), che saranno caratterizzati dall’intestazione e da un certo payload.

Cosa succede quando questo pacchetto entra su uno switch di tipo INT e le sue caratteristiche matchano una particolare regola?

Lo switch aggiunge un’ulteriore intestazione INT hdr e poi un certo numero di dati INT data, che caratterizzano l’attraversamento di quello switch.

Inserisco questi dati (INT hdr e INT data) tra l’intestazione originale del pacchetto e il payload del pacchetto: questa è una tecnica attiva perché va a modificare il pacchetto.

Questo è quello che fa il nodo 1, che è un nodo di inserzione.

Il nodo 2 è un nodo di transito, che esaminerà il pacchetto e o 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 di un ulteriore set di dati. Anche il nodo 2, mantenendo la stessa intestazione, inserisce i suoi dati di telemetria (INT data #2).

Infine, il pacchetto arriverà al terzo nodo: il sink, quindi il collettore, ma il collettore per quanto riguarda la telemetria, non il destinatario dell’informazione.

Il collettore aggiungerà al pacchetto i dati relativi al collettore (INT data #3), solo che riconosce di essere l’ultimo nodo e di fatto invia questo blocco di dati (INT header, INT data #1,#2,#3) al collettore e riporta il pacchetto alla situazione originale; quindi toglie i dati di telemetria e consegna il pacchetto a destinazione.

Questi nodi (mittente e destinatario) non si accorgono neanche che una o più reti intermedie applicano tecniche di telemetria che vanno a modificare il pacchetto perché queste tecniche prevedono che il pacchetto venga riconsegnato inalterato dopo il nodo di uscita.

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

Le informazioni tipiche sono: l’ID del nodo che viene attraversato, l’ID della porta di ingresso, l’ID della porta di uscita e l’utilizzazione della porta d’uscita (quanti byte sono in coda o quanti byte sono stati trasmessi diviso la capacità del link di trasmissione).

In questo disegno consideriamo uno scenario tipico da data center, all’interno del quale ho due macchine virtuali che sono eseguite all’interno di due diversi server. All’interno di un server avrò sia la macchina virtuale sia lo switch virtuale.

Le operazioni di telemetria possono anche essere supportate dagli switch virtuali, dipende se chi gestisce i server vuole partecipare o meno alla campagna di telemetria.

Il primo switch sarà quello che inserisce le istruzioni e aggiungerà un certo numero di informazioni: ad esempio, L1 (che è l’ID dello switch), P1 (porta di ingress0), P2 (porta di uscita) e poi l’utilizzazione della porta di uscita.

Una volta che il pacchetto arriva allo switch S2, sulla base dell’intestazione aggiunta dallo switch 1, vengono aggiunte in testa le nuove informazioni.

Infine, sullo switch numero 3, lo switch L2, vengono aggiunte queste informazioni e poi verranno rimosse andando a scorporare il payload dalla parte del monitoraggio attivo.

* Abbiamo quindi switch, o dispositivi software o hardware, che supportano schemi avanzati sul data plane. Schemi che permettono di andare a fare un monitoraggio più fine del traffico che passa.
* Chiaramente non posso applicare etichette diverse a tutti i pacchetti altrimenti inonderò la rete di un sacco di traffico aggiuntivo.
* Problema: aumenta la dimensione del pacchetto e quindi ho il problema della MTU (Maximum Transmission Unit), che è la massima dimensione del pacchetto. Se ho già un pacchetto di dimensione massima non posso aggiungerci etichette. Quindi al fine di supportare questo schema dovrei utilizzare un MTU all’interno dei dispositivi un po’ più bassa.
* L’efficienza di uso di un collegamento è tanto più grande quanto più grandi sono le dimensioni di un pacchetto. Più è veloce l’interfaccia, più bit riesco a trasmettere nella stessa finestra temporale. Maggiori sono i bit del payload e minori sono i bit di intestazione: se io riduco la dimensione del payload l’intestazione pesa sempre di più e avrò uno schema molto performante, avanzato, ma scarsamente efficiente.
* Una delle regole base 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.

Le soluzioni che abbiamo a disposizione sono:

* Dispositivi di rete che supportano in maniera nativa INT
* Schede di rete programmabili che supportano INT (con P4)
* Server che supportano eBPF (Extended Berkeley Packet Filter). BPF sono strutture di filtraggio che stanno nel kernel unix e linux associati ai primi socket inventati all’università di Berkeley, quindi agli arbori. eBPF sono dei nuovi paradigmi che permettono di fare cose più sofisticate a livello di kernel, tra cui le tecniche INT
* Server utilizzato come hypervisor, ad esempio di macchine virtuali. Se come soluzione di rete per queste macchine virtuali utilizzo OvS (Open vSwitch, v sta per virtual), allora questo package supporta la telemetria di rete

Immagine che contiene testo, schermata, design

Descrizione generata automaticamente

Adesso vediamo due soluzioni: una che riguarda il supporto hardware della telemetria e una sul supporto software.

**Un quadro interessante: P4**

Immagine che contiene testo, schermata, Carattere, diagramma

Descrizione generata automaticamente

Programming Protocol-independent Packet Processors (P4) è un linguaggio specifico del dominio per i dispositivi di rete. Specifica il modo in cui i dispositivi del piano dati elaborano i pacchetti.

Consiste nella possibilità di caricare a run time all’interno di dispositivi predisposti delle routine scritte in linguaggio C, che quindi effettuano l’elaborazione dei pacchetti. Tra le funzioni che posso caricare ci sono anche funzioni di monitoraggio.

Il primo e il terzo blocco di questo ciclo (osserva, analizza e reagisci) riesco a farlo con questi tool. Riesco quindi sia ad osservare sia a reagire, analizzare dipende perché tipicamente l’analisi viene fatta in una diversa sede in quanto i tool hanno capacità di elaborazione limitate a causa della limitazione della dimensione del codice che posso caricare. Riescono quindi a fare molto velocemente sia operazioni di monitoraggio dettagliate e customizzate sia azioni di risposta a eventuali situazioni anomale o per ottimizzare il funzionamento della rete.

Sono soluzioni molto interessanti perché permettono di aggiungere delle funzioni che non sono supportate in maniera nativa dal costruttore.

Come funziona un programma P4?

Vediamo l’esempio viola in alto. Innanzitutto i pacchetti vengono classificati per capire se bisogna applicare una qualsiasi delle funzioni definite all’interno di P4 a quel pacchetto oppure no. Supponiamo di si, queste funzioni potranno riguardare sia il routing che il forwarding e potranno riguardare delle regole di tipo attivo (le ACL, ad esempio la direzione del pacchetto, il pacchetto può essere scartato o può essere inviato ad un controller remoto che decide cosa farci).

Come faccio a caricare un programma di questo tipo?

Tipicamente quello che riesco a fare è creare un mio software, da qualche altra parte non all’interno dello switch, poi dovrò compilarlo e ottengo un modulo eseguibile. Questo modulo sarà caricato all’interno dello switch attraverso delle interfacce API dedicate. Contemporaneamente, oltre che caricare il modulo devo anche preparare il dispositivo per accogliere questo modulo, quindi andrò a generare una sorta di contratto per il data plane; di fatto è un manifesto che spiega in cosa consiste il modulo che andrò a caricare. Il modulo può essere caricato anche tramite un unload, quindi posso disattivarlo o toglierlo dall’esecuzione.

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

Tutti questi framework supportano quindi una validazione preliminare della bontà del codice: se per errore viene scritto un codice che ha un loop infinito, sicuramente non passerà il test di validazione.

L’elaborazione avviene attraverso una cosiddetta pipeline, un certo numero di stage che sono:

Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamente

1. Il Parser iniziale, che estrae i bit dalle intestazioni dei pacchetti. Si guardano le intestazioni e non i payload perché spesso vengono utilizzate tecniche di cifratura.
   * All’interno di un nodo di un data center avrò a disposizione lo strato 2, che è Ethernet, lo strato 3, che è IPv4 o IPc6, e lo strato 4, che è TCP/UDP
2. Match Action: una classificazione del pacchetto e l’identificazione dell’azione che, sulla base di uno specifico match, deve essere eseguita.

I pacchetti passano attraverso la pipeline di ingresso per le modifiche dell'intestazione e la selezione della porta di uscita

1. Inoltro sull’interfaccia di uscita e accodamento su opportuni buffer. I pacchetti vengono inviati alle code/buffer per l'accodamento, la pianificazione e la replica dei pacchetti
2. Una nuova azione da matchare sull’interfaccia di uscita. I pacchetti passano attraverso la pipeline Egress per ulteriori modifiche
3. I pacchetti passano attraverso il Deparser per la serializzazione e poi escono dal dispositivo

Un programma P4 può quindi effettuare operazioni di modifica del pacchetto sia in ingresso che in uscita, sempre sulla base di specifiche condizioni che devono essere all’interno del programma, tipicamente organizzato in una pipeline.

Per far funzionare P4 abbiamo bisogno o di un dispositivo che lo supporti o di una scheda fisica che lo supporti. Esistono anche emulatori software open source, lo svantaggio enorme è la velocità: l’emulatore di un dispositivo va molto molto più lento del dispositivo stesso, e quindi è possibile utilizzarlo per avere una dimostrazione di tipo funzionale, ma non prestazionale.

**Un'ulteriore opzione in Linux: eBPF**

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

eBPF 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. Quindi, una scatoletta isolata dove non posso fare danni e all’interno del quale posso parlare con lo user space: porto programmi dello user space dentro il kernel.

In che punto del kernel? Prima ancora che vengano eseguite le regole di iptables, quindi prima del prerouting.

Posso andare a osservare, monitorare, modificare un pacchetto con eBPF.

Posso programmare un programma in eBPF in vari modi, in C, in python, ecc, perché ci sono poi delle librerie che convertono il programma in bytecode. Quando viene effettuata la conversione viene effettuato anche un processo di verifica per evitare che il software comprometta il corretto funzionamento del calcolatore. Assumendo che il software passi la verifica, viene fatto il **JIT**, just-in-time compiler: viene fatta una compilazione run time in cui il compilatore decide quale parte del bytecode andare a compilare in maniera più veloce (ad esempio perché quella parte di codice viene ripetuta in continuazione) e quale invece andare a interpretare per un’esecuzione immediata, anche se meno veloce.

Si pone questo problema perché creo il software e poi lo voglio eseguire immediatamente, quindi se la compilazione richiede 10 minuti magari non è compatibile con il mio uso.

Applicazioni tipiche: osservabilità e tracciamento (monitoraggio), soluzioni di rete tipo instradamento custom o load balancing oppure tecniche di sicurezza (incapsulamento di pacchetti, tunnelling, cifratura).

All’interno del kernel, una volta che ho il modulo, posso avere accesso a delle strutture dati che prendono il nome di mappe (**Maps**), che sono le strutture dati con cui i vari pezzettini del software si scambiano i dati e li mettono a disposizione dello user space. Se vengono estratti dati di telemetria, vanno anche esportati in qualche modo, ad esempio in questo modo si mettono a disposizione questi dati che sono presenti all’interno del kernel.

Il numero di funzioni che possono essere richiamate da dentro il kernel è limitato e prendono il nome di **Kernel Helper API**.

L’eBPF viene usata anche spesso come tecnica di accelerazione.

Nel mondo SDN andiamo verso una softwarizzazione dei dispositivi di rete perché più mantenibile, più semplice, aggiornabile e costa meno. Il prezzo da pagare è che il software è molto meno veloce dell’hardware. Se però mi spingo a livelli molto bassi della pila protocollare, quindi sotto il kernel, prima ancora che riesco a processare i pacchetti, riesco ad accelerare in maniera significativa queste operazioni. Quindi anche se adotto tecniche basate sulla softwarizzazione riesco comunque ad avere velocità molto importanti, siile a quelle che riesco ad avere in hardware.

Siccome eBPF viene seguito sul server, a seconda della potenza di calcolo del calcolatore potrei avere anche in parte delle capacità di analisi. Sicuramente ho delle capacità di osservazione e di reagire (generare e caricare un nuovo software che permette di eseguire delle modifiche speciali a seguito della rilevazione di malfunzionamento o di una situazione particolare che richiede una nuova funzione di rete)

**Formato INT**

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

I pacchettini INT sono costituiti dall’intestazione, che di solito è molto semplice, in questo caso sono 8 byte (due righe da 4 byte): la versione, i flag, i contatori di istruzioni, il numero massimo di salti, il numero di salti totali e poi i metadati, cioè quelli che vengono inseriti da ciascun nodo che è attraversato in un framework in-band network telemetry, in testa quello più recente e in coda il primo.

Di solito questo è il formato classico.

Questa intestazione e questo payload (cioè i metadati) vengono inseriti tra l’intestazione e il payload del pacchetto che viene trasportato.

**INT raccoglie lo stato della rete**

Meccanismi per la raccolta dello stato della rete nel dataplane:

* Il più vicino possibile al tempo reale
* Alle tariffe di linea attuali e future
* Con un framework capace di adattarsi nel tempo, anche a cambiamenti di tipo tecnologici

Esempi di stato della rete:

* ID dello switch (per switch si intende un dispositivo di commutazione, quindi switch di strato 2 o router), ID della porta di ingresso/uscita
* Utilizzo del collegamento in uscita
* Latenza del salto, quindi quanto tempo ci metto per attraversare il dispositivo (riesco a misurare facilmente il tempo perché utilizzo lo stesso orologio, che è quello locale del dispositivo)
* Occupazione della coda in uscita
* Stato di congestione della coda in uscita

**Dati INT**

La rete dovrebbe rispondere a queste domande:

* Quale percorso ha preso il mio pacchetto?
* Quali regole ha seguito il mio pacchetto?
* Con chi (quali flussi) ha condiviso le code?

Questo perché ci possono essere dei fenomeni transitori, il cui impatto è però non trascurabile e che sono difficili da identificare con soluzioni classiche orientate al flusso.

Se ad esempio mi accorgo che un’interfaccia di uscita è congestionata potrei avere 2 soluzioni a disposizione: cambio l’instradamento al mio flusso oppure cambio l’instradamento ai flussi che attraversano lo stesso link, a seconda dello stato generale della rete.

Analisi approfondita in tempo reale:

* Anomalie:
  + Flussi congestionati
  + Latenza hop/e2e elevata
  + Cambio di percorso/loop
* Eventi:
  + Nuovo flusso/terminazione
  + Modifica della latenza
  + Switch/link non utilizzato
* Rilascia report con metadati avanzati:
  + Timestamp
  + Motivo dello scarto del pacchetto
  + Pacchetto da 5 tuple/metadati
  + ID dello switch
  + ID della porta di ingresso/uscita
  + ID della coda

**Analisi del flusso INT**

INT può acquisire informazioni più dettagliate sullo stato del flusso individuale. Ad esempio, ritardo, dimensione del buffer, caratteristiche del sotto percorso della trasmissione multi-percorso, ecc.

Come sfruttare appieno l'enorme quantità di dati nell'ambiente di rete reale? Gli amministratori di rete possono stabilire in modo rapido e intuitivo modelli statistici accurati per ciascun flusso e progettare soluzioni di ottimizzazione.

Strumenti: analisi di serie temporali, analisi di regressione, analisi di correlazione, ecc.

Risolvere anche i problemi prima che le prestazioni della rete si deteriorino troppo.

**Modalità INT**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Per implementare la in-band network telemetry ho 3 macro-soluzioni.

**INT-XD (EXport Data)**

Non comporta una vera modifica del pacchetto. All’interno del pacchetto metterò solo delle istruzioni, una cosiddetta watchlist, non metterò i metadati.

Sulla base della watchlist esporto tutta una serie di informazioni.

* Configuro i nodi con la watchlist

Immagine che contiene testo, schermata, diagramma, schermo

Descrizione generata automaticamente

Il pacchetto è quello grigio (Header Payload) e non viene mai modificato.

All’interno dei nodi vado a configurare la watchlist: di fatto sto andando a creare dei filtri che triggerano delle operazioni di monitoring. Quindi quando mi arriva un pacchetto che matcha le istruzioni della watchlist genero un pacchetto di metadati: invio un pacchetto che riporta l’intestazione del pacchetto più i metadati del nodo A (info A).

Il pacchetto poi passa al prossimo switch, e se anche lui ha una watchlist invierò di nuovo l’intestazione del pacchetto più i metadati del nodo B.

Infine, la stessa cosa per lo switch C.

Come faccio poi a risalire al percorso, alle regole e a con chi ho condiviso la coda? Tutte queste informazioni le posso dedurre perché ho una copia sia 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. Guardando poi dentro ai metadati riesco a risalire all’interfaccia di ingresso e di uscita, allo stato delle code e così via. Tutto questo lo invio al collettore.

Vantaggi: non modifico i pacchetti e quindi il problema della MTU non esiste. Potremmo quindi classificarla come una soluzione puramente passiva. Sfrutta la programmabilità dei dispositivi, perché posso fare la watchlist complicata come voglio, ma è puramente passiva.

Enorme difetto: genera una quantità di dati di monitoraggio enorme. Per ogni dispositivo che attraverso genero un pacchetto di report e questo può creare un overhead significativo. Questo nel caso in cui il pacchetto che vado a monitorare contiene i dati. Se consideriamo una sessione tcp, che è bidirezionale, e sono interessato ad analizzare entrambi i versi. Parecchi pacchetti tcp saranno solo ack, cioè pacchetti vuoti con intestazione ip e tcp. Genero una di queste, che prende il nome di postcard (cartolina), per ogni pacchetto vuoto; quindi il pacchetto con i metadati è più grande del pacchetto che vado a monitorare. Posso perciò usare questa soluzione solo per brevi intervalli di tempo, altrimenti avrò un sovraccarico della rete enorme, andando a consumare più risorse di quante sono quelle che dovrei monitorare.

**INT-MX (EMbed Instruct(X)ions)**

In questo caso vado a scrivere sul pacchetto solo le istruzioni, poi sulla base di queste istruzioni vengono catturate un certo numero di informazioni che vengono esportate verso un collettore.

* Non configuro i nodi, ma scrivo sul pacchetto le istruzioni per i nodi

**Immagine che contiene testo, schermata, software, design

Descrizione generata automaticamente**

Creo la watchlist solo sul primo nodo. Questo nodo con la watchlist è un modulo, che quando il pacchetto matcha alcune condizioni, una o più regular espression, andrà a inserire una specifica istruzioncina sul pacchetto.

Ciascun nodo non avrà bisogno di configurare una watchlist ma dovrà semplicemente eseguire le istruzioni scritte sul pacchetto. Le istruzioni sono tipicamente molto limitate perché non riguardano i metadati.

All’interno del pacchetto aggiungo quindi un ulteriore intestazione e così ciascuno switch che è in grado di interpretare queste istruzioni invierà questi pacchetti.

Vantaggio: devo conoscere solo il primo nodo, mentre nel caso INT-XD devo conoscere tutti i nodi. In questo caso il primo nodo ha la watchlist e tutti gli altri sono innescati dalla presenza di queste istruzioni. L’ultimo nodo, che deve avere la capacità di capire di essere l’ultimo, dovrà rimuovere l’intestazione, che non deve essere visibile all’host finale.

Svantaggio: anche qui l’overhead è enorme, ogni nodo attraversato genera una postcard, quindi una copia dell’intestazione del pacchetto più i metadati.

Però è più flessibile, perché di fatto devo conoscere solo il punto di ingresso del mio flusso, non devo conoscere tutti i punti intermedi, anzi serve proprio per capire i punti intermedi.

Nel caso precedente, o la configuro su tutta la rete questa watchlist e l’events detection, oppure devo conoscere il routing, che potrebbe non essere banale.

**INT-MD**

In questo caso scrivo direttamente i metadati sul pacchetto, che è quello visto finora.

Quindi sarà solo l’ultimo pacchetto, quello che rimuove le etichette, che poi invia i metadati al collettore.

* Scrivo i metadati sul pacchetto

**Immagine che contiene testo, schermata, software, numero

Descrizione generata automaticamente**

In questo schema metto la watchlist solo sul primo nodo e l’event detector solo sull’ultimo nodo. Devo quindi conoscere il primo e l’ultimo nodo. Con la watchlist non solo inserisco le istruzioni ma inserisco anche i metadati, e in questo caso i metadati li metto sul pacchetto.

Lo switch intermedio, switch B, osserverà le istruzioni e aggiungerà i suoi metadati, tipicamente in testa, dopo le istruzioni.

Sull’ultimo switch, che devo conoscere, devo mettere la rilevazione dell’evento: l’evento qui è che siamo arrivati alla fine del percorso e quindi devo rimuovere le informazioni collezionate sul percorso, le devo inviare a un nodo di monitoring e devo ripristinare il pacchetto nella sua forma originale.

Vantaggi: devo inviare un solo pacchetto di monitoraggio per ogni pacchetto che attraversa la rete. Il pacchetto di monitoraggio potrebbe non essere piccolo, ma la sua intestazione sarà usata una volta sola, non per ogni nodo che attraversa.

Svantaggi:

* MTU: devo avere una stima di quanto è grande la dimensione massima delle istruzioni più i metadati e diminuire l’MTU dei dispositivi attraversati di conseguenza.
* Se il pacchetto non arriva al nodo finale perdo tutto il monitoraggio, è come se il flusso non esistesse.

Quindi è molto più efficiente dal punto di vista dell’impatto del traffico di monitoraggio però è anche più sensibile alle perdite, quindi se ho un problema di congestione all’interno della rete e anche il traffico di monitoraggio ne venisse affetto è un bel problema.

**Svantaggi e sfide dell'INT**

Il rapporto di carico utile del pacchetto normale è ridotto a causa dell'incapsulamento delle istruzioni di telemetria e dei metadati.

Limitazione dell'unità di trasmissione massima del pacchetto (MTU) causata dal lungo percorso di telemetria.

La costruzione, l'incapsulamento, il riempimento e l'estrazione delle istruzioni di telemetria e dei metadati aumentano il carico di elaborazione dello switch.

INT è sensibile alla perdita di pacchetti e non può risolvere il problema dei dati di telemetria mancanti a causa della perdita di pacchetti.

La telemetria di rete in banda consumerà parte della larghezza di banda della rete.

La maggior parte dei protocolli di incapsulamento come TCP contengono checksum. L'inserimento delle meta-misure richiede l'aggiornamento dei campi checksum. Aumenterà il costo di elaborazione del passaggio.

La telemetria della rete in banda può generare una grande quantità di dati di telemetria. La lunghezza totale dei dati di telemetria di un pacchetto è proporzionale al numero di punti di telemetria. Potrebbe anche superare la dimensione del pacchetto originale.

* In generale quello che si ha è che la telemetria è uno strumento di debugging molto raffinato che può dare informazioni molto precise ma che va usato in maniera parsimoniosa, proprio perché abbiamo visto che la quantità di dati generata può essere molto importante, soprattutto su reti ad alta velocità.

Questa considerazione vale fino ad un certo punto all’interno dei data center, soprattutto se le regole di monitoraggio riguardano flussi molto brevi. Se vado ad osservare solo i flussi attraverso specifici microservizi magari la quantità di traffico di monitoraggio che vado a generare potrebbe essere più modesta, ma non è detto, dipende anche da come è configurato l’instradamento all’interno dei microservizi.

Uno dei problemi potrebbe essere proprio l’instradamento che per qualche errore di configurazione può essere non ottimale e attraverso queste tecniche si può cercare di risolvere.

Fine 17/10