Università degli Studi di Pisa

Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea Specialistica in Tecnologie Informatiche

Tesi di Laurea

Disegno e validazione di un'architettura distribuita per il monitoraggio del traffico di rete basata sul protocollo Cisco NetFlow

Candidata
Maria Teti

Relatore
Dott. Luca Deri

Controrelatore
Prof. Marco Danelutto

Anno Accademico 2003/2004

Indice

Introduzione

La gestione delle reti e in particolare la misurazione del traffico di una rete, è stata considerata un'attività necessaria fin dai primi tempi della nascita delle reti. Gli amministratori hanno sempre dovuto tener traccia del traffico delle rete per diverse ragioni, che vanno dalla scoperta di colli di bottiglia alla vera e propria pianificazione delle rete per un futura estensione, alla scoperta di pattern di traffico anomalo.

Tuttavia, negli ultimi anni, questa attività ha dovuto far fronte a dei nuovi problemi ed è diventata un'attività sempre più complessa. Le reti, infatti, hanno conosciuto un forte sviluppo in termini di grandezza, velocità e flessibilità. Sono sempre di più le organizzazione che basano la proprie attività sulle reti per le comunicazioni e il commercio elettronico. Le LAN esistenti sono state spesso ampliate, aggiornate ed interconnesse la mondo di Internet. Inoltre l'integrazione di diversi apparati di rete rende difficoltoso il compito di identificare le varie comunicazioni.

In uno scenario dinamico, come quello descritto, si può ben capire come gli amministratori di rete abbiamo bisogno di strumenti automatici, che supportino lo sforzo umano nel raccogliere informazioni sullo stato e sul comportamento degli apparati di rete. Secondo [32], il monitoraggio di rete è l'aspetto fondamentale della gestione delle reti automatico, ecco perché sempre più le azienda si affidano a tali strumenti per poter monitorare l'utilizzo delle proprie reti.

Nel mondo UNIX esistono strumenti che permettono di controllare in modo semplice la connettività e altri che funzionano da *sniffer*, come il tool `tcpdump`. Questi strumenti risultano molto potenti, ma hanno lo svantaggio di aver bisogno di una fase di analisi offline dei dati ottenuti. Allo stesso modo, le sonde per la rete, come RMON (Remote MONitoring), risultano potenti nello svolgere il proprio compito, ma necessitano di protocolli di gestione sofisticati, come SNMP (Simple Network Management Protocol), in modo da poterli configurare opportunamente. Esistono molti altri strumenti per effettuare il monitoraggio, ognuno dei quali cerca di strutturare caratteristiche diverse del traffico generato da una rete (per esempio

l'analisi dei flussi generati da una rete).

Scopo della tesi

Lo scopo di questa tesi è quello di proporre un'architettura distribuita che permetta di ottenere un sistema di monitoraggio del traffico generato da una rete. Tale sistema deve essere in grado di generare delle statistiche di un insieme di metriche su informazioni rilevanti del traffico. La generazione di tali statistiche deve essere fatta analizzando i dati catturati dalla rete, tramite un insieme di sonde e di device opportunamente configurati per effettuare tale raccolta, in modo da non rappresentare un elemento di disturbo, e quindi senza far variare l'effettiva misurazione delle performance dei sistemi controllati. L'analisi del traffico, o parte di esso, tramite le statistiche generate, deve permette all'amministratore di avere una reale comprensione, o la tendenza, della qualità del servizio offerto della rete monitorata. Poiché le reti stanno diventando sempre più veloci, il volume di traffico che le attraversa è maggiore rispetto agli anni passati. Il sistema che si intende implementare in questa tesi deve essere capace di far fronte alla grande quantità di dati che vengono raccolti dalla rete. I dati saranno quindi memorizzati all'interno di un database, in modo temporizzato, cercando cioé di mantenere solo le informazioni più nuove e significative. Inoltre, data la grande quantità di dati che il sistema si troverà ad analizzare, per rappresentare le metriche sarà usata una forma compressa: i dati saranno cioé filtrati e aggregati, in modo da integrare tra di loro i dati con caratteristiche comuni, cercando di perdere meno informazioni possibili e soprattutto comprimendo il volume di dati che andranno a memorizzarsi all'interno del database. Infine trattandosi di una architettura distribuita il sistema dovrà gestire l'integrazione di dati, che spesso contengono informazioni molto differenti, e permettere di modificare in maniera dinamica la propria configurazione in modo da piegarlo alle esigenze del momento.

Struttura della tesi

- **Capitolo 1** In questo capitolo verrà effettuata una panoramica sul significato di *network monitoring*, descrivendo gli ambiti di interesse e alcune tecniche che è possibile adottare.

- **Capitolo 2** Verrà presentata una panoramica sullo stato dell'arte attuale, cercando di capire i trend e le esigenze attuali. Verrà inoltre posta l'attenzione sul protocollo dell Cisco NetFlow, utlilizzato per lo sviluppo dell'architettura proposta.

- **Capitolo 3** Verrà presentata la specifica dell'architettura, descrivendone i requisiti, i vincoli e gli scenari si applicazione.

- **Capitolo 4** In questo capitolo verrà discussa la validazione del sistema, dimostrando come sono stati validati i requisiti individuati per l'architettura, alla luce delle scelte effettuate nel capitolo 3.

- **Capitolo 5** Verrà presentato un possibile scenario di applicazione del sistema, descrivendo il comportamento del sistema in tale contesto.

Capitolo 1

Il Network monitoring

Sommario

In questo capitolo viene fornita una breve introduzione sul network monitoring, sul suo significato e sui diversi approcci possibili.

1.1 Introduzione

All'interno degli ambienti di aziende e organizzazioni che fanno uso di reti per il loro funzionamento, poter effettuare le operazioni di network monitoring può avere un'importanza fondamentale. Se infatti non si effettuano delle misurazioni sulla rete, non è possibile avere della documentazione oggettiva o dei benchmark su come la rete stessa si comporta. La mancanza delle funzionalità di monitoraggio può portare a grosse difficoltà nel valutare se i cambiamenti che vengono attuati aumentino la performance della rete o se la degradano in qualche modo.

Di conseguenza ci sono diversi motivi per effettuare il *network monitoring*.

Prima di tutto, poter effettuare un qualche tipo di network monitoring aiuta una data organizzazione o compagnia ad essere più efficiente. Per esempio, alcuni warning o alcuni errori all'interno di una rete possono essere scoperti e corretti prima di giungere a situazioni più gravi in cui si può avere la perdita a livello di produttività dell'azienda che possiede la rete stessa.

Secondariamente, il network monitoring aiuta ad applicare le politiche di sicurezza all'interno di una data azienda. Poter sondare e controllare i device chiave delle risorse della rete aziendale in determinati intervalli di tempo, assicura che la rete non

arrivi ad un pesante arresto. Il monitoraggio, in questo caso, permette ad un'azienda di essere molto più efficiente. Se si verifica un problema, può essere riconosciuto e può essere facilmente risolto.

Infine, permettere un monitoraggio costante della rete permette di poter meglio pianificare futuri aggiornamenti e configurazioni. Per esempio, le aree dove occorre apportare dei miglioramenti possono essere individuate facilmente e i problemi risolti parzialmente o eliminati tutti insieme.

Il network monitoring è quindi un compito importante che gli amministratori di una rete, all'interno di una azienda, devono svolgere.

Di seguito verranno descritte le aree di interesse del network monitoring, cosa interessa monitorare all'interno di una rete.

1.2 Network monitoring

Il *Network monitoring* è la funzione di collezionamento delle informazioni utili per il network management. Tali tipi di applicazioni sono create per collezionare dati necessari alle applicazioni che effettuano invece la gestione.

Il compito, quindi, del network monitoring è il collezionamento di informazioni da differenti parti della rete in modo che la rete stessa possa essere gestita e controllata usando le informazioni raccolte. Molti dei device di rete sono allocati in parti remote. Inoltre solitamente non sono connessi a dei terminali, cosí che le applicazioni che effettuano la gestione non possono facilmente monitorare il loro stato.

Per questi motivi le tecniche di monitoraggio sono sviluppate per permettere alle applicazioni di network management di controllare lo stato dei loro apparati di rete.

Inoltre, poiché le reti stanno diventando sempre più complesse grazie anche al numero sempre crescente di device tra di loro collegati, le tecniche per il network monitoring sono state espanse in modo da permettere di monitorare una rete come un tutt'uno. In uno scenario di questo tipo, in cui sempre più persone usano le reti, che spesso risultano collegate ad Internet, le applicazione che vengono usate per poter effettuare il monitoraggio hanno bisogno di usare dei metodi effettivi per controllare lo stato delle loro reti e fornire servizi di rete economici, ma di alta qualità. È veramente importante quindi comprendere quali sono glo obiettivi che il network monitoring deve poter raggiungere: conoscendo infatti gli obiettivi, le

applicazioni possono scegliere tra le varie tecniche quelle che meglio le aiutano nel loro compito.

Secondo [32] sono generalmente tre gli obiettivi del network monitoring:

- Performance monitoring;
- Fault monitoring;
- Account monitoring;

Questi obiettivi coprono tre delle cinque aree funzionali per il network management [14] proposti da OSI . Le altre due aree funzionali non riguardano il monitoraggio, e sono configuration management a security management.

1.2.1 Performance monitoring

Con il termine *Performance monitoring* si intende la misurazione della performance della rete. I compiti fondamentali del performance monitoring sono tre. Il primo scopo consiste nell'utilizzare le informazioni di performance monitoring per pianificare una futura espansione della rete e per localizzare i problemi correnti che riguardano l'utilizzo della rete. Il secondo scopo è quello effettuare le misurazioni di performance in un arco di tempo grande a sufficienza, in modo da poter costruire un modello di comportamento per la rete stessa. Come terzo scopo, occorre scegliere quali sono le misure importanti, quali sono cioé le cose che interessa conoscere. In una rete sono molte le cose che si possono misurare, ma devono essere scelte in modo da risultare significative. Le cose che ci interessa misurare vengono solitamente chiamate *indicatori*, perché indicano degli attributi delle rete. Un esempio di indicatori per la performance sono i seguenti attributi:

- **Availability**: è la percentuale di tempo in cui un sistema di rete, un componente o un applicazione è disponibile per un utente.
- **Response time**: è il tempo impiegato da un sistema e reagire ad un dato input.
- **Accuracy**: è la percentuale di tempo in cui non occorrono errori durante la trasmissione e l'invio delle informazioni.

- **Throughput**: è la misura della quantità di dati che può essere inviata su un link in una quantità di tempo.

- **Utilisation**: è la percentuale della capacità teorica di una risorsa (es. multiplexer, switch) che si sta usando.

Funzioni per il performance monitoring

Di seguito verranno descritte quelle che sono le azioni che solitamente vengono intraprese per poter effettuare le operazioni di misurazione della performance. Tali azioni consistono nelle azioni di misurazione della performance vera e propria in cui avviene la raccolta delle informazioni delle statistiche del traffico di una rete. Solitamente nella LAN tale operazione viene effettuate semplicemente raccogliendo le informazioni con un monitor remoto, che semplicemente osserva il traffico che passa sulla rete. Abbiamo poi l'analisi della performance riscontrata, che si ottiene tramite la riduzione a livello software e la rappresentazione dei dati raccolti. In questo modo è possibile dare risposte a domande del tipo si sono verificati errori o altre inefficienze oppure qual è l'impatto dell'incremento del traffico o della variazione della grandezza dei pacchetti che sono trasmessi. Infine vi sono le funzioni che riguardano la generazione in modo sintetico del traffico, in modo da poter osservare il comportamento della rete sotto un carico di traffico ben controllato.

1.2.2 Fault monitoring

Con il termine *Fault monitoring* si intende la misurazione dei problemi che occorrono in un rete. Gli scopi fondamentali in questo caso sono due. Il fault monitoring riguarda la possibilità di poter rilevare errori nei vari livelli della rete: è quindi importante sapere in quali livelli si sta realmente presentando un problema. Il secondo scopo riguarda la capacità di poter stabilire quelle che sono le caratteristiche normali di una rete in un esteso periodo di tempo. In una rete, infatti, occorrono sempre degli errori, ma quando tali errori si presentano, non significa necessariamente che la rete abbia dei problemi persistenti: ci si aspetta che alcuni errori si presentino. Per esempio, la presenza di un rumore in un link di rete può causare un errore di trasmissione: la rete ha dei problemi solamente quando il numero degli errori cresce immediatamente sopra il suo normale valore, che identifica il suo comportamento

normale della rete. Per questi motivi occorre poter stabilire qual è il comportamento normale di una rete.

Funzioni per il Fault monitoring

Le azioni solitamente adottate per poter effettuare le operazioni di fault monitoring sono le seguenti. Mantenere dei file di log degli eventi significativi e degli errori che sono occorsi. Rispondere agli eventuali polling[1] richiesti dai manager. Riportare gli errori riscontrati ad uno o più manager.

Altre operazioni che è possibile intraprendere sono cercare di anticipare i fault, settando in maniera opportuna i valori di soglia e generando dei report quando tali soglie vengono superate. Aiutare nell'isolare e nel diagnosticare i fault con test standard, come la connettività e il response time. Fornire una buona interfaccia agli utenti, in modo che possano comprendere bene tali situazioni.

1.2.3 Account Monitoring

Con il termine *Account monitoring* si intende il modo in cui gli utenti usano la rete. La rete tiene traccia infatti di quali dei suoi device sono utilizzati dagli utenti e quanto spesso sono utilizzati. Questo tipo di informazioni sono utilizzate solitamente per poter effettuare delle operazioni di billing dell'utilizzo della rete da parte di un utente, e per predire il futuro utilizzo di una rete.

1.3 Cosa occorre monitorare

Ci sono diverse aree di interesse quando si vuole monitorare una data rete. La prima area riguarda i device (o i nodi) connessi alla rete stessa, quali sono questi device e come sono in relazione gli uni con gli altri all'interno della rete. La seconda area di interesse riguarda le applicazioni, le risorse, e i servizi disponibili su un particolare tipo di device. Entrambe queste aree sono importanti all'interno del processo di monitoraggio.

Prima di iniziare tale processo è quindi necessario scegliere che cosa interessa monitorare, anche perché quando le reti sono grandi, tale processo può risultare molto complesso se non opportunamente pianificato.

[1] Per polling si intende la tecnica di interazione tra *manager* e *agent* in cui il manager chiede esplicitamente all'agent di segnalare la sua presenza o il suo funzionamento.

È consigliabile monitorare solo quei device che si ritengono importanti e non tutti quelli che sono presenti all'interno di una rete: monitorare tutto potrebbe portare alla generazione di una grande quantità di traffico in aggiunta a quello già presente in un rete, apportando anche un grosso overhead.

1.3.1 Device critici

All'interno di una rete, i device più importanti sono sicuramente quelli che sono *critici*, cioé quei device che sono di vitale importanza e senza i quali la rete diventa instabile, non risponde correttamente o addirittura smette di funzionare o senza i quali una azienda o organizzazione non è in grado di lavorare. Diversi sono i device che possono essere inclusi in questa categoria. Alcuni esempi sono:

- **Server**: sono quei nodi che forniscono un qualche tipo di servizio di rete o altri servizi ad ogni altro host nella rete. Nelle reti odierne hanno un ruolo fondamentale. Uno dei problemi in cui si potrebbe incorrere è il fatto che il server è andato giù , cioé non è più disponibile e non è possibile usufruire dei servizi forniti.

- **Router**: sono quei nodi che dirigono il traffico attraverso la rete usando gli indirizzi IP. Richiedono la conoscenza della rete e di come determinare il nodo. In caso di un loro fallimento, è possibile che non si possano raggiungere alcuni nodi.

- **Switch**: sono dei device che selezionano un path o un circuito per inviare pacchetti di data alla loro prossima destinazione. Possono includere al loro interno anche la funzione di router. In caso di fallimento di questo device è possibile che i pacchetti di dati non siano inoltrati correttamente alla loro destinazione.

- **Firewall**: sono dei computer che hanno lo scopo di proteggere una rete locale, come una Intranet dal mondo esterno: in particolare possono scoprire accessi non autorizzati alle risorse. Il loro corretto funzionamento ha quindi un alto valore.

- **Hub**: sono dei comuni punti di connessione di una rete. Il loro corretto funzionamento è di vitale importanze per il funzionamento della rete.

1.3.2 Altri device

Oltre ai device critici descritti sopra, è possibile monitorare qualsiasi altro device all'interno di una rete. Tali device possono essere utili da monitorare per un amministratore di rete per diversi motivi. I seguenti device possono essere inclusi in questa categoria:

- **Workstation**: è un personal computer connesso alla rete, che condivide le risorse di uno o più computer più grandi.
- **Terminal**: è un device end-use con poco o nessun software proprio che solitamente si appoggia ad altri computer come un server per operazioni di connessione (non solo monitor e tastiere ma anche scanner o stampanti).

1.3.3 Applicazioni

Quando si monitora una rete non è importante soltanto tenere traccia dei differenti tipi di device che costituiscono la rete, ma anche della applicazioni che sono disponibili e che girano su di essi. Per esempio un server può fornire un file system comune o la possibiltà di condividere le stampanti. Diverse sono quindi la applicazioni che un amministratore di rete potrebbe voler monitorare. Alcune di queste applicazioni sono le seguenti:

- **Sistemi Operativi**: sono quei programmi che permettono di gestire tutti gli aspetti su un computer, incluso altri programmi. Per un amministratore di rete può essere necessario dover monitorare il funzionamento di sistemi operativi di rete.
- **File Server**: sono dei server utilizzati per memorizzare programmi e dati condivisi da differenti utenti della rete. Per questa ragione un amministratore può decidere di monitorarne il funzionamento.
- **Database Server**: sono dei server dedicati alla memorizzazione e alla ricerca di informazioni su database su una rete come una LAN.
- **Application Server**: sono dei server utilizzati dagli utenti di una rete per eseguire programmi e processare dati.

- **Print Server**: sono computer o altri device sulla rete che controllano le stampanti, per poter fornire servizi di stampa a tutti gli host connessi ad una rete.

1.3.4 Network Service

Oltre ai device e alle applicazioni, può esser utile tenere traccia del comportamento di alcuni semplici servizi di rete forniti dai device della rete: tali servizi potrebbero infatti giocare un ruolo di grande importanze sull'intera performance della rete. Alcuni di questi servizi sono:

- **FTP**: File Transfer Protocol usato per trasferire file sulla rete.
- **HTTP**: HyperText Transport Protocol usato per comunicazioni e connessioni ai server su World Wide Web.
- **POP3**: PostOffice Protocol 3 usato per le email in arrivo sulla rete.
- **IMAP4**: Internet Messaging Access Protocol 4 usato per le email in arrivo sulla rete.
- **TELNET**: protocollo che permette l'emulazione di un terminale in modo da permettere agli utenti della rete di andare su altri computer usando un terminale.

1.4 Come monitorare

È importante capire anche come poter effettuare il monitoraggio di una rete, capire quali sono cioé le tecniche che è possibile adottare. Per esempio, su una rete piccola il processo di monitoraggio viene solitamente effettuato con modalità differenti da quello per una rete che presenta invece grosse dimensioni. Allo stesso modo alcune volte si può usare il monitoraggio attivo, mentre altre volte è consigliabile l'uso del monitoraggio passivo e vice versa.

1.4.1 Reti piccole e reti grandi

Sulle reti di piccole dimensioni, quelle che solitamente presentano un numero di nodi connessi inferiore a 1000, il monitoraggio tende a non essere molto attivo sul

computer dove il monitoraggio ha luogo. Inoltre speso è sufficiente utilizzare una singola applicazione per svolgere tutto il lavoro.

Nella reti grandi, quelle in cui il numero di nodi connessi supera i 2000, il problema cambia molto. Se si usa una applicazione per effettuare il monitoraggio di tutti o quasi tutti i nodi, l'applicazione potrebbe creare molto traffico in aggiunta a quello già presente sulla rete, producendo quindi dei risultati che non rispecchiano la corrente e reale situazione della rete. Nei casi peggiori tale quantità di traffico potrebbe addirittura portare a un blocco della rete stessa. Inoltre, per monitorare reti di questa grandezza non sarebbe sufficiente una sola applicazione in esecuzione su un computer. In generale, per reti di questo tipo si tende a fare una scelta su cosa monitorare, scegliendo quegli elementi che risultano importanti, in modo da diminuire in carico di lavoro per le applicazioni.

1.4.2 Monitoraggio passivo

È utile ora distinguere tra monitoraggio passivo e monitoraggio attivo.

Il monitoraggio passivo significa che semplicemente si intercetta il normale traffico in entrata e in uscita da uno o più nodi. Ciò significa che il traffico di rete non è disturbato perché dai nodi non sono inviati sulla rete pacchetti di test.

Questo tipo di monitoraggio viene utilizzato solitamente per misurare i flussi di traffico e delle loro caratteristiche quali il numero di pacchetti o di byte che viaggiano attraverso particolari device.

1.4.3 Monitoraggio attivo

Il monitoraggio attivo consiste invece nell'inviare sulla rete dei pacchetti di test. In questo modo però il monitoraggio va ad impattare sul traffico stesso della rete: questo è uno dei motivi che deve indurre a stare molto attenti se si vuole usare questo tipo di monitoraggio. Infatti, più sono i nodi da monitorare, più è grande la quantità di pacchetti che occorre inviare sulla rete e quindi maggiore sarà l'impatto sul suo comportamento

1.4.4 Misurazioni in un solo punti e in più punti

Quando effettuiamo il monitoraggio, un'altra distinzione che occorre fare è quella della misurazione in un solo punto della rete e in più punti allo stesso momento.

Misurare in un solo punto della rete significa che viene utilizzato un solo nodo per effettuare il monitoraggio. Solitamente il monitoraggio passivo tende ad utilizzare la misurazione in un solo punto, poiché la rete viene monitorata controllando il traffico in uscita e in entrata ad un singolo nodo.

La misurazione in più punti, d'altra parte, implica la possibiltà di monitorare diversi nodi allo stesso tempo. Solitamente il monitoraggio attivo utilizza questo tipo di misurazioni.

1.5 Metriche comuni

La performance di una rete può essere verificata usando diverse metriche, usando di volte in volte quelle che meglio si adattano alle necessità. Di seguito verranno presentate alcune metriche comuni. Tale lista può naturalmente essere espansa

Latenza In termini semplici può essere definita come il tempo che impiega un pacchetto per andare da una sorgente ad una destinazione e ritornare indietro.

Packet Loss È definita come la parte di pacchetti perduta durante il transito tra un nodo sorgente e un nodo destinazione durante un intervallo di tempo specificato. Solitamente è espressa come percentuale dei pacchetti che sono stati inviati da una sorgente ad una destinazione.

Throughput Può essere definito come la quantità di dati trasferiti da un posto ad un'altro durante una specificata quantità di tempo. Solitamente si misura in bit per second (bps) o byte per second (Bps).

Availability È la percentuale di intervalli, su un intervallo di tempo specificato, in cui un particolare device sulla rete è disponibile per l'uso.

Utilization È la percentuale di tempo che una risorsa è utilizzata in un dato periodo.

Capitolo 2

Stato dell'arte

Sommario

Nel capitolo precedente è stata fornita una breve introduzione al significato di network monitoring. In questo capitolo verrà posta l'attenzione sulle tecniche flow-based, cioé basate sul concetto di flusso. Verrà posta in particolare l'attenzione sulla tecnologia della Cisco System, NetFlow e sullo standard IPFIX.

2.1 Introduzione

Uno dei compiti più critici nel mantenere sotto controllo una rete è catturare e analizzare il suo traffico. La complessità di questi compiti sta crescendo poiché le reti stanno diventando sempre più veloci.

Le fasi di cattura e di analisi dei dati possono essere descritte come un insieme di passi, come mostrato nella figura 2.1. Tutti i passi sono ugualmente critici quando ci si trova ad operare su una grande quantità di dati.

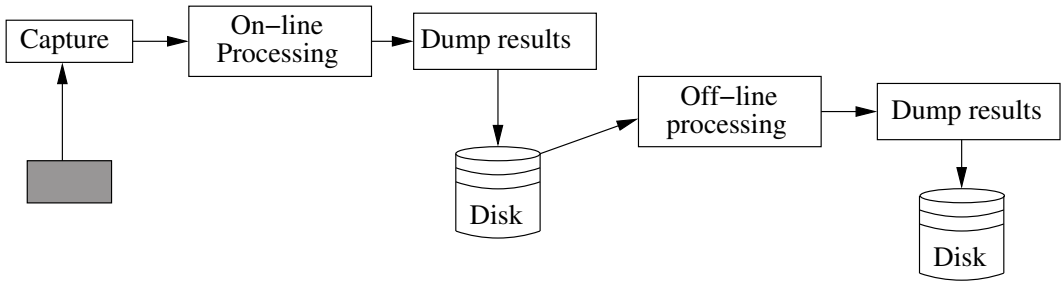

Figura 2.1: Passi base per la cattura e l'analisi del traffico di rete

Mentre le soluzioni ad hoc basate sull'utilizzo di hardware avanzato possono mitigare i problemi relativi ai primi due passi mostrati in figura 2.1, non esistono soluzioni altrettanto chiare per poter ridurre la criticità dei passi successivi. Su reti come quelle odierne dove la quantità di dati che transita è notevole, si riscontrano infatti due grossi problemi: da una parte, l'infrastruttura che occorre utilizzare per poter memorizzare una tale quantità di dati risulta essere costosa e complessa, e dall'altra parte, capire quali sono le informazioni rilevanti contenute all'interno dei dati che sono stati memorizzati è un'operazione costosa in termini di tempo e di risorse computazionali utilizzate. La complessità dell'elaborazione dei dati off-line, dipende quindi dalla quantità di dati e dal formato in cui tali dati sono memorizzati.

Diversi sono i metodi presenti nella letteratura che possono essere utilizzati per poter ridurre la quantità di informazioni relative al traffico di rete che devono essere memorizzate prima di una successiva elaborazione, ma in questa sede ci soffermeremo sulle tecniche di *Packet Sampling* e sulle tecniche *Flow-based*[1]. Di seguito vengono presentati alcuni esempi di approcci relativi a questi due metodi.

2.2 Packet Sampling

La tecnica del *Packet Sampling*[12] consiste nel catturare solo un sottoinsieme dei pacchetti, cioé uno ogni N pacchetti che transitano sulla rete.

In [12] vengono proposti tre differenti metodi per effettuare tale campionamento. Tutti e tre possono essere implementati sia in modalità *packet-driven*, in cui un contatore associato ai pacchetti fa scattare la scelta del pacchetto, che *timer-driven*, in cui è lo scadere di un timer che fa scattare la scelta del prossimo pacchetto da selezionare.

Il primo metodo descritto è il *systematic sampling*, in cui i pacchetti vengono scelti in modo deterministico ogni k elementi dell'insieme. Il secondo metodo, il *stratified random sampling*, è simile la primo metodo, ma nell'intervallo considerato, non viene prelevato il primo pacchetto, ne viene prelevato ogni volta uno scelto in modo casuale. Infine l'ultimo metodo, il *simple random sampling*, sceglie uniformemente n pacchetti tra tutti quelli che transitano sulle rete, senza considerare nessun intervallo.

[1] La nozione di flusso, anche se utilizzata da molti anni, non ha una singola definizione, ma ogni sistema tende ad utilizzare una propria definizione.

Sebbene differenti studi statistici hanno dimostrato che le proprietà statistiche possono essere ricavate dal traffico così campionato, senza una rilevante perdita di informazioni, questo approccio non può essere utilizzato nei casi un cui sia necessario analizzare tutti i pacchetti. Un esempio di tale situazione è la scoperta di attacchi alla rete che sono solitamente basati su un piccolo numero di pacchetti che sfruttano dei security bug.

2.3 Tecniche Flow-Based

Queste tecniche sono basate sul fatto che ogni pacchetto può esser associato ad un flusso (per esempio una connessione TCP). Un flusso può essere definito come un insieme di pacchetti che condividono il valore di alcuni campi all'interno dei loro header (per esempio indirizzo IP sorgente ed indirizzo Ip destinazione, porte sorgente e porta destinazione, etc.). Bisogna tuttavia ricordare che anche se in uso da molti anni, il concetto di flusso non ha ancora oggi una definizione standard, ma ogni sistema tende a personalizzare secondo le proprie esigenze tale definizione, considerando differenti campi.

Le tecniche Flow-based usano il valore comune di questi campi come unità di memorizzazione delle informazioni di traffico: in questo modo gli amministratori di rete non vedranno più i singoli pacchetti. Tuttavia questo approccio non può esser utilizzato in quei casi in cui sia necessario accedere alle informazioni di payload, per esempio nel caso di applicazione che scoprono gli attacchi di rete basandosi su alcuni dati contenuti nel payload dei pacchetti.

2.3.1 NetFlow e IPFIX

Una delle più comuni tecniche Flow-based è un protocollo della Cisco NetFlow [6][5][2], che sta per essere standardizzato dall'IETF IPFIX Working Group[9]. In NetFlow un probe, solitamente ospitato all'interno di un network device come un router o uno switch, analizza il traffico e crea un record per ogni flusso. I record per questi flussi sono quindi esportati ad un collector per una successiva elaborazione.

Sviluppato nel 1996 alla Cisco System, NetFlow è una tecnologia parte integrante del Cisco ISO Software, un software sviluppato per collezionare e misurare i dati in entrata su specifiche interfacce dei router e degli switch, permettendo in particolare di accedere alle informazioni dei flussi IP. Gli elementi di rete (routers e switch)

raccolgono i dati dei flussi e li esportano ai collector. I dati cosí collezionati possono essere usati per scopi differenti: analizzando i dati di NetFlow, un amministratore di rete può per esempio identificare la causa di una congestione, può determinare tipo di servizio utilizzato da ogni utente e da ogni applicazione, può identificare la sorgente e la destinazione del traffico della rete.

Tradizionalmente NetFlow può essere utilizzato per sviluppare diverse applicazioni, tra le quali:

Network Monitoring I dati di NetFlow permettono di avere la possibilità di monitorare le reti in modo ampio e quasi real-time. Le tecniche di analisi basate sui flussi possono essere utilizzate per visualizzare pattern di traffico associate sia con router e switch individuali, che con un'intera rete per fornire la possibilità di scoprire i problemi, e di risolverli in maniera efficiente.

Application Monitoring e Profiling I dati raccolti con NetFlow permettono agli amministratori di rete di avere una dettagliata visione dell'utilizzo delle applicazioni nella rete. Tali informazioni possono essere utilizzate per pianificare e capire nuovi servizi e per allocare le applicazioni e le risorse di rete , in modo da rispondere alle esigenze degli utenti della rete.

User Monitoring I dati di NetFlow permettono anche di comprendere qual è l'utilizzo delle rete e delle applicazioni disponibili da parte degli stessi utenti della rete. Le informazioni raccolte in questo modo possono essere utilizzate per pianificare gli accessi alle risorse e alle applicazioni. Inoltre è possibile utilizzare tale informazioni per risolvere potenziali violazioni delle politiche di sicurezza.

Network Planning NetFlow può essere utilizzato per catturare i dati su un lungo periodo di tempo ottenendo in questo modo la possibilità di tracciare e anticipare la crescita della rete e di pianificarne gli aggiornamenti.

Security Analysis NetFlow identifica e classifica gli attacchi di tipo DOS, i virus e i worm in real-time. I cambiamenti indicano anomalie che sono chiaramente dimostrati all'interno dei dati di NetFlow.

Accounting/Billing I dati NetFlow permettono di ottenere delle misurazioni a grana fine (e.g i dati sui flussi includono dettagli come gli indirizzi IP, il numero dei pacchetti e dei byte, il type-of.service, etc.) che possono essere utilizzate

per le funzioni di accounting delle risorse. I Service Provider possono utilizzare tali misurazioni per le funzioni di billing, distinguendo per ore, utilizzo della banda, utilizzo delle applicazioni, quality of service, etc.

NetFlow ha due componenti chiave, la NetFlow cache o data source, che memorizza le informazioni sui flussi IP, e il meccanismo di export che invia i dati NetFlow ad un collector, per il reporting dei dati. In maniera riassunta il funzionamento di NetFlow può essere schematizzato come quello di una sonda, solitamente ospitata all'interno di network device, che analizza il traffico e crea all'interno di una cache, un record per ogni flusso. I record creati sono quindi esportati ad un collector per un'ulteriore fase di elaborazione.

2.3.2 Definizione ed export dei flussi

Poiché si tratta di un protocollo flow-based, anche NetFlow adotta una propria definizione di flusso al suo interno. In NetFlow un flusso è definito come una sequenza unidirezionale di pacchetti tra una data sorgente ed un data destinazione, definita da un indirizzo IP e da una porta sorgente e destinazione del transport layer. In particolare un flusso è definito come la composizione di sette campi chiave:

- Indirizzo IP sorgente
- Indirizzo IP destinazione
- Numero porta sorgente
- Numero porta destinazione
- Tipo di protocollo (Layer 3)
- ToS
- Interfaccia logica di input (ifIndex)

Questi sette campi identificano un unico flusso. Se un flusso ha un solo campo differente da un altro flusso, allora viene considerato come un nuovo flusso.

I flussi, una volta memorizzati all'interno delle cache vengono esportati ad un collector quando accade uno di questi eventi:

- Il flusso è terminato (per esempio è stato catturato un segmento TCP contenente un flag FIN o RST);
- Il flusso era inattivo da un certo lasso di tempo, cioé nessun pacchetto che gli appartiene è stato catturato in un intervallo di tempo (solitamente 15 secondi);
- Il flusso è ancora attivo, ma è scaduto il timeout, solitamente di 30 minuti. Tale timeout è utile soprattutto per esportare in modo regolare anche flussi di lunga durata.

2.3.3 Il formato di export di NetFlow

I flussi vengono esportati in un formato ben definito, chiamato NetFlow Export Datagram. Tale datagram consiste in un header e in una sequenza di flow record. L'header contiene informazioni come il sequence number, il numero dei record e il sysuptime. Il flow record contiene, invece, le informazioni riguardanti i flussi, per esempio indirizzi IP, porte etc. Un esempio di datagram è mostrato nella figura 2.2.

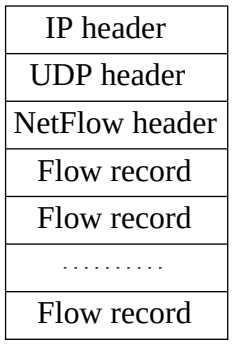

Figura 2.2: Formato del NetFlow Export Datagram

Esistono diverse versioni di questo formato di export, ognuna corrispondente ad una versione di NetFlow. La prima versione di NetFlow è la versione 1, oggi raramente utilizzata. Di seguito è stata rilasciata la versione 5, in cui al formato originale sono state aggiunte informazioni come il flow sequence number. Si è passati poi al rilascio della versione 7 e della versione 8: in quest'ultima versione sono presenti delle aggregazioni. L'ultima versione, la versione 9 [7], differisce moltissimo dei precedenti formati di export. Tale versione è stata adottata come formato di export dei dati all'interno dell'architettura definita in questa tesi.

2.3.4 NetFlow versione 9

L'ultima versione proposta dalla Cisco è la versione 9 [7], descritta anche in una rfc rilasciata dall'IETF [8]. La particolarità di questa versione è quella di essere basata sull'utilizzo dei *template*, in modo da poter fornire accesso alle informazioni sui flussi in maniera flessibile ed estendibile.

Un template definisce una collezione di campi, con la corrispondente struttura e semantica. Tale approccio ha i seguenti vantaggi:

- I nuovi campi possono essere aggiunti ai flow record senza cambiare la struttura di export dei record[2];
- I template che sono inviati ad un NetFlow Collector contengono le informazioni sulla struttura dei campi del flow record: in questo modo il NetFlow Collector può interpretarne i campi.
- La sua flessibilità permette di esportare solo quei campi che sono necessari, o che sono stati richiesti dal NetFlow Collector. Questo permette di ridurre il volume dei dati esportati.

Sulla base delle versione 9 di NetFlow, l'IPFIX Working Group dell'IETF [9] sta sviluppando un nuovo protocollo.

Layout del pacchetto di export

Il formato di export di NetFlow v9 consiste in un header seguito da almeno uno o più template o dati (data FlowSet). Un template fornisce una descrizione dei campi che saranno presenti nei successivi FlowSet. I dati possono seguire immediatamente dopo il template o in pacchetti successivi. In figura 2.7 viene mostrato un esempio di questo pacchetto di export. Differenti sono le combinazioni di template e dati che è possibile avere.

L'header del pacchetto è rimasto sostanzialmente inalterato rispetto alle versioni precedenti, ed è basato sull'header della versione 5. In figura 2.4 viene mostrato la struttura di tale header. La tabella 2.1 descrive il significato dei campi.

Il formato del template è mostrato in figura 2.5. La tabella 2.2 ne descrive il significato dei campi.

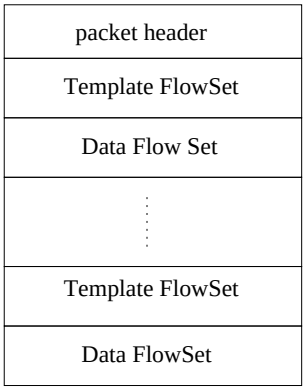

Figura 2.3: Struttura del pacchetto di export dei dati

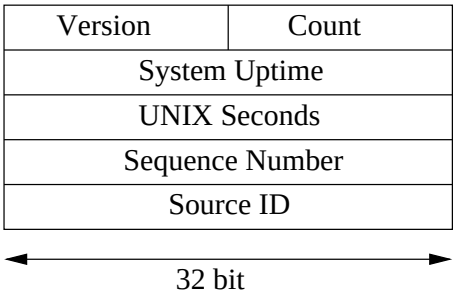

Figura 2.4: Struttura dell'header del pacchetto di export

Nome campo	**valore**
Version	Rappresenta la versione del record NetFlow esportato in questo pacchetto
Count	Numero dei record, sia template che dati, che sono contenuti all'interno del pacchetto
System Uptime	Tempo, in millisecondi, da quando il device ha subito il primo boot
UNIX Seconds	Tempo in secondi dal 1-1-1970
Sequence Number	Numero di sequenza incrementale di tutti i pacchetti che sono inviati da questo device
Source ID	Valore usato per garantire l'unicità per tutti i flussi esportati da un particolare device

Tabella 2.1: Descrizione header pacchetto di export per la versione 9

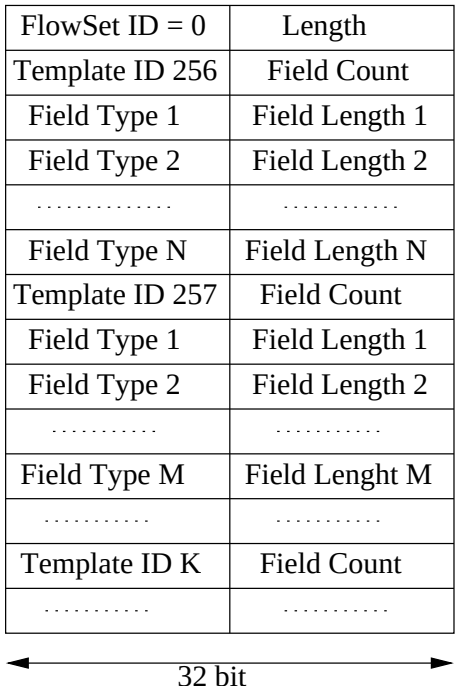

Figura 2.5: Struttura del template del pacchetto di export dei dati

Nome campo	**Valore**
FlowSet ID	E' usato per distinguere i template dai dati. Un template ha sempre un FlowSet ID che varia in un range da 0-255. Al momento per i template il valore di questo campo è 0
Length	Rappresenta la lunghezza totale di questo FlowSet. Poiché all'interno possiamo avere la definizione di più template, tale valore deve essere utilizzato per determinare la posizione del prossimo FlowSet. E' espressa come somma delle lunghezze del campo FlowSetID, del campo Length stesso e di tutti i campi del template.
Template ID	Valore che identifica univocamente il template. L'unicità è garantita rispetto all'entità che esporta il pacchetto
Field Count	Numero dei campi presenti nel template. Poiché all'interno è possibile avere più definizioni di template, il valore di questo campo permette di sapere quando un template è terminato
Field Type	E' un valore numerico che rappresenta il tipo di campo. I valori sono riportati in [7] e [8].
Field Length	E' la lunghezza, espressa in bytes, del tipo di campo corrispondente

Tabella 2.2: Descrizione struttura template

I tipi di campi che è possibile specificare all'interno del template non saranno descritti tutti. Per una lista completa si faccia riferimento a [8] e [7]. Nell'appendice A viene presentata una lista dei campi di NetFlow che l'architettura descritta è in grado di riconoscere.

Il formato della parte dati viene invece descritto in figura 2.6 (la tabella 2.3 descrive il significato dei campi che la compongono)

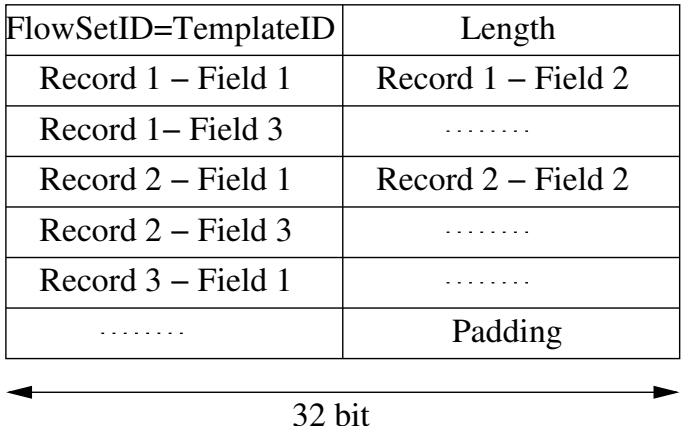

Figura 2.6: Struttura dei dati del pacchetto di export dei dati

Nome campo	**Valore**
FlowSet ID = Template ID	Indica il valore del template ID al quale i dati si riferiscono
Length	Indica la lunghezza di questo FlowSet. E' la somma delle lunghezze dei campi FlowSet ID, Length stesso e di tutti gli altri campi presenti
Record N - File M	Indicano i valori dei campi, il cui tipo e la cui lunghezza sono stati specificati nel template
Padding	Serve per un eventuale allineamento della fine del FlowSet al 32-esimo bit.

Tabella 2.3: Descrizione struttura dei dati

Nella figura 2.7 viene mostrato un esempio completo del formato di export della versione 9.

Ricezione del pacchetto di export

Di seguito verrà descritto il modo in cui avviene la ricezione dei pacchetti di export.

[2]Nelle precedenti versioni di NetFlow l'aggiunta di nuovi campi in un flusso comportava una nuova versione del formato di export e del Collector che doveva supportare il parsing del nuovo formato.

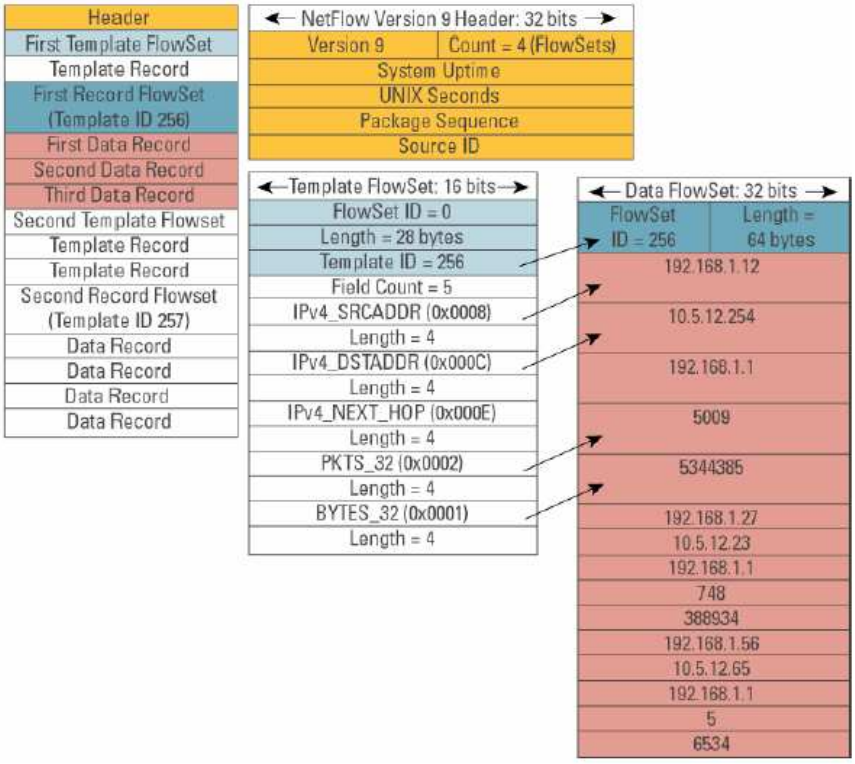

Figura 2.7: Struttura del pacchetto di export dei dati

Un'entità collector riceverà la definizione dei template da un'entità che fa da exporter, normalmente prima di ricevere la parte contenente i dati relativi ai flussi. Tali dati, indicati come *Flow Record*, possono essere quindi codificati e memorizzati localmente all'interno dei device. Nel caso in cui la definizione dei template non è stata ancora ricevuta al momento della ricezione dei dati, il collector metterà da parte i dati per una successiva codifica quando le definizioni dei template saranno finalmente ricevute. Questo significa che il collector non deve assumere che all'interno di uno stesso pacchetto siano presenti la sia il template che i dati ad esso associati.

I template vivono per un certo periodo di tempo. La durata di un template è ricavata dal collector basandosi sul tempo in cui l'ultimo template è stato ricevuto dall'exporter. Se il collector riceve una nuova definizione di template, questa immediatamente sovrascriverà la definizione esistente.

2.3.5 Osservazioni

Il vantaggio più grande offerto da NetFlow, rispetto ad altre soluzioni simili esistenti, è il fatto di essere supportato da molti produttori, tra i quali la stessa Cisco Systems,

che fornisce anche delle soluzioni per l'analisi e la visualizzazione [4, 3] dei risultati ottenuti tramite NetFlow stesso.

L'utilizzo dei template nella versione 9 ha diversi vantaggi:

- Quasi ogni tipo di informazione può essere esportata insieme alle informazioni sui flussi, incluse le informazioni di routing, IPv6, IPv4, informazioni sul multicast e sul MPLS. Queste nuove informazioni permettono nuove applicazioni dei dati relativi ai flussi e permettono di avere una nuova visione del comportamento della rete.

- Gli sviluppatori di applicazioni che permettono il collezionamento, l'analisi e la visualizzazione dei dati di NetFlow, non dovranno ricompilare totalmente le loro applicazioni ogni volta che viene aggiunto un nuovo campo al formato di export.

- E' possibile aggiungere nuove caratteristiche, in modo più veloce, senza compromettere le implementazioni correnti.

- Il formato di export della versione 9 può esser adattato per fornire supporto nel caso di viluppo di nuovi protocolli.

I campi che possono essere aggiunti a quelli che solitamente identificano un flusso possono essere utilizzati per diversi tipi di analisi. Per esempio, utilizzare i numeri delle porte presenti nei flussi esportati, permette di produrre una classificazione del traffico in base al tipo di applicazione; utilizzare invece gli indirizzi IP, permette di ottenere una classificazione del traffico in base alla sua sorgente e alla sua destinazione [26]. Combinando i dati raccolti da più router permette di avere una visione del traffico rispetto a tutta la rete, cosa che solitamente viene richiesta dagli ISP.

Inoltre, poiché su reti ad alta velocità, il costo a livello di risorse computazionali e di memoria per poter mantenere la cache contenente i flussi può essere alto, la Cisco ha introdotto un *sampled NetFlow* [25], in cui l'aggiornamento della cache viene fatto solo per i pacchetti che sono stati campionati. Per un certo valore N configurabile, solo uno ogni N pacchetti sarà raccolto e utilizzato per aggiornare la cache.

Tuttavia NetFlow presenta anche degli aspetti negativi, legati soprattutto al grande numero di eventi possibili che provocano l'export di un flusso e la durata di

vita variabile dei record che identificano i flussi. Questo comporta un aumento della complessità della fase di elaborazione dei dati.

Molti tool di analisi dividono il traffico in intervalli di tempo. Sfortunatamente i record di NetFlow possono andare oltre questi intervalli di tempo, causando una non necessaria complessità e inaccuratezza nell'analisi del traffico. Se il timestamp che indica l'inizio e la fine di un flusso ricadono all'interno di uno stesso intervallo, l'analisi risulta semplice. Ma, d'altro canto, se un flusso inizia in un intervallo e finisce in un altro intervallo, bisogna stimare quanto traffico appartiene ad un intervallo e quanto appartiene ad un altro. Questa complica la fase di elaborazione e introduce inesattezze.

Inoltre quando viene utilizzato il sampled NetFlow, vengono introdotte altre inesattezze nel conteggio di flussi non TCP. Infatti, mentre il conteggio di flussi TCP viene facilitato dalla presenza dei flag TCP[3], lo stesso non avviene per i flussi UDP e ICMP. E questi protocolli sono ugualmente importanti in quanto utilizzati per scanning e altri tipi di attacchi come i worm.

Per far fronte e questi problemi, in [1] vine proposta una soluzione chiamata Adaptive NetFlow.

Nonostante questi problemi, NetFlow rimane la soluzione per il monitoraggio e la misurazione del traffico più utilizzata.

2.3.6 Realtime Traffic Flow Measurement

Un'altra tecnologia [24] basata sul sampling dei flussi è stata proposta nel passato dal Realtime Traffic Flow Measurement (RTFM) Working Group dell'IETF [10]. Rispetto a NetFlow, questa tecnologia risulta più avanzata, in quanto la fase di elaborazione a livello delle sonde può essere parzialmente personalizzata. Il cuore dell'architettura proposta in [24] sono delle entità chiamate Meters. Il compito di queste entità è quello di osservare i pacchetti che passano in un punto e di classificarli in gruppi. Per ogni gruppo vengono accumulati alcuni attributi, per esempio il numero dei pacchetti. Un aspetto importante di queste entità è che forniscono un modo per poter ottenere un prima aggregazione del traffico. Come parte della sua configurazione, un meter ha un insieme di 'regole' che specificano quali sono i flussi di un certo interesse, in termini di valori dei loro attributi. Tali regole, che

[3] Il primo pacchetto di ogni flusso ha il flag SYN settato.

permettono anche di personalizzare la definizione di flusso e le azioni che possono essere intraprese per ognuno di essi, sono descritte in un particolare linguaggio, il Simple Ruleset Language, descritto in [22]. All'interno dell'architettura di RTFM, inoltre, i flussi sono bidirezionali, permettendo quindi di mantenere facilmente sotto controllo le due direzioni di una connessione. L'interazione tra il probe e il flow collector avviene tramite delle queries SNMP. L'unica implementazione public-domain disponibile è NeTraMet [23].

2.3.7 Tecniche di Data Mining

Presso il Politecnico di Torino, il NetGroup ha presentato un nuovo approccio alla cattura e all'analisi del traffico per reti ad alta velocità. L'approccio fornito è un approccio flow-based: un probe, che colleziona i dati, salva un dato insieme di informazioni relative ad ogni flusso. La definizione di flusso adottata è molto generale, e diversi sono i campi che possono essere inclusi nell'insieme che meglio caratterizza il flusso[4].

Allo scopo di sopportare un numero variabile di campi all'interno di ogni flusso, i dati sono organizzati in tre tabelle:

- Una tabella mantiene le informazioni che non cambiano, relative ad ogni flusso;
- Una tabella che per ogni flusso, contiene la lista dei campi che devono essere memorizzati;
- Una tabella che elenca tutti i campi validi.

Le tabelle sono inserite all'interno di un database di tipi SQL-Lite. Allo scopo di ottimizzare il tempo di inserzione, i dati non sono inseriti direttamente all'interno del database, ma vengono prima memorizzati all'interno di flat file.

Il modulo NetLogger include un'interfaccia grafica che permette di estrarre le statistiche attraverso delle specifiche query SQL. Il modulo è fornito di una lista di query predefinite.

Il sistema non fornisce tuttavia statistiche real-time, poiché i dati che corrispondono all'intervallo attuale non sono ancora nel database.

L'estrazione dei dati e delle statistiche avviene tramite l'applicazione delle tecniche di data mining per l'estrazione di Frequent Itemset e Association Rule.

[4] I campi che è possibile inserire sono completamente personalizzabili.

Tuttavia l'approccio dell'utilizzo delle tecniche di data mining deve ancora essere ben sperimentato. Inoltre tale approccio pone il problema del'interpretazione della grande quantità di dati che possono essere estratte tramite il data mining: gli amministratori di rete non sono degli esperti di data mining, e tale interpretazione non è quindi facile ed ovvia.

2.4 Altri approcci

Esistono naturalmente altri approcci per il monitoraggio di rete. Tra questi ricordiamo RMON [31], uno standard proposto dall'IETF. Nel caso di RMON, un agent, solitamente inserito all'interno di router o di switch, implementa l'RMON MIB. In questo modo un amministratore di rete di determinare il livelli di traffico nel network segment, il carico del traffico totale da/per host, suddiviso per tipo di protocollo, etc. RMON, comunque, non fornisce nessuna misurazione del traffico inteso come flusso.

Un altro approccio è quello che viene offerto da `ntop` [15]. Questa applicazione open source ha la capacità di monitorare e gestire una rete da una postazione remota senza il bisogno si eseguire specifiche applicazioni cliente per analizzare il traffico. Permette di catturare i pacchetti dalla rete in maniera indipendente dal sistema operativo e dall'interfaccia usata per catturare i pacchetti stessi. Fornisce un'analisi di base del traffico e offre la possibilità di caratterizzare il traffico in base al tipo di protocollo. I dati ricevuti possono essere visualizzati tramite un server HTTP incluso, senza utilizzare un'applicazione ad hoc.

Infine ricordiamo la tecnologia di sFlow [27] che risulta essere un mix tra la tecnica del packet-sampling e l'estrazione dei flussi. Il sistema di sFlow si consiste in un sFlow Agent, inserto all'interno di router o switch, o all'interno di singole probe, e di un collector centrale, sFlow Analyzer. L'sFlow Agent usa le tecniche di sampling per catturare il traffico dai device che sta monitorando. Tali dati sono inviati al sFlow Analyzer in un formato chiamato sFlow datagram, in modo da poter analizzare i dati raccolti.Le tecniche di packet sampling sono state utilizzate per poter raggiungere la scalabilità. sia i pacchetti così raccolti, che le informazioni su i relativi flussi possono essere esportati.

L'approccio utilizzato rende sFlow adatto per un grande insieme di ambien-

ti poiché permette sia l'analisi dei pacchetti (sebbene limitata alle prime poche centinaia di bytes) che quella dei flussi.

La più grande limitazione risulta però essere lo scarso supporto da parte dei produttori.

2.5 Conclusioni

Dalle tecnologia citate in precedenza si nota una tendenza ad utilizzare per il monitoraggio e la misurazione del traffico di rete le tecniche flow-based. Tali misurazioni sono risultate la maggiore sorgente di informazioni sul traffico della rete.

Tuttavia, le tecniche presentate, non permettono di personalizzare l'insieme dei campi che è possibile l'insieme dei campi memorizzati per ogni flusso.

NetFlow, e con esso lo standard di IPFIX, utilizzando la tecnica dei template, permettono un buon grado di personalizzazione tra i campi che è possibile memorizzare.

Inoltre, rispetto ad altre tecnologie presenti, permettono di ottenere un'aggregazione dei dati, permettendo una riduzione del volume dei dati[5].

Proprio per questi motivi si è deciso, per il sistema sviluppato, di utilizzare i dati che provengono da misurazioni flow-based e di utilizzare un formato dei dati facilmente estendibile e flessibile come quello dei template (all'interno del sistema si farà uso del formato sviluppato per NetFlow v9)

[5]Su reti a alte velocità come quelle odierne, la quantità di dati da gestire può essere molto alta.

Capitolo 3

Specifiche architettura

Sommario

In questo capitolo verrà presentata una descrizione generale dell'architettura del sistema di monitoraggio che si intende implementare, analizzandone i requisiti, i vincoli e le caratteristiche principali. La validazione del sistema verrà discussa nel capitolo successivo.

3.1 Descrizione generale

L'architettura che verrà discussa in questo capitolo è un'architettura distribuita che effettua il monitoraggio di una rete analizzando il traffico che viene generato da quest'ultima. E' costituita da una gerarchia di livelli: nel livello più basso vengono raccolte le informazioni dalla rete, mentre nei livelli superiori quest'ultime vengono analizzate e integrate in modo da poter avere una visione completa di quello che succede in una sottoarea.

Lo schema dell'architettura viene mostrato nella figura 3.1

La figura mostra una gerarchia ad albero, anche se non è necessario che sia tale. Ogni livello ha al suo interno un insieme di blocchi: a sua volta, ogni blocco, al suo interno, svolge un insieme predefinito e completo di funzioni, in modo che il blocco possa essere considerato una piccola scatola chiusa. I blocchi sono architetturalmente identici tra di loro. Facendo questa scelta ogni livello è ottenuto replicando un blocco più volte, seguendo le caratteristiche della rete da monitorare. Questo approccio permette anche di ottenere un buon livello di ***replicabilità***: se si vuole espandere un livello basta semplicemente aggiungere un nuovo blocco.

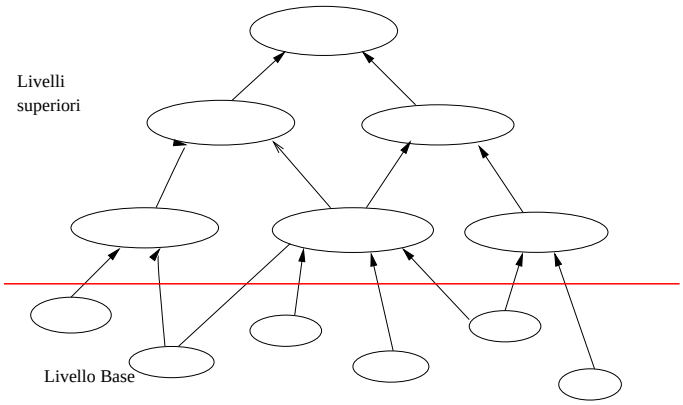

Figura 3.1: Schema architettura

3.2 Requisiti

In questa sezione verranno descritti e analizzati singolarmente i requisiti che l'architettura del sistema che si intende implementare deve possedere.

Replicabilità

Il sistema deve adattarsi facilmente alla rete da monitorare. Quindi se occorre monitorare nuovi elementi bisogna essere in grado di espandere il sistema, senza però andare a modificare la struttura di quello che già esiste. Un modo per ottenere questo è costruire il sistema, usando dei blocchi elementari che forniscono un insieme di funzionalità completo. Espandere o creare il sistema corrisponderà a replicare più volte uno stesso blocco, seguendo le esigenze del momento. La replicabiltà deve essere garantita sia per i blocchi che appartengono al livello base, che per i blocchi che appartengono ai livelli superiori.

Scalabilità

Un altro requisito importante è la ***scalabilità***, passando da un sistema più piccolo ad un sistema con più elementi, cioé aumentando il numero dei blocchi o il numero dei livelli, le prestazioni dell'architettura devono continuare a mantenersi costanti, o devono continuare ad essere paragonabili a quelle precedenti.

Fault tolerance

Trattandosi di una architettura distribuita a più livelli, una delle situazioni in cui ci si potrebbe trovare è quello di un malfunzionamento di uno o più blocchi di uno o più livelli. In tali situazioni il sistema deve avere un comportamento ***fault tolerant***, deve cioé essere in grado di continuare normalmente il proprio lavoro, ribilanciando il sistema e modificandone lo schema di invio delle informazioni. Dove si è verificato il problema, l'utente vedrà una sorta di macchia nera, ma il sistema continuerà a svolgere il proprio lavoro.

Affidabilità

Il sistema sfrutta le tecnologie del mondo di Internet per poter effettuare il monitoraggio di una rete. Può quindi succedere che la rete usata per le comunicazioni si saturi, portando ad una congestione. Ma può anche succedere che uno o più componenti di un livello si trovino in una situazione di overload. In questo caso il sistema deve essere in grado di reagire, cercando si alleviare tale situazione. Occorre quindi garantire che l'architettura abbiamo un buon livello di ***affidabiltà***. Le politiche che saranno adottare per poter garantire questo requisito saranno discusse nel capitolo dedicato alla validazione del sistema.

Sicurezza

Poiché le informazioni sono spostate tra i vari livelli tramite lo scambio di messaggi, occorre garantire un certo livello di ***sicurezza***: si tratta in questo caso di poter riconoscere in modo sicuro i partner della comunicazione, per esempio tramite l'utilizzo di un meccanismo basato sui certificati, ma anche di nascondere mediante l'anonimato le informazioni che possono coinvolgere la privacy dell'utente (dietro ogni indirizzo IP vi è un utente la cui privacy va tutelata).

Aggregazione informazioni

La quantità di informazioni che vengono raccolte dalla rete è molto grande, soprattutto su reti ad alta velocità. Solitamente in reti di una certa grandezza, come le reti universitarie, si va da poche unità a decine di migliaia di flussi al secondo: è improponibile quindi spostare una tale quantità di dati che andrebbe a saturare in poco tempo i sistemi di memorizzazione (dischi e DB) del sistema e ad occupare

molta della banda di comunicazione. Le informazioni devono quindi essere inviate in un formato compresso, che permetta di non perdere molte informazioni e che permetta una buona integrazione di dati compressi con caratteristiche differenti. Si rimanda alle sezioni 3.8 e 4.1.5 per ulteriori precisazioni sulle aggregazioni.

Altri requisiti

Per poter monitorare i servizi offerti su TCP/IP, occorre che il sistema non sia un elemento di disturbo, occorre cioé che non faccia variare l'effettiva misurazione del servizio, e creare una diminuzione dei performance dei sistemi controllati: per poter ottenere questo il sistema deve catturare il traffico prodotto dal servizio, analizzandone i dati per poter generare metriche necessarie all'analisi dei servizi erogati.

Ci si può chiedere, a questo punto, cosa esattamente questo sistema deve essere in grado di fare. Il sistema deve essere in grado di poter effettuare azioni come l'identificazione di servizi basati su IP, e deve essere in grado di assegnare nomi ai servizi IP basandosi sulle porte o su indirizzi di server utilizzati.

Per quanto riguarda i dati che vengono elaborati dal sistema, il sistema deve poter creare raggruppamenti basati su classi di indirizzi IP o liste di indirizzi IP, per server , servizi e URLs in report che rappresentano il servizio erogato o la piattaforma da cui che viene erogato il servizio. Le metriche che verranno calcolate devono essere aggregate ogni 5 minuti; si deve mantenere il dettaglio per almeno 15 giorni; i dati devono essere poi aggregati per giorno ed essere così mantenuti per 30 giorni, quindi devono essere aggregati per settimana/ mese e così mantenuti per almeno 12 mesi[1].

Infine la reportistica deve essere modificata in base all'account con cui si accedere al sistema, permettendo in questo modo di tutelare l'accesso al sistema.

3.3 Livello Base

Il primo livello dell'architettura è responsabile della raccolta delle informazioni dalla rete e del loro invio ai livelli superiori

[1] Queste azioni corrispondono a mantenere i dati all'interno del sistema solo in forma compressa, generando aggregazioni.

3.3.1 Architettura livello base

Il livello base dell'architettura è costituito da un insieme di sonde che raccolgono le informazioni dalla rete e le inviano ad un'entità che fa da collector. Le sonde sono posizionate quindi sui luoghi fisici dove transita il traffico: può trattarsi quindi anche di altri apparati di rete opportunamente configurati per effettuare la raccolta delle informazioni.

L'entità che fa da *collector* provvede quindi a raccogliere i dati esportati da una o più sonde. Il suo compito è di filtrare e aggregare i dati in modo da riorganizzarli e inviarli ai livelli superiori. L'invio può essere fatto ad uno o più di blocchi, secondo le esigenze riscontrate. La figura seguente mostra uno schema di tale livello.

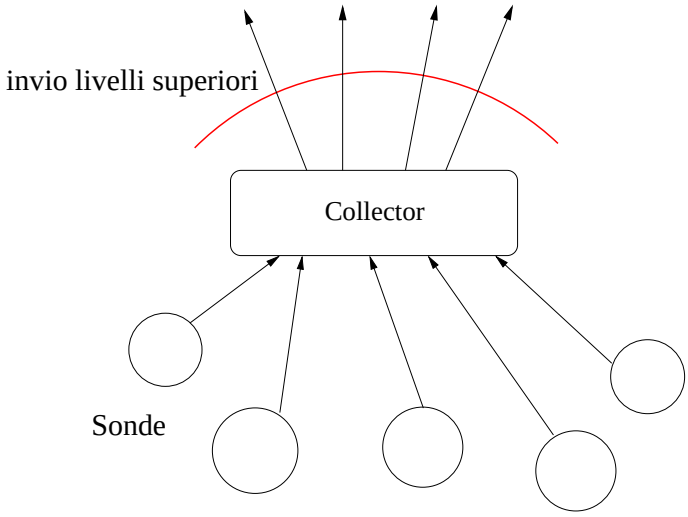

Figura 3.2: Schema livello base

3.4 Livelli Superiori

Come già detto prima, i livelli superiori al primo, sono costituiti da uno stesso blocco replicato più volte. E' quindi interessante andare a vedere come è stato disegnato tale blocco.

3.4.1 Architettura livelli superiori

I blocchi del livello superiore al primo sono il cuore vero e proprio del sistema. I dati, filtrati una prima volta dai collector del livello base, sono raccolti ai livelli successivi,

analizzati, filtrati, aggregati ed infine inviati ai livelli superiori.

Vediamo ora da vicino come sono costruiti questi blocchi. All'interno di ogni blocco sono presenti delle entità che collaborando tra di loro forniscono un insieme completo di funzionalità. Nei capitoli successivi verranno descritte le scelte che sono state adottate per poter costruire queste entità. All'interno di questo capitolo ci riferiremo in termini generali, senza soffermarci sulle varie possibiltà implementative, in modo da dare una descrizione generale delle caratteristiche che tali entità devono possedere e delle interazioni e comunicazioni che avvengono tra di loro. La figura seguente ne mostra uno schema.

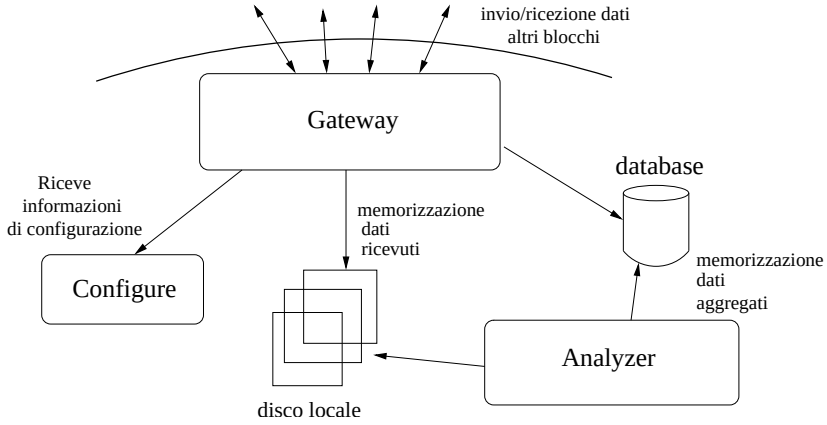

Figura 3.3: Schema livelli superiori

3.5 Gateway

Gateway è l'entità che si occupa di fornire un'interfaccia al blocco verso il mondo esterno. Tutte le comunicazioni da/per il mondo esterno che coinvolgono il blocco sono gestite in questo punto dell'architettura. I compiti che deve svolgere sono i seguenti

- Ricezione delle nuove informazioni di configurazione per le altre entità interne al blocco;

- Ricezione dei dati dal mondo esterno da altri blocchi;

- Memorizzazione in locale dei dati su disco;

- Invio dei dati elaborati all'interno del blocco agli altri blocchi del livello superiore.

3.6 Configure

Configure rappresenta l'entità, all'interno dell'architettura, che ha il compito di mantenere lo stato del blocco, mantenendo aggiornata la configurazione delle singole entità. Le nuove configurazioni sono ricevute dall'entità Gateway, e sono aggiornate da Configure. Facciamo un esempio: se Configure riceve l'aggiornamento della configurazione dell'entità Analyzer, provvederà ad arrestarne l'esecuzione e a riattivarla con i nuovi parametri di configurazione.

Le nuove configurazioni vengono propagate dai livelli più alti dell'architettura verso i livelli più bassi. Questo è dato dal fatto che per come è costruita l'architettura, in cui si suppone che ai livelli più alti siano presenti gli amministratori della rete, e quindi si decidono quali sono le politiche da adottare per gli schemi di aggregazione, per i filtri e per tutte le altre informazioni di configurazione che è possibile specificare per le varie entità che appartengono ai livelli inferiori.

3.7 Analyzer

L'entità Analyzer ha il compito di effettuare le aggregazioni ha il compito di aggregare i dati che gli sono inviati, in un formato definito e di memorizzarli all'interno di un Database. La politiche da usare durante la fase di aggregazione (e di filtraggio) sono contenute nei file di configurazione e sono aggiornabili. All'interno di un file di questo tipo troviamo la definizione di :

- Schemi di aggregazione, che definiscono le regole in base alle quali riassumere le informazioni sul traffico raccolto.
- Filtri, che permettono di discriminare tra i dati ricevuti riducendo il volume dei dati, in modo, per esempio, da applicare uno schema di aggregazione sui dati per cui è stato specificato il valore di alcuni dei parametri presenti nello schema.

Gli schemi di aggregazione e i filtri saranno descritti in dettaglio nel capitolo successivo, dedicato alla validazione.

L'Analyzer legge dal disco locale i dati inviati dall'esterno e vi applica gli schemi di aggregazione e i vari filtri. In una prima fase, si ha una riduzione del volume dei dati dovuta all'applicazione dei filtri specificati. I dati così ottenuti subiscono una ulteriore riduzione dovuta in questo caso all'applicazione degli schemi di aggregazione.

I dati, dopo aver subito questa compressione, vengono memorizzati all'interno delle tabelle di un database. Ogni blocco, che contiene un'entità di questi tipo può effettuare le aggregazioni con schemi differenti, ottenendo in questo modo un insieme non omogeneo di informazioni.

3.8 Schemi di aggregazione

Gli schemi di aggregazione giocano un ruolo importante all'interno dell'intero sistema, in quanto permettono di poter mantenere in forma compressa i dati che sono inviati all'interno del sistema.

Uno schema di aggregazione, nel sistema che si sta definendo, corrisponde ad un insieme di proprietà comuni che i dati devono possedere, permettendo in questo modo di integrare tra di loro dati con proprietà simili come uno stesso indirizzo IP sorgente o una stessa sottorete di destinazione. Le proprietà di cui si parla non sono quindi altro che informazioni ritenute rilevanti, estratte dai dati ricevuti. Un esempio banale è quello di aggregare i dati in base all'indirizzo IP sorgente e al tipo di protocollo. Naturalmente questo schema è molto banale, in realtà possono essere più complessi di quello descritto sopra.

Solitamente gli schemi vengono propagati dall'alto verso il basso: le informazioni vengono inviate per sottoaree, cioè a quei blocchi che in un livello individuano una sottorete. Da queste sottoaree verranno infine propagate le informazioni in modo dettagliato ai singoli blocchi. La aggregazioni devono essere personalizzabili, chi utilizza il sistema deve cioé essere in grado di poter specificare le proprietà che vuole considerare per l'aggregazione, in modo da poterle adattare alle esigenze del momento e ai diversi scenari in cui il sistema può essere utilizzato. La possibilità di modificare gli schemi fa sì che questi non siano unici: ogni Analyzer può utilizzare uno schema differente. La conseguenza di tutto questo è che quando le informazioni vengono raccolte dai blocchi risulta tra di loro differenti e debbano essere opportunamente integrate. Inoltre, un blocco può inviare i dati a più blocchi al livello

superiore al suo. Questa fa sì che si possano avere anche delle informazioni in parte duplicate.

Ogni blocco è in grado di analizzare solo un sottoinsieme di proprietà, esattamente quelle che sono state specificate all'interno degli schemi. Alcuni esempi di tali proprietà sono:

- Indirizzo IP sorgente
- Indirizzo IP destinazione
- Tipo di protocollo (TCP,UDP, ICMP..)
- Porta sorgente
- Porta di destinazione
- Numero dei pacchetti
- Numero dei byte....

3.9 Memorizzazione dei dati sul database

I dati che sono analizzati e aggregati all'interno di un blocco devono essere integrati e memorizzati al'interno di un unico database all'interno del blocco stesso. La memorizzazione deve avvenire solo dopo che i dati sono stati aggregati, e quindi quando si presentano in un forma compressa. Questo tipo di precauzione nella memorizzazione deve essere preso, in quanto, in fase di analisi del problema, si è giunti alla conclusione che memorizzare tutti i dati inviati all'interno del database, occuperebbe tutto lo spazio disponibile. Se pensiamo infatti che mediamente, in una rete come quelle universitaria, i flussi che vengono generati vanno da poche unità a decine di migliaia al secondo, la quantità dei dati che si dovrebbe memorizzare è enorme, e ciò porterebbe ad un utilizzo massiccio (con conseguente degrado delle prestazioni) del database.

3.10 Vincoli

In questa sezione verranno presentati i vincoli che sono stati imposti all'architettura descritta. Il sistema fa uso di schemi di aggregazione, per poter mantenere i dati in

forma compressa: questo fa sì che il livello di dettaglio con cui si possono analizzare le statistiche generate sia in un qualche modo limitato. Inoltre, ogni blocco può effettuare solo un sottoinsieme di aggregazioni, quelle che sono specificate all'interno dei file di configurazione: questo fa sì che le informazioni che possono essere estratte dai dati sono limitate dagli schemi. Una successiva applicazione di altri schemi di aggregazione potrebbe quindi non permettere di ottenere le informazioni desiderate.

Inoltre il sistema non è in grado di affrontare il problema della sicurezza in termini di scoperta di attacchi alla rete: in questo caso sarà compito dell'amministratore riuscire ad identificare nelle statistiche pattern anomali che permettano di comprendere se è presente o meno un'anomalia.

3.11 Scenari di applicabilità

In questa sezione verranno presentati alcuni scenari in cui un sistema, come quello proposto può essere utilizzato, alla luce dei vincoli che per necessità sono stati imposti al sistema. Il sistema può essere utilizzato nei casi in cui sia necessario avere una panoramica di metriche per la caratterizzazione del traffico di rete. Il sistema può essere cioé utilizzato per effettuare misurazioni del volume del traffico di rete di un certo numero di organizzazioni connesse alla rete. Le informazioni ottenute del traffico catturato possono essere divise in sottocategorie, permettendo quindi di poter usare il sistema per effettuare l'accounting e il billing di una rete. Ma il sistema permette anche di effettuare il monitoraggio della rete e delle applicazioni, per permettere di comprendere la performance di applicazioni che usano la rete. Le informazioni ottenute in questi modo possono essere utilizzate per una migliore allocazione delle risorse e della applicazioni di rete, come per esempio l'allocazione di un server Web.

Un altro possibile scenario di applicazione del sistema è per l'ingegnerizzazione della rete e quindi per una sua futura pianificazione e estensione: il sistema può infatti fornire informazioni sull'andamento a lungo tempo del traffico, in modo da permettere all'amministratore di comprendere e anticipare la crescita e l'aggiornamento della rete.

In generale, quindi, il sistema può essere utilizzato all'interno di quelle organizzazioni, come le banche o le reti universitarie, che desiderano monitorare l'uso dei

propri link sia in termini di connettività che in termini di accounting del volume di traffico generato.

Capitolo 4

Validazione

Sommario

In questo capitolo verranno descritte in dettaglio tutte le scelte adottate per soddisfare i requisiti e le caratteristiche dell'architettura, descritti all'interno del capitolo 3.

4.1 I livelli dell'architettura

Nelle specifiche dell'architettura, le funzionalità di un blocco sono svolte tramite le interazioni di entità generiche. Di tali entità sono state date le caratteristiche generali, i requisiti che devono possedere. Nella fase di validazione verranno specificate dettagliatamente come saranno soddisfatte le caratteristiche e i requisiti.

4.1.1 Livello base

Nel livello base è presente una sola entità, *Collector* che ha il compito di raccogliere informazioni sul traffico della rete che sono inviate dalle sonde e dagli apparati di rete opportunamente configurati. Tale processo ha anche il compito di effettuare una prima elaborazione dei dati filtrando i dati in base a delle regole che gli vengono inviati dai livelli superiori. Le convenzioni usate per specificare i filtri e il modo in cui tali filtri sono propagati verranno descritti in seguito all'interno di questo capitolo.

4.1.2 I livelli superiori

Nei blocchi dei livelli superiori a quello base, sono presenti tre entità che collaborano tra di loro per fornire le funzionalità. Avremo quindi le seguenti entità:

- Gateway
- Configure
- Analyzer

4.1.3 Gateway

Da quanto detto nelle specifiche dell'architettura, il compito di tale entità è quello di interfaccia, verso il mondo esterno, di ciascuno dei blocchi dei livelli superiori. Avrà quindi il compito di ricevere i dati dell'esterno e le configurazioni degli altri demoni. I dati sul traffico della rete sono inviati nel formato dei dati di NetFlow v.9 [2]. La scelta di utilizzare tale formato per l'invio dei dati è stata motivata dal fatto che essendo un formato basato sui template, permette di personalizzare l'invio delle informazioni secondo schemi definibili dall'utente del sistema: esiste infatti al possibilità di definire dei template personalizzati e di specificare campi, all'interno degli stessi template interamente definiti dall'utente. Inoltre si occupa anche di memorizzare in locale i dati che vengono ricevuti, effettuando quindi una sorta di bufferizzazione.

Memorizzazione locale dei dati

I dati che sono scambiati tra i vari livelli non vengono elaborati subito, ma appena sono ricevuti vengono memorizzati in locale su disco dall'entità Gateway. Tale memorizzazione è dovuta al fatto che dopo un'analisi della grandezza dei dati che vengono inviati, si è giunti alla conclusione che un immediata elaborazione dei dati, applicando cioè gli schemi di aggregazioni definiti, comporta un ritardo che potrebbe portare alla perdita di parte delle informazioni inviate. Memorizzando invece in locale mi permette di salvare le informazioni e di posticipare l'elaborazione ai soli dati memorizzati.

Gateway provvede a memorizzare i dati, in locale, generando due insiemi di file distinti: un insieme conterrà i template, mentre l'altro conterrà i dati. I template sono memorizzati nei file con estensione *.tmpl*, in modo da poter ricordare i campi che li compongono. I dati relativi al template sono memorizzati, invece, in un insieme di file la cui estensione è *.flw*.

La memorizzazione, sia dei dati che dei template, è in un certo qual modo temporizzata. I template, infatti, vivono solamente per un certo periodo di tempo, dopo

il quale possono essere eliminati. Tale periodo può essere dedotto all'interno del blocco, basandosi sul tempo di arrivo dell'ultimo flusso relativo al template. Se il periodo di tempo supera un valore di soglia, il template è scartato perché considerato troppo vecchio. Se, dopo la cancellazione del template, altri flussi, che lo riguardano sono ricevuti, non saranno considerati e saranno innvece eliminati.

I nuovi template andranno in questo modo a sovrascrivere quelli vecchi.

I file contenenti le informazioni sui flussi saranno suddivisi per tempo: ogni file conterrà, infatti, le informazioni riguardanti un certo intervallo di tempo, in questo modo all'interno di un file è possibile memorizzare i dati ad intervalli di tempo, per esempio, di 5 minuti. Ogni 5 minuti, o secondo un differente intervallo di tempo specificato dall'utente del sistema, viene generato un nuovo file per memorizzare i dati. Il nome dei file creati dovranno rispecchiare la sequenza temporale di ricezione: per poter fare cioò il sistema genera i il nome dei file appendendo al nome standard *file_ data* il System Uptime. Quando tali file saranno troppo vecchi, e cioé dopo che vi saranno applicati tutti gli schemi di aggregazione previsti all'interno del blocco, verranno cancellati, in modo da rendere disponibile nuovo spazio per nuove memorizzazioni.

Una volta che sarà avvenuta la fase di memorizzazione, l'Analyzer potrà iniziare la prima fase di aggregazione dei dati.

Rimane ancora da spiegare il perché della memorizzazione temporizzata. Una volta che i dati sono stati filtrati, e soprattutto, una volta che sono stati aggregati, vengono memorizzati come entry all'interno di un database. I dati presenti all'interno di un file, dopo la fase di aggregazione, perdono il loro valore iniziale (non sono più significativi). Per evitare di riempire il disco locale con informazioni inutili, i file dopo un intervallo di tempo t vengono eliminati, permettendo i questo modo di avere dello spazio libero disponibile.

I file, inoltre, rispecchiano in un certo qual modo la sequenza temporale di ricezione: ogni file contiene i dati, cosí come gli sono arrivati, di un certo intervallo di tempo. Si genera in questo modo una successione, dove ogni elemento ha nel proprio nome il tempo di inizio dell'intervallo (tempo espresso come secondi dal momento di avvio del sistema). In questo modo è facile effettuare sia le aggregazioni, sia la cancellazione dei file vecchi.

4.1.4 Configure

L'entità Configure si occupa di mantenere aggiornata la configurazione degli altri demoni presenti all'interno di un blocco. Le nuove configurazioni vengono propagate dall'alto verso il basso (vedi sez. 3.6) e sono ricevute dall'entità Gateway che provvede a segnalarne la ricezione a Configure. In questa fase si è deciso di considerare come informazioni di configurazione anche l'invio dei nuovi schemi di aggregazione. Le informazioni di configurazione sono scambiate, esattamente come per i file delle aggregazioni, in formato *xml*. Un esempio di file di configurazione è mostrato nella figura 4.1. Come si può notare nell'esempio fornito, la struttura di

```
<configurazione_analyzer>
        <file_aggr>...</file_aggr>
        <file_tmpl>...</file_tmpl>
        <file_dati>...</file_data>
        <soglia_canc_tmpl>...</soglia_canc_tmpl>
        <timer_aggr_avanzate>...</timer_aggr_avanzate>
        <timer_compressione>...</timer_compressione>
        <limitazione_invio>...</limitazione_invio>
        <partner>
            <blocco>...</blocco>
            <blocco>...</blocco>
            ....
            <blocco></blocco>
        </partner>
</configurazione_analyzer>
```

Figura 4.1: Esempio file di configurazione

questo file risulta essere molto semplice. I tag rappresentano i parametri, mentre i valori sono i nuovi valori dei parametri. Tra i parametri che si può notare come compaia anche il nome del file contenete gli schemi di aggregazione. Il significato dei singoli tag sarà chiarito di seguito, nelle sezioni successive, dove verranno discussi i vari parametri che è possibile specificare nelle configurazioni. All'interno di ciascun blocco, i valori di tali parametri sono impostati ai loro valori di default: in questo modo all'interno del file è possibile specificare solo valori per i parametri che si intende modificare. Il parsing di questo file durante la fase di inizializzazione dell'entità Analyzer, permette di inizializzare correttamente tutti i parametri.

Le convenzioni adottate per specificare le aggregazioni verranno discusse in segui-

to in questo capitolo. L'aggiornamento della configurazione avviene come descritto nella sezione 3.6

4.1.5 Analyzer

L'entità Analyzer si occupa di elaborare i dati, memorizzati in locale, filtrandoli e aggregandoli, in base agli schemi di cui dispone. I filtri e le aggregazioni sono applicati in cascata. Entrambi sono memorizzati in file di estensione *xml*. Una volta che i dati sono stati filtrati e aggregati, lo stesso Analyzer provvederà a memorizzarli all'interno di tabelle nel database che si trova in ogni blocco. Di seguito vengono descritte le scelte e le convenzioni adottate per gli schemi di aggregazione (e i filtri) e per la memorizzazione nel database.

Filtri

I filtri sono delle clausole logiche, formate da più predicati, che i dati raccolti devono possedere. I predicati sono legati tra di loro in *and*, in modo da poter selezionare solo i dati che soddisfano la clausola, ed effettuare una scrematura. Come per le aggregazioni, anche i filtri sono descritti in file in formato *xml*. Un esempio di filtro è il seguente:

```
<?xml version="1.0" encoding="utf-8"?>
<filter>
    <discard>
    <IPv4_SRCADDR>192.168.1.12</IPv4_SRCADDR>
    <IPv4_DSTADDR>10.2.12.254</IPv4_DSTADDR>
    <\discard>
<\filter>
```

Il filtro sopra descritto permette di scartare tutti i dati che hanno come indirizzo IP sorgente e indirizzo IP destinazione rispettivamente 192.168.1.12 e 10.2.12.254 [1]. Naturalmente, proprio come per le aggregazioni, è possibile filtrare i dati in maniera differente all'interno di uno stesso blocco.

Una volta che i dati sono stati filtrati, si passa alla fase vera e propria di aggregazione dei dati.

[1] Se avessimo sostituito in tag `<discard>` con il tag `<permit>`, avremmo preso in considerazione solo i dati che hanno come indirizzo IP sorgente e indirizzo IP destinazione rispettivamente 192.168.1.12 e 10.2.12.254.

Descrizione schemi di aggregazione

Vediamo ora in che cosa consiste esattamente uno schema di aggregazione e quali sono le regole e le convenzioni che occorre seguire.

E' stato già detto che gli schemi di aggregazione sono descritti all'interno dei uno dei file di configurazione: il formato di tale file è di tipo *XML*[11], mediante i quale è facile descrivere le informazioni di cui abbiamo bisogno. L'utilizzo del formato XML permette inoltre di modificare, ampliare o aggiornare facilmente gli schemi. Alcuni esempi dei campi da usare nelle aggregazioni sono:

- indirizzo IP origine
- indirizzo IP destinazione
- protocollo
- porta di origine
- porta di destinazione, etc..

I campi sopra descritti non sono fissi, ma variano a seconda delle aggregazioni che si vogliono applicare. Ogni aggregazione, infatti, permette di avere rappresentazioni delle informazioni, relative al traffico in transito sulla rete, differenti. Nell'esempio fatto prima si parla di indirizzo IP sorgente e indirizzo IP destinazione: questi campi potrebbero essere usati per aggregare i dati sul traffico in transito, tra una sorgente e una destinazione, in generale. Se si specifica nella stessa aggregazione anche il campo che indica il tipo di protocollo, significa che si vuole suddividere il traffico in base ai tipi di protocolli supportati. Quindi, andando a specificare sempre più campi, si ottiene un rappresentazione sempre più in dettaglio delle informazioni raccolte.

Un'aggregazione costituisce una sorta di maschera, da applicare ai file dei template e dei dati, per ricavare le informazioni che ci interessano. Per poter descrivere meglio come si effettua un'aggregazione, è opportuno mostrare come è costruito il file xml. I file di aggregazione vengono generati usando il formato XML unitamente al formato di export dei dati di NetFlow v.9[2][7][6]. Ogni file è accoppiato ad un *DTD*, che fornisce la lista degli elementi, dei tag e degli eventuali attributi che sono contenuti nel file xml. L'utilizzo dei DTD fa sí che si possa usare sia la sintassi di NetFlow v.9, sia che si possa espanderla con campi definiti dall'utente, definendo in questo modo delle aggregazioni più ricche e complesse.

Tuttavia, se l'utente vuole utilizzare dei campi che sono già presenti all'interno della sintassi definita da NetFlow v.9, occorre che i campi seguano la sintassi di quest'ultimo. Per esempio, poiché in NetFlow v.9 sono definiti i campi per indirizzo sorgente e destinazione, rispettivamente `IPv4_SRC_ADDR` e `IPv4_DST_ADDR`, se vogliamo aggregare usando questi campi, il file conterrà i tag con questa sintassi.

Prendiamo in considerazione uno schema di aggregazione in cui occorra aggregare i dati in base all'indirizzo IP sorgente e al tipo di protocollo. Di seguito viene riportato il *Data Type Definition* per il file di aggregazione.

```
<?xml version="1.0"?>
<!DOCTYPE schema [
    <!ELEMENT aggregazione (name,field+,rule+)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT field (#PCDATA)>
    <!ELEMENT rule (#PCDATA)>
]>
```

Questo DTD ci dice che un'aggregazione è costituita da più `field` e da più `rule`, dove con field si indicano i campi che si intendono conosce dall'aggregazione, mentre con rule si indicano i campi rispetto ai quali aggregare. Il file xml corrispondente al DTD è il seguente:

```
<?xml version="1.0" >
<! DOCTYPE schema SYSTEM "aggr.dtd">
<aggregazione>
    <name>AGG_SRC</name>
    <rule>OUT_PKTS</rule>
    <rule>OUT_BYTES</rule>
    <field>IPv4_SRC_ADDR</field>
    <field>PROTOCOL </field>
</aggregazione>
```

Dal parsing del file xml si ricavano le informazioni necessarie per poter aggregare. Con tali informazioni si accede ai file contenti i template e si calcolano le posizioni in cui sono presenti sia campi per cui devo aggregare sia quelli che voglio conoscere.

In questo modo ottengo degli insiemi di offset[2], uno per ogni template, che mi permettono di accedere alle posizioni corrispondenti nei file dei dati, permettendo quindi di aggregare i dati.

Poiché ogni blocco del sistema ha la capacità di aggregare i dati in maniera differente, i dati non sono omogenei, e occorre integrarli, senza perdere informazioni.

Si è detto che le aggregazioni vengono effettuate analizzando i dati che sono presenti nei file memorizzati in locale, all'interno del blocco, riducendo in questo modo l'accesso al db, che potrebbe non riuscire a mantenere delle buone performance con degli accessi continuati.

Tuttavia non è possibile effettuare tutte le aggregazioni con i dati presenti sul disco. Alcune aggregazioni potrebbero infatti richiedere una visione più ampia del comportamento della rete, rispetto a quello offerto dai file. Si è deciso, quindi, di effettuare questo tipo di aggregazioni solo in un secondo momento.

Per questi motivi, all'interno dell'architettura si è deciso di fare uso della seguente convenzione:

1. Le aggregazioni che è possibile fare subito, con i dati che sono memorizzati nei file, vengono definite *di base*;
2. Le aggregazione che occorre fare sul database, perché hanno bisogno di ulteriori informazioni sono *avanzate*.

Esempi di aggregazioni *di base* sono:

- numero di flussi, pacchetti e byte per indirizzo sorgente

Esempi di aggregazioni *avanzate* sono:

- Durata media di un flusso;
- Traffico incoming/outcoming per poter stabilire se si tratta di un server o di un client;
- Utilizzo di un mirror, etc...

La figura 4.2 mostra come sono effettuate le aggregazioni di base all'interno di un blocco.

[2]L'invio dei dati avviene informato NetFlow, di cui si conoscono le varie grandezze dei campi e degli header, permettendo quindi di calcolare facilmente gli offset come bits.

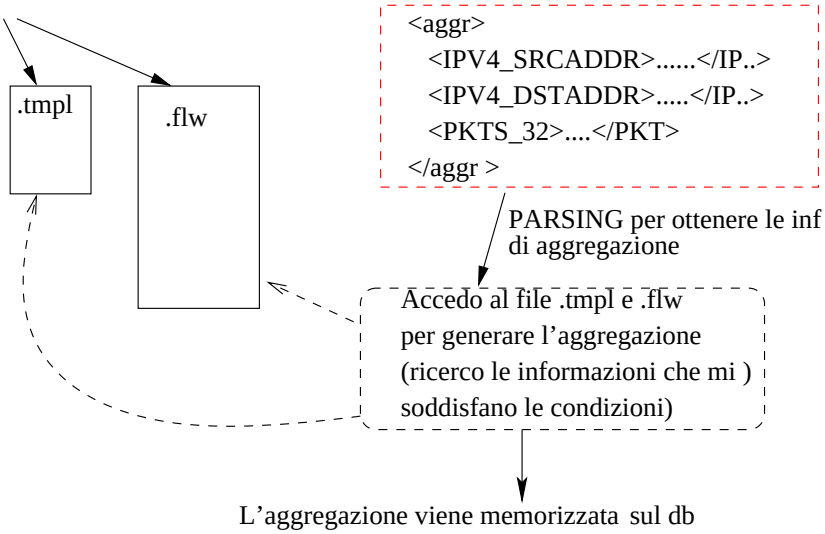

Figura 4.2: Schema aggregazione di base

Durante la fase di applicazione degli schemi di aggregazione di base, ad ognuno dei file che contengono i dati, vengono applicati tutti gli schemi di aggregazione di base che sono presenti nel blocco, ottenendo differenti visioni delle stesse informazioni. Una volta che l'entità Analyzer ha effettuato la prima fase di aggregazione, applicando le aggregazioni di base, si effettua una seconda fase, applicando le aggregazioni avanzate e accedendo alle tabelle del database. Le informazioni cosí ottenute andranno a memorizzarsi all'interno del database, creando altre entry e tabelle. Diversamente dalle aggregazioni di base, quelle avanzate, sono effettuate solo dopo un intervallo di tempo Δt_1 da quando si inizia a memorizzare i primi dati aggregati nel database: in questo modo si spera che, scegliendo un opportuno intervallo, le informazioni necessarie saranno presenti nel database. Dopo un intervallo di tempo Δt_2 le aggregazioni vengono esportate verso gli altri blocchi dell'architettura (e verranno successivamente compresse)[3]. La figura 4.3 mostra lo schema per le aggregazioni avanzate.

Sostanzialmente, le aggregazioni avanzate, sono tutte quelle aggregazioni che richiedono il calcolo di valori, che dipendono da fattori il cui valore dipende a sua volta da una visione più ampia nel tempo del traffico: ciò significa che tutte quelle aggregazioni che richiedono per esempio il valore medio di un qualche parametro, o

[3]Le informazioni non rimango nel database per sempre, ma vengono comunque eliminate se troppo vecchie. I timer in base ai quali effettuare tali operazioni si trovano nel file di configurazione di Analyzer.

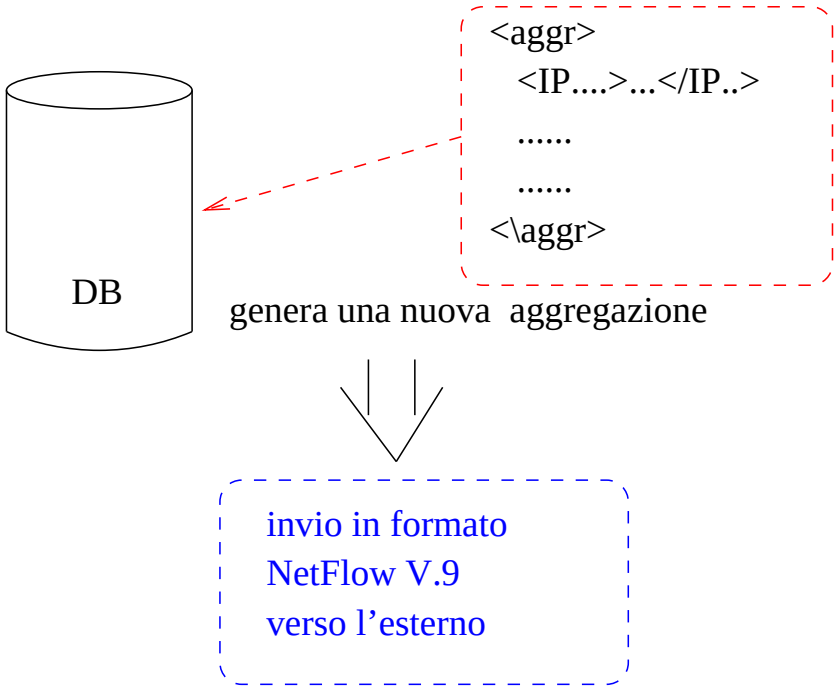

Figura 4.3: Schema aggregazione avanzato

un'analisi in un intervallo di tempo molto ampio, sono da considerarsi *aggregazioni avanzate*.

Il motivo che ha spinto a questa suddivisione dei dati è che si potrebbe voler avere una visione più ad ampio raggio di quello che avviene nella rete. Questo perché la granularità delle informazioni dopo una prima fase di aggregazione potrebbe essere troppo fine e perché si vogliono avere delle informazioni che necessitano di informazioni distribuite su un intervallo di tempo più ampio rispetto a quello fornito dai file in cui sono memorizzati[4].

Una volta che i dati sono stati aggregati, il risultato verrà inserito nel database, creandone una nuova entry. La struttura della nuova entry sarà esattamente quella dell'aggregazione.

La scelta di attenersi il più possibile alla sintassi di NetFlow v.9 è dovuta al fatto che si vuole mantenere una struttura simile a quella di utilizzata per l'export dei dati (si faccia riferimento alla sezione dedicata all'invio dei dati). Ma è dovuta anche al fatto che NetFlow offre un insieme di campi abbastanza ricco, e si può evitare di costruire una nuova sintassi.

[4]L'invio è subordinato ad un intervallo di export che, anche se configurabile, potrebbe essere troppo piccolo rispetto alle necessità dell'aggregazione desiderata.

Le aggregazioni di base

Come detto nella sezione precedente, le aggregazioni si dividono in aggregazioni di base e aggregazioni avanzate. Di seguito verranno descritte le aggregazioni di base che il sistema supporta, cioé quelle che sono rese disponibili all'utente del sistema. Naturalmente l'utente può scegliere di personalizzare anche quest'ultime. Nelle descrizioni successive si farà riferimento ad alcuni campi di NetFlow che sono utilizzati per descrivere gli schemi. La lista completa di tutti i campi che il sistema è in grado di riconoscere è fornita nell'appendice A.

Le aggregazioni supportate sono:

- **Traffico verso un indirizzo IP destinazione**: questa aggregazione permette di conoscere il numero dei byte, dei flussi e dei pacchetti relativi al traffico verso un indirizzo IP destinazione. Il campo rispetto a cui aggregare sarà quindi l'indirizzo IP di destinazione. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi:

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo IP destinazione	n. byte n. flussi n. pacchetti

Tabella 4.1: Aggregazione per indirizzo IP destinazione

- **Traffico generato da un indirizzo IP sorgente**: questa aggregazione permette di conoscere il numero dei byte, dei flussi e dei pacchetti relativi al traffico generato da un indirizzo IP sorgente. Il campo rispetto a cui aggregare sarà quindi l'indirizzo IP sorgente. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi:

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo IP sorgente	n. byte n. flussi n. pacchetti

Tabella 4.2: Aggregazione per indirizzo IP sorgente

- **Traffico tra un specifico IP sorgente e uno specifico IP destinazione**: questa aggregazione permette di conoscere il numero dei byte, dei flussi e dei pacchetti relativi al traffico generato tra uno specifico indirizzo IP sorgente ed uno specifico IP destinazione. Questa aggregazione permette di conoscere, per esempio, le metriche sul traffico di un cliente della rete. I campi rispetto a cui aggregare saranno quindi l'indirizzo IP sorgente e quello di destinazione. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi:

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo IP destinazione indirizzo IP sorgente	n. byte n. flussi n. pacchetti

Tabella 4.3: Aggregazione per indirizzo IP sorgente e destinazione

- **Traffico suddiviso in base al protocollo**: questa aggregazione permette di conoscere il numero dei byte, dei flussi e dei pacchetti del traffico, suddiviso per tipo di protocollo utilizzato. In questo modo è possibile esaminare l'utilizzo della rete in base al tipo di traffico. I campi rispetto cui aggregare saranno il protocollo. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi.

Campi rispetto a cui aggregare	Valori da conoscere
protocollo	n. byte n. flussi n. pacchetti

Tabella 4.4: Aggregazione per tipo di protocollo

- **Traffico suddiviso in base all'applicazione**: questa aggregazione permette di conoscere il numero dei byte, dei flussi e dei pacchetti del traffico, suddiviso per tipo di applicazione. In questo modo è possibile esaminare l'utilizzo della rete in base al tipo di applicazione. I campi rispetto cui aggregare saranno il protocollo, la porta sorgente e quella di destinazione. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi.

Campi rispetto a cui aggregare	Valori da conoscere
protocollo n. porta sorgente n. porta destinazione	n. byte n. flussi n. pacchetti

Tabella 4.5: Aggregazione per tipo di applicazione

- **Traffico suddiviso per protocollo, considerando l'indirizzo IP sorgente**: questa aggregazione per permette di conoscere l'utilizzo della rete, suddiviso per tipo di traffico generato, in base all'utilizzo IP. In questo modo è possibile conoscere l'utilizzo della rete da parte di un singolo cliente della rete. I campi rispetto ai quali aggregare saranno quindi l'indirizzo IP sorgente, il protocollo. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi:

Campi rispetto a cui aggregare	Valori da conoscere
protocollo indirizzo IP sorgente	n. byte n. flussi n. pacchetti

Tabella 4.6: Aggregazione per protocollo e indirizzo IP sorgente

Lo stesso tipo di aggregazione può essere applicata, sostituendo l'indirizzo IP sorgente con l'indirizzo IP di destinazione, permettendo in questo modo di conoscere l'utilizzo della rete verso un indirizzo IP. In questo modo è possibile analizzare il tipo di traffico, per esempio, verso un server in particolare.

- **Traffico suddiviso per applicazione, considerando l'indirizzo IP sorgente**: questa aggregazione per permette di conoscere l'utilizzo della rete, suddiviso per tipo di applicazione utilizzata, in base all'utilizzo IP. In questo modo è possibile conoscere l'utilizzo della rete da parte di un singolo cliente della rete. I campi rispetto ai quali aggregare saranno quindi l'indirizzo IP sorgente, il protocollo, la porta di destinazione e quella sorgente. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi:

Lo stesso tipo di aggregazione può essere applicata, sostituendo l'indirizzo IP sorgente con l'indirizzo IP di destinazione.

Campi rispetto a cui aggregare	Valori da conoscere
protocollo indirizzo IP sorgente n. porta sorgente n. porta destinazione	n. byte n. flussi n. pacchetti

Tabella 4.7: Aggregazione per tipo di applicazione e indirizzo IP sorgente

- **Traffico in dettaglio**: questa aggregazione permette di avere un'analisi più in dettaglio del traffico, tra un indirizzo IP sorgente e uno di destinazione. I campi che vengono coinvolti aggregano in dettaglio il traffico. Quelli rispetto a cui aggregare sono indirizzo IP sorgente e destinazione, protocollo, porta sorgente e destinazione e tos (type of service). I valori da conoscere saranno invece il numero di byte, pacchetti, flussi, startime (il timestamp del primo pacchetto) e l'endtime (il timestamp dell'ultimo pacchetto).

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo IP destinazione indirizzo IP sorgente protocollo n. porta destinazione n. porta sorgente tos	n. byte n. flussi n. pacchetti startime endtime

Tabella 4.8: Aggregazione in dettaglio

- **Traffico suddiviso per Autonomous System**: questa aggregazione permette di conoscere il numero dei byte, dei flussi e dei pacchetti relativi al traffico generato tra Autonomous System, discriminando protocollo. I campo rispetto ai quali aggregare saranno quindi l'indirizzo IP sorgente, indirizzo IP destinazione, AS sorgente, AS destinazione, protocollo, porta sorgente e porta destinazione. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi. La tabella 4.9 mostra uno schema di questa aggregazione.

- **Traffico suddiviso per sottorete**: questa aggregazione permette analizzare il traffico in base alla sottorete sorgente e alla sottorete di destinazione. I

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo IP destinazione	n. byte
indirizzo IP sorgente	n. flussi
AS sorgente	n. pacchetti
AS destinazione	
protocollo	
n. porta sorgente	
n. porta destinazione	

Tabella 4.9: Aggregazione per AS

campi rispetto a cui aggregare saranno la sottorete sorgente, la maschera che permette di identificare la sottorete[5], la sottorete destinazione, la maschera che permette di identificare la sottorete. I valori che si vorranno conoscere saranno invece il numero di byte, pacchetti e flussi. La tabella 4.10 mostra uno schema di questa aggregazione.

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo sottorete destinazione	n. byte
maschera sottorete destinazione	n. flussi
indirizzo sottorete sorgente	n. pacchetti
maschera sottorete destinazione	

Tabella 4.10: Aggregazione per sottorete

Le aggregazioni di base presentate in questa sezione sono state scelte in modo da poter avere una visione il più possibile completa di ciò che avviene nelle rete monitorata.

Le aggregazioni avanzate

In questa sezione verranno invece elencate le aggregazione avanzate che vengono fornite dal sistema all'utente. Come detto nelle sezioni precedenti, queste aggregazioni sono applicate ai dati che sono contenuti all'interno del database. Questo significa che il parsing del file che le contiene genera un insieme di query con sui ricercare i dati, a cui applicare le aggregazioni, all'interno del db. Naturalmente quello che viene fornito è solo un piccolo insieme di possibile aggregazione. Nel sistema viene lasciata la libertà di definire aggregazioni personalizzate.

[5]Numero dei bit contigui nell'indirizzo della rete che identificano.

- **Durata media di un flusso**: questa aggregazione permette di calcolare la durata media di un flusso. Questa aggregazione è però vincolata dal fatto che per come è costruita è necessario che sia stata prima applicata l'aggregazione di base indicata come *aggregazione in dettaglio*. La durata media di un flusso viene calcolate in base allo startime e all'endtime riportati dall'aggregazione di base.

Campi rispetto a cui aggregare	Valori da conoscere
indirizzo IP sorgente indirizzo IP destinazione protocollo n.porta sorgente n.porta destinazione	durata media

Tabella 4.11: Aggregazione per la durata media di un flusso

- **Utilizzo medio di un protocollo**: questa aggregazione permette di conoscere la media dell'utilizzo dei protocolli in un giorno, per poter capire per che tipo di traffico è stata utilizzate mediamente la rete, per esempio in una giornata.

Campi rispetto a cui aggregare	Valori da conoscere
n. flussi protocollo n.porta sorgente n.porta destinazione	utilizzo medio

Tabella 4.12: Utilizzo medio di un protocollo

Le aggregazioni sopra descritte sono state considerate aggregazioni avanzate perché il calcolo dei valori richiesti ha bisogno di fattori che dipendono da un arco di tempo più grande di quello che viene offerto dai file in cui sono memorizzati i dati. Quando di parla di utilizzo medio, come per i protocolli o per la durata media di un flusso, l'arco di tempo a cui si fa riferimento sarà per esempio il giorno.

Gli intervalli di aggregazione

All'interno del sistema l'intervallo temporale di aggregazione gioca un ruolo molto importante, in quanto determina il grado di dettaglio che avranno le aggregazioni

generate all'interno del sistema. Tuttavia si tratta di un'arma a doppio taglio. Se l'intervallo di aggregazione è corto, all'interno del sistema si avranno delle aggregazione con un grado di dettaglio tale da fornire una visione molto accurata di quello che avviene nella rete monitorata. Questo corrisponde ad avere però una grande quantità di dati da inserire all'interno del database. L'inserzione di tale quantità di dati potrebbe portare alla degradazione delle performance del database e sicuramente riempirebbe velocemente lo spazio in esso disponibile.

D'altra parte scegliere un intervallo di aggregazione troppo grande porta ad avere una minore quantità di dati che vengono generate dalle aggregazioni. Quindi si ottiene una minore quantità di dati da memorizzare all'interno del database. Tuttavia, un intervallo troppo grande fa perdere alle informazioni generate il loro grado di dettaglio, con il risultato di avere delle informazioni che riassumono troppo il comportamento della rete.

Un altro problema che sorge quando si vuole scegliere un opportuno intervallo di aggregazione, nei livelli più bassi dell'architettura, è dovuto alla presenza di flussi che hanno una durata maggiore dell'intervallo di tempo.

I flussi hanno solitamente un tempo di vita che viene stabilito dal probe che raccoglie il traffico dalla rete. L'intervallo su cui effettuare l'aggregazione potrebbe essere più piccolo del lifetime del probe: in questo caso si rischia di aggregare dei dati su flussi che ancora non sono terminati.

Le soluzione che solitamente vengono proposti in questi casi sono due: nella prima soluzione non si tiene conto del fatto che un flusso non è ancora terminato e si continua ad aggregare secondo l'intervallo scelto. Nella seconda soluzione, invece, si tiene traccia del fatto che il flusso non è ancora terminato e quindi si effettua una sorta di aggregazione parziale: si continua ad aggregare sull'intervallo scelto, ma il risultato non viene inserito immediatamente all'interno del database, viene mantenuto da parte e dalle aggregazioni parziali si genera un'aggregazione totale, che copra tutto l'intervallo di durata del flusso.

Prima di descrivere qual è la soluzione che è stata scelta all'interno del sistema, è opportuno analizzare quali sono i pro e i contro di entrambe le soluzioni alla luce delle scelte fatte all'interno di questa tesi.

Nella seconda soluzione, cioé nel caso in cui tengo conto del fatto che il flusso non è terminato, occorre che i livelli inferiori, e in particolare il livello base dell'architettura, devono segnalare il fatto che il flusso non è terminato. In questo modo,

possono mantenere da parte i dati e generare solo alla fine un'aggregazione su tutta la durata del flusso. Adottare una soluzione del genere permette di mantenere un buon dettaglio sui flussi, e permette di fare delle aggregazioni come la durata media dei flussi della rete, secondo lo schema riportato nella tabella 4.11. Tuttavia, trattenere le informazioni su flussi che non sono ancora terminati, non mi permette di segnalare ai livelli superiori il fatto che ci sono dei flussi che hanno una durata molto lunga.

La prima soluzione non tiene conto del fatto che un flusso è terminato o meno. Adottare un soluzione di questo tipo all'interno del sistema significa che invio sempre le informazioni sui flussi ai livelli superiori, anche se tali informazioni sono solo parziali. Tuttavia, questo spezzettamento delle informazioni non mi permette di poter effettuare delle aggregazioni che hanno bisogno di informazioni più dettagliate sui flussi, come per esempio quelle necessarie un'aggregazione come quelle nella tabella 4.11.

Alla luce di quanto detto, all'interno del sistema si è deciso di adottare la prima soluzione, di non tenere quindi conto se un flusso è realmente terminato o meno. Infatti, per quanto detto prima, i flussi hanno un lifetime stabilito del probe stesso che li raccoglie. Inoltre l'intervallo di aggregazione è stato stabilito nell'ordine dei minuti, ed esattamente il suo valore è di cinque minuti. Tale intervallo è sufficientemente ampio. Per questo motivo si è deciso di settare opportunamente il lifetime del probe, che risulta essere un paramento configurabile. In particolare, in base al valore che è stato attribuito all'intervallo di aggregazione, si è deciso che i flussi scadono dopo un valore massimo di tre minuti. Allo scadere di tale intervallo, tutti i flussi, anche se non terminati vengono esportati.

Adottare questa soluzione permette di mantenere i livelli superiori costantemente informati su quello che accade nella rete monitorata, anche se si rischia di avere un situazione non reale nel caso di flussi che durano più del lifetime del probe. Tuttavia, l'intervallo è comunque abbastanza ampio e molti dei flussi saranno terminati prima dello scadere del lifetime.

4.1.6 Esempio

Vediamo ora un esempio pratico di schema di aggregazione, per capire meglio cosa significa aggregare i dati e qual è l'output che gli schemi generano.

Supponiamo di voler aggregare i nostri dati in base all'indirizzo IP di desti-

nazione, calcolando per ogni indirizzo IP il numero di pacchetti e di byte inviati. Supponiamo inoltre di disporre, all'interno di un blocco dei seguenti dati:

SRC_ADDR	DST_ADDR	PKTS	BYTES	SRC_PORT	DST_PORT	PROTOCOL
10.12.241.91	172.22.5.161	8	2036	2192	80	6
10.12.241.91	172.22.5.161	8	2036	2194	80	6
10.12.241.91	172.22.5.161	9	1277	2191	80	6
10.12.241.91	172.22.5.161	9	1277	2193	80	6
10.12.241.42	172.22.5.161	6	861	3375	80	6
10.12.241.91	172.22.5.161	2	548	2192	80	6
172.22.6.169	81.37.15.129	40	28400	4345	4662	6
172.22.6.169	81.37.15.129	47	38328	4345	4662	6
10.41.51.174	172.22.5.96	4	349	1521	1568	6

Aggregare i dati secondo lo schema definito significa, in questo caso sommare il numero di pacchetti e il numero dei byte, per un indirizzo IP destinazione, il tutto ripetuto su tutti i dati di cui disponiamo. Le altre informazioni contenute nei dati non vengono considerate dall'aggregazione.

Nell'esempio specifico, lo schema di aggregazione corrisponde a sommare i valori contenuti nella terza e nella quarta colonna, per gli indirizzi IP. Una volta applicato lo schema, quello che si ottiene sono i seguenti valori:

SRC_ADDR	PKTS	BYTES
172.22.5.161	42	8035
81.37.15.129	87	66728
172.22.5.96	4	349

Il risultato dell'aggregazione sarà inserito all'interno del database presente nel blocco. Il nome dato allo schema di aggregazione sarà il nome della tabella in cui andare a memorizzare i dati. In questo modo otterremo le aggregazioni, nel database, classificate in base al nome che dovrà indicare all'utente il significato dell'aggregazione stessa. La struttura sarà invece generata dalla struttura dello schema.

In questo esempio, il risultato dell'aggregazione è costituito da una serie di numeri. Tuttavia, non sempre ci si trova in questo caso. Se infatti volessimo aggregare i dati, con lo stesso schema, ma suddividendo i dati anche i base al protocollo, il risultato potrebbe non essere solo una serie di numeri, ma una struttura un pò più complessa, se non si vogliono ripetere le informazioni comuni. Supponiamo di voler applicare tale schema ai seguenti dati:

SRC_ADDR	DST_ADDR	PKTS	BYTES	SRC_PORT	DST_PORT	PROTOCOL
172.22.4.21	172.22.31.255	4	564	1025	2071	17
172.22.4.21	172.22.31.255	4	564	1025	2071	17
172.22.4.10	172.22.5.150	15	12833	139	2538	6
172.22.4.34	172.22.5.150	1	90	137	137	17
172.22.4.10	172.22.5.219	1	60	0	0	1
172.22.4.10	172.22.5.219	2	268	445	1210	6

L'output generato dallo schema di aggregazione in questo caso è il seguente: Come si vede dal risultato ottenuto, le informazioni aggregate hanno una parte co-

DST_ADDR	PKTS	BYTES	PROTOCOL
172.22.31.255	8	1128	17
172.22.5.150	15	12833	6
172.22.5.150	1	90	17
172.22.5.219	1	60	1
172.22.5.219	2	268	6

mune che viene ripetuta. Se andassimo a memorizzare i dati, cosí come sono all'interno del database occuperemmo dello spazio utile con delle informazioni ripetute.

Un modo per poter evitare questo spreco di spazio è cercare di comprimere il risultato di aggregazioni che generano questo tipo di risultati.

Nell'esempio proposto un modo potrebbe essere quello di costruire tabelle di tabelle,in modo da mantenere solo le parti comuni.

Riprendendo l'esempio, quello che si otterrebbe in questo caso, dando una rappresentazione grafica, è:

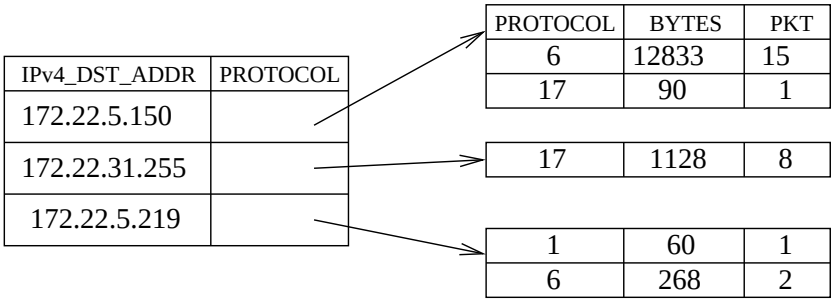

Figura 4.4: Memorizzazione in tabelle annidate

Su una grande quantità di dati questa tecnica permette di risparmiare spazio all'interno dei database.

4.2 Memorizzazione dei dati sul database

I dati che vengono memorizzati sul database sono tutti quei dati che hanno già subito una prima aggregazione, in modo da diminuire il volume di dati che vengono memorizzati, evitando che le tabelle diventino troppo grandi.

Per il momento si è fatta la scelta di utilizzare come database un database di tipo SQL e in particolare i database di MySQL[20]. Poiché le aggregazioni non sono statiche, ma possono essere personalizzate secondo il bisogno del momento, la struttura delle tabelle sarà dinamica. In particolare, si sono adottate le tabelle MyISAM [21] dinamiche (quindi non transactional-safe), sfruttando quindi i loro vantaggi, tra i quali:

1. maggiora velocità dovuta alla mancanza dell'overhead delle transazioni;
2. Minore memoria richiesta per gli update;
3. Minore spazio occupato sul disco rispetto alle tabelle statiche: ogni record usa solo lo spazio che è richiesto e, se diventa troppo grande, viene diviso in pezzi più piccoli.

Tuttavia le tabelle di questo tipo risultano più complesse, in quanto ogni riga deve avere un header che indichi quanto è lunga. Inoltre le tabelle possono essere corrotte poiché non godono dei benefici delle tabelle che usano le transazioni: in una situazione di questo tipo ci si può trovare nel caso in cui il processo che effettua la scrittura sul db viene ucciso o si ha un improvviso shutdown del sistema.

Le tabelle, quindi, risultano essere non omogenee, sia per grandezza (aggregazioni differenti, possono generare tabelle di grandezza differenti), sia come struttura: tipicamente, infatti, le aggregazioni hanno come risultato dei numeri (come avviene se si vuole calcolare il totale del traffico in entrata o in uscita), il cui significato è dato dalla struttura della tabella. Ma si possono generare anche dei valori differenti, che non sono dei semplici numeri,come descritto nella sezione precedente.

La struttura delle tabelle, quindi quali e quanti dati le tabelle possiedono, è generata dagli schemi di aggregazione: dal parsing del file .xml si genera la struttura.

4.2.1 Esempio

Come già fatto nelle sezione precedenti, presentiamo un esempio pratico di come le informazioni vengono memorizzate all'interno del database. Supponiamo di voler

applicare lo schema di aggregazione in figura 4.5 e lo schema di aggregazione in figura 4.6 e di voler quindi aggregare i dati sia in base agli indirizzi IP sorgente che destinazione. Supponiamo inoltre di disporre dei dati mostrati nella tabella 4.13.

```
<?xml version="1.0" >
<! DOCTYPE schema SYSTEM "aggr.dtd">
<schema>
 <aggregazione>
    <name> AGG_SRC</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>IPv4_SRC_ADDR</field>
 </aggregazione>
</schema>
```

Figura 4.5: Aggregazione in base all'indirizzo IP sorgente

```
<?xml version="1.0" >
<! DOCTYPE aggregazione SYSTEM "aggr.dtd">
<schema>
 <aggregazione>
    <name>AGG_DST</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>IPv4_DST_ADDR</field>
 </aggregazione>
</schema>
```

Figura 4.6: Aggregazione in base all'indirizzo IP destinazione

Vediamo ora come l'entità Analyzer lavora per poter memorizzare i dati all'interno delle tabelle.

Dal parsing dei file xml che descrivono lo schema si estrae la struttura delle tabelle. Nel caso specifico, se le tabelle non sono già presenti nel database[6], l'Analyzer creerà due tabelle:

- La prima tabella sarà AGG_SRC, dal nome dato all'aggregazione. I campi della tabella saranno invece `IPv4_SRC_ADDR`, `PKT` e `BYTE`.

[6]Se le tabelle sono già presenti all'interno del database significa che in precedenza altri dati sono stati aggregati con questi schemi e memorizzati nel database. In questo caso l'Analyzer si limiterà a creare una nuova entry nella tabella corrispondente.

IPv4_SRC_ADDR	IPv4_DST_ADDR	PKTS	BYTES	PROTOCOL
172.22.4.21	172.22.31.255	4	564	17
172.22.4.10	172.22.5.150	15	12833	6
172.22.4.34	172.22.5.150	1	90	17
172.22.4.10	172.22.5.219	1	60	1
172.22.4.10	172.22.5.219	2	268	6

Tabella 4.13: Dati disponibili

- La seconda tabella sarà AGG_DST, al nome dato alla seconda aggregazione. I campi della tabelle saranno invece `IPv4_DST_ADDR`, `PKT` e `BYTE`.

Alla struttura sopra descritta, durante la costruzione delle tabelle, viene aggiunto un ulteriore campo che inserisce all'interno del db il concetto di tempo. Il nome del campo è StartTime e il valore che viene inserito per questo campo è il tempo, espresso come secondi dal 1-1-1970, in cui inizia la fase di inserzione. Ogni volta che l'Analyzer inizia ad inserire i dati all'interno del database, il valore del campo viene settato al tempo corrente. Tale accorgimento è stato preso per poter permette di inserire all'interno della tabelle dei record multipli. Allo scadere, infatti, di ogni intervallo di aggregazione, si andranno a memorizzare i risultati degli stessi schemi di aggregazione. Può quindi accadere di dover memorizzare dati che hanno gli stessi valori per le chiavi: per poter distinguere quindi tali valori, anche per poter tracciare un andamento temporale, si tiene conto del fattore tempo. Un esempio di tale situazione e di come il fattore tempo permette di risolverla, è mostrato all'interno dell'appendice B, nella tabella B.4, che fa riferimento all'esempio descritto nella sezione 4.2.2. Analizzando tale tabella si può vedere come ci sono due record con lo stesso valore della chiave[7], ed esattamente quelli che si riferiscono all'aggregazione per l'indirizzo IP destinazione 172.22.6.169 e 172.25.5.255. Poiché si riferiscono ad aggregazioni che sono state fatte in momenti differenti, i due record si distinguono tra di loro per il valore che è stato assegnato al campo StartTime.

Una volta create la tabelle e aggregati i dati secondo gli schemi proposti, come descritto nelle sezioni precedenti, l'Analyzer provvederà a inserire il risultato delle aggregazioni nelle rispettive tabelle, ottenendo i risultati mostrati in figura 4.7.

La figura mostra la struttura e il contenuto delle tabelle descritte dopo che i dati sono stati aggregati e memorizzati all'interno del database. In particolare, come si

[7]Nella costruzione della tabelle la chiave è costituita dai campi rispetto a cui si aggrega. Nell'esempio descritto il campo è costituito dall'indirizzo IP destinazione.

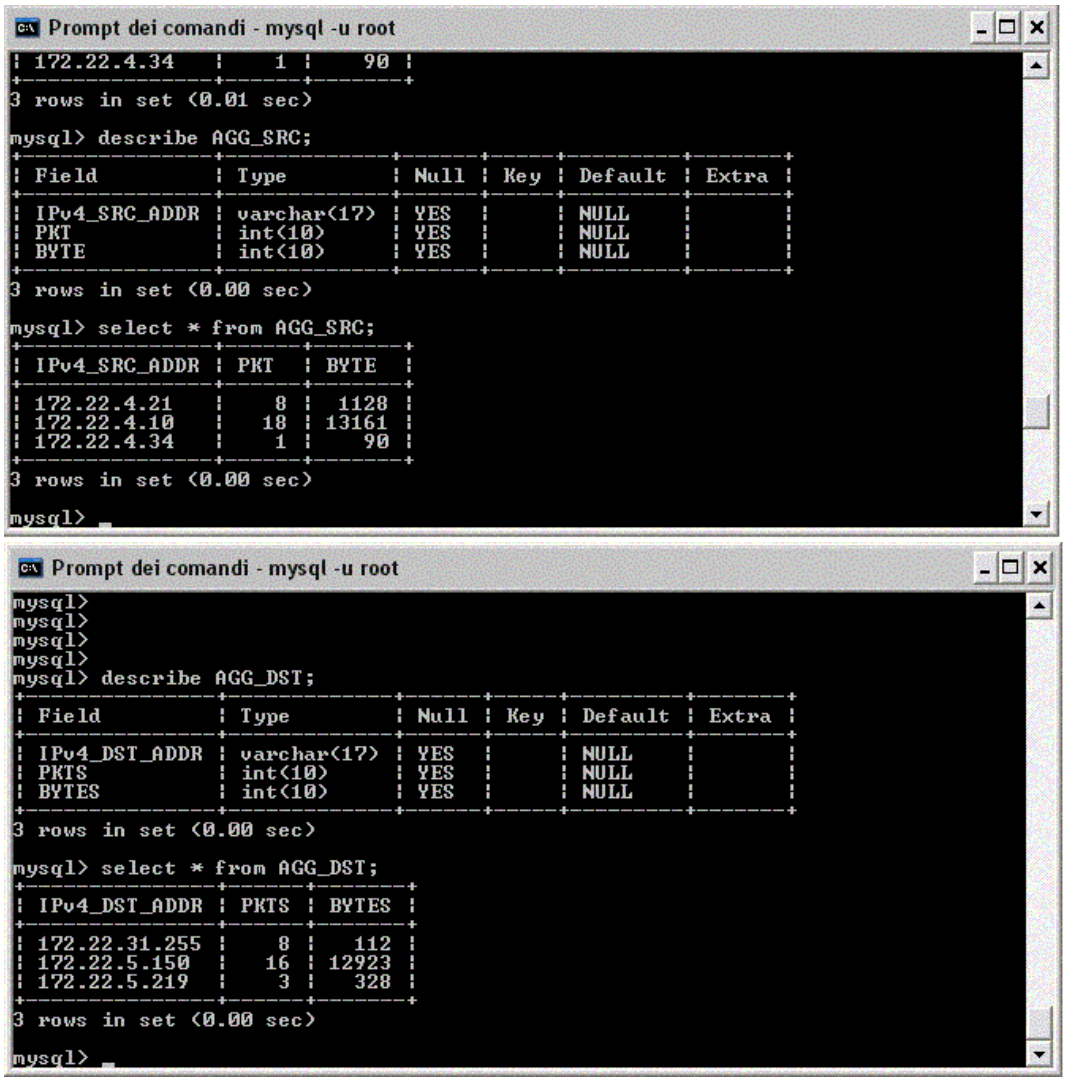

Figura 4.7: Contenuto delle tabelle dopo la memorizzazione dei dati

vede dalla figura stessa, questo è il contenuto della tabelle create con MySQL. Da notare è anche il fatto di come la struttura sia perfettamente concordante, anche nella sintassi dei campi, con gli schemi di aggregazione descritti. Quest è anche una agevolazione nella successiva fase di export dei dati: mantenendo i campi identici a quelli di NetFlow v9, sarà più facile costruire i pacchetti dei dati da inviare. Per maggiori dettagli si rimanda alla sezione 4.3.

4.2.2 Creazione delle tabelle

In questa sezione verrà presentato un esempio per permettere di comprendere meglio qual è il meccanismo che il sistema utilizza per poter creare la tabelle dinamiche all'interno del database e qual è la politica che viene adottata quando si ha un aggiornamento degli schemi di aggregazione. Le motivazioni che hanno spinto all'utilizzo di tali tipi di tabelle è stato discusso all'interno delle sezioni precedenti.

Per maggiore chiarezza, consideriamo uno solo dei blocchi che sono presenti all'interno di un livello, ed in particolare consideriamo in momento in cui l'entità Analyzer sta applicando le proprie aggregazioni e riceve un aggiornamento degli schemi. Per quanto detto nelle sezioni precedenti, il cambiamento degli schemi di aggregazione corrisponde ad un cambiamento della configurazione di Analyzer. Quando l'entità Analyzer sarà riattivata dall'entità Configure, tra le varie operazioni che dovrà compiere per ripristinare il suo funzionamento, dovrà effettuare nuovamente il parsing del file di configurazione che contiene gli schemi.

E' stato detto più volte che il parsing di questo file genererà la struttura delle tabelle.

Consideriamo il caso in cui il nuovo file sia quello mostrato in figura 4.9, mentre il vecchio file contenente i vecchi schemi sia mostrato in figura 4.8.

Consideriamo inoltre le due tabelle B.1 B.2 presenti nell'appendice B.

L'entità Analyzer, dal vecchio file delle aggregazione ha ricavato la struttura di tre tabelle, una per ognuno degli schemi di aggregazione che sono presenti all'interno del file. La struttura di queste tabelle, che riguardano rispettivamente il primo, il secondo e il terzo schema, è la seguente:

- **Tabella AGG_SRC**: i campi di questa tabella sono `IPv4_SRC_ADDR`, `PKTS` e `BYTES`. In particolare, poiché il database che viene utilizzato all'interno

```
<?xml version="1.0" >
 <! DOCTYPE schema SYSTEM "aggr.dtd">
<schema>
 <aggregazione>
    <name> AGG_SRC</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>IPv4_SRC_ADDR</field>
 </aggregazione>
 <aggregazione>
    <name>AGG_DST</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>IPv4_DST_ADDR</field>
 </aggregazione>
 <aggregazione>
    <name>AGG_PROTO</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>PROTOCOL</field>
 </aggregazione>
 </schema>
```

Figura 4.8: Vecchi schemi di aggregazione

```
<?xml version="1.0" >
 <! DOCTYPE schema SYSTEM "aggr.dtd">
<schema>
 <aggregazione>
    <name> AGG_SRC</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>IPv4_SRC_ADDR</field>
 </aggregazione>
 <aggregazione>
    <name>AGG_DST</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>IPv4_DST_ADDR</field>
 </aggregazione>
 <aggregazione>
    <name>AGG_PROTO_SRC_172_22_6_169</name>
    <rule>PKTS</rule>
    <rule>BYTES</rule>
    <field>PROTOCOL</field>
    <field field_val=172.22.6.169>IPv4_SRC_ADDR</field>
 </aggregazione>
 </schema>
```

Figura 4.9: Nuovi schemi di aggregazione

del blocco è un database di tipo MySQL, l'entità Analyzer effettuerà una operazione di `CREATE` della tabella, con i campi sopra descritti

- **Tabella AGG_DST**: i campi di questa tabelle sono invece `IPv4_DST_ADDR`, `PKTS` e `BYTES`
- **Tabella AGG_PROTO**: i campi di quest'ultima tabella sono invece `PKTS`, `BYTES` e `PROTOCOL`

Tutte e tre le tabelle sono costruite tramite delle operazione di CREATE, specificando tra i vari parametri che le tabelle che si intende creare sono tabelle dinamiche. Inoltre, per quanto detto nella sezione precedente, viene inserito in ogni tabella anche un campo che indichi il momento in cui inizia la fase di memorizzazione dei dati all'interno del database.

L'entità Analyzer aggrega così i suoi dati e li inserisce all'interno delle tabelle create. Applicando i tre schemi ai dati della tabella B.1, si ottengono i risultati mostrati nelle tabelle 4.14, 4.15 e 4.16. Il valore del campo StratTime è settato per questo esempio al valore 1095941580.

A questo punto, il blocco riceve un cambiamento di configurazione, dovuto all'aggiornamento del file che contiene le aggregazioni. Quando l'entità Analyzer verrà riattivata dell'entità Configure, tra le operazioni preliminari necessarie per riprendere il corretto funzionamento, effettuerà il parsing del nuovo file (vedi figura 4.9) con gli schemi di aggregazione.

Dal parsing di tale file, Analyzer capirà che ci sono tre schemi e che quindi è necessario costruire tre nuove tabelle. Viene quindi ricavata la seguente struttura delle tabelle:

- **Tabella AGG_SRC**: i campi di questa tabella sono `IPv4_SRC_ADDR`, `PKTS` e `BYTES`.
- **Tabella AGG_DST**: i campi di questa tabella sono invece `IPv4_DST_ADDR`, `PKTS` e `BYTES`
- **Tabella AGG_PROTO_SRC_172_22_6_169**: i campi di quest'ultima tabella sono invece `PKTS`, `BYTES`, `PROTOCOL`.

Prima di effettuare la creazione di queste nuove tabelle, l'entità Analyzer controlla se le tabelle sono già presenti nel database: Nell'esempio che stiamo facendo

StratTime	IPv4_SRC_ADDR	PKTS	BYTES
1095941580	80.180.205.119	2	92
1095941580	172.183.195.184	2	92
1095941580	217.122.233.133	2	92
1095941580	217.127.237.146	2	92
1095941580	220.240.123.157	2	92
1095941580	10.12.241.91	34	6626
1095941580	10.41.51.174	4	349
1095941580	149.156.88.48	2	92
1095941580	151.37.16.23	6	401
1095941580	151.37.16.30	4	357
1095941580	172.22.17.58	3	156
1095941580	172.22.4.10	15	12833
1095941580	172.22.4.118	4	512
1095941580	172.22.4.121	1	78
1095941580	172.22.4.18	3	324
1095941580	172.22.4.21	4	564
1095941580	172.22.4.34	2	174
1095941580	172.22.4.42	2	94
1095941580	172.22.4.43	16	6896
1095941580	172.22.4.53	1	84
1095941580	172.22.5.129	6	468
1095941580	172.22.5.134	1	237
1095941580	172.22.5.14	1	244
1095941580	172.22.5.149	11	1202

Tabella 4.14: Risultato primo schema di aggregazione del file vecchio

StartTime	IPv4_DST_ADDR	PKTS	BYTES
1095941580	172.22.31.255	4	92
1095941580	172.22.4.102	1	92
1095941580	172.22.4.32	1	92
1095941580	172.22.4.62	1	92
1095941580	172.22.5.149	1	92
1095941580	172.22.5.150	16	92
1095941580	172.22.5.161	34	92
1095941580	172.22.5.35	2	92
1095941580	172.22.5.96	4	92
1095941580	172.22.6.169	23	92
1095941580	172.22.7.255	27	92
1095941580	235.0.0.1	16	92

Tabella 4.15: Risultato secondo schema di aggregazione del file vecchio

StartTime	PROTOCOL	PKTS	BYTES
1095941580	1	4	240
1095941580	6	77	21212
1095941580	17	49	10699

Tabella 4.16: Risultato terzo schema di aggregazione del file vecchio

ben due delle tabelle presenti nel database, AGG_SRC e AGG_DST sono già presenti nel database. In casi come questo, la creazione di una nuova tabella non ha luogo, ma si continua a mantenere la vecchia tabella, inserendo semplicemente delle nuove entry all'interno. Le altre due tabelle però non coincidono. In questo caso bisogna quindi abbandonare la vecchia tabella e costruire quella nuova. Prima però di effettuare questa nuova creazione, l'Analyzer provvederà a comprimere la vecchia tabella AGG_PROTO, liberando in questo modo delle spazio per nuove creazioni e inserzioni di dati.

Esattamente come per le precedenti, tali tabelle saranno create tramite una operazione di create specificando i parametri che permettono di costruirla come tabella dinamica.

A questi punto, dopo aver effettuato le corrette operazioni di aggiornamento delle tabelle, l'Analyzer può iniziare ad aggregare i nuovi dati, mostrati nella tabella B.2 in appendice B. Il risultato delle nuove operazioni di aggregazione, è mostrato in nelle tabelle 4.17, 4.18 e 4.19. Il valore del campo StarTime è settato al valore 1095941760.

StartTime	IPv4_SRC_ADDR	PKTS	BYTES
1095941760	172.22.5.173	1	246
1095941760	172.22.5.182	3	234
1095941760	172.22.5.199	1	229
1095941760	172.22.5.29	1	242
1095941760	172.22.5.54	6	276
1095941760	172.22.6.169	143	79226
1095941760	216.94.119.45	2	92
1095941760	217.95.192.164	2	92
1095941760	62.219.64.111	2	92
1095941760	80.185.249.97	10	460
1095941760	81.37.15.129	79	3634
1095941760	81.53.167.161	2	92
1095941760	82.55.200.80	2	92

Tabella 4.17: Risultato primo schema di aggregazione del file nuovo

StratTime	IPv4_DST_ADDR	PKTS	BYTES
1095941760	80.180.205.119	2	92
1095941760	172.183.195.184	2	92
1095941760	217.122.233.133	2	92
1095941760	217.127.237.146	2	92
1095941760	220.240.123.157	2	92
1095941760	149.156.88.48	2	92
1095941760	151.37.16.23	8	92
1095941760	151.37.16.30	8	505
1095941760	172.22.4.53	4	84
1095941760	172.22.6.169	99	4554
1095941760	172.22.7.255	6	951
1095941760	217.95.192.164	2	92
1095941760	224.0.0.22	2	276
1095941760	62.219.64.111	2	92
1095941760	80.185.249.97	19	10408
1095941760	81.37.15.129	87	66728
1095941760	81.53.167.161	2	92
1095941760	82.55.200.80	2	92

Tabella 4.18: Risultato secondo schema di aggregazione del file nuovo

StartTime	PROTOCOL	PKTS	BYTES
1095941760	1	1	84
1095941760	6	142	79142

Tabella 4.19: Risultato terzo schema di aggregazione del file nuovo

La nuova situazione delle tabelle B.3, B.4 e B.5 del database è mostrata all'interno della tabella in appendice B.

4.3 Export delle aggregazioni

I dati che vengono inviati all'esterno sono quelli che sono presenti all'interno del database. Si tratta quindi di intere tabelle (o di alcune righe) del database.

Nasce quindi il problema di scegliere un opportuno formato di invio di tali dati. E' necessario infatti che la costruzione del messaggio di invio non sia costosa in termini di tempo, in modo da non introdurre un ulteriore ritardo nella trasmissione dei dati, in aggiunta a quello che viene introdotto durante la trasmissione vera e propria sul mezzo fisico.

Le informazioni vengono esportate per tabelle, inglobate nel formato di export dei dati di NetFlow v.9[7][6]. L'id del template viene generato in locale dal blocco (in NetFlow v.9 l'unicità è garantita rispetto al device che genera il template stesso).

Il template del messaggio viene generato dalla struttura della tabella, mentre la successiva parte di dati viene generata dai quelli veri e propri memorizzati all'interno del db. In questo modo, per costruire un messaggio, occorre accedere al database e ottenere la struttura della tabella e poi effettuare una sorta di dump delle informazioni.

Esempio

Diamo un esempio di come effettivamente viene costruito il pacchetto per l'invio dei dati. Consideriamo le tabelle che sono state costruite nell'esempio della sezione 4.2.1. Effettuando un dump delle tabelle, l'Analyzer può ricavarne la struttura. Consideriamo la tabella `AGG_SRC`, dal dump di questa tabella, si ricava la seguente struttura:

```
IPv4_SRC_ADDR
PKTS
BYTES
```

Poiché il sistema fa uso di NetFlow v9, occorre per prima cosa costruire il template che descrive i dati. Il formato, in generale, di un template è mostrato nella sezione (mettere riferimento alla sezione in cui descrivi NetFlow v9) e in [7] [6]. L'Analyzer

provvederà a calcolare l'id del template che deve essere unico, la lunghezza totale del template stesso, il numero di campi presenti all'interno del template stesso. A questo punto si può passare a costruire il template aggiungendo il nome dei campi e la loro lunghezza in byte. Il risultato è mostrato in figura 4.10 Poiché il pacchetto

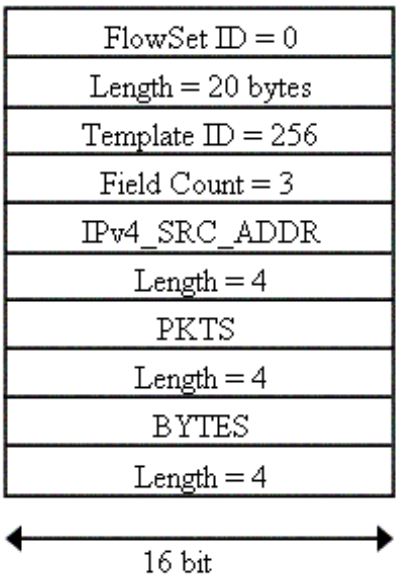

Figura 4.10: Template costruito dalla struttura della tabella AGG_SRC

contiene anche un header, l'Analyzer provvederà a costruire anche quest'ultimo. La struttura dell'header cosí come quella del template è descritta nella sezione 2.3.4 e in [7] [6]. Per costruire il template occorre calcolare il numero di template e dati[8]. Occorrerà quindi specificare il valore del campo sequence number, che permette di capire l'esatta sequenza di invio dei pacchetti e il source ID che identifica l'entità che esporta.

Alla fine l'Analyzer dovrà costruire la parte dati relativa al pacchetto. Prima di inserire i dati, ottenuti dal dump delle tabelle del db, seguendo la struttura del template occorrerà calcolare la lunghezza totale in byte dell'intera parte dati. Il risultato ottenuto in questo esempio è mostrato in figura 4.11.

[8] In NetFlow v9 corrisponde a calcolare il numero di FlowSet che sono presenti nel pacchetto.

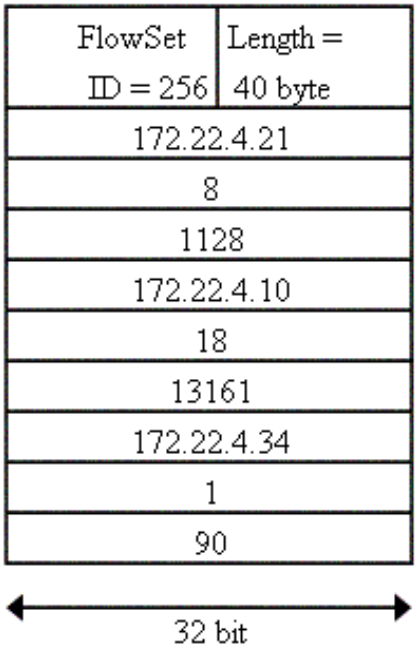

Figura 4.11: Parte dati del pacchetto costruito in base ai dati della tabella AGG_SRC

Il pacchetto cosí ottenuto sarà inviato al livello superiore, secondo i criteri descritti nella sezione successiva.

Se le tabelle sono troppo grandi, questo meccanismo può produrre un certo overhead, che va ad impattare sull'overhead totale di invio dei dati.

L'invio avviene ad intervalli di tempo configurabili, in modo da non generare un flusso continuo di dati, che andrebbe ad impattare sulla performance della rete, in quanto produrrebbe traffico aggiuntivo a quello che già si trova sulla rete stessa.

Una volta che la tabella è stata esportata, viene compressa, in modo da liberare dello spazio per la memorizzazione di nuovi dati.

Un modo per poter ridurre l'overhead della costruzione del messaggio, sarebbe quello di evitare la trasformazione dei dati in formato NetFlow v.9, e di inviare direttamente i dati presenti nel database. Tuttavia, per come sono costruite le aggregazioni, le informazioni sono in un formato molto simile a quello di NetFlow. In questo modo, le aggiunte dei campi necessari ad effettuare la trasformazione, non dovrebbero produrre un ritardo tale da andare da andare ad impattare in modo considerevole sull'overhead totale di trasmissione.

Inoltre, la scelta di effettuare la trasmissione in NetFlow v.9, è stata fatta per poter mantenere uniformità nel formato di invio/ricezione dei dati in modo da favorire anche la replicabilità: in questo modo non occorre modificare nessuna interfaccia tra i vari livelli, potendo in questo modo mantenere uguali tutti i blocchi.

4.3.1 Metodo di export

Nella sezione precedente è stato detto che le informazioni contenute all'interno dei database, presenti in un livello vengono esportate ai livelli superiori ad intervalli di tempo configurabili. In questo modo i livelli superiori dispongono continuamente di nuovi dati su cui applicare le aggregazioni. Tale approccio è in contrasto, naturalmente, con un approccio che prevede l'invio dei dati su richiesta dei livelli superiori. Si potrebbe obiettare che una politica di questo tipo può portare dei problemi di congestione di un blocco. Si potrebbe infatti rischiare di inviare continuamente troppi dati che il blocco presente nel livello successivo non è in grado di ricevere e a processare ed elaborare velocemente. Una rete di media grandezza, come una rete universitaria, genera mediamente migliaia di flussi al secondo: anche se compressi tramite gli schemi di aggregazione, il volume di questi dati risulta comunque molto consistente. Problemi in cui non si rischia di incorrere con una politica di invio su richiesta esplicita. Al contrario, se si optasse per una politica di solo invio su richiesta esplicita, si rischierebbe di non avere i dati disponibili per controllare il comportamento della rete, in quanto bisogna attendere l'invio della richiesta e e la ricezione delle informazioni.

Per ovviare a questi problemi è stato adottato un approccio misto, che permetta di fondere insieme l'approccio di invio si richiesta e quello che prevede l'invio continuo di informazioni, in modo da avere sempre disponibili le informazioni sul comportamento della rete, senza rischiare di congestionare il blocco che le riceve.

Il sistema, infatti, invierà ad intervalli di tempo configurabili solo un sottoinsieme dei dati. Tale sottoinsieme di dati, che può essere scelto da che utilizza il sistema, dovrà essere scelto in modo da dare comunque una visione di quello che avviene nella rete. Questo corrisponde a inviare una parte delle aggregazioni che sono state definite *aggregazioni di base*. In particolare il sistema deve inviare i dati riguardanti le seguenti aggregazioni:

- Traffico diviso in base al protocollo, in quanto permette di avere una visione dell'utilizzo della rete, in termini di tipo di traffico generato.

- Traffico generato da un indirizzo IP sorgente, in quanto permette di controllare nello specifico il comportamento di indirizzi IP di particolare interesse.

- Traffico generato da un indirizzo IP destinazione.

Gli altri dati verranno inviati solo su richiesta esplicita degli altri livelli o in caso di eventi eccezionali come l'arresto dell'entità Analyzer per un cambiamento di configurazione. Altri eventi che possono far scattare questa seconda fase di invio senza richiesta esplicita è il superamento di eventuali soglie di guardia impostate nel sistema (valori calcolati nelle aggregazioni, che superano un certo valore, tabelle che superano una certa grandezza, etc.).

L'invio dei dati su richiesta è subordinato allo scambio di particolari messaggi tra i livelli coinvolti. Il livello superiore infatti richiede l'invio di dati aggiuntivi rispetto all'insieme minimo, inviando al livello inferiore un messaggio in cui vengono specificati quali dati[9] il livello inferiore deve inviare. I dati sono identificati dal nome dell'aggregazione che li ha generati. Il formato del messaggio che viene inviato è mostrato in figura 4.12.

```
<richiesta>
    <id_blocco_sup>35</id_blocco_sup>
    <id_richiesta>35_1</id_richiesta>
    <numero_dati>5</numero_dati>
    <nome_dati>....</nome_dati>
    <nome_dati>....</nome_dati>
    <nome_dati>....</nome_dati>
    <nome_dati>....</nome_dati>
    <nome_dati>....</nome_dati>
</richiesta>
```

Figura 4.12: Formato per la richiesta di dati aggiuntivi

Il significato dei campi presenti all'interno del messaggio è il seguente:

- **<id_blocco_sup>**: indica quale dei blocchi del livello superiore, ai quali è possibile inviare i dati, ha effettuato la richiesta di invio di nuove informazioni.

- **<id_richiesta>**: unitamente al campo precedente serve per identificare in modo univoco la richiesta, in modo da poter discriminare tra richieste ricevute

[9] Ricordiamo che i dati, nei livelli superiori al primo, corrispondono ai dati già aggregati.

da blocchi differenti. Il solo id permette anche di discriminare tra più richieste effettuate da uno stesso blocco.

- <**numero_dati**>: indica quanti dati sono stati richiesti.
- <**nome_dati**>: indica il nome dei dati richiesti. Poiché i dati in questo caso sono tabelle all'interno del db, il nome corrisponde al nome della tabelle[10].

La ricezione, nel livello inferiore, di questo messaggio provoca l'invio di un nuovo messaggio, questa volta dal livello inferiore al livello superiore: l'invio di tale messaggio corrisponde ad accettare la richiesta ed indica al livello superiore che fino alla ricezione di un nuovo messaggio dal livello inferiore, ciò che viene inviato sono i soli dati richiesti. Quando il livello inferiore ha terminato di inviare i dati al livello superiore, invia un esplicito messaggio di end: da ora in poi la situazione ritorna alla normalità e i dati scambiati sono solo quelli corrispondenti all'insieme minimo.

I dati che possono essere richiesti dai livelli superiori devono tuttavia essere già 'presenti' al momento della richiesta. Questo significa che i dati richiesti, anche se non presenti, non vengono calcolati dall'entità Analyzer al momento della richiesta. La situazione in cui i dati non sono presenti, è descritta di seguito.

Nella situazione in cui i dati richiesti non fossero disponibili, cancellati o persi, il livello inferiore invia subito il messaggio che indica la fine dell'invio su richiesta. Il formato del messaggio di accettazione della richiesta e di fine invio è lo stesso. Nelle figure 4.13, 4.14 e 4.15 vengono mostrati il formato del messaggio di accettazione e di fine dell'invio dei dati e uno schema della sequenza in cui avviene la fase di handshake e di trasmissione.

```
<stato_richiesta>
    <id_blocco_inf>24</id_blocco_inf>
    <status>accepted</status>
    <id_richiesta>1_35</id_richiesta>
</stato_richiesta>
```

Figura 4.13: Formato del pacchetto per l'accettazione dell'invio

Nella figura 4.13 il significato dei campi presenti è il seguente:

[10]Il nome delle tabelle viene ricavato del nome dato allo schema di aggregazione che le ha generate.

- **<id_blocco_inf>** : identifica il blocco del livello inferiore al quale è stata inviata la richiesta di invio di dati aggiuntivi.

- **<status>** : indica se la richiesta è stata accettata o se l'invio dei dati è terminato. Nel caso in cui il blocco non sia in grado di inviare i dati, viene inviato il valore end.

- **<id_richiesta>** : identifica la richiesta che è stata gestita, sia nel caso positivo, in cui i dati sono inviati, sia nel caso negativo, in cui la richiesta è rifiutata.

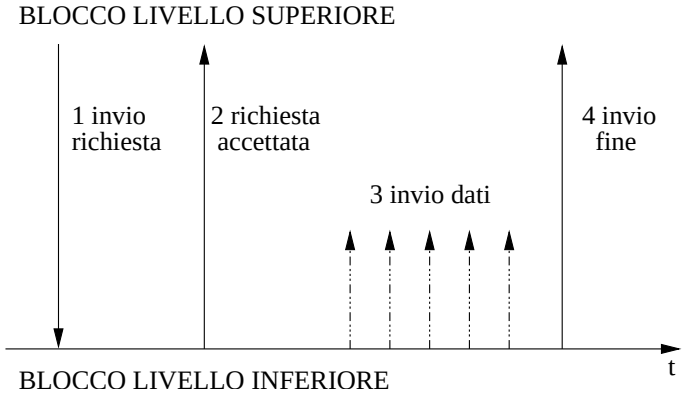

Figura 4.14: Schema di invio con accettazione richiesta

Figura 4.15: Schema di invio con rifiuto della richiesta

Prelievo dei dati

Oltre all'invio periodico dei dati dal livello inferiore al livello superiore, il sistema mette a disposizione la possibilità di effettuare un prelievo periodico dei dati, effettuato in questo caso dai blocchi dei livelli superiori dai blocchi dei livelli inferiori.

Diversamente dall'invio periodico, dove i dati sono prelevati dal database, in questo caso i dati sono prelevati da una directory al di fuori dal database, nel filesystem del blocco del livello da cui si preleva.

Tale meccanismo è stato adottato per rendere più autonomi i livelli superiori: l'invio periodico dei dati obbliga i livelli a subire i dati, mentre in questo modo sono i livelli superiori a decidere quando prelevare i dati.

Per rendere ancora più autonomo questo meccanismo i dati presenti all'interno della directory da cui prelevare non sono necessariamente gli stessi che il livello otterrebbe con l'invio dai livelli inferiori. Ogni blocco mette infatti a disposizione un insieme di dati tra i quali i blocchi del livello superiore possono scegliere, seguendo ognuno una propria politica. Inoltre per rendere ancora più autonoma il meccanismo, i dati verranno prelevati tramite il protocollo SCTP[29][28].

I dati prelevati in questo modo possono essere utilizzati all'interno del blocco con due scopi differenti: possono cioè essere trattati come gli altri dati ricevuti quindi saranno filtrati e aggregati, oppure possono essere memorizzati all'interno del database del blocco senza subire nessuna modifica. Quest'ultimo uso può essere utilizzato quando si vuole mantenere lo stesso grado di granularità e di dettaglio delle informazioni.

Tuttavia, poiché il sistema ha fondamentalmente il compito di ridurre il volume dei dati per poterli analizzare al fine di generare statistiche che permettano di tracciare l'andamento del comportamento della rete monitorata, il secondo utilizzo sarà meno frequente, anche perché potrebbe portare ad una forte duplicazione dei dati tra i livelli della gerarchia.

Il formato dei dati contenuti all'interno della directory sarà lo stesso formato in cui i dati vengono memorizzati nei file quando vengono ricevuti. Questo significa che un insieme di dati, per esempio quelli che descrivono il traffico in termini di byte e pacchetti inviati tra una sorgente e una destinazione vengono descritti da due tipi di file. Il primo tipo di file conterrà la struttura dei dati, mentre il secondo tipo di file conterrà i dati stessi. Per costruire il file che contiene la struttura dei

dati viene costruito un template che ne descriva la struttura, mentre i dati saranno memorizzati seguendo questa struttura.

Esempio

Supponiamo di voler rendere disponibile all'interno della directory per il prelievo, i dati descritti nella tabella 4.13. Per renderli disponibili occorre memorizzarli nel formato descritto in precedenza. Bisogna quindi, per prima cosa, creare il template che descriva la struttura dei dati. Si genera quindi un id del template, la sua lunghezza totale in byte e il numero dei campi contenuti del template stesso. Il file risultante, che esattamente come tutti gli latri dati ricevuti ha estensione *.tmpl* è mostrato in figura 4.16. Facendo riferimento al formato di export dei dati di NetFlow

```
0               28
256              5
IPv4_SRC_ADDR    4
IPv4_DST_ADDR    4
PKTS             4
BYTES            4
PROTOCOL         4
```

Figura 4.16: Esempio template dei dati che vengono resi disponibili per il prelievo

v9[7][6], il significato dei primi valori rispettivamente è FlowSet ID , lunghezza totale del template in byte, Template ID, numero dei campi presenti. Seguono poi i nomi dei campi seguiti ognuno dalla propria lunghezza in byte.

I dati seguiranno la struttura descritta nel file che contiene il template. Esattamente come per il file dei template, questo file ha la stessa estensione dei dati inviati e cioé *.flw*. La struttura del file è mostrata in figura 4.17

```
172.22.4.21  172.22.31.255  4  564  17
172.22.4.10  172.22.5.150   15 12833 6
172.22.4.34  172.22.5.150   1  9  6
172.22.4.10  172.22.5.219   1  60 1
172.22.4.10  172.22.2.219   2  268 6
```

Figura 4.17: Esempio dei dati che vengono resi disponibili per il prelievo

Il formato di memorizzazione di questi dati è lo stesso dei dati inviati, perché si suppone che tali dati saranno filtrati e aggregati esattamente come tutti gli altri dati scambiati attraverso i livelli della gerarchia. Non viene previsto un formato nel caso di inserzione diretta nel database: nel caso in cui un blocco decida di memorizzare i dati direttamente all'interno del database, sarà cura delle entità presenti nel blocco costruire le opportune query per la memorizzazione nel database. Tuttavia, il formato dei dati, per come costruito facilita questo compito: il file contenente il template descrive bene la struttura dei dati, fornendo anche la loro grandezza in byte.

Il prelievo dei dati può essere utilizzato all'interno del sistema insieme all'invio dei dati, senza che la presenza di una politica escluda la presenza dell'altra. Il sistema può infatti essere configurato in modo da utilizzare per alcuni blocchi il prelievo dei dati, mantenendo per gli altri la politica che prevede l'invio periodico dei dati. La scelta di quando sostituire l'invio dei dati con il prelievo viene lasciato all'utente del sistema, che potrà scegliere secondo le necessità riscontrate.

Esempio

Vediamo ora con un esempio quali sono le informazioni che vengono realmente inviate ai livelli superiori, a intervalli di tempo, senza attende l'esplicita richiesta del livello superiore. Supponiamo di disporre dei dati elencati nella tabella 4.20

IPv4_SRC_ADDR	IPv4_DST_ADDR	PKTS	BYTES	SRC_PORT	DST_PORT	PROTOCOL
172.22.17.58	172.22.4.102	1	52	0	2048	1
172.22.17.58	172.22.4.32	1	52	0	2048	1
172.22.17.58	172.22.4.62	1	52	0	2048	1
172.22.4.28	172.22.7.255	1	249	138	138	17
172.22.4.30	172.22.7.255	1	245	138	138	17
172.22.4.31	172.22.7.255	1	78	137	137	17
172.22.4.31	172.22.7.255	1	249	138	138	17
172.22.4.34	172.22.5.149	1	84	137	137	17
172.22.4.34	172.22.5.150	1	90	137	137	17
172.22.4.34	172.22.7.255	1	229	138	138	17
172.22.4.37	172.22.7.255	1	248	138	138	17
172.22.4.38	172.22.7.255	1	245	138	138	17
172.22.6.169	81.37.15.129	1	28400	4345	4662	6
172.22.6.169	81.37.15.129	1	38328	1345	4662	6

Tabella 4.20: Esempio dati presenti in un blocco

Applichiamo a questi dati i tre schemi di aggregazione descritti nella sezione precedente. I risultati, che saranno memorizzati all'interno dei database, sano rappresentati nelle tabelle 4.21, 4.22 e 4.23.

E saranno proprio questi i dati che verranno inviati al livello successivo. Tutte

IPv4_SRC_ADDR	PKTS	BYTES
172.22.17.58	3	156
172.22.4.28	1	249
172.22.4.30	1	245
172.22.4.31	2	327
172.22.4.34	3	403
172.22.4.37	1	248
172.22.4.38	1	245
172.22.6.169	87	66728

Tabella 4.21: Dati aggregati per indirizzo IP sorgente

PROTOCOL	SRC_PORT	DST_PORT	PKTS	BYTES
1	0	2048	3	156
6	4345	4662	87	494
17	137	137	3	411
17	138	138	91	67540

Tabella 4.22: Dati aggregati per protocollo

IPv4_DST_ADDR	PKTS	BYTES
172.22.4.102	1	52
172.22.4.32	1	52
172.22.4.62	1	52
172.22.5.149	1	84
172.22.5.150	1	90
172.22.7.255	7	1223
81.37.15.129	87	66728

Tabella 4.23: Dati aggregati per indirizzo IP destinazione

le altre aggregazioni che saranno applicate agli stessi dati forniti nella tabella 4.20 saranno inviati, a meno di situazioni eccezionali, solo su esplicita richiesta del livello superiore.

4.3.2 Informazioni comuni

Per come sono state definite le aggregazioni, i dati, una volta subito il processo di aggregazione, possiedono un insieme di informazioni comuni. Un esempio di tale situazione è il seguente: supponiamo di voler aggregare secondo il seguente schema:

- Aggregare i dati, separandoli solo per la loro porta di destinazione, rispetto all'indirizzo sorgente IP 192.168.0.0/23.

I dati di cui si dispone sono i seguenti:

IPv4_SRC_DST	IPv4_DST_ADDR	SRC_PORT	DST_PORT	PKTS
192.168.0.201	194.95.211.10	64235	80	10
192.168.0.202	194.95.211.11	64236	80	10
192.168.0.203	194.95.211.12	64237	110	10
168.192.0.204	194.95.211.13	64238	80	10
168.195.0.205	194.95.211.14	64239	80	10

L'applicazione della schema di aggregazione genera il seguente insieme di dati:

IPv4_SRC_DST	IPv4_DST_ADDR	SRC_PORT	DST_PORT	PKTS
192.168.0.0/23	*	*	80,110	20
168.192.0.204	194.95.211.13	64238	80	10
168.195.0.205	194.95.211.14	64239	80	10

Il modo più semplice di inviare tali dati è quello di inviarli all'interno di ogni pacchetto di NetFlow, durante la fase di export. Tuttavia, in questo modo, si rischia di trasmettere una grossa quantità di dati ripetuti, con un inutile spreco di risorse e di tempo.

All'utente viene però messa a disposizione la possibilità di scegliere tra due modi differenti di invio di tali informazioni comuni. In entrambi i casi i dati comuni sono inviati una sola volta in un pacchetto separato invece di ripeterle in ogni export.

La prima alternativa proposta consiste nell'inviare prima tutti i dati che non sono stati coinvolti nell'aggregazione. Successivamente si inviano i dati coinvolti nelle aggregazioni, considerando però solo le informazioni discriminati. Infine si inviano

tutte le proprietà comuni, avendo sempre cura di mantenere i giusti riferimenti tra le informazioni inviate (es. gli id dei template di NetFlow). Dall'esempio fatto sopra, si inviano in un pacchetto NetFlow prima le informazioni comuni come descritto nella tabella 4.24.

IPv4_SRC_DST	IPv4_DST_ADDR	SRC_PORT	DST_PORT	PKTS
168.192.0.204	194.95.211.13	64238	80	10
168.195.0.205	194.95.211.14	64239	80	10

Tabella 4.24: Informazioni discriminanti

Successivamente, in un altro pacchetto le informazioni non comuni, come descritto nella tabella 4.25.

DST_PORT	PKTS
80	20
110	10

Tabella 4.25: Informazioni non comuni

In un altro pacchetto ancora le informazioni comuni dei flussi aggregati, come mostrato nella tabella 4.26.

IPv4_SRC_ADDR
192.168.0.0/23

Tabella 4.26: Informazioni comuni

Naturalmente l'ultimo pacchetto inviato avrà un riferimento all'id del template el pacchetto contente le informazioni comuni.

La seconda alternativa è simile alla prima, ma in questo caso è il pacchetto con le informazioni discriminanti che mantiene il riferimento a quello contenente le informazioni comuni.

Inviare ogni volta tutte le informazioni comuni, non è la migliore strada, anche se appare la più semplice. Infatti in questo caso si incorre in un grosso overhead nell'inviare gli stessi valori costanti in continuazione.

Le due alternative proposte sono più promettenti, in quando permettono di non ripetere le informazioni. Tuttavia possono nascere dei problemi se non si sono ancora ricevuti tutti i template necessari: potrebbe non essere possibile infatti capire quali sono i campi associati tra i vari template.

```
<limitazione>
    <id_blocco_superiore>...</id_blocco_superiore>
    <limite_dati>....<limite_dati>
</limitazione>
```

Figura 4.18: Esempio limitazione di invio dei dati

4.3.3 Limitazioni di invio

Al fine di evitare possibili casi di congestione dei blocchi presenti nei livelli, il sistema limita l'invio dei dati dai livelli inferiori ai livelli superiori, limitando la quantità di dati che è possibile inviare ogni minuto.

La quantità di dati che è possibile inviare varia, infatti, tra un valore minimo ed un valore massimo, seguendo anche la disponibilità di dati di cui si dispone all'interno del blocco. La quantità minima corrisponde naturalmente a non inviare nessun dato, situazione che corrisponde alla mancanza di dati da inviare. La quantità massima può essere definita dall'utente, anche se nel sistema viene definito un insieme di valori di default, uno per ogni livello[11].

Una volta che il sistema viene mandato in esecuzione, il cambiamento della limitazione nell'invio dei dati viene affidato ai livelli stessi. Ogni livello, infatti, invia al livello sottostante un valore che indica quanto al massimo il blocco può ricevere. Tale valore viene stabilito all'interno di un blocco, in base alla quantità di spazio disponibile sui suoi dischi e sul suo database: se lo spazio disponibile risulta piccolo, si richiede una quantità minima di dati. Al contrario, se la disponibilità di spazio è maggiore, il blocco può richiedere una quantità maggiore di dati.

La quantità richiesta è definita in funzione delle righe presenti nel db. In questo modo un blocco conosce ogni minuto il numero di righe delle tabelle del database di cui può fare il dump e che può inviare al livello superiore.

La segnalazione del cambiamento della quantità di dati che è possibile inviare provoca il cambiamento di uno dei parametri di configurazione dell'entità Analyzer, ma non ne provoca l'arresto per il cambiamento di configurazione. Il formato con cui viene scambiata tale informazione continua ad essere mediante un file di tipo xml. La figura 4.18 mostra un esempio.

Il significato dei campi presente nella figura 4.18 è il seguente:

[11] I blocchi di uno stesso livello condividono lo stesso valore.

- <**id_blocco_superiore**>: identifica in modo univoco il blocco del livello superiore che invia tale limitazione. L'id unico è necessario, in quanto un livello inferiore può inviare i dati a più blocchi del livello superiore: ognuno di questi blocchi può a sua volta inviare una propria limitazione. In questo modo il blocco del livello inferiore che riceve tale informazione è in grado di distinguere a chi appartiene il valore ricevuto e può gestirlo correttamente.
- <**limite_dati**>: è il valore che indica quanti dati il blocco del livello inferiore può inviare al blocco bel livello superiore identificato dall'id inviato.

Capitolo 5

Scenari di applicazione

Sommario

In questo capitolo viene proposto uno scenario di applicazione del sistema descritto, analizzando in dettaglio i compiti di ciascuna delle entità introdotte nell'architettura. Verrà inoltre presentato un esempio dettagliato del funzionamento del sistema.

5.1 Definizione di uno scenario

In questa tesi è stato descritto un sistema che permette di monitorare il comportamento di una rete. Può quindi essere usato da quelle organizzazioni, che desiderano monitorare l'uso della loro rete sia in termini di connettività che in termini di volume di traffico generato.

Sostanzialmente il sistema può essere usato da tutte quelle organizzazioni che hanno il bisogno di una visione gerarchica della propria rete. Esistono molti esempi di reti con questa caratteristica, per esempio le reti di interconnessione delle banche, quella della poste italiane o quelle delle università e così via.

Un banca, per esempio, vuole avere una visone di quello che avviene nella proprie filiali sparse sul territorio. In questo caso abbiamo il livello base costituito dalle singole filiali, i livelli superiori dai raggruppamenti reali o meno, della filiali, fino ad ottenere un ultimo livello da cui controllare il funzionamento di tutta la rete sottostante.

Nel caso delle poste italiane, lo scenario che si delinea è molto simile. Il livello base del sistema sarebbe anche in questo caso rappresentato dalle singole filiali. I

livelli superiori vengono invece ottenuti dai raggruppamenti di queste filiali per zone, provincie, regione, etc.

Molto simile, anche se in scala più ridotta, risulta lo scenario per un'università.

Questi sono alcuni esempi dei possibili casi in cui si può applicare il sistema discusso all'interno della tesi. Per semplicità e maggiore chiarezza di seguito verrà presentato il caso in cui il sistema sia applicato ad un'università.

5.2 Applicazione del sistema

Schematicamente possiamo pensare una rete universitaria come divisa in aule, laboratori, dipartimenti e facoltà. Quindi anche in questo caso abbiamo una visione gerarchica, come mostrata in figura 5.1 .

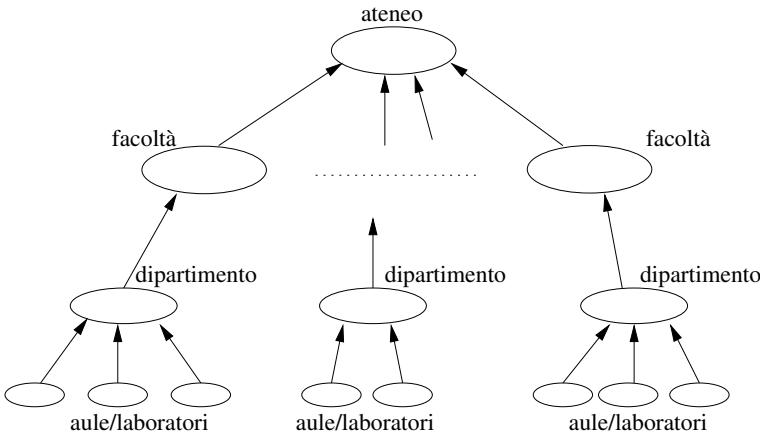

Figura 5.1: Scenario di applicazione

Per poter applicare il sistema occorre individuare quali saranno i livelli base del sistema.

All'interno di questa tesi un blocco del livello base dell'architettura è stato definito come un insieme di sonde che catturano il traffico e lo inviano ad una entità che fa da collector. Quindi i blocchi del livello base saranno posizionati là dove verrà realmente generato il traffico. Nel caso di una rete universitaria come quella presa in considerazione, questo corrisponde a posizionare i blocchi che andranno a costituire livello base, per esempio, nelle singole aule, laboratori e uffici. Si potrebbe pensare che ogni singolo computer connesso alla rete avrà una sonda che catturerà il traffico, ma che le sonde saranno inserite in punti interessanti della rete da monitorare: per esempio basterà posizionare la sonde su tutti quei device che fanno da interfaccia

da/verso l'esterno. Nella prima soluzione infatti si avrebbe lo svantaggio di avere una grossa quantità di dati da gestire con un costo elevato.

Le informazioni catturate in questo modo saranno le informazioni di base sulle quali effettuare la fase di analisi e saranno inviate ai livelli superiori.

I livelli superiori, in questo scenario, rispecchiano la gerarchia della rete che già esiste: questo significa che per esempio un primo livello superiori potrebbe essere quello che raggruppa le aule, i laboratori e gli uffici per dipartimento. Salendo nella gerarchia un altro livello sarà quello che raggruppa i vari dipartimenti per facoltà e infine per ateneo.

5.3 Scenari tipici

Vediamo ora qual è il comportamento del sistema, applicato allo scenario, in alcune situazioni tipiche, in modo da avere una comprensione di quello che è il comportamento reale del sistema.

5.3.1 Propagazione dei dati

Per quanto discusso all'interno della tesi, le informazioni che sono state raccolte ai livelli superiori, saranno inviate ai livelli superiori per poter essere analizzate. Vediamo come si comporta esattamente il sistema applicato allo scenario. Consideriamo il caso mostrato in figura 5.2, in cui abbiamo un insieme di aule/laboratori indicate con le lettere A,B...N. Tali aule/laboratori, fanno parte di un dipartimento X. Il dipartimento, insieme ad altri, farà parte della facoltà W.

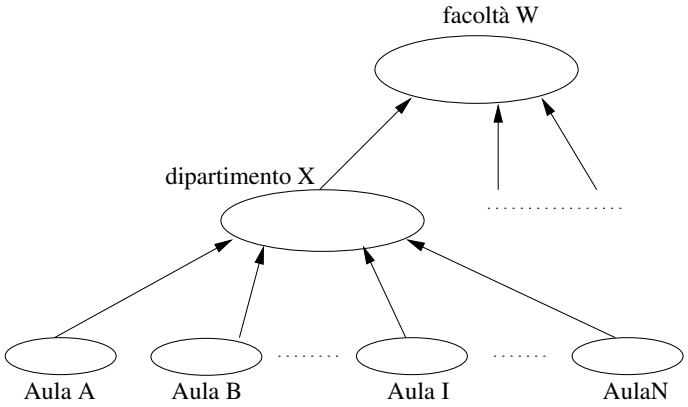

Figura 5.2: Scenario propagazione dati

Le informazioni raccolte nel livello base saranno esportate, dopo un prima fase di filtraggio, al livello superiore costituito del dipartimento X. Le informazioni che sono qui ricevute saranno filtrate e analizzate secondo gli schemi di aggregazioni presenti nel blocco. Per esempio, in questo dipartimento i dati saranno aggregati per indirizzi IP sorgente, per indirizzo IP destinazione e per tipo di protocollo.

Dopo la fase di aggregazione, i dati, che sono ora all'interno del database, saranno inviati al livello superiore, che permette di raggruppare per facoltà. quindi quello che questo blocco riceve saranno delle informazioni riassunte dei quelle catturate.

Tale invio dei dati in forma più ridotta rispetto a quello che si cattura nei livelli base permette di evitare di inviare una grande di dati. L'invio delle informazioni in forma non riassunta, infatti comporta lo spostamento di una grande quantità di dati che oltre a riempire velocemente i supporti di memorizzazione occupa in modo non necessario parte della bande di invio, provocando un degrado delle performance della rete stessa.

5.3.2 Propagazione schemi di aggregazione differenti

La propagazione di schemi di aggregazione differenti, all'interno di questo tipo di scenario, può essere utile quando si vuole differenziare tra le situazioni. Per esempio, consideriamo il caso in cui un dipartimento, che permetta di raggruppare le aule, i laboratori e gli atri uffici. In un caso come questo probabilmente, l'amministratore della rete vuole avere delle visioni differenti di quello che avviene all'interno dei tre raggruppamenti. Per poter fare questo basterà semplicemente specificare per ogni blocco degli schemi di aggregazione differenti, seguendo per ogni blocco le esigenze riscontrate.

Per esempio, se l'amministratore vuole capire quali sono risorse, più richieste da parte degli utenti delle aule e di che tipo è il traffico generato negli altri casi, allora potrà decidere di aggregare le informazioni del traffico in base agli indirizzi IP destinazione per le aule. Per gli altri casi invece, aggregherà il traffico in base ai protocolli e alle porte sorgente e destinazione in modo da poter caratterizzare il traffico.

5.3.3 Aggiunta blocchi/livelli

Il meccanismo di aggiunta di blocchi o di interi livelli è stato progettato in modo da soddisfare principalmente due requisiti: l'aggiunta sia di uno nuovo blocco che di un nuovo livello deve essere facile e intuitiva e non deve sconvolgere l'architettura del sistema precedente.

La necessità di soddisfare tali requisiti nasce dopo un'analisi di un possibile scenario di applicazione del sistema: in un sistema già applicato ad una situazione reale, non si vuole che l'aggiunta di un livello o di un blocco comporti grosse modifiche o addirittura comporti l'arresto del sistema esistente, portando quindi costi aggiuntivi a quelli dell'aggiunta.

Nel caso della nostra università scelta come scenario di applicazione, supponiamo di trovarci nel caso in cui vengo costituito un nuovo dipartimento (o che questo dipartimento venga finalmente connesso alla rete dell'ateneo). Indichiamo questo dipartimento come A. Il dipartimento A farà capo ad un insieme di aule, laboratori e uffici: quindi quello che occorrerà inserire è un mini sistema costituito da livelli superiori e inferiori.

La situazione è mostrata in figura 5.3

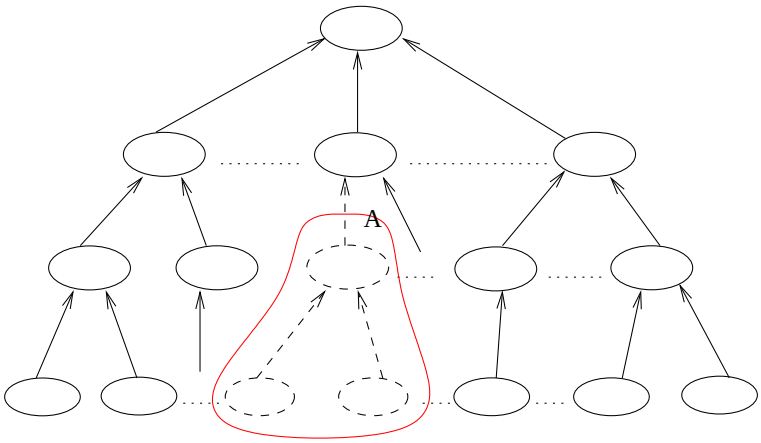

Figura 5.3: Aggiunta di un livello

Il livello base del blocco che occorre aggiungere è ottenuto replicando più volte il blocco che lo costituisce. Naturalmente tali blocchi saranno configurati in modo differente secondo le varie situazioni. Il o i livelli superiori saranno ottenuti (come descritto nel capitolo 4) replicando il blocco del livello superiore.

per poter ottenere giusto comportamento del sistema, occorre che il blocco ag-

giunto sia integrato all'interno del sistema già funzionante. Oltre ai collegamenti fisici necessari, occorre opportunamente configurare le parti aggiunte. Poiché le configurazioni sono solitamente propagate dall'alto verso il basso, la giusta configurazione sarà inviata ai nuovi pezzi con un semplice cambio di configurazione.

5.3.4 Prelievo e Richiesta Dati

Il prelievo e la richiesta dei dati sono due meccanismi adottati all'interno del sistema per cercare di rendere un pò più indipendente il lavoro dei livelli superiori.

La richiesta dei dati può essere utilizzata quando l'amministratore della rete di ateneo vuole avere informazioni aggiuntive a quelle già inviate dal livello inferiore. Tale situazione può presentarsi in caso di problemi rilevati sulla rete: per identificare meglio il problema potrebbero essere necessari dato aggiuntivi rispetto a quelli di base inviati dai livelli inferiori[1]

Il meccanismo di prelievo dei dati può essere usato nel nostro caso per avere i dati prima dell'invio in modo da non dipendere dai livelli inferiori.

5.3.5 Limitazioni di invio

Il meccanismo di limitazione di invio permette di mantenere sotto controllo la quantità di dati che viene spostata tra i livelli, in base alla disponibilità dei singoli livelli o blocchi[2].

Consideriamo, all'interno del nostro ateneo preso come esempio, due diverse facoltà, che indichiamo con A e B. Supponiamo che la facoltà A riceva i dati da sei dipartimenti, mentre la facoltà B riceve i dati solo da tre dipartimenti. La quantità di dati che viene quindi spostata risulterà differente. Per evitare di soffocare i livelli superiori con l'invio dei dati occorre limitare l'invio nei livelli inferiori. come descritta nel capitolo 4, la limitazione avviene in base alla quantità di spazio disponibile nel livello superiore.

Supponiamo, per semplicità, che sia la facoltà A che la facoltà B abbiano le stesse caratteristiche in termini di spazio di memorizzazione. Tuttavia la due facoltà ricevono una quantità di dati differenti. Questo significa, per esempio, che la facoltà A, che riceve da sei dipartimenti, limiterà l'invio di ogni dipartimento a x dati al

[1] Per esempio le aggregazioni avanzate sono un esempio di dati che è possibile richiedere dopo.
[2] Il parametro di invio è configurabile.

minuto, mentre la facoltà B limiterà l'invio ad un valore y, dove tale valore sarà presumibilmente più alto di x.

In questo modo, in ogni istante , le due faco0ltà possono ricevere solo la quantità di dati che è possibile elaborare per loro, cercando di evitare situazioni di overload.

Conclusioni

L'architettura proposta all'interno di questa tesi è un'architettura che presenta diversi vantaggi rispetto allo stato dell'arte attuale.

Per i protocolli utilizzati, quest'architettura si colloca nell'ambiente flow-based. Come discusso infatti all'interno dei capitoli precedenti, l'architettura fa uso del protocollo della Cisco NetFlow, che risulta essere il protocollo maggiormente diffuso e maggiormente supportato per il monitoraggio delle reti.

In particolare, l'uso della versione 9 di tale protocollo ha permesso all'architettura di soddisfare uno dei requisiti che le odierne architetture di questo tipo devono possedere, e cioé la possibilità di personalizzare le informazioni da memorizzare. NetFlow v9 ha infatti la caratteristica di utilizzare i *template* per descrivere i dati, tecnica che permette anche di decidere quali informazioni sui dati raccolti si vogliono memorizzare.

L'architettura proposta fa inoltre fronte al problema della grande quantità di dati che su reti come quelle odierne si possono raccogliere. Le informazioni sul traffico vengono infatti comprese all'interno dei blocchi facendo uso delle tecniche di aggregazione. Aggregare i dati ha permesso infatti di poterli mantenere, in forma compressa e significativa, e di poterli inserire in un database per un successivo ritrovamento.

L'inserzione all'interno di un database ha permesso anche di poter avere una visione sia a breve che lungo termine del comportamento della rete: i dati vengono infatti mantenuti compressi in diversi intervalli temporali in modo da poter mantenere i dati raggruppati per giorni, settimane e mesi.

Trattandosi inoltre di un'architettura distribuita, in cui esistono più punti di raccolta e di analisi delle informazioni del traffico, e gerarchica, viene offerta all'utente la possibilità di avere una visone del comportamento della rete con diversi gradi di dettaglio. Il fatto che l'architettura permette di avere una visione dall'alto verso

il basso della rete, permette di avere una visione che, scendendo nei livelli, abbia diversi gradi di dettaglio. Scendendo nella gerarchia è possibile avere ogni volta una visione per sottoreti sempre più piccole, fin quasi ad arrivare ai singoli punti in cui viene generato il traffico.

Ma un grande vantaggio che viene fornito dall'architettura proposta, è quello di poter personalizzare la fase di analisi dei dati, personalizzando gli schemi di aggregazione che è possibile applicare. Il cambiamento degli schemi di aggregazione nel tempo permette di volta in volta di soddisfare nuove esigenze della fase di analisi. Inoltre, la possibilità di modificare gli schemi tra i diversi livelli e blocchi dell'architettura, fa sì che si possa avere in ogni livello, una visione del comportamento sotto differenti caratteristiche, ritenute importanti dall'utente.

Di seguito viene proposto uno schema riassuntivo delle principali caratteristiche dell'architettura proposta.

- **Replicabilità**: l'elemento base dell'architettura è un blocco al cui interno le entità svolgono un insieme completo di funzioni. L'architettura è costruita replicando più volte il blocco.
- **Personalizzazione**
  - della fase di analisi, personalizzando gli schemi di aggregazione:
  - della memorizzazione dei dati, tramite l'uso del protocollo NetFlow v9;
- **Riduzione dei dati**: utilizzo delle tecniche di aggregazione per la riduzione del volume dei dati.
- **Diversificazione visione**
  - temporale, modificando nel tempo gli schemi di aggregazione;
  - per sottoaree, modificando per sottoaree gli schemi secondo le esigenze riscontrate.
- **Visione temporale a breve e a lungo termine**.

L'architettura proposta permette di monitorare il traffico generato, permette cioé di ottenere delle statistiche di metriche ritenute rilevanti. Questo significa che non è un sistema per la sicurezza delle reti, in quanto non offre nessuno strumento

in questo senso. Sicuramente le statistiche generate possono essere utilizzate per la successiva scoperta di pattern di traffico anomalo. Tuttavia, essendo propriamente un'architettura per il monitoraggio, si è deciso di rimanere in quest'ambito e di tralasciare questa possibile applicazione.

Infine rimane da espandere l'implementazione del sistema.

Appendice A

Lista dei campi per le aggregazioni

La tabella successiva fornisce la liste completa dei campi per le aggregazioni che il sistema è in grado di poter riconoscere. La sintassi di tali campi, per le scelte fatte all'interno di questa tesi durante la fase di specifica e di validazione, rispecchiano fedelmente la sintassi prevista in NetFlow v9 per i campi [7] [8].

A causa della natura unidirezionale del flusso nella definizione utilizzata, all'interno di NetFlow si distingue tra flusso in ingresso e in uscita. In particolare, all'interno di [8], viene inserito un campo, `DIRECTION`, il cui valore indica se il flusso è da considerarsi in ingresso o in uscita. In particolare il valore `1` indica che il flusso deve essere considerato come flusso in uscita, mentre il valore `0` indica che il flusso deve essere considerato come flusso in entrata.

Tipo campo	Descrizione
IN_BYTES	Numero dei byte associati con un flusso IP, considerati come in ingresso
IN_PKTS	Numero dei pacchetti associati con un flusso IP, considerati come in ingresso
OUT_PKTS	Numero dei pacchetti associati con un flusso IP, considerati come in uscita
OUT_BYTES	Numero dei byte associati con un flusso IP, considerati come in uscita
FLOWS	Numero dei flussi che sono aggregati
PROTOCOL	Tipo di protocollo IP
TOS	Type of Service
TCP_FLAG	Indicatore cumulativo di tutti i flag TCP visti fino ad ora
L4_SRC_PORT	Numero per la porta sorgente per i protocolli TCP/UDP etc.
IPV4_SRC_ADDR	Indirizzo IP v4 sorgente
SRC_MASK	Numero di bit contigui nell'indirizzo sorgente che individuano la maschera per la sottorete sorgente
L4_DST_PORT	Numero per la porta destinazione per protocolli TCP/UDP etc.
IPV4_DST_ADDR	Indirizzo IP v4 destinazione
DST_MASK	Numero di bit contigui nell'indirizzo destinazione che individuano la maschera per la sottorete destinazione
SRC_AS	Autonomous System sorgente
DST_AS	Autonomous System destinazione
MIN_PKT_LNGTH	Lunghezza minima del pacchetto IP del flusso
MAX_PKT_LNGTH	Lunghezza massima del pacchetto IP del flusso
TOTAL_BYTES_EXP	Numero dei byte totali esportati
TOTAL_PKTS_EXP	Numero dei pacchetti totali esportati
TOTAL_FLOWS_EXP	Numero dei flussi totali esportati
MIN_TTL	Minimo TTL dei pacchetti in ingresso per un flusso
MAX_TTL	Massimo TTL dei pacchetti in uscita per un flusso
ICMP_TYPE	Tipo del pacchetto per l'Internet Control Message Protocol (ICMP)
LAST_SWITCHED	sysUptime in msec dell'ultimo pacchetto inoltrato
FIRST_SWITCHED	sysUptime in msec del primo pacchetto inoltrato

Appendice B

Tabelle citate negli esempi

All'interno di questa appendice sono presentate della tabelle contenenti i dati che sono stati utilizzati in alcuni degli esempio proposti nella tesi.

IPv4_SRC_ADDR	IPv4_DST_ADDR	PKTS	BYTES	L4_SRC_PORT	L4_DST_PORT	PROTOCOL
80.180.205.119	172.22.6.169	2	92	4662	1640	6
172.183.195.184	172.22.6.169	2	92	4662	1639	6
217.122.233.133	172.22.6.169	2	92	4662	1641	6
217.127.237.146	172.22.6.169	2	92	9000	1637	6
220.240.123.157	172.22.6.169	2	92	4662	1638	6
10.12.241.91	172.22.5.161	9	1277	2191	80	6
10.12.241.91	172.22.5.161	8	2036	2192	80	6
10.12.241.91	172.22.5.161	9	1277	2193	80	6
10.12.241.91	172.22.5.161	8	2036	2194	80	6
10.41.51.174	172.22.5.96	4	349	1521	1568	6
149.156.88.48	172.22.6.169	2	92	4662	1632	6
151.37.16.23	172.22.6.169	6	401	4662	1646	6
151.37.16.30	172.22.6.169	4	357	4662	1645	6
172.22.17.58	172.22.4.102	1	52	0	2048	1
172.22.17.58	172.22.4.32	1	52	0	2048	1
172.22.17.58	172.22.4.62	1	52	0	2048	1
172.22.4.10	172.22.5.150	15	12833	139	2538	6
172.22.4.118	172.22.7.255	4	512	631	631	17
172.22.4.121	172.22.7.255	1	78	137	137	17
172.22.4.18	172.22.7.255	3	324	1076	135	17
172.22.4.21	172.22.31.255	4	564	1025	2071	17
172.22.4.34	172.22.5.149	1	84	137	137	17
172.22.4.34	172.22.5.150	1	90	137	137	17
172.22.4.42	172.22.5.35	2	94	32767	1521	6
172.22.4.43	235.0.0.1	8	3448	6881	6881	17
172.22.4.43	235.0.0.1	8	3448	6881	6881	17
172.22.4.53	172.22.6.169	1	84	0	2048	1
172.22.5.129	172.22.7.255	6	468	137	137	17
172.22.5.134	172.22.7.255	1	237	138	138	17
172.22.5.14	172.22.7.255	1	244	138	138	17
172.22.5.149	172.22.7.255	9	702	137	137	17
172.22.5.149	172.22.7.255	2	500	138	138	17

Tabella B.1: Sequenza dati 1

IPv4_SRC_ADDR	IPv4_DST_ADDR	PKTS	BYTES	L4_SRC_PORT	L4_DST_PORT	PROTOCOL
172.22.5.173	172.22.7.255	1	246	138	138	17
172.22.5.182	172.22.7.255	3	234	137	137	17
172.22.5.199	172.22.7.255	1	229	138	138	17
172.22.5.29	172.22.7.255	1	242	138	138	17
172.22.5.54	244.0.0.22	6	276	0	0	2
172.22.6.169	172.22.4.53	1	84	0	0	1
172.22.6.169	80.180.205.119	2	92	1640	4662	6
172.22.6.169	172.183.195.184	2	92	1639	4662	6
172.22.6.169	217.122.233.133	2	92	1641	4662	6
172.22.6.169	217.127.237.146	2	92	1637	9000	6
172.22.6.169	220.240.123.157	2	92	1638	4662	6
172.22.6.169	149.156.88.48	2	92	1632	4662	6
172.22.6.169	151.37.16.23	8	581	1646	4662	6
172.22.6.169	151.37.16.30	8	505	1645	4662	6
172.22.6.169	217.95.192.164	2	92	1636	9000	6
172.22.6.169	62.219.64.111	2	92	1642	9767	6
172.22.6.169	80.185.249.97	10	5224	4750	4662	6
172.22.6.169	80.185.249.97	9	5184	4750	4662	6
172.22.6.169	81.37.15.129	40	28400	4345	4662	6
172.22.6.169	81.37.15.129	47	38328	4345	4662	6
172.22.6.169	81.53.167.161	2	92	1644	4662	6
172.22.6.169	82.55.200.80	2	92	1633	4662	6
216.94.119.45	172.22.6.169	2	92	4662	1630	6
217.65.192.164	172.22.6.169	2	92	9000	1636	6
62.219.64.111	172.22.6.169	2	92	9767	1642	6
80.185.249.97	172.22.6.169	10	460	4662	4750	6
81.27.15.129	172.22.6.169	35	1610	4662	4345	6
81.27.15.129	172.22.6.169	44	2024	4662	4345	6
81.53.167.161	172.22.6.169	2	92	4662	1644	6
82.55.200.80	172.22.6.169	2	92	4662	1633	6

Tabella B.2: Sequenza dati 2

StartTime	IPv4_SRC_ADDR	PKTS	BYTES
1095941580	80.180.205.119	2	92
1095941580	172.183.195.184	2	92
1095941580	217.122.233.133	2	92
1095941580	217.127.237.146	2	92
1095941580	220.240.123.157	2	92
1095941580	10.12.241.91	34	6626
1095941580	10.41.51.174	4	349
1095941580	149.156.88.48	2	92
1095941580	151.37.16.23	6	401
1095941580	151.37.16.30	4	357
1095941580	172.22.17.58	3	156
1095941580	172.22.4.10	15	12833
1095941580	172.22.4.118	4	512
1095941580	172.22.4.121	1	78
1095941580	172.22.4.18	3	324
1095941580	172.22.4.21	4	564
1095941580	172.22.4.34	2	174
1095941580	172.22.4.42	2	94
1095941580	172.22.4.43	16	6896
1095941580	172.22.4.53	1	84
1095941580	172.22.5.129	6	468
1095941580	172.22.5.134	1	237
1095941580	172.22.5.14	1	244
1095941580	172.22.5.149	11	1202
1095941760	172.22.5.173	1	246
1095941760	172.22.5.182	3	243
1095941760	172.22.5.199	1	229
1095941760	172.22.5.29	1	242
1095941760	172.22.5.54	6	276
1095941760	172.22.6.169	143	79226
1095941760	216.94.119.45	2	92
1095941760	217.95.192.164	2	92
1095941760	62.219.64.111	2	92
1095941760	80.185.249.97	10	460
1095941760	81.37.15.129	79	3634
1095941760	81.53.167.161	2	92
1095941760	82.55.200.80	2	92

Tabella B.3: Contenuto tabella AGG_SRC dopo l'applicazione di tutti gli schemi

StartTime	IPv4_DST_ADDR	PKTS	BYTES
1095941580	172.22.31.255	4	564
1095941580	172.22.4.102	1	52
1095941580	172.22.4.32	1	52
1095941580	172.22.4.62	1	52
1095941580	172.22.5.149	1	84
1095941580	172.22.5.150	16	12923
1095941580	172.22.5.161	34	6626
1095941580	172.22.5.35	2	94
1095941580	172.22.5.96	4	349
1095941580	172.22.6.169	23	1394
1095941580	172.22.7.255	27	3065
1095941580	235.0.0.1	16	6896
1095941760	80.180.205.119	2	92
1095941760	172.183.195.184	2	92
1095941760	217.122.233.133	2	92
1095941760	217.127.237.146	2	92
1095941760	220.240.123.157	2	92
1095941760	149.156.88.48	2	92
1095941760	151.37.16.23	8	92
1095941760	151.37.16.30	8	505
1095941760	172.22.4.53	1	84
1095941760	172.22.6.169	99	4554
1095941760	172.22.7.255	6	951
1095941760	217.95.192.164	2	92
1095941760	224.0.0.22	6	276
1095941760	62.219.64.111	2	92
1095941760	80.185.249.97	19	10408
1095941760	81.37.15.129	87	66728
1095941760	81.53.167.161	2	92
1095941760	82.55.200.80	2	92

Tabella B.4: Contenuto tabella AGG_DST dopo l'applicazione di tutti gli schemi

StratTime	PROTOCOL	PKTS	BYTES
1095941760	1	1	84
1095941760	64	142	79142

Tabella B.5: Contenuto tabella AGG_PROTO_172_22_6_169 dopo l'applicazione di tutti gli schemi

Bibliografia

[1] C.Estan, K.Keys, D.Moore, and G. Varghese. Building a better netflow. In *Proceedings of SIGCOMM 2004*, August 2004.

[2] Cisco Systems,Inc. *http://www.cisco.com.*

[3] Cisco Systems,Inc. *NetFlow Analyzer Installation and User Guide.*

[4] Cisco Systems,Inc. *NetFlow Collector Installation and User Guide.*

[5] Cisco Systems,Inc. *NetFlow Services and Applications. White paper*, 1999.

[6] Cisco Systems,Inc. *NetFlow Services Solutions Guide*, 2004.

[7] Cisco Systems,Inc. *NetFlow version 9 Flow-Record format. White paper*, 2004.

[8] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes. *Cisco Systems NetFlow Services Export V9 rfc 3954.* IETF Network Working Group, October 2004.

[9] Internet Engineering Task Force. *IP Flow Information Export (IPFIX) http://www.ietf.org/html.charters/ipfix-charter.html.*

[10] Internet Engineering Task Force. *Realtime Traffic Flow Mesaurement Working Group http://www.ietf.org/html.charters/rtfm-charter.html.*

[11] Elliotte Rusty Harold. *XML : eXtensible Markup Language.* IDG Books, 1998.

[12] K.C.Claffy, G.C.Polyzos, and H.W.Braun. Application of sampling methodologies to network traffic characterization. In *Proceedings of SIGCOMM*, pages 194–203, 1993.

[13] Tom Kosnar. Notes to flow-based traffic analysis system design. Technical report, CESNET, 2004.

[14] L.Deri. Sistemi di Elaborazione dell'Informazione:Elementi di Gestione di Rete 2003-2004, 2004.

[15] L.Deri and S.Suin. Effective traffic measurement using ntop. *IEEE Communications Magazine*, 38(5):138–145, May 2000.

[16] Simon Leinen. *Fluxoscope a System for Flow-based Accounting.* Charging and Accounting Technology for the Internet (CATI), 2000.

[17] Simon Leinen. *Flow-Based Traffic Analysis at SWITCH*, 2001.

[18] M.Baldi, E.Baralis, and F.Risso. Data mining techiques for effective flow-based analysis of multi-gigabit network traffic. In *12th IEEE International Conference on Software, Telecommunications and Computer Network (SoftComm 2004)*, 2004.

[19] M.Baldi, E.Baralis, and F.Risso. Data mining techiques for effective and scacable multi-gigabit traffic analysis. In *9th IFIP/IEEE International Symposium on Intregated Network Management (IM 2005)*, May 2005.

[20] MySQL. *http://www.mysql.com.*

[21] MySQL. *Technical Reference for Version 4.1.1-alpha.*

[22] N.Brownlee. *SRL: A Language for Describing Traffic Flow Measurement rfc 2723.* IETF Network Working Group, October 1999.

[23] N.Brownlee. Using netramet for production traffic measurement. In *Proceedings of the 2001 IEEE/IFIP Intrenational Symposium on Intregated Network Management*, pages 213–336, May 2001.

[24] N.Brownlee, C.Mills, and G.Ruth. *Traffic Flow Mesaurement Architecture rfc 2722.* IETF Network Working Group, October 1999.

[25] Sampled NetFlow. *http://www.scisco.com/univercd/cc/td/doc/product/software/ios120/120nweft/120limit/120s/120s11/12s_sanf.htm.*

[26] David Plonka. Flowscan. a network traffic flw reporting and visualization tool. *USENIX LISA*, pages 305–317, December 2000.

[27] P.Phall, S.Panchen, and N.McKee. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks rfc 3176.* IETF Network Working Group, September 2001.

[28] Stream Control Transmission Protocol. *http://www.sctp.org.*

[29] Stream Control Transmission Protocol. *rfc.3286*, Maggio 2002.

[30] Switch. *http://www.switch.ch.*

[31] Wladbusser. *Remote Network Monitoring Managemet Infromation Base rfc 2819.* IETF Network Working Group, May 2000.

[32] W.Stalling. *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2.* Addison Wesley, third edition, 1999.

Elenco delle figure

Glossario

AS Autonomous System, 62

DTD Data Type Definition, 54

FTP File Transfer Protocol, 18

HTTP HyperText Transport Protocol, 18

ICMP Internet Control Message Protocol, 45
IMAP4 Internet Messaging Access Protocol 4, 18
IP Internet Protocol, 39
IPFIX IP Flow Information eXport, 27

LAN Local Area Network, 7

MPLS Multiprotocol Label Switching, 32

OSI Open System Interconnect, 13

POP3 PostOffice Protocol 3, 18

RMON Remote MONitoring, 7

SCTP Stream Control Transmission Protocol, 87
SNMP Simple Network Management Protocol, 34
SNMP Simple Network Monitoring, 7
SQL Structured Query Language, 69

TCP Transmission Control Protocol, 40

UDP User Datagram Protocol, 45
URL , 40

XML eXtensible Markup Language, 54