I[redacted]

[redacted]e.

[redacted], i[redacted]

[redacted]

[redacted]

[redacted]**to**.

[redacted]

[redacted]

- [redacted]
- [redacted]

[redacted]

U[redacted]

G[redacted]

[redacted]

- [redacted]
- [redacted]
- [redacted]
- [redacted]

[redacted]

[redacted]

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

[redacted]

[redacted]ta).

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]OID.

[redacted]

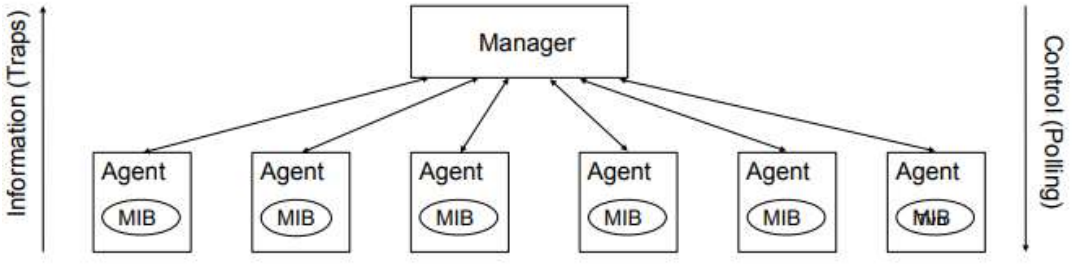
[redacted]

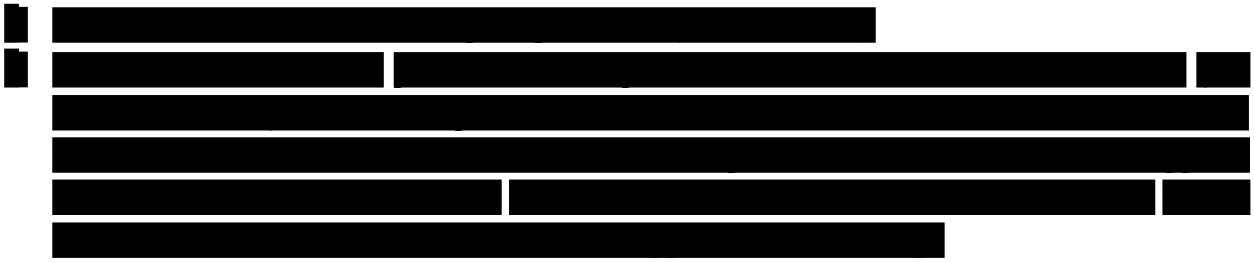
- [redacted]
- [redacted]
- [redacted]
- [redacted]

[redacted]

- [redacted]
- [redacted]

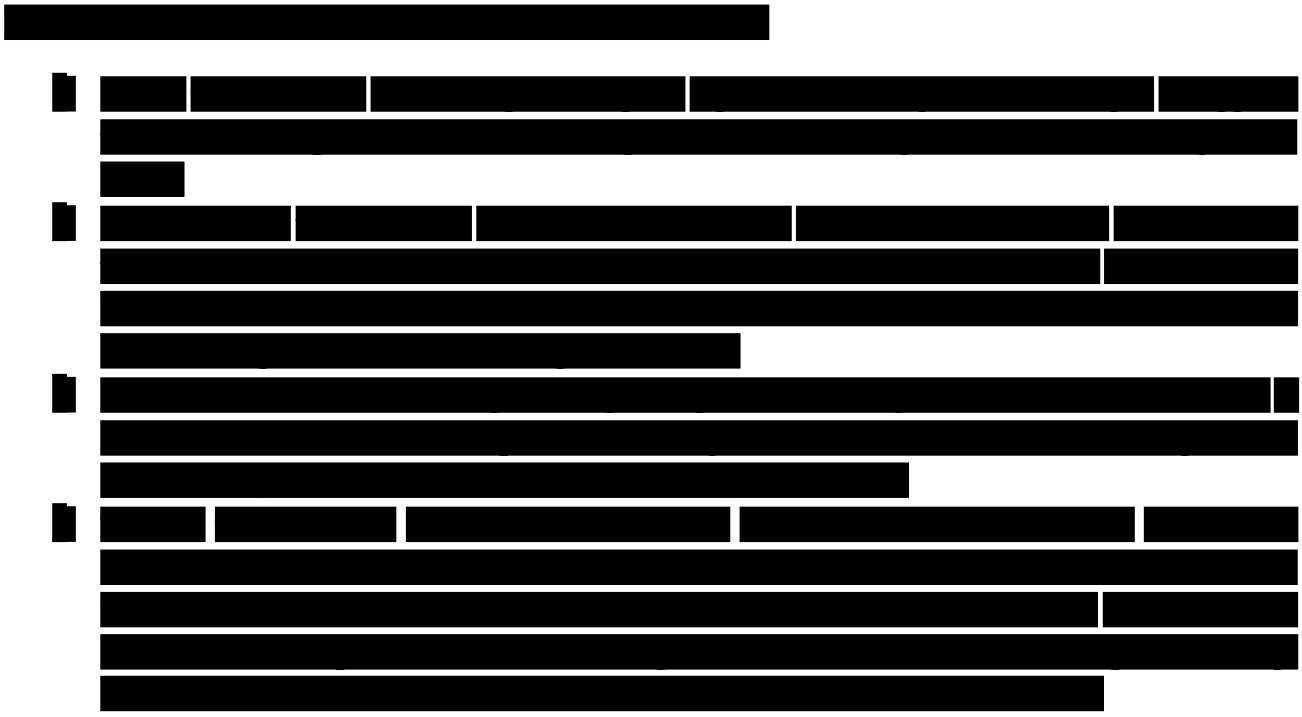
Trap Directed Polling





Le **informazioni scambiate tra agent e manager si mantengono nel MIB**, rappresentato come un file testuale con alcune macro:

- **Module identity**: **prima parte**, **unica, per ogni MIB**, da informazioni sulla versione Ethernet in uso, la tipologia di oggetti che compongono l'agent e la versione del MIB in uso. Per garantire la compatibilità tra MIB di versioni diverse, gli oggetti non vengono eliminati ma marcati con dei tag appositi.
- **Object identity**: Gli **oggetti intermedi** per percorrere l'ISO Registration Tree **si definiscono con questa macro,** contenente uno stato ed una descrizione per gli utenti.
- **Object type**: Dopo aver creato **un oggetto**, esso **viene definito con questa macro** ed un nome simbolico con iniziale minuscola. Ogni oggetto ha una **sintassi**, uno **stato**, una **descrizione** ed un **insieme di operazioni consentite**
- **Notification type**: Per le **notifiche** di cambio di stato si utilizza questa **macro.** L'invio della notifica avviene tramite UDP per motivi di semplicità e robustezza. In questa macro troviamo **uno o più oggetti necessari al manager** per fare una diagnostica, lo **stato** ed una **descrizione in cui si specificano gli OID o la causa della notifica**



Un pacchetto SNMP è formato dalla versione SNMP, dalla community e da un campo PDU (Protocol Data Unit): per quanto riguarda le **operazioni get, get_next e set**, sia il **campo PDU** sia il **pacchetto di risposta** sono formati dagli **stessi campi**. Le operazioni di **get_next e set** si dicono **atomiche**, perché se si eseguono passando loro più OID come argomento, è come se acquisissero una lock e la rilasciassero alla fine di tutto. Per la **trap**, il **PDU del pacchetto** contenente la richiesta è leggermente **diverso**, poiché **ci sono campi in più**, come **ad esempio l'indirizzo IP nativo dell'agent**: è importante che ci sia questo campo, poiché l'agent potrebbe trovarsi dietro un NAT e in quel caso l'IP pubblico sarebbe diverso dall'IP vero e proprio del device. Il manager quindi dovrà conoscere l'indirizzo IP nativo per poter effettuare interventi mirati a seguito di eventi che l'agent ha notificato.

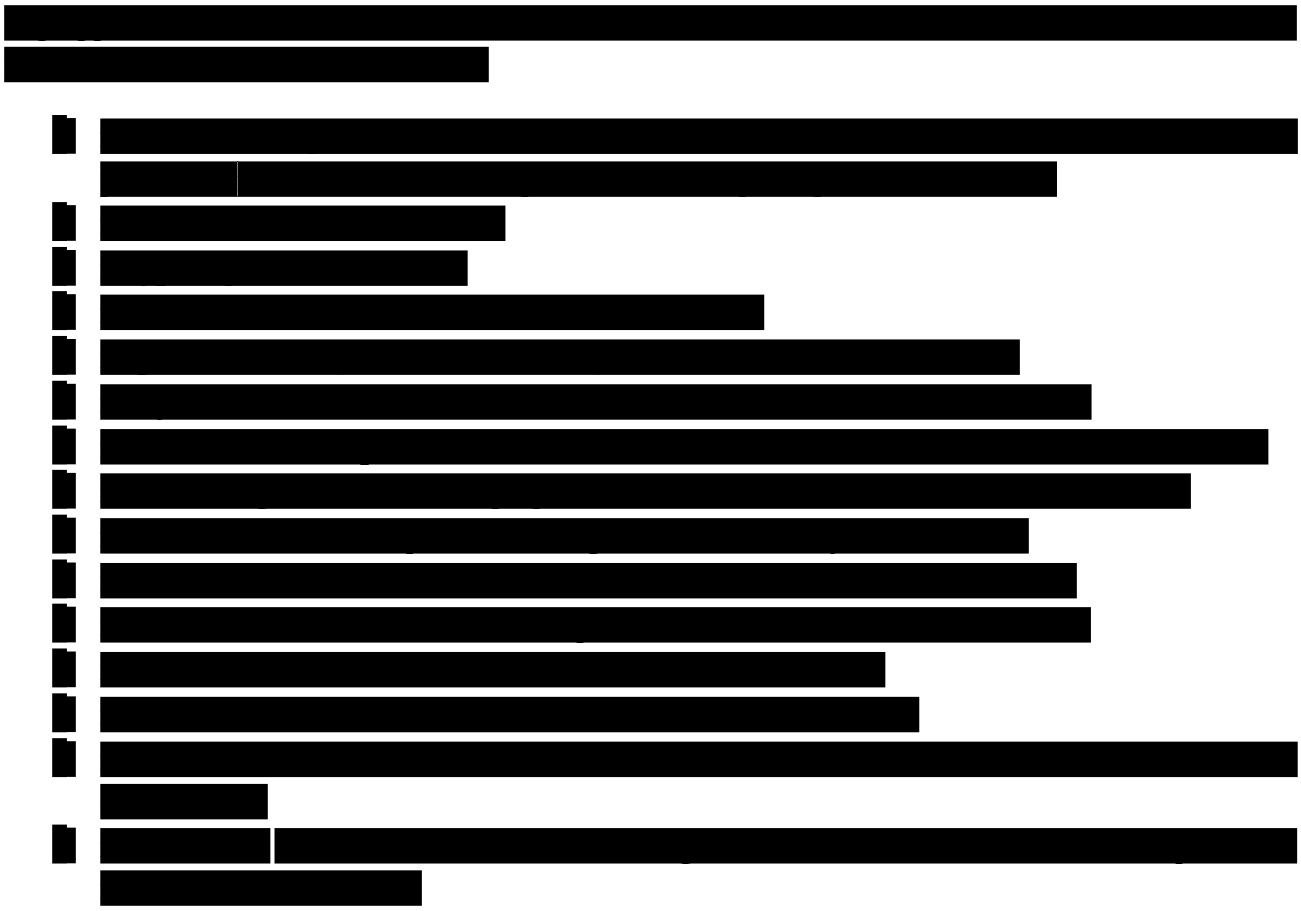
MIB-II E GRUPPI

MIB-II definisce gli object types per IP, ICMP, UDP, TCP, etc… I suoi **obiettivi** consistono nel definire la **gestione dei protocolli di rete**, **informazioni base sugli errori**, pochi oggetti di controllo, **no informazioni ridondanti** e **non deve interferire con la normale attività di rete**. Il MIB-II sta alla base del monitoraggio di rete, contiene circa 170 oggetti, è diviso in gruppi ed è presente su ogni macchina.

Il gruppo "**system**" contiene i seguenti campi:

- **sysDescr:** breve descrizione del sistema
- **sysObjectID:** modello dell'agent e informazioni sul costruttore
- **sysUptime:** indica da quanto è attivo l'agent, quindi fornisce un'idea se l'agent è stato riavviato o meno
- **sysContact:** contatto del gestore dell'agent
- **sysName:** nome del device

- **sysLocation:** locazione fisica del device
- **sysServices:** bitmap che indica quali livelli dello stack TCP/IP implementa l'agent



Un altro MIB importante è il **Bridge MIB**, utile per **controllare gli stati e le interfacce degli switch di livello 2 e 3**. È quasi complementare al MIB-II, poiché **fornisce informazioni sugli host connessi alle porte dello switch** esaminato, **tra cui il loro indirizzo MAC**. Con l'associazione <MAC, porta> **è possibile ricostruire una sorta di topologia della rete** e conoscere la locazione fisica di un dato host

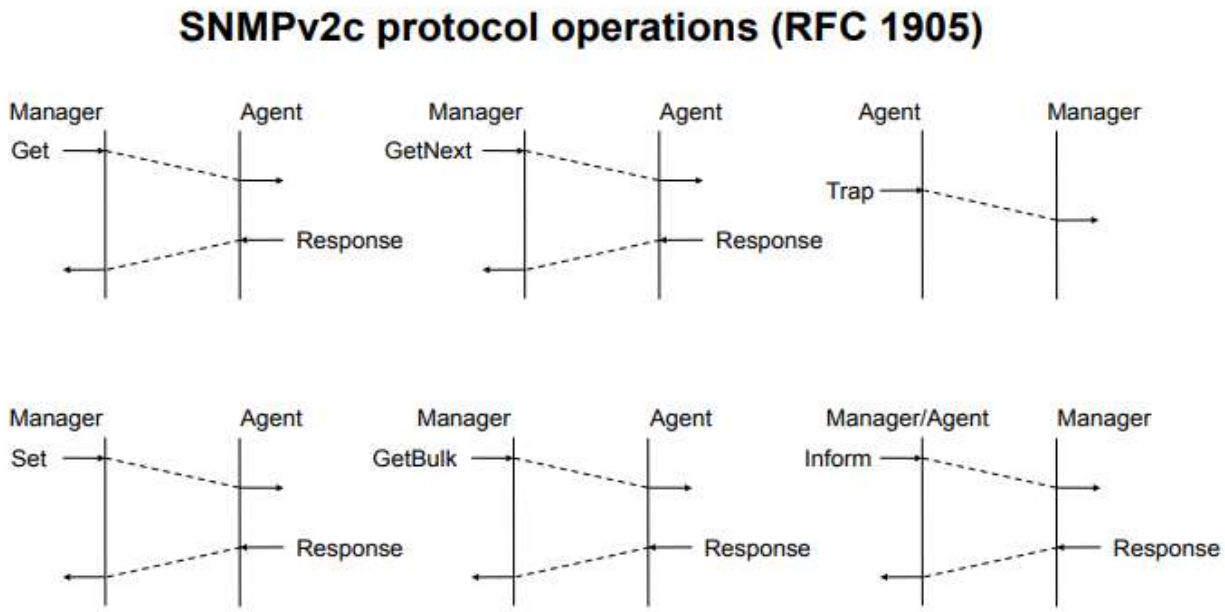
SNMPv2

La **versione 2 di SNMP**, detta appunto SNMPv2, è la più semplice versione che **ha risolto alcuni problemi di SNMPv1** e ha reso efficienti altre operazioni. Nella versione 2 sono state introdotte due **nuove primitive**, i **contatori a 64 bit**, **eccezioni più specifiche** ed espressive, tipi di dati nuovi, etc…

Una primitiva introdotta è la **GET_BULK**: si tratta di una richiesta che va **dal manager all'agent** e prevede una risposta. Serve per rendere più efficiente la get, è una sorta di **unione tra get e get_next**. Oltre a prendere come argomento degli OID, ha anche un **parametro "no repeaters"** che indica su quanti degli OID passati va fatta la get, mentre sui restanti viene fatta la get_next fino ad un limite **"max-repetitions"**. Il **vantaggio** di questa nuova primitiva è che **si riduce nettamente il**

n° di messaggi scambiati. Di **contro** però **se le richieste sono troppe, la risposta potrebbe eccedere la dimensione massima del pacchetto SNMP** e quindi non verrebbe trasmessa

L'altra primitiva introdotta è la **INFORM**, simile alla **trap** ma **confermata dal manager**. Inoltre, questa primitiva **non viene inviata esclusivamente dall'agent**, ma può accadere che in reti complesse con una gerarchia ben delineata ci siano più manager a vari livelli, quindi **può anche essere che la invii un manager ad un altro di livello superiore**.



[redacted]

[redacted].

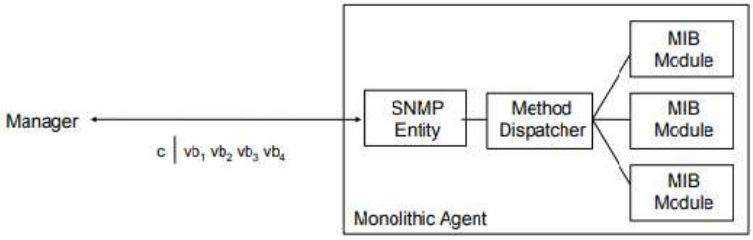
[redacted]

[redacted]ta.

[redacted]

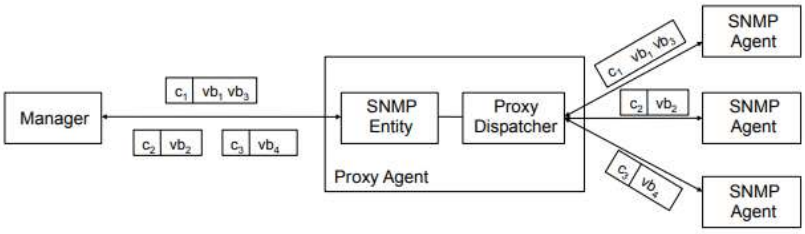
- [redacted]

Monolithic Agents



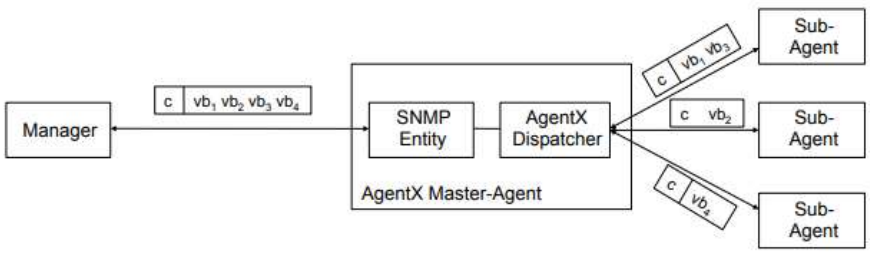
- [redacted]

Proxy-Agents

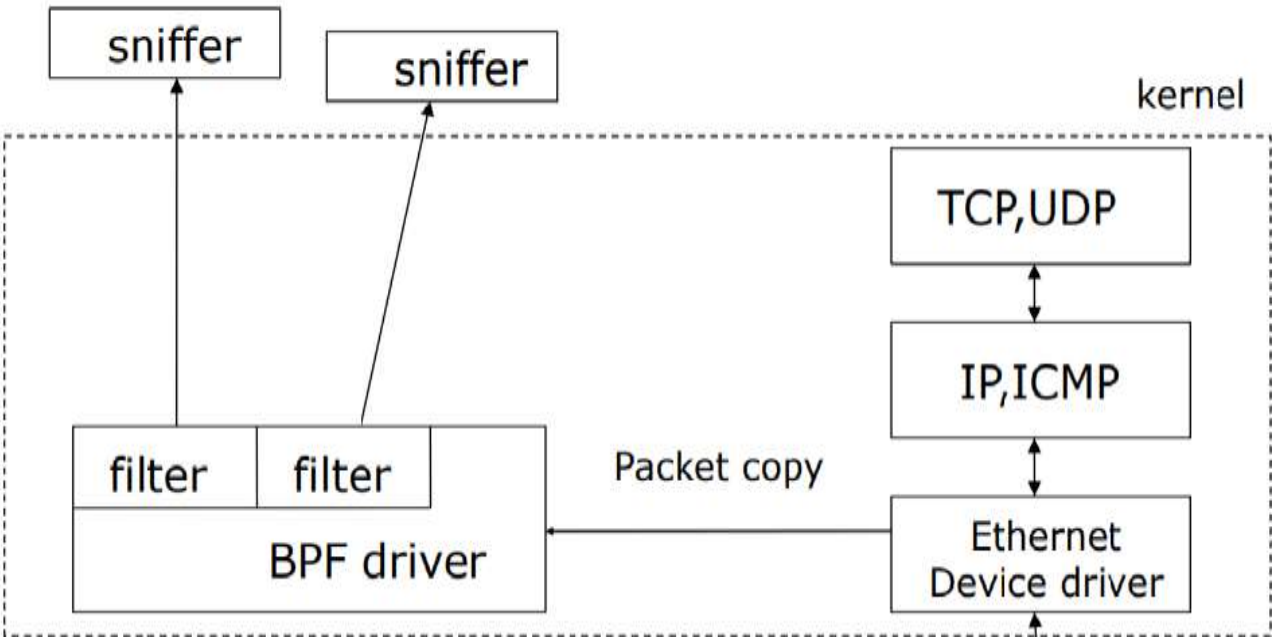


- [redacted]

Extensible Agents



[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

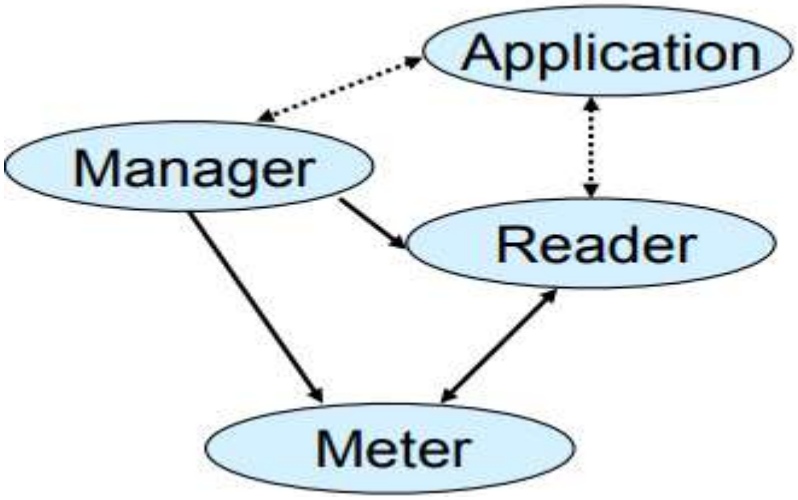
- [redacted]
- [redacted]
- [redacted]

[redacted]

[redacted]e[redacted] S[redacted]

[redacted]

[redacted]



- **T**[redacted]

[redacted]

[redacted] I[redacted]. [redacted] Due flussi monodirezionali opposti sono un unico flusso bidirezionale, su cui ho alcune informazioni aggiuntive. Mantenere flussi separati aumenta la granularità delle informazioni ma anche il carico di lavoro della CPU e del device per il monitoraggio.

Il **probe** presente sul router **non ha capacità di memorizzazione**, quindi un flusso quando termina o resta inattivo viene esportato. I **flussi** sono **esportati verso un collector**, che ha il compito di **assemblare ed interpretare i flussi ricevuti** al fine di **stilare un report** utile al gestore della rete. Generalmente, **in reti di dimensioni elevate** con molti router e probe, quello che si fa è **mantenere i flussi esportati in un collector locale** per evitare che le informazioni viaggino in rete o siano mantenute tutte in un collector centrale. Inoltre, **per evitare la duplicazione del traffico**, nelle reti solitamente **si posizionano i probe sui router di confine e su di essi si monitora solo il traffico in entrata e fino al livello 3.**

Negli anni sono state realizzate diverse versioni di NetFlow, ognuna con il proprio formato di pacchetto. Ogni pacchetto inoltrato dal router viene analizzato secondo i criteri stabiliti e si va a controllare se nella cache è già presente un flusso per quelle particolari caratteristiche. In base ai criteri scelti può cambiare il n° di flussi che si vanno a generare. Un **flusso termina** quando si trova il **flag di fine sessione**, **dura troppo**, **resta inattivo troppo a lungo** o si è **raggiunto la capienza massima della cache**. Il software all'interno del router andrà ad esaminare la cache per trovare i flussi da esportare verso il collector. Un flusso lungo potrà essere suddiviso in tanti sotto-flussi che verranno riassemblati dal collector. Il motivo per cui i flussi hanno una durata massima è dovuto al fatto che **non si può trasferire il flusso alla fine della sessione, ma si rende necessario fornire aggiornamenti continui**. Infatti, il collector necessita di dati sempre aggiornati da mostrare alle applicazioni a lui collegate. **L'esportazione dei dati dipende dalla configurazione del probe** e si basa su un **protocollo connectionless** per essere veloce, leggera ed indipendente da altre entità coinvolte nella comunicazione.

In genere, **un probe al suo interno ha un agent SNMP** e vale che il **traffico SNMP è maggiore o uguale al traffico NetFlow**. Questo perché **SNMP tratta informazioni di livello 2**, quindi **considera anche pacchetti multicast e broadcast** che non escono dalla rete **ed il payload di livello 2**. In **NetFlow** si considera il **traffico a livello 3** e si esamina solo il traffico **che transita su un router**, ossia che viene inoltrato **da un'interfaccia di input ad una di output**.

NETFLOW v5 PACKET FORMAT

Nei pacchetti di ogni versione ci sono campi comuni e campi specifici per ciascuna versione. In un **pacchetto di NetFlow** v5 si ha un **header ed un insieme di flussi esportabili, detti flow records** (nella v5 max. 30). L'**header di un pacchetto NetFlow** contiene

- versione corrente di NetFlow
- n° di flussi del PDU
- sysuptime del router
- ora in cui i flussi sono stati esportati
- numero di sequenza utile al collector per capire se ha perso flussi precedenti
- engine_type e engine_id

Il **contenuto di un flow record** è invece formato da:

- indirizzi IP sorgente e destinazione
- next hop
- id delle interfacce di input e output del router
- n° di pacchetti e bytes inviati
- tempo del primo e dell'ultimo pacchetto del flusso
- porte sorgente e destinazione (se è un protocollo senza porte si mette 0)
- or dei flag TCP
- protocollo di livello di trasporto
- AS sorgente e AS destinazione
- netmask dell'IP sorgente e destinazione

NETFLOW v9

I **formati fissi dei pacchetti** furono definiti in **NetFlow dalla v1 alla v8**, in ogni versione sono state introdotte delle piccole modifiche, ma si è reso necessario ridefinire un formato più libero e flessibile. Inoltre, si volevano introdurre modalità di supporto per il livello 2, vlan, IPv6, etc…

Con **NetFlow v9** si è deciso di introdurre il concetto in cui il **formato del pacchetto** viene **deciso dal probe in base ai dati che deve esportare** in quel momento. Il **formato** è **detto template** e **specifica la codifica dei dati e la loro struttura al collector**, il quale non potrà più basare le proprie operazioni su una struttura fissa standard come nelle precedenti versioni. Il nuovo

pacchetto in NetFlow v9 è quindi **composto da un header, un flow template e i flow records**. **Se il collector** per qualche motivo **non riceve il flow template non sarà in grado di decodificare i dati**, quindi il **template** viene **inviato ogni n secondi oppure ogni n flussi**. Altri due campi sono option template e option record, che forniscono informazioni sulla configurazione del probe.

In **situazioni in cui il traffico da analizzare è elevato** possiamo andare ad operare un fenomeno detto **sampling**. Esso è utile perché **permette di ridurre il carico di lavoro della CPU e l'utilizzo della memoria**, a patto di perdere alcune informazioni. Esistono **2 tipologie** di sampling:

- **Packet sampling rate:** indica quanti pacchetti considero sul totale (es. 1:100). **Serve a ricevere meno pacchetti da analizzare**, potrei però ottenere dati non corretti poiché **i flussi che vado a creare avranno i contatori dei bytes sballati**. Si utilizza in casi limite dove ci basta avere un'approssimazione del traffico suddiviso in flussi e **si usa per alleggerire il carico di lavoro del probe**
- **Flow sampling rate: limita il n° di flussi da esportare**, i pacchetti vengono ricevuti e analizzati tutti. In seguito, **si esportano meno flussi di quelli che abbiamo per alleggerire il carico di lavoro del collector.**

[redacted]**TI**

[redacted]

- **R**[redacted]zi IP ad esse.

[redacted]

[redacted]

[redacted]

FLOW AGGREGATION E FLOW FILTERING

In generale i **flussi raw** sono **utili ma limitanti**, nel senso che non rispondono a domande del tipo quanto traffico è web/mail, quanto traffico va verso il provider x, etc … Per rispondere a queste domande ci interessa effettuare delle **aggregazioni**, che possono essere fatte **sia dal collector che dal probe.**

- **Probe aggregation: si aggregano i flussi secondo criteri stabiliti dall'utente**, andando a migliorare l'utilizzo della memoria per quanto riguarda la Netflow cache. Di contro **il collector non potrà risalire alle informazioni dei singoli flussi** iniziali
- **Collector aggregation: l'aggregazione viene fatta dal collector**, che dovrà svolgere più lavoro ma potrà prima salvarsi i singoli flussi per risalire alle informazioni in essi contenute

Prima di effettuare l'aggregazione dei flussi, si potrà effettuare un **filtraggio** secondo criteri stabiliti dall'utente. Il filtraggio è un **processo diverso dall'aggregazione, poiché esso si applica prima di aggregare i flussi.**

SFLOW

NetFlow per sua natura è **nato per i router e per traffico** che generalmente viene routato, ossia **intra-LAN**. Quindi il suo grosso limite è appunto quello di essere **nato per analizzare il traffico in reti WAN o geografiche**, non si vede il traffico LAN. Si potrebbe quindi mettere una sonda su ciascun device per analizzare il traffico LAN. I **principali problemi** potrebbero essere **l'ammontare di traffico da analizzare**, il **costo dei dispositivi di rete** utilizzati per il monitoraggio, l'**aumentare della velocità delle reti**, **scalabilità**, etc…

Quindi entra in gioco **sFlow**, il quale non parte dal principio di NetFlow per cui va visto tutto il traffico che transita. Si può applicare **sia su reti WAN che su reti LAN** ed i **principi di sFlow** sono i seguenti:

- **Non pretende di essere veloce quanto la rete monitorata**, poiché anche se riuscisse a rilevare tutto il traffico, avrebbe comunque problemi nel gestire i flussi generati
- Effettua il **sampling sui pacchetti**, sapendo che più pacchetti vede più i suoi report saranno precisi. Il sampling **si effettua su tutte le porte del device ed in modo da avere sempre la stessa proporzione ma in modo casuale**, per non rischiare di avere un traffico periodico e di analizzare solo quello
- Se la **velocità della rete** è comunque **alta**, applica un **sampling maggiore**
- La **perdita dei pacchetti** e del traffico deve essere **controllata**, nel senso che va saputo il motivo di quella perdita

L'**architettura** su cui si basa sFlow è composta da un **probe che cattura il traffico**, da un **collector che riceve i pacchetti esportati** ed **un'applicazione che riceverà i report**. Il **probe sFlow** opera secondo principi diversi da quelli del probe NetFlow, poiché **prende un pacchetto da ciascuna porta dello switch su cui si trova in base al sampling rate che si sta applicando**. Una volta presi i pacchetti, **li incapsula e li invia al collector, quindi il probe non analizza il pacchetto ma si limita ad inviarlo al collector**: è questo modo di fare che influisce sulle prestazioni e consente di andare più veloce.

Inoltre**, sFlow consente di risolvere un problema di SNMP**. Esso infatti fornisce solo informazioni quantitative, quindi **gli sviluppatori di sFlow hanno deciso di inserire all'interno dei pacchetti sFlow anche alcune statistiche e contatori del MIB-II riguardanti le interfacce di rete** del collector, detti **counter samples**. In questo modo non c'è più bisogno di andare a fare polling sullo switch. sFlow è descritto nell'RFC3176 e ha varie versioni, la più usata è la v5. Inoltre, essendo in esecuzione sullo switch, nel pacchetto sFlow si può trovare informazioni ulteriori sullo switch stesso o in più rispetto alle statistiche di SNMP.

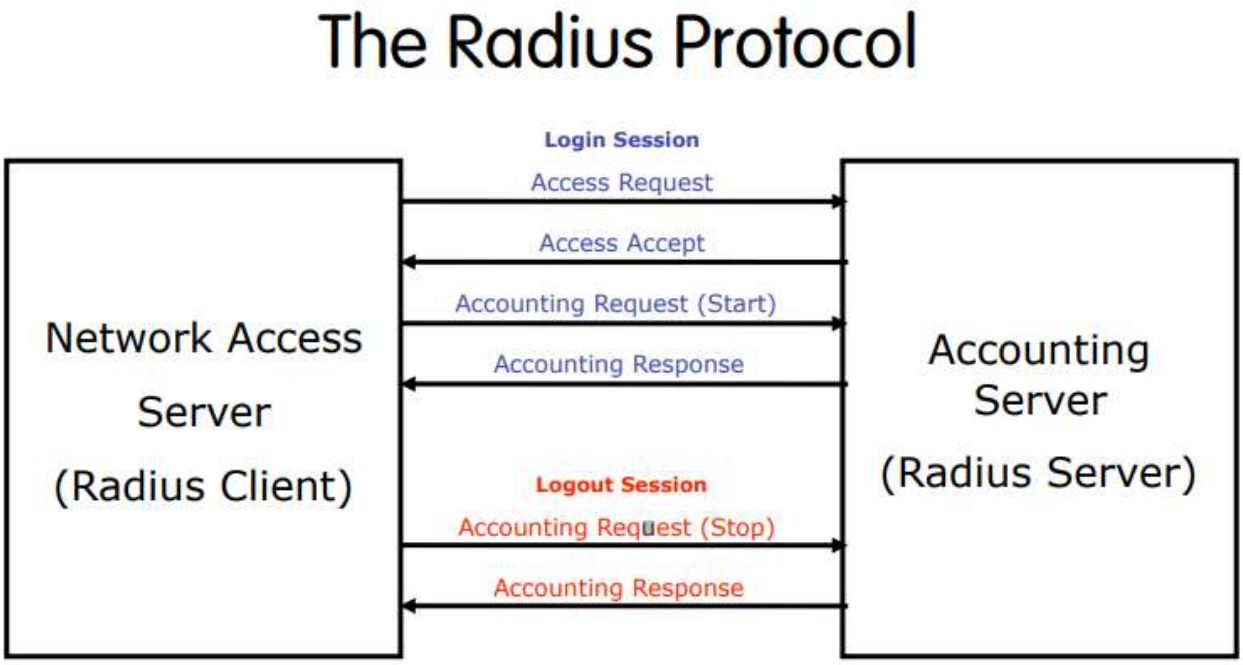
Quindi **in NetFlow un flusso è generalmente un insieme di pacchetti con stessa quintupla**, mentre **in sFlow un flusso è solamente un pacchetto, che viene preso dal probe ed inviato al collector senza essere analizzato**. Un pacchetto sFlow è composto da:

- Versione di sFlow in uso
- Indirizzo IP dell'agent che ha catturato ed esportato il pacchetto (serve perché il traffico potrebbe passare attraverso un NAT, etc…)

- Numero di sequenza per sapere se ho perso qualche pacchetto
- Numero di samples
- Sysuptime dello switch
- Flow sample (pacchetto)

P

[redacted]



[redacted]