



UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Dipartimento di Ingegneria e Scienze dell'Informazione
e Matematica

Tesi di Laurea Triennale in Ingegneria dell'Informazione

Controllo Software Defined di una Rete Ottica Passiva

Anno Accademico 2020-2021

Indice

Elenco delle figure	iii
Elenco delle tabelle	v
1 Introduzione	1
2 Network Slicing	5
2.1 Introduzione al Network Slicing	5
2.2 Tecniche abilitanti lo Slicing	8
2.2.1 Software Defined Networking	8
2.2.2 Network Function Virtualization	10
3 NETCONF-YANG Solution	14
3.1 Considerazioni introduttive sulla NETCONF-YANG solution	14
3.2 NETCONF	14
3.2.1 Alternative a NETCONF	16
3.2.2 Stack Protocolare	16
3.2.3 Esempio di una sessione NETCONF	18
3.2.4 Tool per l'utilizzo di NETCONF	23
3.3 YANG	27
3.3.1 Introduzione	27
3.3.2 Il Linguaggio	28
3.3.3 Descrizione di un modulo YANG e utilizzo del tool pyang	29
4 Rete ottica	38
4.1 La fibra ottica	38
4.1.1 Introduzione	38
4.1.2 Tipi e composizione delle fibre ottiche	38
4.2 Rete ottica	39
4.2.1 Rete Ottica di Accesso: AON vs PON	40
4.2.2 Rete Ottica Passiva (PON)	41
4.2.3 Dynamic Bandwidth Allocation nelle GPON	45
5 Sezione Sperimentale	49
5.1 Introduzione	49
5.2 Architettura NG-PON2	49
5.2.1 Setup Laboratorio	50
5.3 Monitoraggio del traffico della PON senza politiche di controllo	51
5.4 Monitoraggio del traffico nella PON e configurazioni di controllo	56
6 Conclusioni	63

bibliography	64
A Appendix	66
A.1 Codici Python	66
A.2 I tool tshark e pyshark	71
A.2.1 tshark	71
A.2.2 pyshark	72

Elenco delle figure

1.1	Evoluzione delle reti	2
1.2	Scenario del mercato delle auto con il 5G	4
2.1	Ciclo di vita di una Network Slice	6
2.2	Schema delle diverse Slices esaminate associate ai casi d'uso 5G	8
2.3	Networking classico vs Software Defined Networking	9
2.4	Architettura del paradigma SDN	10
2.5	Modello di rete tradizionale VS Modello di rete virtualizzata	11
2.6	Scenario descrittivo di Mobile Edge Computing	12
2.7	Struttura logica di una Network Function Virtualization	13
3.1	Scenario generale NETCONF	15
3.2	Stack Protocollare di NETCONF	17
3.3	Comando di inizio sessione e primo elenco delle capabilities del server	20
3.4	Terzo elenco delle capabilities del server	20
3.5	Hello Message lato client	21
3.6	<rpc> e <rpc-reply> per l'operazione di <get-config>	21
3.7	Visualizzazione di una <rpc-reply> attraverso Tree View	21
3.8	<rpc> e <rpc-reply> per l'operazione di <lock>	22
3.9	<rpc> e <rpc-reply> per l'operazione di <edit-config>	22
3.10	<rpc> e <rpc-reply> per l'operazione di <unlock>	23
3.11	<rpc> e <rpc-reply> per l'operazione di <close-session>	23
3.12	Cattura dell'interfaccia fornita da Advanced NETCONF Explorer	24
3.13	Esempio di <get> con Advanced NETCONF Explorer	25
3.14	Output della <get> utilizzando la libreria Python ncclient	27
3.15	Esempio di validazione del modulo YANG con pyang	32
3.16	Generazione del modello ad albero del modulo YANG con pyang	33
3.17	Comandi per la generazione del modulo YANG in altri formati con pyang	33
3.18	Vaidazione del modello XML sempliceProva.xml	37
4.1	Sezione di una fibra ottica	39
4.2	Esempio delle diverse strategie FTTx	40
4.3	Struttura generale di una PON	42
4.4	Esempio di Time Division Multiplexing	43
4.5	Esempio di Wavelength Division Multiplexing	43
4.6	Scenario della DBA in una GPON	46
4.7	Struttura di una Bandwidth Map	48
5.1	Architettura di una NG-PON2	50
5.2	Setup presente in Laboratorio	51
5.3	Plot dei throughput nel caso di semplice monitoraggio	52
5.4	Cattura effettuata con Wireshark della sessione NETCONF descritta	53
5.5	Output di statistiche.py	54

5.6	Generazione traffico tramite iperf3, nel caso di monitoraggio	56
5.7	Generazione traffico tramite iperf3, nel caso di monitoraggio e controllo. Prima cattura	58
5.8	Generazione traffico tramite iperf3, nel caso di monitoraggio e controllo. Seconda cattura	59
5.9	Plot dei throughput nel caso di monitoraggio e controllo	60
A.1	Risposta al comando Wireshark da pc remoto	71
A.2	Comandi cattura tshark e traffico iperf3	72
A.3	Comandi linux scp e ls	72
A.4	File "flow.pcap" visualizzato con Wireshark	73
A.5	Accesso tramite pyshark alle informazioni di un pacchetto catturato	74
A.6	Comando per eseguirlo e output iniziale di pacchettiCodice.py	74
A.7	Output finale di pacchettiCodice.py	75

Capitolo 1

Introduzione

All'interno del seguente lavoro verranno descritti prima formalmente, seguendo un approccio teorico, e poi in modo pratico, grazie agli esempi forniti, tutti gli ingredienti necessari per poter comprendere l'esempio conclusivo, in cui verranno implementate semplici politiche per il controllo via software di una rete ottica passiva. Il paradigma di rete definita via software, o Software Defined Network (SDN), rivoluziona il modo di monitorare e configurare la rete, offrendo strategie di gran lunga più efficienti rispetto alle reti delle generazioni passate. Dovendo continuamente riconfigurare un numero sempre maggiore di dispositivi, effetto dovuto alla continua espansione delle reti, e con condizioni da rispettare sempre più stringenti, per garantire all'utente le alte prestazioni sottoscritte, diventa impossibile pensare di farlo ricorrendo ai metodi classici, come la Command Line Interface (CLI). Inoltre, grazie alla crescente adozione della tecnologia ottica, come fibre e dispositivi ottici, inizialmente nelle reti core e metro, ed in seguito anche nelle reti di accesso, la rete è in grado di offrire potenzialità elevatissime. Non è difficile intuire che con esse, aumenta il livello di affidabilità che gli operatori delle telecomunicazioni devono garantire ed il grado di complessità che devono fronteggiare. Al contrario, diminuisce il tempo a disposizione per riconfigurare la rete, a cui si richiede reattività e programmabilità, in seguito a guasti o a particolari eventi (si pensi ad esempio alle variazioni di traffico da fronteggiare in occasione di avvenimenti straordinari, come il “Super Bowl” negli Stati Uniti, o la finale di “Champions League” in Europa), ricorrendo ad avanzate strategie di backup e tecniche definite via software per il reindirizzamento del traffico. Negli ultimi decenni si è assistito ad un notevole sviluppo tecnologico, per rispondere alle esigenze di una società che evolve a ritmi incessanti. Il passaggio dall'analogico al digitale, dalle reti di prima generazione 1G a quelle di seconda generazione 2G, nei primi anni '90, ha cambiato per sempre il mondo delle telecomunicazioni. L'impiego di strategie di trasmissione ed elaborazione di segnali digitali, oltre ad offrire una migliore efficienza spettrale e ad introdurre possibilità di cifratura (migliorando la sicurezza delle conversazioni), ha reso possibile l'utilizzo di servizi dati come gli SMS. A partire dai primi anni 2000, con le reti di terza generazione 3G, accanto al trasferimento di dati voce (telefonate digitali), è avvenuto parallelamente anche quello di dati “non-voce”, come ad esempio inviare e ricevere e-mail o eseguire download da internet. La quarta generazione, 4G, ha consentito lo sviluppo su scala mondiale di applicazioni multimediali avanzate e di collegamenti dati con banda elevata. La figura 1.1 mostra l'evoluzione dei dispositivi utilizzati nelle diverse generazioni di rete. Il susseguirsi di questi standard sempre più performanti, oltre a cambiare il modo di concepire e di utilizzare la rete, ha trasformato anche il ruolo rivestito dagli operatori di telecomunicazioni. Per



Figura 1.1: Evoluzione delle reti

decenni il modello di business delle Telco non ha registrato importanti variazioni. In questo periodo, le società di telecomunicazioni hanno offerto prestazioni legate alla connettività, come semplice servizio voce per la clientela standard, e servizio voce e trasmissione dati per gli utenti business. Tuttavia, con la diffusione dei servizi internet sul mercato di massa, e con l'avvento degli Over-The-Top (OTT), il ruolo degli operatori di telecomunicazioni è stato ridimensionato, e sostanzialmente si è trasformato in quello di semplici fornitori di infrastruttura, a scapito della funzione originaria di azienda di servizi. Funzione da allora ricoperta sempre più dagli OTT, che utilizzano le infrastrutture degli operatori di telecomunicazioni in modo perlopiù gratuito (infrastructure-as-a-free-service) [20], al fine di erogare un'ampia varietà di servizi ai clienti finali. Se da un lato le Telco hanno tutte le caratteristiche delle società di infrastrutture, essendo caratterizzate da intensità di capitale molto elevata, dall'operare su mercati locali, e dal servire una clientela che attribuisce il massimo valore all'affidabilità e alla disponibilità del servizio, dall'altro, gli OTT, il cui compito è quello di sviluppare l'idea di un servizio su cui costruire in modo rapido un'applicazione per attrarre il maggior numero di utenti, operano su scala globale, si appoggiano alle infrastrutture delle Telco in modalità perlopiù gratuita e investono inizialmente poco capitale. Se per le prime il tasso di natalità e mortalità delle aziende è molto basso, per i secondi è elevato. Diretta conseguenza del fatto che in caso di errori si deve far fronte a costi considerevoli per la prima categoria, contro un costo di fallimento più che contenuto per la seconda. Ma come monetizzano la loro posizione i grandi player OTT? Il primo modo è la remunerazione diretta da parte del cliente (servizio a pagamento, per esempio Netflix e Spotify); il secondo consiste nell'aumentare la capacità di profilazione degli utenti che si registrano e accedono ad una piattaforma (in questo caso in modalità gratuita, per esempio Facebook e Instagram), per poi monetizzare le informazioni raccolte (a fini pubblicitari o di direct marketing). Si intuisce che il modello di business degli OTT non può che mettere in crisi quello degli operatori Telco. La linea difensiva adottata dal mondo delle Telco è risultata piuttosto debole. È stato semplice per gli OTT argomentare che la connettività veniva già pagata dal cliente finale, dovendo quindi restare fruibile dallo stesso con politiche non discriminatorie. Inoltre, anche il mancato riconoscimento della natura prevalentemente infrastrutturale del business delle TLC ha portato ad una regolamentazione del settore ispirata ad un modello cost oriented, piuttosto che ad

uno RAB¹ oriented (diversamente da quanto accaduto per esempio per il sistema autostradale e di distribuzione dell’energia). Si è assistito a vere e proprie gare al ribasso dei prezzi che hanno messo in crisi l’attuale modello business delle società di telecomunicazioni che, come risultato, hanno visto ridursi notevolmente i propri flussi di cassa. In che modo le reti di quinta generazione, 5G, possono restituire agli operatori di telecomunicazioni un ruolo centrale dal punto di vista economico? Grazie all’ultima tecnologia di rete, oltre alla piattaforma Over-The-Top, diventa realtà il concetto di piattaforma Over-The-Network [20]. Mentre le prime sono perlopiù globali, economiche, “best effort” e non gestite (unmanaged), le seconde sono solitamente locali, meno economiche, ma a qualità garantita e gestite (managed). Ma la differenza sostanziale tra le due sta nella relazione che lega la piattaforma che eroga il servizio e l’infrastruttura su cui ci si appoggia per erogarlo. Se nel primo caso la piattaforma è tanto più performante quanto più riesce ad ottimizzare i servizi offerti dalle infrastrutture, nel secondo i servizi vengono direttamente progettati al fine di utilizzare nel modo più efficiente possibile le risorse di rete (motivo per cui sono tipicamente sviluppabili dagli operatori di Telecomunicazioni). Si intuisce che gli operatori Telco in alcune aree di mercato possono trarne un notevole vantaggio competitivo. Per capire in quali situazioni un approccio OTN può essere più adatto rispetto a quello OTT (e viceversa), bisogna calarsi nei panni del consumatore, e capire quanto alcuni parametri siano più o meno rilevanti quando si fa uso di un preciso servizio. Ne consideriamo tre: Quality of Service (QoS) e/o Quality of Experience (QoE), Sicurezza e Proximity al cliente. Quanto più queste specifiche sono richieste dal consumatore, tanto più l’approccio OTN è più congeniale, risultando più efficiente rispetto a quello OTT. Viceversa, se questi parametri sono poco rilevanti agli occhi del cliente, le probabilità che hanno le società di telecomunicazioni di competere con i grandi Player OTT tendono ad azzerarsi. Non bisogna però pensare che tutti i servizi OTT saranno sostituiti. In alcuni casi essi rappresentano una soluzione più conveniente ed efficace, e si ipotizza una coesistenza con i nascenti OTN. Per quanto detto, si può intuire che per rimettersi in gioco, gli operatori delle telecomunicazioni devono concentrare la loro attenzione su aree di attività in cui le caratteristiche dei servizi offerti determinino un vantaggio nei confronti dei grandi player OTT. Aree di attività in cui non basta banalmente una semplice SIM con un piano dati per usufruire del servizio, ma in cui è richiesta una stretta collaborazione tra i service providers e gli operatori di telecomunicazioni per sfruttare al massimo tutte le potenzialità di cui la rete dispone, in modo da offrire al cliente la miglior esperienza possibile. È importante osservare che questo nuovo modo di concepire la rete, oltre a restituire un ruolo centrale agli operatori di telecomunicazioni, favorisce lo sviluppo di imprese locali sui territori nazionali, dando loro la possibilità di realizzare il proprio modello di business. Nella figura 1.2 è schematizzato il modello di business per il mercato dei veicoli autoconnessi, in uno scenario 5G, in cui il ruolo degli operatori di telecomunicazioni torna ad essere centrale. Possiamo concludere affermando che il 5G nasce con l’obiettivo di supportare i servizi, cioè di mettere la rete all’interno del servizio.

¹RAB: Regulatory Asset Base, o in italiano Capitale Investito netto Riconosciuto, rappresenta una grandezza di riferimento primaria per la determinazione dei ricavi annui di molteplici aziende operanti in settori regolati in regime di monopolio. Si tratta tipicamente di settori per i quali, alla luce di vincoli strutturali e/o della rilevanza sociale del servizio offerto, lo Stato ha optato per un regime di monopolio anziché di libero mercato concorrenziale

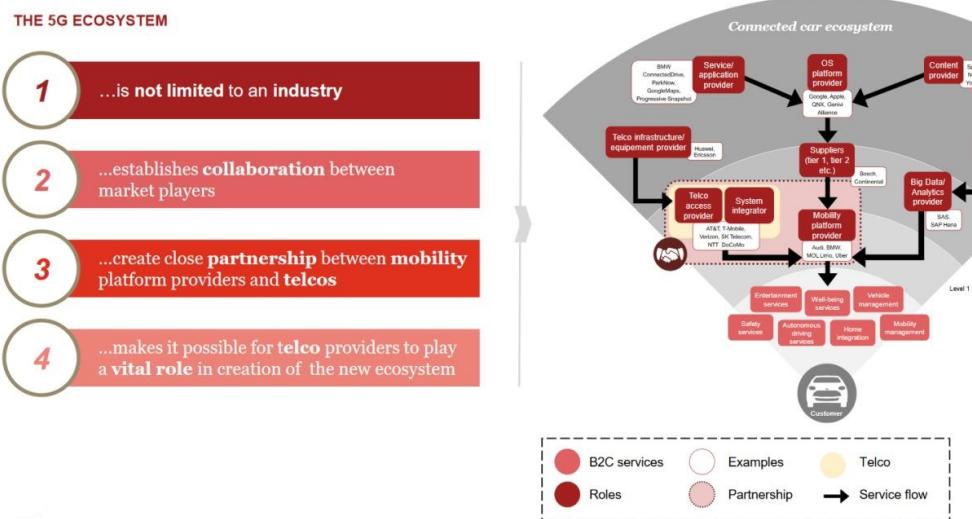


Figura 1.2: Scenario del mercato delle auto con il 5G

Capitolo 2

Network Slicing

In questa sezione del lavoro di tesi è stato presentato il concetto di Network Slicing, e le tecniche di cui ci si serve per implementarlo, note come: Software Defined Networking e Network Function Virtualization. Attraverso il Network Slicing implementato via software, si conferisce alla rete maggiore versatilità, proprietà necessaria per riuscire a supportare diverse classi di servizio in maniera efficiente, sulla medesima infrastruttura fisica.

2.1 Introduzione al Network Slicing

Per garantire ad un servizio massima efficienza e versatilità nell'utilizzo delle risorse di rete, in modo da offrire all'utente la miglior esperienza possibile, c'è bisogno di introdurre nuove funzionalità di rete. Temi come latenza, disponibilità, affidabilità e scalabilità, sono fondamentali per rendere implementabili e funzionanti al massimo dell'efficienza alcune classi di servizi. Una rete poco flessibile e che possiede pochi strumenti di controllo, non consente di utilizzare le risorse fisiche in modo ottimale. Essa dovrà soddisfare Quality of Service (QoS) diverse in base al tipo servizio che dovrà supportare. Affrontare questo problema con un approccio architettonale in cui una sola rete riesca a soddisfare tutti i requisiti che le vengono richiesti, non è la strategia migliore. Questa architettura non consentirebbe alla rete di riuscire a supportare le richieste dei diversi servizi, e non permetterebbe un rapido dispiegamento delle relative configurazioni di rete. Una singola architettura non è in grado di supportare tutti questi requisiti allo stesso tempo, in quanto dovrebbe bilanciare tra soluzioni contrastanti tra loro. In aggiunta, in una società così dinamica, risulta estremamente difficile prevedere i requisiti di cui necessiteranno i futuri servizi, pertanto la rete non può essere "statica" come lo è stata fino ad ora. Ciononostante, le aziende hanno sempre adattato le loro richieste di connettività alla cosiddetta "one-size-fits-all" mobile network, in maniera non congeniale e non redditizia (se non per gli OTT). Per ovviare a questo problema, la comunità delle telecomunicazioni ha incluso nell'attuale generazione di mobile networking, il 5G, la possibilità di effettuare network slicing via software. Infatti, per gestire e modellare le diverse slice, non è più necessaria una riconfigurazione dell'hardware, ma queste operazioni possono essere eseguite direttamente nei livelli applicativi, e non in quelli di rete e trasporto. Definendo un sottoinsieme isolato di risorse virtuali (computazionali, di rete e di archiviazione) e un insieme di regole per identificare il traffico che verrà eseguito su di esse, il paradigma del network slicing fornisce reti logiche personalizzate e adattate alle richieste di una classe di servizi, permettendone la realizzazione su un'unica infrastruttura di

rete comune [23]. Possiamo pensare una Network Slice, come un'istanza virtualizzata indipendente, definita dall'allocazione di un opportuno sottoinsieme di risorse di rete disponibili in termini di:

- Bandwidth: ogni slice dovrebbe avere la propria porzione di bandwidth su un collegamento. Fanno eccezione i casi in cui la slice gestisce traffico a bassa priorità, ed in tal caso, potrebbe non avere alcuna banda garantita
- Topologia: Ogni slice dovrebbe avere la propria visione dei nodi della rete (switch, router) e della connettività che esiste tra di loro. Ad esempio, nel caso in cui la slice supporti un servizio che richieda massima affidabilità e tolleranza ai guasti, i nodi fisici, ed i link che li collegano, possono essere organizzati via software, al fine di ottenere una rete virtuale con topologia ad anello, o parzialmente magliata. Al contrario, per una slice che supporta un servizio best effort, che non ha specifiche così stringenti in termini di affidabilità e tolleranza ai guasti, può essere realizzata una rete virtuale caratterizzata da una topologia a bus o ad albero, coinvolgendo un minor numero di collegamenti e nodi
- CPU del dispositivo: Ad ogni slice possono essere assegnate risorse computazionali adeguate
- Archiviazione: Ogni slice potrebbe avere diversi livelli di capacità di archiviazione
- Tabelle di inoltro e altre risorse del piano di controllo: dovrebbero essere anch'esse suddivise in Slice
- Traffico: Una porzione specifica del traffico verso una rete virtuale dovrebbe essere associata ad una slice per restare isolata dalla restante rete sottostante

Essa può essere dunque vista come una rete logica end-to-end che può essere dinamicamente creata, gestita e una volta obsoleta, disattivata, rilasciando le risorse di rete precedentemente utilizzate [17]. Questo processo di design, attivazione, controllo, e finale disattivazione di una slice, è riportato in figura 2.1.

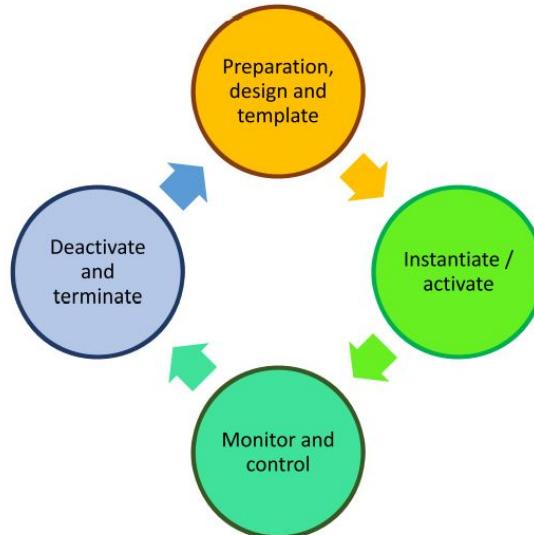


Figura 2.1: Ciclo di vita di una Network Slice

In questo modo è possibile allocare dinamicamente ed efficientemente le risorse di rete ad ogni singola slice in base alle QoS richieste, e l'infrastruttura di rete ne trae un notevole guadagno in termini di flessibilità e scalabilità. Dopo aver introdotto formalmente i concetti di Network Slicing e Network Slice, analizziamo tre classi di servizio che il 5G può supportare:

- enhanced mobile broadband (eMBB)
- massive Machine type communications (mMTC)
- ultra reliable low latency communications (URLLC)

La prima categoria richiede data rate molto elevate, alta mobilità, sicurezza ed estesa copertura (esempi tipici sono lo streaming video ad altissima definizione su smartphone o tablet e il Tactile Internet). Nella seconda, un gran numero di dispositivi ad alta ed altissima densità, solitamente fissi e che richiedono una buona efficienza energetica, trasmette bassi volumi di dati non sensibili al ritardo. Per evitare che questa quantità di dati sovraccarichi la rete core, è possibile elaborare e processare tali dati, prima che essa venga raggiunta (questa categoria contiene l'IoT, che a sua volta include le reti di sensori per attività di monitoraggio dello stato di edifici, e di infrastrutture, o di piante e terreni nella smart agricoltura). La terza categoria contiene servizi critici che sono caratterizzati da bassissima latenza, massima affidabilità e disponibilità (esempi tipici sono la telechirurgia, veicoli connessi e le reti di comunicazione per i servizi di emergenza in caso di disastri). La figura 2.2 mostra come la stessa infrastruttura fisica, mediante il Network Slicing, viene condivisa dalle tre reti distinte, ognuna delle quali progettata e gestita per garantire le specifiche, richieste da ciascun servizio. Pur condividendo le medesime risorse fisiche, esse lavorano in parallelo su reti logiche diverse. Per sfruttare questa proprietà è necessario che le slice siano tra loro isolate, e questo isolamento consiste nell'imporre che le performance di ogni slice, non abbiano alcun impatto sulle performance delle restanti altre. Tuttavia, se non ci limitiamo a considerare slice dedicate a servizi pregiati, ma inseriamo nello scenario anche slice che supportano servizi meno pregiati, la proprietà di isolamento può venir meno. Infatti esse potrebbero non aver risorse garantite nella rete fisica, ma usufruire di tali risorse solo quando disponibili.

Tale proprietà di isolamento migliora l'architettura della slice in termini di:

- Sicurezza: eventuali cyberattacchi o guasti restano confinati nella slice colpita
- Privacy: le informazioni private relative ad ogni slice, il suo stato, il suo traffico, non sono condivise con le restanti slice

Anche per ciò che riguarda la gestione delle slice, ognuna di esse deve essere gestita indipendentemente da tutte le altre, e il dover garantire questo isolamento rende necessari meccanismi e politiche ben definite ad ogni livello protocollare. Per creare, personalizzare e gestire queste reti logiche operanti sulla medesima infrastruttura fisica, è richiesta alla rete una notevole flessibilità, programmabilità e modularità, tutte specifiche difficili da soddisfare tramite un approccio Hardware. Proprio per questo, una delle principali novità introdotte dal 5G, è la cosiddetta network softwerization. Le tecniche abilitanti per il Network Slicing di cui ci serviamo e che introduremo sono:

- Software Defined Networking (SDN)
- Network Function Virtualization (NFV)

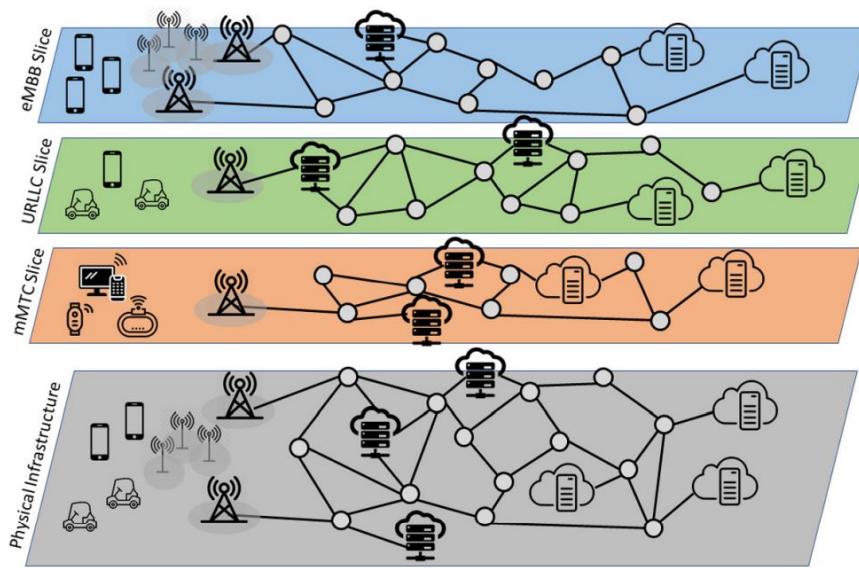


Figura 2.2: Schema delle diverse Slices esaminate associate ai casi d'uso 5G

2.2 Tecniche abilitanti lo Slicing

2.2.1 Software Defined Networking

L’idea di fondo del paradigma del Software defined networking (SDN) è la separazione tra il forwarding plane (o data plane) e il control plane [19]. Il control plane, che si occupa della gestione delle regole di inoltro dei pacchetti, possiamo considerarlo, in un parallelismo tra rete di telecomunicazioni e corpo umano, il cervello della rete, ovvero l’elemento dotato di logica. Esso sarà gestito da un controllore centralizzato, detto SDN controller. Per analogia, il forwarding plane, il cui compito è quello di attuare le decisioni logiche prese dal control plane, può essere accostato ai muscoli del nostro corpo, privi di logica ma necessari per eseguire i compiti impartiti dal nostro cervello. L’obiettivo è quello di estrarre più logica possibile dall’hardware, il cui unico compito sarà quello di attuare le decisioni prese dal controllore SDN centralizzato. Funzioni come inoltro selettivo degli switch, o instradamento dei router, proprie di questi dispositivi nelle precedenti generazioni di rete, non sono più “embedded”. Queste funzionalità saranno implementate direttamente dal controllore SDN, che impartirà le regole ai network device, il cui unico compito sarà quello di eseguirle. La figura 2.3 permette di confrontare le due tecniche per effettuare networking, riportando l’approccio tradizionale, e quello definito via software.

Per gli amministratori di reti software defined diventa possibile scrivere propri programmi SDN per configurare, gestire, proteggere e ottimizzare le risorse di rete tramite script automatici. Il controllo della rete diventa direttamente programmabile, in modo da regolare dinamicamente il traffico, traendone vantaggio in termini di flessibilità e agilità nella sua gestione. Nel paradigma SDN l’attore principale diventa il software. Bisogna sottolineare che se da un lato è corretto dedurre che i dispositivi di inoltro non dovranno più essere intelligenti e complessi come in passato, diventando più economici, è altresì vero che dovranno essere in grado di implementare il paradigma SDN per interagire con il controllore SDN.

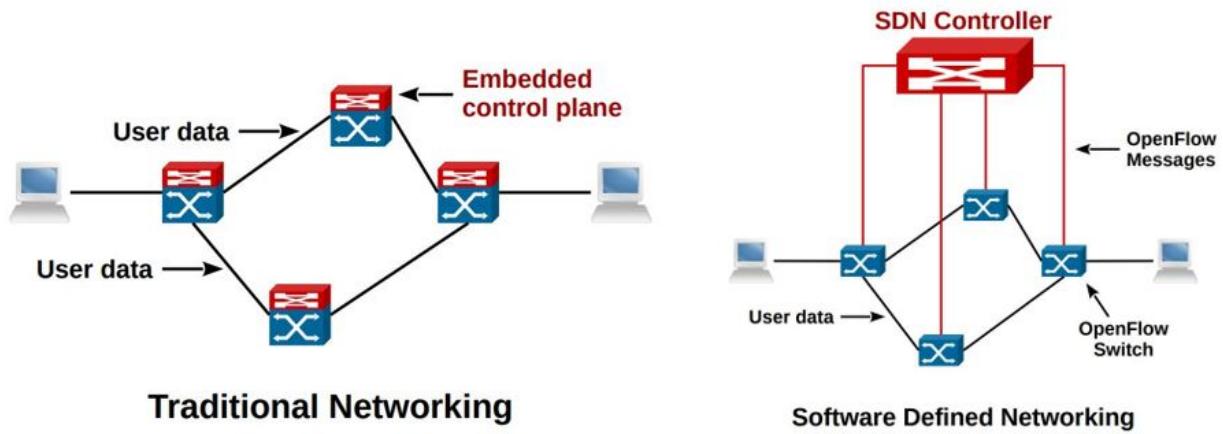


Figura 2.3: Networking classico vs Software Defined Networking

Architettura SDN

Nel modello architettonico mostrato in figura 2.4, oltre al control plane e al forwarding plane, esiste anche un terzo livello detto application plane [16]. Il suo compito è quello di comunicare le proprie necessità al control plane, ed è costituito da tutte le componenti applicative necessarie per gestirlo ad un livello d'astrazione molto alto. È proprio nel suddetto layer che viene implementata l'intera logica. In questo modello, ogni livello si relaziona con quello adiacente attraverso un'apposita interfaccia. Quella che permette l'interazione tra control plane e data plane prende il nome di Southbound Interface, mentre quella che consente la comunicazione tra application plane e control plane, è detta Northbound Interface, come a voler sottolineare la loro posizione nello stack. La seconda, attraverso l'uso di driver, lato application layer, ed agent, lato controller, permettere alle applicazioni di comunicare i loro requisiti alla rete. La prima invece, utilizzata dal controllore SDN per programmare il piano dati, è standardizzata da alcuni protocolli, come OpenFlow e NETCONF. Possiamo considerare questi protocolli come i driver con cui il sistema operativo della rete (il controllore SDN) interagisce con l'hardware (dispositivi di rete). Ricapitolando, per ottenere tale separazione sono necessari tre componenti:

- Un controllore SDN centralizzato
- Dispositivi che supportano il paradigma SDN
- Un protocollo di gestione (come OpenFlow o NETCONF)

Benefici del paradigma SDN

Implementando il paradigma SDN, i benefici che si possono ottenere non si limitano ad avere un'elevata flessibilità, agilità e programmabilità nel gestire e riconfigurare rapidamente la rete. Grazie alla gestione centralizzata che questo paradigma rende possibile, un solo controllore (o più ridondanti, per evitare di avere un unico punto di fallimento), che ha una visione completa dell'intera topologia della rete e di tutti i dispositivi che la compongono, può opportunamente riconfigurarli al fine di ottenere il comportamento desiderato del flusso dei pacchetti. Un altro risultato importante

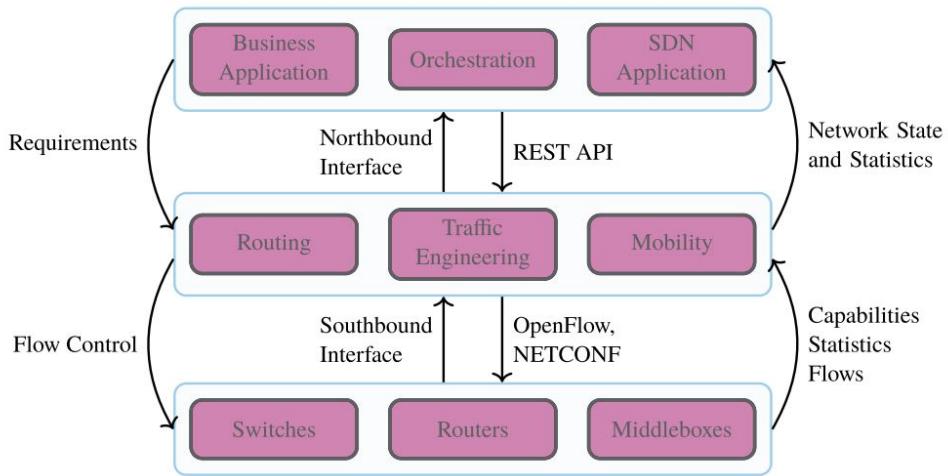


Figura 2.4: Architettura del paradigma SDN

è quello di limitare il vendor lock-in. Un singolo controllore SDN può gestire dispositivi prodotti da aziende diverse, utilizzando istruzioni che non sono specifiche del venditore. Un parallelismo interessante può essere fatto osservando quanto accaduto per i Personal Computers. Nel mondo dei PC il software è sviluppato per essere compatibile con un sistema operativo che ha il compito di interfacciarsi con l'hardware. In questo modo, lo stesso software può funzionare su PC di vendor diversi, con il risultato di essere non specifico per un determinato hardware. È esattamente quello che sta accadendo nel mercato delle reti. In passato esistevano pochi grandi attori, i quali dettavano i tempi dello sviluppo tecnologico, e producevano hardware verticalmente integrato, cioè con software specifico e quindi funzionalità specifiche per l'hardware da loro prodotto. Con il paradigma SDN, i dispositivi di rete diventano hardware generico, controllabile con sistemi operativi di rete diversi, e diventa finalmente possibile, così come per i PC, sviluppare funzionalità di rete specifiche per sistemi operativi specifici, e non per hardware specifico.

2.2.2 Network Function Virtualization

L'obiettivo del paradigma del Network Function Virtualization (NFV) è quello di disaccoppiare le funzionalità di rete basate sul software, dai dispositivi fisici di rete basati sull'hardware. Queste funzioni di rete implementate via software, che prendono il nome di Virtual Network Function (VNF), saranno disponibili su hardware generico, detto anche general-purpose (come i server) [9]. La figura 2.5 mette a confronto lo scenario classico di rete, raffigurando hardware specifico, con il nuovo modello di rete, ritraendo hardware generico, e le funzioni di rete virtuali, implementate via software. Questo paradigma consente un dispiegamento flessibile delle funzioni di rete, in modo da offrire una gestione e un'assegnazione più semplice delle risorse di rete a specifici servizi. Sarà dunque possibile adattare le risorse di rete in modo dinamico e flessibile, in base all'esigenza del servizio. Non dovendo più preoccuparci della configurazione degli specifici dispositivi di rete, cioè di ciò che sono o non sono in grado di fare (per esempio in base al produttore, o alla versione del sistema operativo su essi installato), forniti dai diversi vendor, aumenta il grado di libertà nella creazione, nell'implementazione e nella gestione dei servizi di rete. Un preciso servizio può essere ottenuto combinando tra loro più VNF, facendo un'operazione comunemente detta VNFs chaining. Le funzionalità di rete, essendo ora implementate in software ed eseguibili su hardware generico,

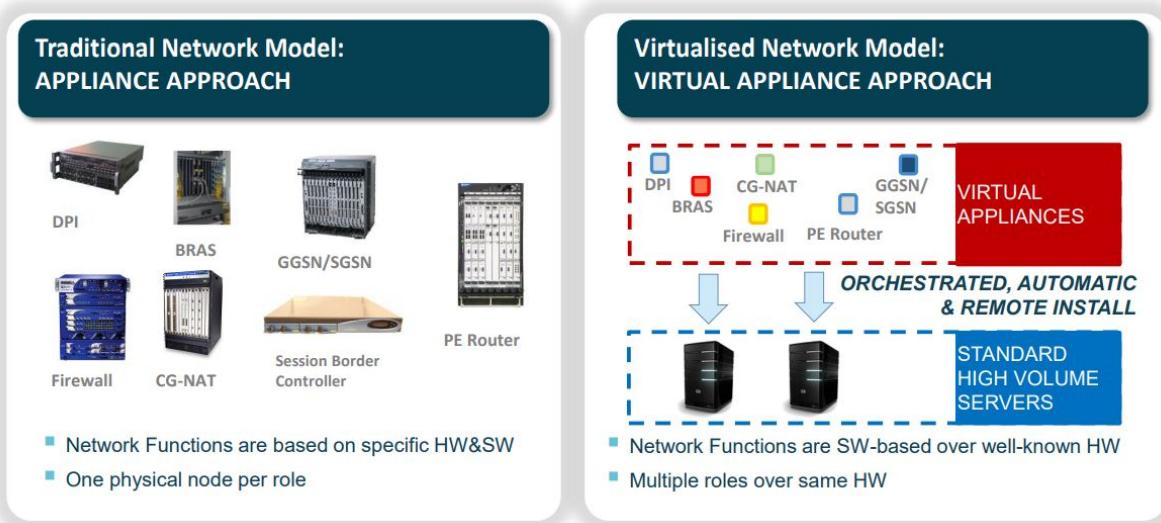


Figura 2.5: Modello di rete tradizionale VS Modello di rete virtualizzata

possono essere dislocate in diverse posizioni della rete senza dover necessariamente, come avveniva in passato, installare nuovo hardware specifico. È interessante osservare che con questa strategia è possibile eseguire la VNF che implementa un servizio in un punto più vicino all'utilizzatore. Questo importante concetto è alla base del Mobile Edge Computing, schematizzato in figura 2.6. Per Mobile Edge Computing, si intende l'architettura di rete caratterizzata dalla presenza di risorse di computazione sul bordo della rete mobile. Se non ci limitiamo solo alla rete mobile, ma consideriamo anche per esempio le PON¹, parliamo di Multi Access Edge Computing. Nel caso specifico di reti ottiche passive, questa operazione possiamo vederla come l'aggiunta di funzionalità sull'OLT (optical line termination), che possono consentirci di recuperare le informazioni di cui abbiamo bisogno senza dover andare nella data network. Questa soluzione prende il nome di "MEC at OLT", in cui il nodo MEC si trova all'interno, o è direttamente connesso alla rete di accesso dell'Internet Service Provider (ISP). Essendo necessaria la condivisione delle risorse ottiche all'interno della PON per raggiungerlo, potrebbe esserci una riduzione di bandwidth, con una latenza leggermente inferiore ad 1 ms [12]. Due possibili altenrative sono:

- MEC at customer premise: il nodo MEC è all'interno della rete del cliente, ed è la strategia migliore in termini di throughput e latenza, ma a fronte di un costo maggiore
- MEC at Cloud: il nodo MEC è al di fuori della rete di accesso dell'ISP, e per questo si ha un elevato Round Trip Time (RTT), nell'ordine dei 10-100 ms, ed una riduzione della bandwidth. Più economica, ma la peggiore tra le tre dal punto di vista prestazionale

Questa operazione porta un duplice beneficio:

- riduzione della latenza, con un Round Trip Time (RTT) minore
- si evita di intasare la rete Core, elaborando il traffico prima che essa venga raggiunta

¹PON: Passive Optical Network, descritta con dettaglio nel seguito della trattazione

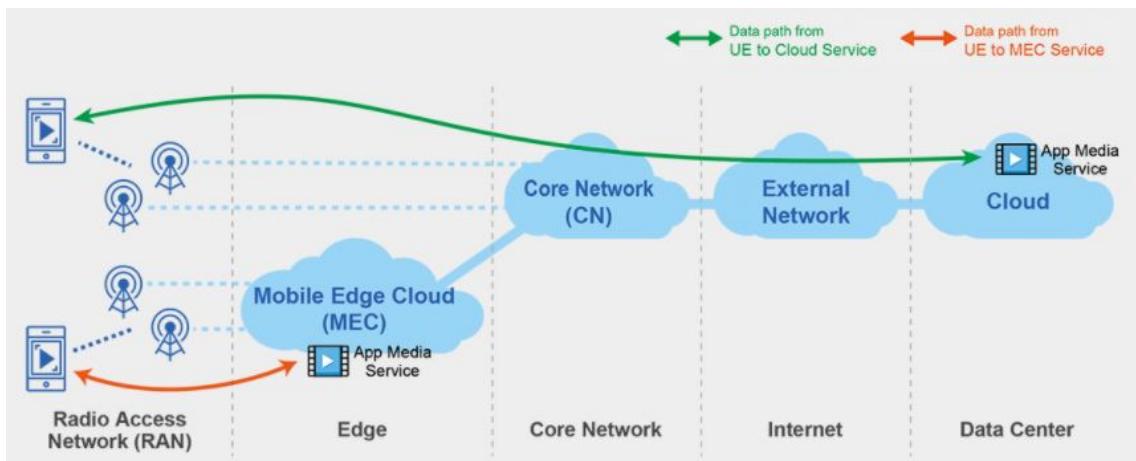


Figura 2.6: Scenario descrittivo di Mobile Edge Computing

Benefici del paradigma NFV

L'introduzione di nuove funzionalità ai servizi di rete può essere ottenuta direttamente via software, senza la necessità di dover installare hardware proprietario e senza dover riorganizzare il livello fisico della rete. Per lo stesso principio, se un servizio diventa obsoleto, può essere rimosso semplicemente arrestando la Virtual Function (VF) che lo implementava, liberando successivamente le risorse fisiche, che tornano ad essere a disposizione di altri servizi. Si intuisce che l'hardware general-purpose, come server, assumerà sempre più un ruolo centrale nelle future generazioni di rete, diventando sostanzialmente l'infrastruttura di base. La sua caratteristica principale, nel contesto del NFV, rispetto ad hardware dedicato, è la sua capacità di fornire maggiori risorse ad un minor costo. Questo disaccoppiamento tra software e hardware consente anche una più agevole manutenzione sia dell'hardware, che del software. Uno scenario che possiamo immaginare è quello di poter acquistare del software per un router ad esempio, da installare su hardware generico, senza essere costretti ad acquistare il router stesso, per poter usufruire delle medesime funzioni fornite dal software su di esso installato. In questo scenario si ha un'innovazione decisamente più rapida, considerando che il software presenta inevitabilmente cicli di sviluppo inferiori rispetto a quelli necessari per innovare l'hardware. Oltre al guadagno inteso in termini di tempi di innovazione, il paradigma NFV ha ridotto drasticamente i cosiddetti CAPEX² (CAPital EXPenditure) e OPEX³ (OPeration EXPense), garantendo agli operatori di telecomunicazioni un guadagno anche dal punto di vista economico.

Architettura NFV

L'architettura logica del paradigma NFV, descritta in figura 2.7, è anch'essa organizzata in livelli. Nello strato più basso si trovano le risorse fisiche dei dispositivi, generici, che compongono la rete. Queste risorse possono essere suddivise in risorse computazionali, di archiviazione e di rete. Il layer superiore contiene un'astrazione delle risorse fisiche, ovvero una mappatura delle effettive risorse in risorse che sono dette virtuali. Salendo ulteriormente lo stack, si arriva al livello che contiene tutte le funzioni di rete virtuali. Mentre le risorse computazionali e di memoria sono necessarie per il corretto funzionamento di una VNF, quelle di rete permettono la comunicazione e la collaborazione

²CAPEX: spese legate alle infrastrutture e alle immobilizzazioni di un'azienda, come edifici, terreni o attrezzature

³OPEX: spese legate alla gestione delle attività e dei sistemi di un'azienda

tra esse. Il livello verticale, oltre ad occuparsi della gestione e dell'orchestrazione di come le risorse hardware vengono mappate nelle risorse virtuali, è responsabile anche di come le VNF comunicano e collaborano tra loro.

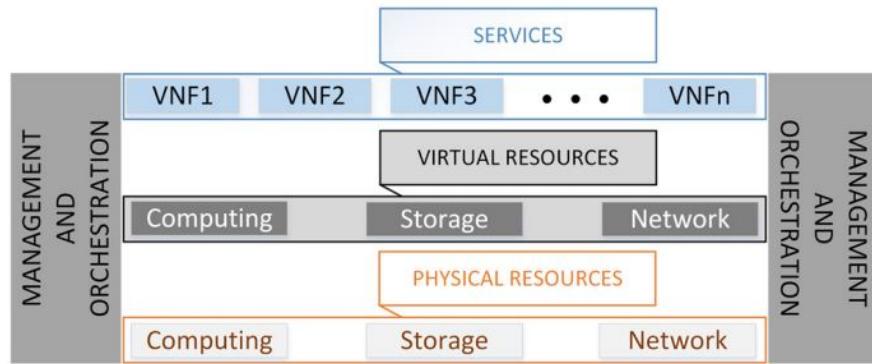


Figura 2.7: Struttura logica di una Network Function Virtualization

Un sistema NFV è composto da tre principali componenti:

- Virtual Network Functions (VNFs): funzioni di rete implementate in software che possono essere dispiegate sull'infrastruttura NFV (NFVI)
- NFV Infrastructure (NFVI): l'insieme di risorse sia hardware che software necessarie per le funzionalità interne delle VNF e per la comunicazione tra le VNF
- NFV Management and Orchestration (MANO): Ha il ruolo di coordinatore dell'intero sistema NFV

Capitolo 3

NETCONF-YANG Solution

In questo capitolo del lavoro di tesi sono stati presentati il protocollo NETCONF, per esteso Network Configuration Protocol, ed il linguaggio di modellazione dati YANG, per esteso Yet Another Next Generation. Dopo una loro descrizione formale, sono stati forniti esempi per comprenderne le caratteristiche e le funzioni principali. Infine, per completare la trattazione, sono stati introdotti alcuni tool per semplificarne l'utilizzo, mostrandone le funzionalità più importanti.

3.1 Considerazioni introduttive sulla NETCONF-YANG solution

La natura estremamente dinamica delle reti virtuali, con funzionalità che cambiano continuamente, ne richiede una pronta ed efficace gestione quanto più possibile automatizzata. Questo problema trova risposta nella soluzione NETCONF-YANG, in grado di fornire un alto livello di flessibilità nella gestione dei dispositivi di rete. Non a caso, molte aziende operanti nel settore del Software Defined Networking e molti progetti riguardanti questo tema, stanno promuovendo questa “ricetta” come protocollo universale per la Southbound Interface, per configurare e gestire sia le VNF, che i dispositivi di rete fisici nello scenario SDN. Molti vendor hanno iniziato a supportare la soluzione NETCONF-YANG nelle loro ultime generazioni di dispositivi, riconoscendone le grandi potenzialità. Attraverso il suo utilizzo, si ha la possibilità di configurare direttamente un servizio sulla rete, e non solo i singoli dispositivi. Una funzionalità chiave che offre questa tecnologia è il cosiddetto supporto delle transazioni. Attraverso questo strumento i Service Network Providers hanno la certezza che qualora non venissero applicate correttamente tutte le configurazioni programmate, l'intero aggiornamento sarà annullato, senza conseguenze più o meno gravi sul comportamento della rete. Il protocollo NETCONF ed il linguaggio YANG, risultano essere perfettamente complementari. Infatti, attraverso YANG si descrivono le modifiche delle configurazioni che si desidera apportare, ed utilizzando NETCONF, vengono applicate ad un preciso datastore presente all'interno del dispositivo da configurare [13].

3.2 NETCONF

NETCONF, per esteso Network Configuration Protocol, è un protocollo di gestione di rete sviluppato e standardizzato ufficialmente nel dicembre del 2006, nel documento rfc4741 dall'IETF, e rivisitato poi nel 2011 attraverso l'rfc6241 [15]. NETCONF fornisce meccanismi per installare,

manipolare e cancellare le configurazioni dei dispositivi di rete. Utilizza una codifica dati basata su un Extensible Markup Language (XML) sia per i dati di configurazione che per i messaggi del protocollo. Le operazioni sono realizzate secondo il paradigma del RPC (Remote Procedure Call), che facilita la comunicazione tra un client e un server. Per client si intende uno script o un'applicazione di gestione di rete in esecuzione, mentre il server corrisponde al dispositivo di rete con cui si vuole interagire. Una sessione NETCONF può essere intesa come la connessione logica tra un amministratore di rete o un'applicazione di configurazione di rete, e il network device, e lo scenario di lavoro tipico prevede l'invio di una RPC codificata in XML da parte del client verso il server, il quale a sua volta gli fornirà una risposta, detta RPC-Reply, attraverso un messaggio sempre codificato in XML. La figura 3.1 mostra un client NETCONF interagire con cinque server NETCONF, situati all'interno di diversi dispositivi (per esempio poichè prodotti da vendor diversi), ma che supportano tutti il protocollo, per essere monitorati e gestiti attraverso esso.

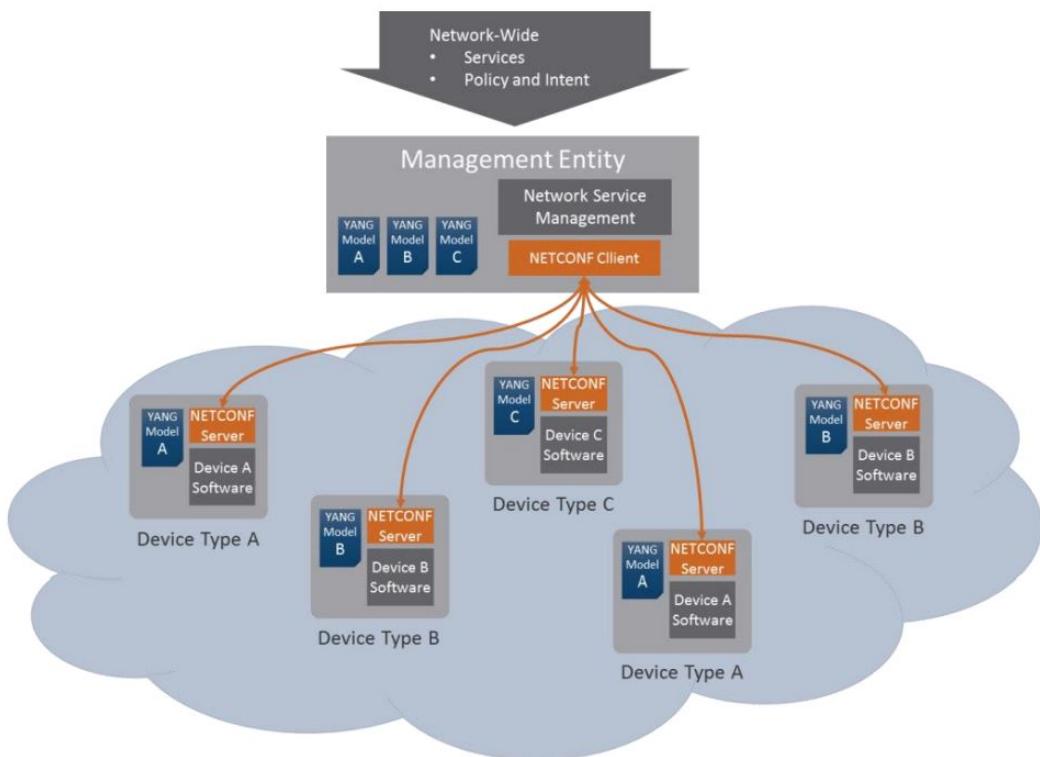


Figura 3.1: Scenario generale NETCONF

XML cenni

L'Extensible Markup Language (XML), standard approvato da W3C, è un metalinguaggio, ovvero un linguaggio a partire dal quale vengono creati nuovi linguaggi. Esso definisce le regole per creare linguaggi di Markup, cioè linguaggi il cui contenuto è strutturato tramite particolari delimitatori detti tag. Ogni tag indica un dato, ed in XML è possibile definirne un numero illimitato. Semplificato notevolmente rispetto a SGML, oltre che per memorizzare dati, XML ha l'obiettivo di semplificare la condivisione tra diversi sistemi, e per questo è molto utilizzato quando si scambiano informazioni. Essendo completamente text-based, risulta leggibile anche dagli umani oltre che dalle macchine, e editabile anche a mano, utilizzando un semplice editor di testo. Per questo motivo,

e poiché sempre richiesto il tag di fine, oltre alla macchina anche lo sviluppatore può individuare eventuali errori nella stesura, come ad esempio i comuni annidamenti errati. Attraverso l'utilizzo di XML, gli sviluppatori possono creare linguaggi ad-hoc per contenere informazione strutturata, in base alle proprie esigenze [11]. Per introdurlo in modo più pratico e poter comprenderne le caratteristiche principali, è riportato un breve esempio di file XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<utenti>
    <utente anni="20">
        <nome>Ema</nome>
        <cognome>Princi</cognome>
        <indirizzo>Torino</indirizzo>
    </utente>
    <utente anni="54">
        <nome>Max</nome>
        <cognome>Rossi</cognome>
        <indirizzo>Roma</indirizzo>
    </utente>
</utenti>
```

Listing 3.1: Esempio di file XML

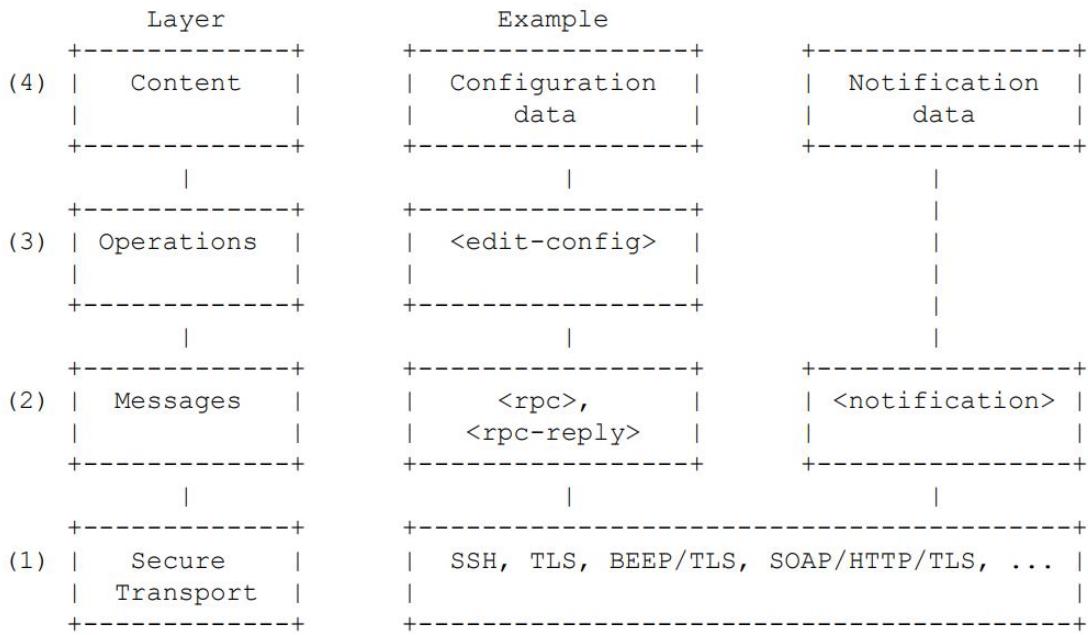
3.2.1 Alternative a NETCONF

Possibili alternative a NETCONF potrebbero essere il Simple Network Management Protocol (SNMP), che però non supporta la funzionalità delle transazioni, non restituisce testo in chiaro ma messaggi codificati con la codifica BER e, inoltre, non ha la capacità di distinguere i dati operativi da quelli di configurazione salvati su un dispositivo [18]. Questo protocollo resta comunque una valida soluzione per il semplice monitoraggio della rete. Per quanto riguarda l'alternativa classica della CLI, se da un lato offre il vantaggio di presentare il testo in chiaro, quindi direttamente comprensibile per un umano, di rendere possibile l'operazione di cut-and-paste tra dispositivi simili, e di avere la capacità di distinguere tra dati operativi e di configurazione, dall'altro è fortemente limitante sia per via dei frequenti errori di battitura commessi dagli operatori, sia per la molteplicità delle sintassi proprietarie, da conoscere per interagire con il device. Aspetto che limita fortemente l'interoperabilità tra dispositivi di vendor diversi.

3.2.2 Stack Protocollare

Il protocollo NETCONF, come riportato in figura 3.2, può essere concettualmente partizionato in 4 layer:

- 1. Secure Transport Layer:** si occupa del trasporto dei messaggi in modo sicuro ed affidabile tra Client e Server. Il protocollo NETCONF non si limita ad un particolare protocollo di trasporto, ma consente una mappatura per definire come può essere implementato su ogni protocollo specifico, a patto che esso fornisca un insieme di funzionalità. L'unica mappatura obbligatoria che un'implementazione NETCONF deve necessariamente supportare, è quella del protocollo di trasporto SSH. Essendo NETCONF un protocollo connection oriented, è

**Figura 3.2:** Stack Protocollare di NETCONF

richiesta una connessione duratura tra client e server, che deve consegnare dati non desenquenziati, garantendone integrità informativa e riservatezza, a seguito di una procedura di autenticazione

2. **Messages layer:** fornisce un meccanismo di framing semplice ed indipendente dal Transport Layer, per codificare le RPC e le notifiche. Client e Server usano un modello di comunicazione basato sul paradigma RPC, utilizzando messaggi di tipo `<rpc>` e `<rpc-reply>`. Il primo è un elemento utilizzato per una NETCONF request (richiesta, modifica o cancellazione di informazioni) dal client al server, mentre il secondo è inviato in risposta dal server al client. Entrambi hanno l'attributo “message-id” obbligatorio, il cui valore coincide quando la `<rpc-reply>` è la risposta ad una precisa `<rpc>`. All'interno di un messaggio `<rpc-reply>` si può trovare l'elemento `<ok>` se non ci sono stati errori nell'elaborazione di una richiesta `<rpc>`, o l'elemento `<rpc-error>`, che ne indica la presenza, con informazioni aggiuntive che ci consentono di individuarlo più agevolmente. Un'altra funzionalità molto utile che il protocollo NETCONF offre ad un client è quella di poter “sottoscrivere un accordo” con un server, da cui il client riceverà, in modo asincrono, notifiche riguardanti particolari eventi (ad esempio il superamento di una soglia prevista per le risorse di archiviazione del server)

3. **Operations layer:** definisce un insieme di operazioni base che il client può utilizzare per recuperare e modificare i dati di configurazione di un server. Esse vengono invocate come metodi RPC, usando parametri codificati in XML, ed eventuali operazioni aggiuntive di cui un dispositivo è dotato, oltre a quelle base del protocollo, sono comunicate attraverso un meccanismo di pubblicizzazione delle cosiddette “capabilities”. Le operazioni base del protocollo sono:

- `<get>`: recupera informazioni sull'attuale configurazione e sullo stato del dispositivo

- <get-config>: recupera tutte o una parte delle informazioni (attraverso un’operazione di filtraggio) di uno specifico datastore di configurazione
- <edit-config>: modifica il contenuto di un preciso datastore di configurazione che, qualora non esistesse, sarà creato. Questa operazione offre le opzioni di aggiunta, sostituzione, creazione, cancellazione o rimozione, completa o parziale, del contenuto
- <copy-config>: crea (qualora non esistesse) o sostituisce un intero datastore di configurazione sovrascrivendo i dati di un altro datastore di configurazione
- <delete-config>: elimina un datastore di configurazione, ad eccezione del datastore di configurazione <running>, che non può essere rimosso
- <lock>: permette ad un client di bloccare un intero datastore di configurazione di un dispositivo. Questa operazione consente al client di poter lavorare indisturbato su quel preciso datastore di configurazione, solitamente per periodi di breve durata
- <unlock>: utilizzata in seguito ad un’operazione di lock per rilasciare un datastore di configurazione precedentemente bloccato
- <close-session>: richiede la terminazione di una sessione NETCONF
- <kill-session>: forza la terminazione di una sessione NETCONF

4. **Content layer:** non descritto nell’rfc6241, contiene dati di configurazione, di stato ed informazioni statistiche

3.2.3 Esempio di una sessione NETCONF

In questa sezione analizzeremo un ipotetico scenario di utilizzo del protocollo, eseguendo le principali <rpc>, riportate nelle catture insieme alle relative <rpc-reply>. Dopo l’instaurazione della sessione NETCONF, si osserva subito lo scambio delle cosiddette “capabilities” tra server, che lo fa in automatico, non appena la sessione è instaurata, e client (in questo caso saremo noi a rispondere), attraverso un messaggio comunemente detto di “Hello Message”. A questo punto si può procedere con la sessione vera e propria, operando attraverso messaggi <rpc> codificati in XML, e ricevendo <rpc-reply>, richiedendo prima informazioni al server, per poi eseguire le modifiche volute. In questo esempio mostrerò come richiedere un semplice parametro come l’hostname del server NETCONF, per poi modificarlo, dopo una preliminare operazione di lock, per lavorare indisturbati sul datastore del server. In seguito alla modifica, prima di terminare la sessione, si effettuerà l’operazione di unlock per liberare il datastore su cui si stava lavorando e che era stato precedentemente monopolizzato. Questo esempio, seppur molto semplice, racchiude gran parte dei concetti principali, e permetterà di concettualizzare le nozioni formali descritte nella sezione precedente, comprendendo in modo pratico gli aspetti più importanti del protocollo NETCONF. Per la visualizzazione gerarchica delle risposte del server, esistono diversi tool, tra i quali la funzione Tree View, disponibile sul sito www.xmlviewer.org, utilizzata nell’esempio seguente.

Instaurazione sessione ed Hello Message

Dopo la prima fase di autenticazione in cui è richiesto l’inserimento della password, viene instaurata la sessione NETCONF, che si appoggia al protocollo SSH, che garantisce cifratura ed integrità

dei dati. In questa fase gli aspetti più interessanti da commentare sono principalmente due. Il primo è il campo “session-id”, fornito dal server al client nell’hello message, che sarà l’identificativo della sessione NETCONF instaurata. Esso è assegnato unicamente dal server, e non può essere in alcun modo deciso dal client. Per il secondo bisogna osservare con attenzione la lista delle capabilities del server. Alcune di esse sono standardizzate dall’ietf, quindi generiche, non dedicate al dispositivo specifico utilizzato per la dimostrazione. Altre invece sono proprietarie, sviluppate in questo esempio da calix, l’azienda produttrice del device utilizzato, specifiche e dedicate per i dispositivi da loro prodotti. Va specificato che le prime sono le NETCONF capabilities vere e proprie del server, cioè le operazioni del protocollo che esso supporta. Quelle che contengono la keyword “module”, rappresentano invece i data models supportati. Gran parte di questi moduli sono reperibili ed analizzabili all’indirizzo www.github.com/YangModels/yang. È importante precisare che una mancata risposta all’hello message del server, con l’hello message del client, comporterà la chiusura immediata della sessione. Inoltre, è utile osservare che la sequenza “[>]]>”, è il campo che indica la fine del messaggio. Per alcuni dispositivi, questo campo è aggiunto di default a conclusione del messaggio, in altri invece, è necessario l’inserimento manuale, altrimenti il server non è in grado di identificare il messaggio, e non fornirà alcuna risposta [3]. Nelle figure 3.3 e 3.4, è riportato prima il comando per instaurare la sessione con il server NETCONF, scegliendo come livello di trasporto sicuro ssh, e la porta 830 (solitamente scelta per il protocollo), supportato dal dispositivo di indirizzo ip 10.10.10.30, e nome utente “sysadmin”, e poi parte della sua risposta, che prevede l’invio in automatico delle proprie capabilities. Per completare il processo di instaurazione della sessione NETCONF, e poter procedere con le vere operazioni da eseguire, si risponde al server con l’hello message lato client, riportato in figura 3.5.

Operazione di <get-config>

Una volta instaurata la sessione, possiamo procedere con le operazioni vere e proprie. La prima che svolgiamo è una get, per interrogare il server NETCONF, sui parametri che ci interessano, per poter analizzarli, confermarli, o, come nell’esempio, modificarli. In questo caso abbiamo richiesto l’hostname del dispositivo. Per questo compito, è stata utilizzata l’operazione <get-config>, che permette di interrogare il server NETCONF riguardo i dati di configurazione, a differenza dell’operazione <get>, che oltre ai dati di configurazione permette di chiedere anche quelli operativi (operational data). È importante sottolineare che a differenza dei dati di configurazione come l’hostname, gli indirizzi IP, ecc, che possono essere sia letti, che scritti, quelli operativi, cioè di stato, dovrebbero solo poter essere letti, attraverso un’operazione di <get>. Osserviamo come nel messaggio <rpc> sia specificato l’attributo obbligatorio “message-id”, a cui farà riferimento la <rpc-reply> di risposta. In questa operazione di <get-config>, si è utilizzata la funzione <filter>, molto utile per circoscrivere la ricerca al parametro cercato. Per utilizzare questa strategia, bisogna però conoscere il data model, per poter specificare il path. La figura 3.6 riporta l’operazione di <get-config> eseguita e la relativa <rpc-reply>. Per visualizzare la risposta in modo gerarchico, tra i diversi tool esistenti, è stato utilizzato come già accennato, la funzione Tree View disponibile su www.xmlviewer.org, facendo un copia e incolla della <rpc-reply>, eliminando la sequenza conclusiva “[>]]>”. L’interfaccia e l’output forniti dal tool, sono mostrati nella figura 3.7.

```

stefano@stefano:~$ ssh sysadmin@10.10.10.30 -p 830 -s netconf
sysadmin@10.10.10.30's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,sftp,file</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=trim&also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=2016-06-21&module-set-id=f66f14f441d80058ec389988a6335a42</capability>
<capability>http://tail-f.com/ns/netconf/extensions</capability>
<capability>http://tail-f.com/ns/aaa/1.1?module=talif-aaa&revision=2013-03-07</capability>
<capability>http://tail-f.com/ns/common/query?module=talif-common-query&revision=2017-04-27</capability>
<capability>http://tail-f.com/ns/webui?module=talif-webui&revision=2013-03-07</capability>
<capability>http://tail-f.com/yang/acm?module=talif-acm&revision=2013-03-07</capability>
<capability>http://tail-f.com/yang/common?module=talif-common&revision=2017-09-28</capability>
<capability>http://tail-f.com/yang/common-monitoring?module=talif-common-monitoring&revision=2013-06-14</capability>
<capability>http://tail-f.com/yang/confd-monitoring?module=talif-confd-monitoring&revision=2013-06-14</capability>
<capability>http://tail-f.com/yang/netconf-monitoring?module=talif-netconf-monitoring&revision=2012-06-14</capability>
<capability>http://www.calix.com/ns/arc?module=arc&revision=2019-03-28</capability>
<capability>http://www.calix.com/ns/erps-g8032-qos?module=erps-g8032-qos&revision=2020-08-18</capability>
<capability>http://www.calix.com/ns/ethernet-std?module=ethernet-std&revision=2021-05-18</capability>
<capability>http://www.calix.com/ns/exa/access-security?module=access-security&revision=2020-10-26</capability>
<capability>http://www.calix.com/ns/exa/access-security-std?module=access-security-std&revision=2020-11-17</capability>
<capability>http://www.calix.com/ns/exa/acl?module=acl&revision=2021-04-28</capability>
<capability>http://www.calix.com/ns/exa/acl-router-lqmp?module=acl-router-lqmp&revision=2021-01-08</capability>
<capability>http://www.calix.com/ns/exa/acl-std?module=acl-std&revision=2020-11-17</capability>
<capability>http://www.calix.com/ns/exa/ae-management?module=ae-management&revision=2021-02-28</capability>
<capability>http://www.calix.com/ns/exa/ae-management-std?module=ae-management-std&revision=2019-04-18</capability>
<capability>http://www.calix.com/ns/exa/axos-netconf-server-base?module=axos-netconf-server-base&revision=2020-09-10</capability>
<capability>http://www.calix.com/ns/exa/axos-netconf-server-std?module=axos-netconf-server-std&revision=2021-01-12</capability>
<capability>http://www.calix.com/ns/exa/base?module=exa-base&revision=2020-03-16&features=ont-simulation,dos-protection,chassis-craft-port</capability>
<capability>http://www.calix.com/ns/exa/bridge?module=bridge&revision=2021-03-31</capability>
<capability>http://www.calix.com/ns/exa/bridge-std-eth?module=bridge-std-eth&revision=2021-04-09</capability>
<capability>http://www.calix.com/ns/exa/bridge-std-lag?module=bridge-std-lag&revision=2021-04-09</capability>
<capability>http://www.calix.com/ns/exa/card-profiles?module=card-profiles&revision=2021-04-30</capability>
<capability>http://www.calix.com/ns/exa/copp?module=copp&revision=2020-10-22</capability>
<capability>http://www.calix.com/ns/exa/copp-std?module=copp-std&revision=2020-10-30</capability>
<capability>http://www.calix.com/ns/exa/crypto-types?module=crypto-types&revision=2018-03-07</capability>
<capability>http://www.calix.com/ns/exa/diagnostics?module=diagnostics&revision=2019-04-30</capability>
<capability>http://www.calix.com/ns/exa/diagnostics-std?module=diagnostics-std&revision=2018-08-17</capability>
<capability>http://www.calix.com/ns/exa/discovery?module=discovery&revision=2021-01-12</capability>
<capability>http://www.calix.com/ns/exa/dot1x-base?module=dot1x-base&revision=2021-03-03</capability>
<capability>http://www.calix.com/ns/exa/dot1x-std-ethernet?module=dot1x-std-ethernet&revision=2020-11-17</capability>
<capability>http://www.calix.com/ns/exa/dscp?module=dscp&revision=2020-05-05</capability>
<capability>http://www.calix.com/ns/exa/dscp-copp?module=dscp-copp&revision=2020-10-22</capability>
<capability>http://www.calix.com/ns/exa/dscp-std?module=dscp-std&revision=2019-12-18</capability>
<capability>http://www.calix.com/ns/exa/equipment-ua?module=equipment-ua&revision=2020-12-14</capability>
<capability>http://www.calix.com/ns/exa/erps?module=erps&revision=2021-01-13</capability>
<capability>http://www.calix.com/ns/exa/erps-rstp?module=erps-rstp&revision=2020-10-22</capability>
<capability>http://www.calix.com/ns/exa/erps-std-eth?module=erps-std-eth&revision=2020-02-25</capability>
<capability>http://www.calix.com/ns/exa/erps-std-lag?module=erps-std-lag&revision=2020-02-25</capability>
<capability>http://www.calix.com/ns/exa/ethernet-link-oam?module=ethernet-link-oam&revision=2017-08-02</capability>
<capability>http://www.calix.com/ns/exa/ethernet-loan-std?module=ethernet-loan-std&revision=2020-11-17</capability>
<capability>http://www.calix.com/ns/exa/ethernet-service-oam?module=ethernet-service-oam&revision=2021-01-14</capability>

```

Figura 3.3: Comando di inizio sessione e primo elenco delle capabilities del server

```

<capability>http://www.calix.com/ns/router-isis?module=router-isis&revision=2020-12-14</capability>
<capability>http://www.calix.com/ns/router-ospf?module=router-ospf&revision=2021-03-03</capability>
<capability>http://www.calix.com/ns/router-ospf-lag-std?module=router-ospf-lag-std&revision=2020-11-20</capability>
<capability>http://www.calix.com/ns/router-ospf-std?module=router-ospf-std&revision=2020-11-20</capability>
<capability>http://www.calix.com/ns/router-ospf-vrf?module=router-ospf-vrf&revision=2021-04-27</capability>
<capability>http://www.calix.com/ns/router-pim?module=router-pim&revision=2020-11-10</capability>
<capability>http://www.calix.com/ns/router-plyc?module=router-plyc&revision=2020-12-14</capability>
<capability>http://www.calix.com/ns/router-rip?module=router-rip&revision=2020-12-14</capability>
<capability>http://www.calix.com/ns/rstp?module=rstp&revision=2020-01-10</capability>
<capability>http://www.calix.com/ns/rstp-std-eth?module=rstp-std-eth&revision=2020-11-17</capability>
<capability>http://www.calix.com/ns/rstp-std-lag?module=rstp-std-lag&revision=2020-07-08</capability>
<capability>http://www.calix.com/ns/sensor-bcm-switch?module=sensor-bcm-switch&revision=2018-08-17</capability>
<capability>http://www.calix.com/ns/vlan-interface-std?module=vlan-interface-std&revision=2021-04-28</capability>
<capability>http://www.calix.com/ns/vrf-routing?module=vrf-routing&revision=2021-04-27</capability>
<capability>http://www.calix.com/ns/wifi-interface?module=wifi-interface&revision=2018-08-17</capability>
<capability>http://www.calix.com/ns/wifi-interface-std?module=wifi-interface-std&revision=2019-12-09</capability>
<capability>http://www.verizon.com/ns/verizon-tpfix?module=verizon-tpfix&revision=2019-10-09</capability>
<capability>urn:dummy-common?module=talif-common-monitoring-ann</capability>
<capability>urn:dummy-ietf-interfaces-ann?module=ietf-interfaces-ann</capability>
<capability>urn:dummy-monitoring?module=ietf-monitoring-ann</capability>
<capability>urn:dummy2?module=talif-confd-monitoring-ann</capability>
<capability>urn:dummy3?module=ietf-netconf-notifications-ann</capability>
<capability>urn:dummy4?module=ietf-netconf-acm-ann</capability>
<capability>urn:dummy5?module=ietf-restconf-ann</capability>
<capability>urn:dummy7?module=ietf-restconf-monitoring-ann</capability>
<capability>urn:dummy8?module=ietf-yang-library-ann</capability>
<capability>urn:ietf:params:xml:yang:ietf-inet-types?module=ietf-inet-types&revision=2013-07-15</capability>
<capability>urn:ietf:params:xml:yang:ietf-interfaces?module=ietf-interfaces&revision=2014-05-08&amp;features=if-mib</capability>
<capability>urn:ietf:params:xml:yang:ietf-ipfix-psamp?module=ietf-ipfix-psamp&revision=2020-12-22&amp;features=udpTransport,timeoutCache,sctpTransport,psampRandoutOfN,psamp SampCountBased,psampFilterMatch,psampFilterHash,permanentCache,naturalCache,meter,immediateCache,fileWriter,fileReader</capability>
<capability>urn:ietf:params:xml:yang:ietf-netconf-acm?module=ietf-netconf-acm&revision=2012-02-22</capability>
<capability>urn:ietf:params:xml:yang:ietf-netconf-monitoring?module=ietf-netconf-monitoring&revision=2010-10-04</capability>
<capability>urn:ietf:params:xml:yang:ietf-netconf-notifications?module=ietf-netconf-notifications&revision=2012-02-06</capability>
<capability>urn:ietf:params:xml:yang:ietf-netconf-server?module=ietf-netconf-server&revision=2015-11-12&amp;features=ssh-call-home</capability>
<capability>urn:ietf:params:xml:yang:ietf-restconf-monitoring?module=ietf-restconf-monitoring&revision=2016-08-15</capability>
<capability>urn:ietf:params:xml:yang:ietf-yang-library?module=ietf-yang-library&revision=2016-06-21</capability>
<capability>urn:ietf:params:xml:yang:ietf-yang-smiv2?module=ietf-yang-smiv2&revision=2011-11-25</capability>
<capability>urn:ietf:params:xml:yang:ietf-yang-types?module=ietf-yang-types&revision=2013-07-15</capability>
<capability>urn:ietf:params:xml:netconf:base?module=ietf-netconf&revision=2011-06-01</capability>
<capability>urn:ietf:params:xml:yang:ietf-netconf-with-defaults?module=ietf-netconf-with-defaults&revision=2011-06-01</capability>
<capability>urn:ietf:params:xml:netconf:notification?module=ietf-notifications&revision=2008-07-14</capability>
</capabilities>
<session-id>172</session-id></hello>]]]>]]>
```

Figura 3.4: Terzo elenco delle capabilities del server

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
</hello>
]]>]]>
```

Figura 3.5: Hello Message lato client

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <get-config>
        <source>
            <running/>
        </source>
        <filter>
            <config xmlns="http://www.calix.com/ns/exa/base">
                <system>
                    <hostname></hostname>
                </system>
            </config>
        </filter>
    </get-config>
</rpc>
]]>]]><?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"><data><config
xmlns="http://www.calix.com/ns/exa/base"><system><hostname>E7-2-olt</hostname></system></config></data></rpc-reply>]]>]]>
```

Figura 3.6: <rpc> e <rpc-reply> per l'operazione di <get-config>

XML Viewer

Free Online XML Viewer & Formatter

AD:Bandwagon Host

The screenshot shows the XML Viewer interface. On the left, there is an 'XML Input' text area containing the XML code from Figure 3.6. Below it are several buttons: 'Browse', 'To Json', 'Format' (which is highlighted in blue), 'Minify', 'Tree View' (which is also highlighted in green), and 'Share'. On the right, the 'Tree View Result' pane displays a hierarchical tree structure of the XML data. The root node is 'rpc-reply', which contains a 'data' node. The 'data' node contains a 'config' node with attributes 'xmlns="http://www.calix.com/ns/exa/base"'. This 'config' node has a child 'system' node, which in turn has a child 'hostname' node with the value 'E7-2-olt'.

Figura 3.7: Visualizzazione di una <rpc-reply> attraverso Tree View

Operazione di <lock>

Prima di procedere con l'operazione di <edit-config>, per essere sicuri di essere gli unici a lavorare sul datastore scelto, effettuando tutte le modifiche necessarie indisturbati, si usa l'operazione

<lock>. In questo esempio, descritto in figura 3.8, è stato scelto di bloccare ad altri client NETCONF il datastore running, specificato nel campo target. Come si vede nella <rpc-reply>, non avendo eseguito una <get> o una <get-config>, si ha solo una conferma come risposta, indicata dalla keyword “ok”. Questa risposta indica anche che non sono stati commessi errori nel comporre il messaggio <rpc>. L’operazione di <lock> non è obbligatoria, ma quando si apportano modifiche è fortemente consigliata.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <lock>
    <target>
      <running/>
    </target>
  </lock>
</rpc>
]]>]]><?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2"><ok/></rpc-reply>]]>]]>
```

Figura 3.8: <rpc> e <rpc-reply> per l’operazione di <lock>

Operazione di <edit-config>

Ora che abbiamo il datastore riservato, possiamo finalmente procedere con l’operazione di <edit-config>, riportata in figura 3.9, per apportare la modifica dell’hostname, o in generale di qualsiasi altro parametro. All’interno del tag <target> specifichiamo il datastore su cui vogliamo lavorare, in questo caso il “running”, lo stesso riservato. Avremmo potuto anche scegliere il datastore “candidate”, senza impattare sul comportamento del dispositivo, per poi, attraverso l’operazione <commit>, riportare le azioni effettuate sul datastore “running”. Il valore che verrà assegnato al tag <hostname> è, come si intuisce, la stringa “E7-2-oltProva”. Ancora una volta la <rpc-reply> ci conferma che l’operazione effettuata è andata a buon fine (come ulteriore verifica, avremmo potuto ripetere la <get-config> chiamata precedentemente). Come per la <get-config>, è necessario conoscere il data model per indicare il path per raggiungere la “variabile”, più precisamente detta nodo, da modificare e/o interrogare.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <config xmlns="http://www.calix.com/ns/exa/base">
        <system>
          <hostname>E7-2-oltProva</hostname>
        </system>
      </config>
    </config>
  </edit-config>
</rpc>
]]>]]><?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3"><ok/></rpc-reply>]]>
```

Figura 3.9: <rpc> e <rpc-reply> per l’operazione di <edit-config>

Operazione di <unlock>

Una volta completate tutte le operazioni necessarie, per rilasciare il datastore riservato precedentemente attraverso l'operazione di <lock>, permettendo ad altri client NETCONF di lavorarci, è richiesta l'operazione duale di <unlock>. All'interno del tag <target> va indicato il datastore precedentemente riservato, nel nostro caso il “running”. Se si prova a terminare l'operazione di <lock>, effettuata da un'altra sessione NETCONF, l'operazione di <unlock> non andrà a buon fine, e si otterrà una <rpc-reply> che indicherà la presenza di un errore. Nel nostro esempio la risposta è positiva, e lo capiamo dalla solita keyword “ok”. La figura 3.10 mostra l'operazione di <unlock> eseguita, e la relativa <rpc-reply> di conferma.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="4" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <unlock>
    <target>
      <running/>
    </target>
  </unlock>
</rpc>
]]]><?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="4"><ok/></rpc-reply>]]>]]>
```

Figura 3.10: <rpc> e <rpc-reply> per l'operazione di <unlock>

Operazione di <close-session>

Concluse tutte le operazioni e le modifiche necessarie, terminiamo la sessione attraverso l'operazione di <close-session>, riportata in figura 3.11. Ancora una volta l'operazione si conclude correttamente, e si riceve una <rpc-reply> contenente la keyword “ok”. Si osservi come nell'esempio, per l'attributo obbligatorio “message-id” delle diverse <rpc>, sono stati scelti numeri consecutivi, crescenti monotonicamente. È una scelta non casuale, in quanto fortemente raccomandata nell'rfc6241, ma non obbligatoria.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="5" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <close-session/>
</rpc>
]]]><?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5"><ok/></rpc-reply>]]>]]>stefano@stefano:~$ █
```

Figura 3.11: <rpc> e <rpc-reply> per l'operazione di <close-session>

3.2.4 Tool per l'utilizzo di NETCONF

In seguito all'esempio riportato, va precisato che non è questa la modalità migliore di utilizzare NETCONF, ma è servito solo come scenario chiarificatore per comprenderne il funzionamento. Per facilitare l'utilizzo del protocollo, lavorando ad un livello applicativo più alto, sono stati ideati diversi tool. Nel presente lavoro ne ho utilizzati due: Advanced NETCONF Explorer e la libreria ncclient di Python. Grazie a questi strumenti si evitano molti errori, soprattutto nella fase di codifica XML per costruire le <rpc>, per esempio dovuti ad annidamenti errati.

Advanced NETCONF Explorer

Advanced NETCONF Explorer è un esploratore grafico, utile per gestire i modelli YANG supportati da un dispositivo controllabile attraverso il protocollo NETCONF [8]. Esso fornisce alcune funzionalità molto utili, tra cui:

- Mostra e permette di scaricare tutti i modelli YANG supportati dal dispositivo
 - Consente di analizzare i modelli YANG fornendo la struttura ad albero del modello, con tutti i nodi che l'utente può espandere o comprimere, a seconda della sezione che vuole studiare
 - Permette di filtrare la struttura ad albero del modello attraverso il nome del modulo e cercando i nomi e le descrizioni dei nodi YANG presenti
 - Consente di visualizzare e generare metadata per un nodo YANG, come la descrizione, il path, e il subtree-filter, utili per strutturare le NETCONF <rpc>

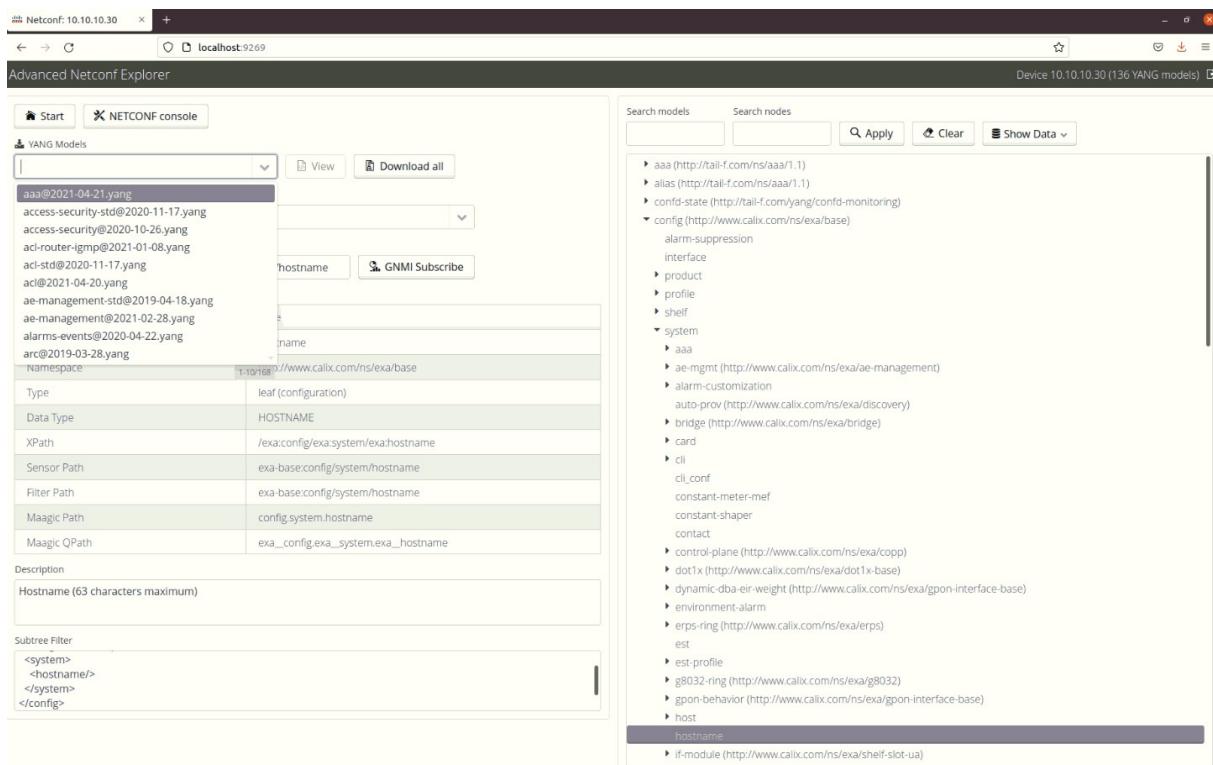


Figura 3.12: Cattura dell’interfaccia fornita da Advanced NETCONF Explorer

Nella figura 3.12 è riportata una cattura dell’interfaccia che Advanced NETCONF Explorer offre agli utenti. In fase di ricerca è possibile selezionare il datastore da analizzare tra: Operational, Running, Startup e Candidate (solo se supportato). Nella sezione di sinistra della finestra, oltre al path espresso in diversi formati, al subtree-filter e alla descrizione del nodo, già anticipati, sono riportati in una tabella anche altri parametri come namespace, type (se leaf, container, ecc), e nome. Grazie alla console NETCONF che esso fornisce, è possibile formulare direttamente <rpc> come <get> ed <edit-config>, per interagire con il dispositivo, scrivendo pochissimo testo in XML o addirittura, in alcuni casi, senza dover scrivere nulla, trovando il messaggio già preparato. Si evitano anche errori dovuti a copia e incolla, da editor di testo a terminale, che si possono commettere

lavorando con il protocollo NETCONF seguendo l'approccio mostrato nell'esempio precedente. La cattura 3.13 mostra un esempio di <get> effettuata direttamente tramite NETCONF console, per la richiesta del parametro “hostname”. Come si osserva, non è stato necessario né lo scambio del messaggio preliminare di <hello>, né di quello conclusivo di <close-session>. Oltre a non aver dovuto scrivere neanche una riga in XML, non è stato necessario neanche l'inserimento manuale dell'attributo “message-id” nella fase di strutturazione della <rpc>. Si intuisce come l'utilizzo del protocollo, grazie all'utilizzo di Advanced NETCONF Explorer sia molto più semplice, intuitivo e veloce, rispetto all'approccio base da terminale.

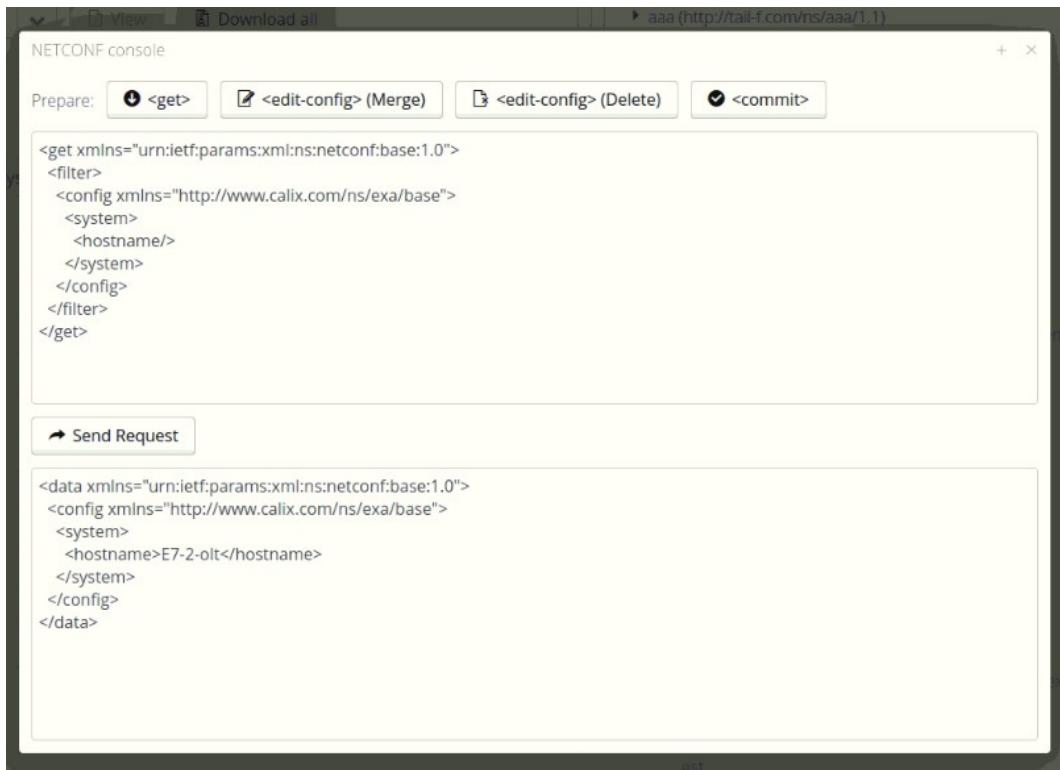


Figura 3.13: Esempio di <get> con Advanced NETCONF Explorer

Libreria Python ncclient

Realizzata da Shikar Bhushan ed attualmente mantenuta da Leonidas Poulopoulos, la libreria ncclient di Python, facilita l'utilizzo del protocollo NETCONF, semplificando il processo di sviluppo di script e applicazioni lato client [22]. Supporta tutte le operazioni e le capabilities definite nell'rfc6241, rendendo l'operazione di scrittura dei cosiddetti network-management script, molto più semplice. Anche in questo approccio si ricorre al formato XML utilizzato dal protocollo per codificare le <rpc> e le <rpc-reply>, solo quando strettamente necessario, per esempio in una <get-config>, per specificare il tag <filter>. Come nel caso di Advanced NETCONF Explorer, non bisogna preoccuparsi di inserire manualmente l'attributo obbligatorio delle <rpc> “id-message”, mentre per instaurare e chiudere una sessione esistono dei metodi dedicati. Optando per questa soluzione, il numero di errori che si verificano in fase di creazione dei messaggi, si riduce notevolmente. Nel lavoro sperimentale, è stata scelta questa alternativa, che consente, insieme a tutte le altre funzionalità che il linguaggio Python offre, di realizzare veri e propri programmi per la gestione ed il controllo

della rete. Infatti, l'utilizzo dei cicli, delle istruzioni condizionali (come if ed else) e delle strutture dati, tipiche di un linguaggio di programmazione, permettono di realizzare codici elaborati, senza limitarsi ad una <rpc> per volta, come nei due esempi illustrati precedentemente di Advanced NETCONF Explorer e dell'utilizzo classico del protocollo tramite terminale. Seguendo questo approccio, le operazioni di monitoraggio e gestione della rete, sono a tutti gli effetti automatizzate. Come si vedrà nella validazione sperimentale, oltre al vantaggio di essere processi automatizzati, sono eseguiti praticamente in tempo reale, con tempi di risposta più che contenuti. Grazie a questa caratteristica, la rete guadagna in termini di programmabilità, flessibilità e prontezza, indispensabili nel nuovo paradigma di rete. Oltre ai singoli dispositivi, diventa quindi possibile configurare direttamente il servizio sulla rete.

```

from ncclient import manager
import xml.dom.minidom
import xml.etree.ElementTree as ET

# Settings
HOST = "10.10.10.30"
PORT = 22
USERNAME = "sysadmin"
PASSWORD = "sysadmin"
m = manager.connect_ssh(
    host=HOST,
    port=PORT,
    username=USERNAME,
    password=PASSWORD
)

QUERY_HOSTNAME = '''
<filter>
<config xmlns="http://www.calix.com/ns/exa/base">
    <system>
        <hostname></hostname>
    </system>
</config>
</filter>
'''

netconf_reply = m.get(filter=QUERY_HOSTNAME)

print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml()) #stampo la <rpc
> nel classico formato XML

root1 = ET.fromstring(netconf_reply.xml)
for hostname in root1.findall('.//{*}hostname'):
    print(hostname.text) # stampo solo il parametro di cui ho bisogno

m.close_session()

```

Listing 3.2: Esempio di una <get> con la libreria ncclient di Python

Nell'esempio di codice riportato per contestualizzare l'argomento, si richiede al server NETCONF, caratterizzato da indirizzo IP "10.10.10.30", porta "22" (porta utilizzata dal protocollo ssh), userna-

me "sysadmin" e password "sysadmin", il valore del parametro "hostname", come fatto nella sezione dedicata al tool Advanced NETCONF Explorer. Dopo aver dichiarato le variabili che accolgono questi dati del dispositivo, attraverso il modulo manager di ncclient, ed il suo metodo "connect_ssh", è stata instaurata la sessione NETCONF, indicata con la variabile m. Come anticipato, il formato XML, passato come stringa alla funzione get, è stato utilizzato solo per istanziare il parametro "filter", richiedendo così solo il valore dell'hostname. Per visualizzare la <rpc-reply> fornita dal server NETCONF, in maniera ordinata o solo il parametro richiesto senza il subtree completo, Python mette a disposizione diversi tool come, il metodo "parseString" della libreria "xml.dom.minidom", e i metodi "fromstring" e "findall" della libreria "xml.etree.ElementTree", rispettivamente. Per terminare la sessione, si invoca il metodo "close_session". La cattura 3.14 mostra l'output dello script Python riportato precedentemente.

```

get_hostname
/home/stefano/PycharmProjects/QueryFlowCounters/venv/bin/python /home/stefano/PycharmProjects/QueryFlowCounters/get_hostname.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:0e8ec83e-5da4-4080-b723-a10f1f51bb73">
  <data>
    <config xmlns="http://www.calix.com/ns/exa/base">
      <system>
        <hostname>E7-2-olt</hostname>
      </system>
    </config>
  </data>
</rpc-reply>

E7-2-olt

Process finished with exit code 0

```

Figura 3.14: Output della <get> utilizzando la libreria Python ncclient

3.3 YANG

3.3.1 Introduzione

Con la crescente adozione del protocollo NETCONF, divenne necessario definire un linguaggio di modellazione dei dati per completarlo. Sebbene alcuni linguaggi di modellazione già esistessero, come UML ed XML Schema, per citarne alcuni, nessuno di essi si sposava con NETCONF. Il motivo principale è che essi non erano facilmente leggibili e comprensibili, e non fornivano meccanismi per convalidare i modelli di dati di configurazione [24]. Venne allora sviluppato il linguaggio di modellazione dati Yet Another Next Generation (YANG), descritto nell'rfc6020 rilasciato nel settembre del 2010 da NETMOD¹, aggiornato poi in YANG 1.1, in seguito all'rfc7950 pubblicato nel 2016. Esso è caratterizzato da alcune funzionalità chiave per "andare a braccetto" con NETCONF, e le principali sono:

- Rappresentazioni leggibili e facili da imparare
- Modelli di dati di configurazione gerarchici
- Vincoli formali per la validazione della configurazione
- Modularità dei dati attraverso moduli e sottomoduli

¹NETMOD: gruppo di lavoro dell'IETF

- Tipi e raggruppamenti riutilizzabili
- Supporta la definizione di operazioni <rpc>

3.3.2 Il Linguaggio

Originariamente ideato per modellare i dati manipolati dal protocollo NETCONF, YANG consente una completa descrizione dei dati scambiati tra un client ed un server [10]. Può essere utilizzato per modellare sia i dati di stato che di configurazione manipolati da NETCONF, le NETCONF <rpc> e le notifiche NETCONF. YANG definisce il modello in maniera modulare, rappresentando la struttura dati tramite un modello ad albero, in cui ogni nodo ha un nome e può avere o un valore o altri nodi detti “figli”, fornendo una descrizione chiara e concisa, di essi e della loro interazione. La struttura dei modelli di dati si sviluppa in moduli, i quali possono importare dati da altri moduli esterni e includere dati da sottomoduli. Come molti linguaggi di programmazione, YANG definisce un insieme di tipi di dato predefiniti (come boolean, decimal64, bits, ...), ma allo stesso tempo fornisce un meccanismo attraverso cui è possibile definire tipi addizionali. Le istanze di dati, pur dovendo seguire i modelli YANG, possono essere codificate in qualsiasi formato che il protocollo di configurazione di rete supporta, ad esempio XML e JSON. Un aspetto interessante è che i moduli YANG possono essere tradotti in una sintassi XML equivalente, chiamata YANG Independent Notation (YIN), consentendo alle applicazioni che utilizzano parser XML e script XSLT, di manipolarli. Questa operazione di conversione tra YANG e YIN (e viceversa), attraverso alcune regole di mappatura tra i due formati, avviene senza perdite, in modo da non modificare il contenuto informativo del modello. Entrambi i formati, pur utilizzando notazioni diverse, contengono informazioni equivalenti. Grazie a questa proprietà, gli sviluppatori possono utilizzare la notazione equivalente YIN per sfruttare un’ampia gamma di funzionalità basate su XML, per esempio per il filtraggio e la validazione dei dati, la generazione automatica di codice, ed altre ancora.

Assodato che il modello sia definito in moduli, possiamo descrivere formalmente un modulo, come l’unità di base di definizione in YANG. Nel definirlo sono previsti tre tipi di statement:

- header: descrive il modulo fornendo alcune informazioni illustrate (per esempio yang-version, namespace, organization, description)
- revision: fornisce informazioni sulla “vita” del modulo, individuandone la versione attraverso una data
- definition: è il corpo del modulo, dove viene definito effettivamente il modulo

Un modulo può includere al suo interno un qualsiasi numero di sottomoduli, che per questo sono definiti come moduli parziali, che contribuiscono alla definizione di un modulo. Attraverso l’istruzione “include”, il contenuto di un sottomodulo diventa disponibile per quel modulo, che viene detto “padre”, mentre l’istruzione “import” rende le definizioni di un modulo disponibili all’interno di un altro modulo o sottomodulo.

YANG definisce quattro tipi di nodi principali, dove per nodi intendiamo gli elementi che contengono i dati. Si distinguono in:

- Leaf: L’istanza di una foglia contiene tipi di dati semplici, come una stringa o un intero. Non possiede nodi figli, ma assume esattamente un valore di un particolare tipo

- Leaf-list: Contiene una sequenza di nodi di tipo Leaf (tutti dello stesso tipo)
- Container: Un nodo contenitore è utilizzato per raggruppare nodi correlati in un sottoalbero, non ha un valore, ma ha una lista di nodi figli. Il numero dei nodi figli può essere qualsiasi, così come il loro tipo
- List: Contiene una sequenza di elementi, ognuno dei quali identificato univocamente attraverso una o più leaf dette "key leaf"

3.3.3 Descrizione di un modulo YANG e utilizzo del tool pyang

Per rendere più chiara ed intuitiva la descrizione di YANG data model, è stato creato un modulo YANG molto semplice ma che tocca più o meno tutti gli argomenti descritti formalmente, chiarificandone la trattazione. Esso, attraverso il linguaggio YANG, descrive semplici interfacce Ethernet, e dopo essere stato analizzato velocemente, sarà utilizzato per mostrare alcune funzioni interessanti, messe a disposizione da pyang.

Descrizione del modulo YANG "semplici-interfacceETH"

All'interno del modulo "semplici-interfacceETH", è possibile notare da subito l'header in cui si specifica la versione YANG, il namespace, il prefisso, l'organizzazione, il contatto (ad esempio per il supporto) e la descrizione del modulo. In questo caso l'ultima revisione coincide con la prima, ma è buona norma, per ogni modifica che viene apportata al modulo, aggiungere un nuovo statement "revision" che riporti la data della modifica e una sua descrizione. Proseguendo nell'analisi, si arriva alla definizione del modulo vera e propria. Per farlo, è stato necessario però definire prima dei tipi, non presenti tra quelli "built-in" che YANG supporta, per descrivere gli indirizzi IPv4, MAC e data e ora. Questo approccio è stato seguito per fornire una descrizione completa, passo passo, partendo da zero, su come costruire un modulo YANG, ma avremmo potuto evitare questa operazione preliminare, come avviene in realtà, ricorrendo allo statement "import", importandoli direttamente dal modulo "ietf-yang-types", in cui sono già definiti. Questo concetto spiega la modularità del linguaggio: per descrivere una qualsiasi entità, come un'interfaccia, un algoritmo di instradamento, o qualsiasi altra cosa, non è necessario scrivere da zero l'intero modulo, ma è possibile importarne alcuni già sviluppati. Essi sono implementati soprattutto da organizzazioni internazionali del settore, come IANA², IEEE³, ETSI⁴ e, come nel nostro caso, IETF⁵. Tuttavia, grandi aziende come Cisco, Nokia, Huawei, Juniper, per citarne alcune, integrano questi moduli standardizzati, con moduli e quindi funzionalità aggiuntive per i dispositivi da loro prodotti, proprietari e specifici. La maggior parte di questi moduli, come già detto nella sezione dedicata alla descrizione dell'hello-message da parte del server, possono essere ispezionati e scaricati online al seguente link: <https://github.com/YangModels/yang>. Un'interfaccia Ethernet, volendo ridurla all'osso, è caratterizzata da un nome, da un indirizzo MAC, da un indirizzo IP e dalla subnet-mask, e dal suo stato di attività. Lo stato di un'interfaccia, volendo sempre semplificare la trattazione al massimo, è descritto, oltre che dal nome che la identifica, dallo stato operativo e dalla data in cui

²IANA: Internet Assigned Numbers Authority

³IEEE: Institute of Electrical and Electronics Engineers

⁴ETSI: European Telecommunications Standards Institute

⁵IETF: l'Internet Engineering Task

è entrata nello stato operativo attuale. Tutti questi parametri sono dei semplici nodi leaf, ma di tipo diverso, sia “built-in”, che appositamente realizzati (o importati), come boolean, enumeration, date-and-time, ecc. Essi sono a loro volta all’interno dei nodi interfacciaETH ed interface-state, entrambi nodi list. La differenza principale tra i due è che uno ha al suo interno dati di configurazione, quindi “read-write”, mentre il secondo ha solo dati operativi che possono essere letti, ma non scritti, detti “read-only”. Attraverso YANG posso distinguere, e lo si fa attraverso lo statement “config false”, utilizzato in interface-state. In entrambi i casi la key leaf è la leaf “nome”, che identifica e permette di distinguere i diversi elementi della lista, e per questo deve essere univoca (non possono esserci due interfacce che abbiano lo stesso nome). È consigliato inserire sempre una descrizione, e magari anche un esempio di inserimento, per permettere a chi legge di comprendere al meglio il significato di ogni nodo. Lo statement “mandatory true” specifica per un nodo leaf che lo contiene, che va assegnato obbligatoriamente un valore a quel campo, che non può essere lasciato vuoto. Lo statement “mandatory” non è obbligatorio, e dove assente, nel nostro modulo nella leaf description, ad esempio, assume il valore “false” di default. Quindi, nel caso specifico si può decidere di non inserire alcuna descrizione dell’interfaccia, mentre nel caso generico è possibile lasciare il campo vuoto, senza dover completarlo obbligatoriamente. Lo statement default “false” assegna il valore logico false se non viene specificato esplicitamente il valore “true”, mentre lo statement “status current”, specifica che la definizione in esame è corrente e valida. Al posto di “current” avremmo potuto trovare “deprecated” o “obsolete”. Sia interfacciaETH che interface-state sono contenuti all’interno del nodo container “interfacceETH”. È bene sottolineare che i moduli YANG definiscono solamente la struttura dei dati operativi e di configurazione, e che quindi non contengono dei valori istanziati di tali dati. Oltre allo statement “import”, nello specificare i tipi di dati importati che un nodo richiede, non è sufficiente scrivere “type mac-address;” come nell’esempio riportato, ma va utilizzata la notazione “type yang:mac-address;”, specificando il prefisso, in questo caso “yang”, del modulo importato. Di seguito si riporta il modulo YANG semplici-interfacceETH, appositamente creato.

```

module semplici-interfacceETH {
    yang-version 1;
    namespace
        "http://example.com/yang/semplici-interfacceETH";
    prefix if;
    organization
        "Stefano Hinic";
    contact
        "Support: stefano.hinic@student.univaq.it";
    description
        "Esempio di una semplice interfaccia Ethernet";
    revision "2021-11-25" {
        description
            "Prima versione";
    }
    typedef mac-address {
        type string {
            pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
        }
        description
            "Tipo per rappresentare un indirizzo MAC IEEE 802.";
    }
    typedef dotted-quad {
        type string {
            pattern '(([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}|' +
                    '([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
        }
        description
            "Tipo per rappresentare un indirizzo ipv4.";
    }
    typedef date-and-time {
        type string {
            pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d+)?|' +
                    '(Z|\+\-\d{2}:\d{2})';
        }
    }
}
```

```

        }
        description
        "Tipo per rappresentare data e ora.";
    }
    container interfacceETH {
        list interfacciaETH {
            key "nome";
            leaf nome {
                type string;
                mandatory "true";
                description
                    "Nome dell'interfaccia, esempio: FastEthernet 0/0";
            }
            leaf description {
                type string;
                description
                    "Descrizione testuale dell'interfaccia";
            }
            leaf indirizzo-MAC {
                type mac-address;
                mandatory "true";
                description
                    "Indirizzo MAC dell'interfaccia. Esempio: aa:bb:cc:dd:ee:ff";
            }
            leaf indirizzo-IP {
                type dotted-quad;
                mandatory "true";
                description
                    "Indirizzo IP dell'interfaccia. Esempio: 10.20.20.1";
            }
            leaf subnet-mask {
                type dotted-quad;
                mandatory "true";
                description
                    "Subnet mask dell'interfaccia. Esempio: 255.255.255.0";
            }
            leaf enabled {
                type boolean;
                default "false";
                description
                    "Indica lo stato di attivita dell'interfaccia. Esempio: true";
            }
        }
        list interface-state {
            config false;
            key "nome";
            leaf nome {
                type string;
                description
                    "Nome dell'interfaccia, esempio: FastEthernet 0/0";
            }
            leaf oper-status {
                type enumeration {
                    enum up;
                    enum down;
                    enum testing;
                }
                mandatory "true";
                description
                    "Indica lo stato operativo dell'interfaccia. Esempio: up ";
            }
            leaf last-change {
                type date-and-time;
                status current;
                description
                    "L'ora in cui l'interfaccia era entrata nel suo attuale stato operativo. Esempio: 2021-11-25T19
                     :30:10.345+09:00";
            }
        }
    }
}

```

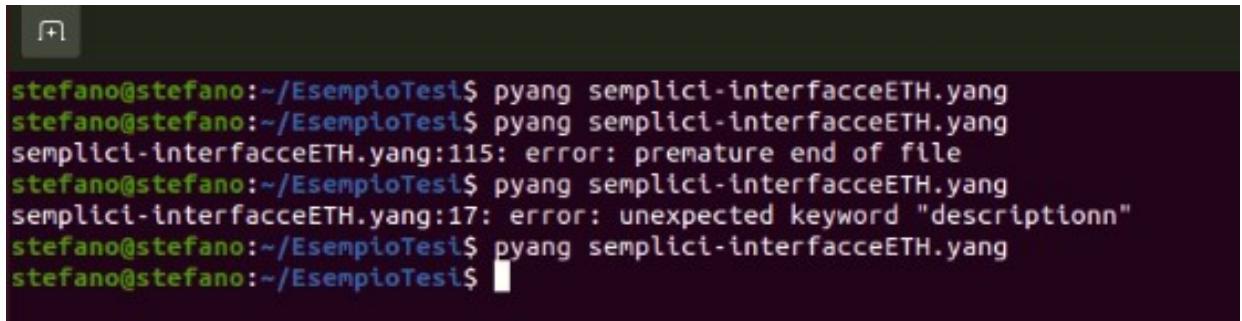
Listing 3.3: semplici-interfacceETH.yang

Il tool pyang

Il tool pyang, scritto in Python, è uno strumento da linea di comando che permette, tra le diverse funzioni, di validare la correttezza di moduli YANG, di trasformarli in altri formati e di generarne

il modello ad albero. Lo utilizzeremo per validare il modulo creato e descritto precedentemente “semplici-interfacceETH”, ne analizzeremo il modello ad albero, ne genereremo l’XML con due diverse opzioni che lo strumento ci offre, e lo YIN. Per concludere, vedremo se un testo XML (che per esempio potrebbe essere necessario in una NETCONF <rpc>) rispetta le regole imposte dal modulo YANG associato.

Validazione modulo YANG con pyang Attraverso il solo comando `pyang`, seguito dal nome del modulo, come mostrato in figura 3.15, si effettua la sua validazione. Se non sono presenti errori, il metodo non restituirà nulla, e sarà dunque corretto. Viceversa, qualora ci siano errori nel modulo, questa funzione ne segnalerà la presenza e darà indicazioni utili per individuarli. Nell’esempio, la prima verifica è andata a buon fine. Nella seconda, la risposta segnala un errore dovuto alla mancanza di una parentesi graffa a conclusione del modulo. Nella terza, simulando un errore nella sintassi dello statement “description”, la funzione lo individua e ce lo riporta. Correggendo questi errori appositamente commessi, ma in uno scenario reale abbastanza frequenti, l’ultima verifica non restituisce output: il modulo è di nuovo sintatticamente corretto.



```
stefano@stefano:~/EsempioTesti$ pyang semplici-interfacceETH.yang
stefano@stefano:~/EsempioTesti$ pyang semplici-interfacceETH.yang
semplici-interfacceETH.yang:115: error: premature end of file
stefano@stefano:~/EsempioTesti$ pyang semplici-interfacceETH.yang
semplici-interfacceETH.yang:17: error: unexpected keyword "descriptionnn"
stefano@stefano:~/EsempioTesti$ pyang semplici-interfacceETH.yang
stefano@stefano:~/EsempioTesti$
```

Figura 3.15: Esempio di validazione del modulo YANG con `pyang`

Generazione del modello ad albero Effettuata la validazione del modulo, ed essendo sicuri della sua correttezza, procediamo con la generazione del modello ad albero, riportato nella cattura 3.16. Esso ci permette di comprenderne più agevolmente la gerarchia dei nodi, e fornisce informazioni descrittive su ognuno di essi. Da come si vede in figura, la prima informazione riportata è il tipo, come string, boolean, ed enumeration “built-in” di YANG, e mac-address, dotted-quad e date-and-time appositamente creati o più intelligentemente importati. La seconda informazione è la caratteristica del dato, se “read-write” per i dati di configurazione contrassegnati con la sigla “rw”, o “read-only” per i dati operativi, contraddistinti dalla sigla “ro”. Il campo tra parentesi quadre “nome”, ci indica la key leaf, mentre il punto interrogativo vicino a description, ad enabled e a last-change, specifica che l’assegnazione di un valore a quei nodi è facoltativo, non obbligatorio. Nel caso delle key leaf, nel nostro esempio “nome”, gli statement mandatory, così come anche qualsiasi tipo di valore di default, sono ignorati, pertanto non è necessario inserirli.

Generazione dei formati XML e YIN A questo punto attraverso le funzioni che `pyang` ci offre, generiamo la codifica in XML del modulo, sia con i valori di default, nel primo caso, che con le annotazioni nel secondo. Non sono le due uniche opzioni che questo tool offre, ma forse le più utili per i nostri scopi. Il vantaggio del primo è che appunto sono già inseriti i valori di default, dove

```
stefano@stefano:~/EsempioTesi$ pyang -f tree semplici-interfacceETH.yang
module: semplici-interfacceETH
  +-rw interfacceETH
    +-rw interfacciaETH* [nome]
      | +-rw nome          string
      | +-rw description?  string
      | +-rw indirizzo-MAC mac-address
      | +-rw indirizzo-IP  dotted-quad
      | +-rw subnet-mask   dotted-quad
      | +-rw enabled?      boolean
    +-ro interface-state* [nome]
      +-ro nome          string
      +-ro oper-status   enumeration
      x--ro last-change? date-and-time
stefano@stefano:~/EsempioTesi$
```

Figura 3.16: Generazione del modello ad albero del modulo YANG con pyang

presenti. Nel secondo invece, le annotazioni ci guidano nell'inserimento, suggerendoci ad esempio, il tipo del valore che un nodo si aspetta. Inoltre, è stato generato anche il modulo YIN equivalente, utile per ulteriori validazioni ed elaborazioni con l'uso di tool che utilizzano parser XML e script XSLT. I comandi usati per generarli, sono riportati nella figura 3.17.

```
stefano@stefano:~/EsempioTesi$ pyang -f sample-xml-skeleton --sample-xml-skeleton-defaults -o interfacceDefault.xml semplici-interfacceETH.yang
stefano@stefano:~/EsempioTesi$ pyang -f sample-xml-skeleton --sample-xml-skeleton-annotations -o interfacceConAnnotazioni.xml semplici-interfacceETH.yang
stefano@stefano:~/EsempioTesi$ pyang -f yln -o semplici-interfacceETH.yln semplici-interfacceETH.yang
stefano@stefano:~/EsempioTesi$
```

Figura 3.17: Comandi per la generazione del modulo YANG in altri formati con pyang
Per visualizzarli, ed osservarne le differenze, sono riportati i moduli prima in formato XML con l'opzione "defaults", poi in XML con l'opzione "annotations", ed infine in formato YIN.

```
<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfacceETH xmlns="http://example.com/yang/semplici-interfacceETH">
    <interfacciaETH>
      <name/>
      <description/>
      <indirizzo-MAC/>
      <indirizzo-IP/>
      <subnet-mask/>
      <enabled>false</enabled>
    </interfacciaETH>
    <interface-state>
      <name/>
      <oper-status/>
      <last-change/>
    </interface-state>
  </interfacceETH>
</data>
```

Listing 3.4: File interfacceDefault.xml generato con pyang

```
<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfacceETH xmlns="http://example.com/yang/semplici-interfacceETH">
    <interfacciaETH>
      <!-- # keys: nome -->
      <!-- # entries: 0.. -->
      <name>
        <!-- type: string -->
      </name>
      <description>
        <!-- type: string -->
      </description>
      <indirizzo-MAC>
        <!-- type: mac-address -->
      </indirizzo-MAC>
```

```

<indirizzo-IP>
    <!-- type: dotted-quad -->
</indirizzo-IP>
<subnet-mask>
    <!-- type: dotted-quad -->
</subnet-mask>
</interfacciaETH>
<interface-state>
    <!-- # keys: nome -->
    <!-- # entries: 0.. -->
<nome>
    <!-- type: string -->
</nome>
<oper-status>
    <!-- type: enumeration -->
</oper-status>
<last-change>
    <!-- type: date-and-time -->
</last-change>
</interface-state>
</interfacceETH>
</data>

```

Listing 3.5: File interfacceConAnnotazioni.xml generato con pyang

```

<?xml version="1.0" encoding="UTF-8"?>
<module name="semplici-interfacceETH"
        xmlns="urn:ietf:params:xml:ns:yang:yin:1"
        xmlns:if="http://example.com/yang/semplici-interfacceETH">
<yang-version value="1"/>
<namespace uri="http://example.com/yang/semplici-interfacceETH"/>
<prefix value="if"/>
<organization>
    <text>Stefano Hinic</text>
</organization>
<contact>
    <text>Support: stefano.hinic@student.univaq.it</text>
</contact>
<description>
    <text>Esempio di una semplice interfaccia Ethernet</text>
</description>
<revision date="2021-11-25">
    <description>
        <text>Prima versione</text>
    </description>
</revision>
<typedef name="mac-address">
    <type name="string">
        <pattern value="[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}" />
    </type>
    <description>
        <text>Tipo per rappresentare un indirizzo MAC IEEE 802.</text>
    </description>
</typedef>
<typedef name="dotted-quad">
    <type name="string">
        <pattern value="(([0-9][1-9][0-9][1[0-9][0-9][2[0-4][0-9][25[0-5]) \. )"
                  {3}([0-9][1-9][0-9][1[0-9][0-9][2[0-4][0-9][25[0-5]) />
    </type>
    <description>
        <text>Tipo per rappresentare un indirizzo ipv4.</text>
    </description>
</typedef>
<typedef name="date-and-time">
    <type name="string">
        <pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d+)?(Z|[+\-]\d{2}:\d{2})"/>
    </type>
    <description>
        <text>Tipo per rappresentare data e ora.</text>
    </description>
</typedef>
<container name="interfacceETH">
    <list name="interfacciaETH">
        <key value="nome"/>
        <leaf name="nome">
            <type name="string"/>
            <mandatory value="true"/>
            <description>
                <text>Nome dell'interfaccia, esempio: FastEthernet 0/0</text>
            </description>
        </leaf>
        <leaf name="description">

```

```

<type name="string"/>
<description>
    <text>Descrizione testuale dell'interfaccia</text>
</description>
</leaf>
<leaf name="indirizzo-MAC">
    <type name="mac-address"/>
    <mandatory value="true"/>
    <description>
        <text>Indirizzo MAC dell'interfaccia. Esempio: aa:bb:cc:dd:ee:ff</text>
    </description>
</leaf>
<leaf name="indirizzo-IP">
    <type name="dotted-quad"/>
    <mandatory value="true"/>
    <description>
        <text>Indirizzo IP dell'interfaccia. Esempio: 10.20.20.1</text>
    </description>
</leaf>
<leaf name="subnet-mask">
    <type name="dotted-quad"/>
    <mandatory value="true"/>
    <description>
        <text>Subnet mask dell'interfaccia. Esempio: 255.255.255.0</text>
    </description>
</leaf>
<leaf name="enabled">
    <type name="boolean"/>
    <default value="false"/>
    <description>
        <text>Indica lo stato di attivita dell'interfaccia. Esempio: true</text>
    </description>
</leaf>
</list>
<list name="interface-state">
    <config value="false"/>
    <key value="nome"/>
    <leaf name="nome">
        <type name="string"/>
        <description>
            <text>Nome dell'interfaccia, esempio: FastEthernet 0/0</text>
        </description>
    </leaf>
    <leaf name="oper-status">
        <type name="enumeration">
            <enum name="up"/>
            <enum name="down"/>
            <enum name="testing"/>
        </type>
        <mandatory value="true"/>
        <description>
            <text>Indica lo stato operativo dell'interfaccia. Esempio: up </text>
        </description>
    </leaf>
    <leaf name="last-change">
        <type name="date-and-time"/>
        <status value="current"/>
        <description>
            <text>L'ora in cui l'interfaccia era entrata nel suo attuale stato operativo. Esempio:
                2021-11-25T19:30:10.345+09:00</text>
        </description>
    </leaf>
</list>
</container>
</module>

```

Listing 3.6: File semplici-interfacceETH.yin generato con pyang

Completamento dell'XML con parametri e validazione finale Adesso, avendo già l'XML ben costruito a disposizione, senza errori, come i comuni annidamenti errati, proseguiamo la trattazione riempiendo i tag con dei valori, per poi sfruttare il tool per validarne la correttezza in riferimento al modulo YANG. Descriviamo due interfacce “FastEthernet 0/0” e “FastEthernet 0/1” con i rispettivi valori per i nodi leaf nome, description, indirizzo-MAC, indirizzo-IP, subnet-mask ed enabled, di interfacciaETH, e per i nodi leaf nome, oper-status, e last-change di interface-state.

```

<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfacceETH xmlns="http://example.com/yang/semplice-interfacceETH">
    <interfacciaETH>
      <nome>FastEthernet 0/0</nome>
      <description>Interfaccia di prova1</description>
      <indirizzo-MAC>aa:bb:cc:dd:ee:ff</indirizzo-MAC>
      <indirizzo-IP>10.20.30.1</indirizzo-IP>
      <subnet-mask>255.255.255.0</subnet-mask>
      <enabled>true</enabled>
    </interfacciaETH>
    <interfacciaETH>
      <nome>FastEthernet 0/1</nome>
      <description>Interfaccia di prova2</description>
      <indirizzo-MAC>aa:bb:cc:dd:fe:ef</indirizzo-MAC>
      <indirizzo-IP>10.20.20.1</indirizzo-IP>
      <subnet-mask>255.255.255.0</subnet-mask>
      <enabled>true</enabled>
    </interfacciaETH>
    <interface-state>
      <nome>FastEthernet 0/0</nome>
      <oper-status>up</oper-status>
      <last-change>2021-11-25T22:11:10.000+00:00</last-change>
    </interface-state>
    <interface-state>
      <nome>FastEthernet 0/1</nome>
      <oper-status>up</oper-status>
      <last-change>2021-11-25T22:11:10.000+00:00</last-change>
    </interface-state>
  </interfacceETH>
</data>

```

Listing 3.7: File sempliceProva.xml generato con pyang

A questo punto, possiamo procedere con la validazione, mostrata in figura 3.18, per essere sicuri di non aver commesso errori, prima di procedere con una NETCONF <rpc>, ad esempio. Nel primo caso, la validazione va a buon fine, in quanto viene restituita dal metodo la stringa “No errors found”, e possiamo essere sicuri che il testo XML è ben formattato, senza errori sintattici. Nei due tentativi successivi, in seguito al voluto inserimento di un indirizzo MAC non valido (aa:bb:cc:dd:ee:fg) prima, e di una subnet-mask errata (255.255.256.0) poi, oltre alla stringa “sempliceProva.xml fails to validate”, sono riportate informazioni che permettono di individuare immediatamente l’errore, come il numero di riga e l’elemento che lo ha generato.

```

stefano@stefano:~/EsempioTest$ yang2dsdl -v sempliceProva.xml semplici-interfacceETH.yang
== Generating RELAX NG schema './sempliци-interfacceETH-data.rng'
Done.

== Generating Schematron schema './sempliци-interfacceETH-data.sch'
Done.

== Generating DSRL schema './sempliци-interfacceETH-data.dsrl'
Done.

== Validating grammar and datatypes ...
sempliceProva.xml validates

== Adding default values... done.

== Validating semantic constraints ...
No errors found.
stefano@stefano:~/EsempioTest$ yang2dsdl -v sempliceProva.xml semplici-interfacceETH.yang
== Generating RELAX NG schema './sempliци-interfacceETH-data.rng'
Done.

== Generating Schematron schema './sempliци-interfacceETH-data.sch'
Done.

== Generating DSRL schema './sempliци-interfacceETH-data.dsrl'
Done.

== Validating grammar and datatypes ...
sempliceProva.xml:7: element indirizzo-MAC: Relax-NG validity error : Error validating datatype string
sempliceProva.xml:7: element indirizzo-MAC: Relax-NG validity error : Element indirizzo-MAC failed to validate content
sempliceProva.xml:4: element interfacciaETH: Relax-NG validity error : Invalid sequence in interleave
sempliceProva.xml:4: element interfacciaETH: Relax-NG validity error : Element interfacciaETH failed to validate content
Relax-NG validity error : Extra element interfacciaETH in interleave
sempliceProva.xml:4: element interfacciaETH: Relax-NG validity error : Element interfacciaETH failed to validate content
sempliceProva.xml fails to validate
stefano@stefano:~/EsempioTest$ yang2dsdl -v sempliceProva.xml semplici-interfacceETH.yang
== Generating RELAX NG schema './sempliци-interfacceETH-data.rng'
Done.

== Generating Schematron schema './sempliци-interfacceETH-data.sch'
Done.

== Generating DSRL schema './sempliци-interfacceETH-data.dsrl'
Done.

== Validating grammar and datatypes ...
sempliceProva.xml:17: element subnet-mask: Relax-NG validity error : Error validating datatype string
sempliceProva.xml:17: element subnet-mask: Relax-NG validity error : Element subnet-mask failed to validate content
sempliceProva.xml:12: element interfacciaETH: Relax-NG validity error : Invalid sequence in interleave
sempliceProva.xml:12: element interfacciaETH: Relax-NG validity error : Element interfacciaETH failed to validate content
Relax-NG validity error : Extra element interfacciaETH in interleave
sempliceProva.xml:12: element interfacciaETH: Relax-NG validity error : Element interfacciaETH failed to validate content
sempliceProva.xml fails to validate
stefano@stefano:~/EsempioTest$ █

```

Figura 3.18: Vaidazione del modello XML sempliceProva.xml

Capitolo 4

Rete ottica

Nella presente sezione del lavoro di tesi, dopo essere stati introdotti i concetti di fibra e rete ottica, concentrando l'attenzione sulla rete ottica di accesso, è stata descritta l'architettura di una rete ottica passiva e le principali dinamiche che si verificano al suo interno. Infine, dopo aver presentato le principali differenze tra le Ethernet PON (EPON) e le Gigabit-capable PON (GPON), è stata descritta la strategia della Dynamic Bandwidth Allocation, con maggiore dettaglio nel caso specifico di una GPON.

4.1 La fibra ottica

4.1.1 Introduzione

Grazie alla fibra ottica, è possibile trasmettere grandi quantità di dati ad altissima velocità. Con il tempo le tecnologie ottiche stanno sostituendo quelle più datate basate sul rame, ed è per questo che la maggior parte delle informazioni che viaggiano su internet, sono trasportate in fibra. Oltre agli altissimi data rate che essa supporta, sono molteplici i vantaggi che essa offre. Una caratteristica fondamentale è la bassissima attenuazione che la contraddistingue. In alcune porzioni di spettro, descritte più comunemente in termini di lunghezze d'onda, e non di frequenze, si può raggiungere un valore di attenuazione prossimo a 0,2 dB/km. Questa proprietà, fa sì che le comunicazioni ottiche possano avvenire a grande distanza, senza il bisogno di dover rigenerare continuamente il segnale. Un'altra peculiarità che ci piace molto è l'immunità ad interferenze elettromagnetiche esterne e, non meno importante, il bassissimo contributo di rumore che essa presenta.

4.1.2 Tipi e composizione delle fibre ottiche

Il cavo, mostrato in figura 4.1, è costituito sostanzialmente da un nucleo di vetro, detto core, da un mantello di vetro “meno puro”¹ detto cladding, che lo riveste, e da una copertura esterna. Il segnale luminoso si propaga nel core, mentre il cladding serve da delimitatore, da guida, rappresentando quella porzione di cavo dove la luce non deve poter fluire. Il principio fisico che ne è alla base è la Legge di Snell. Esistono due tipologie di fibre, le monomodali e le multimodali (Step Index o Graded Index), che offrono caratteristiche e prestazioni diverse. Pur presentando lo stesso spessore totale, cambia il diametro del core in cui si propaga la luce. Se il diametro del cavo è in genere pari a 125 μm ,

¹Con un indice di rifrazione minore

quello del core è indicativamente pari a $50 \mu\text{m}$ per le multimodali e $10 \mu\text{m}$ per le monomodali. Per via del sempre minor spazio disponibile all'interno delle grandi tubature sottomarine che accolgono i cavi in fibra e che ricoprono lunghe distanze, hanno acquisito molta importanza le cosiddette fibre multi-core, che presentano, come il loro nome suggerisce, la presenza di più core all'interno del medesimo cavo.

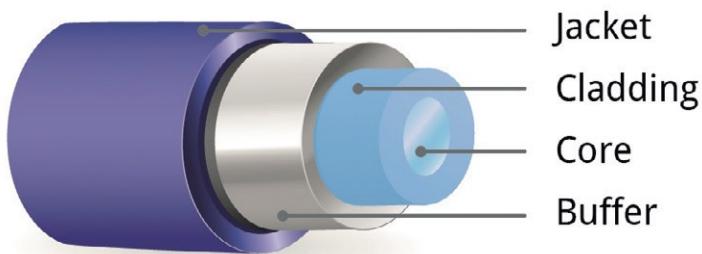


Figura 4.1: Sezione di una fibra ottica

4.2 Rete ottica

Una rete ottica non necessariamente contiene tutti dispositivi puramente ottici. Se per il trasporto delle informazioni si utilizzano le fibre ottiche, per l'operazione di switching esistono diverse soluzioni. Oltre all'impiego degli switch completamente ottici come ad esempio i MEMS (Micro Electro Mechanical System), gli OADM (Optical Add and Drop Multiplexer), e i ROADM (Reconfigurable OADM), si può optare anche per quelli elettronici, o ibridi, che necessitano di una conversione del segnale luminoso, in elettrico, e viceversa. Se nelle reti core e metro la tecnologia ottica è ormai consolidata ed imprescindibile, nelle reti di accesso spesso coesiste con la più datata basata sul rame, per fornire connettività all'utente finale. Per dispiegare la fibra “nell'ultimo miglio”, sostituendo i già presenti cavi in rame, emersero alcune criticità. Questa operazione rese e rende tuttora necessari scavi in aree urbane con alta densità di popolazione, che richiedono specifici permessi ed autorizzazioni, ed un riadattamento dei condotti già esistenti. Inoltre, per via dell'estrema delicatezza dei cavi in fibra, e della loro scarsa elasticità, è spesso difficile raggiungere le abitazioni utilizzando i condotti preesistenti all'interno degli edifici, poiché presentano angoli e spigoli non percorribili dagli stessi. Tuttavia, con il tempo, si è pensato ad alcune strategie per abbattere i prezzi e i tempi di questo processo, ottenendo una copertura capillare soprattutto nelle aree urbane. Si è pensato a diversi compromessi, mediando tra costi dell'operazione e performance aggiuntive che la tecnologia ottica avrebbe offerto. Le diverse tecniche di coesistenza tra ottico ed elettrico si distinguono in base alla distanza dell'EOI² (Electro-Optical-Interface) dall'utente finale. In ordine decrescente in termini di tale distanza, come visibile nella figura 4.2, si classificano in: Fiber to the Exchange (FTTE), Fiber to the Cab (FTTCab), Fiber to the Curb (FTTC), Fiber to the Building (FTTB), Fiber to the Home (FTTH).

²L'interfaccia che trasforma il segnale luminoso in elettrico

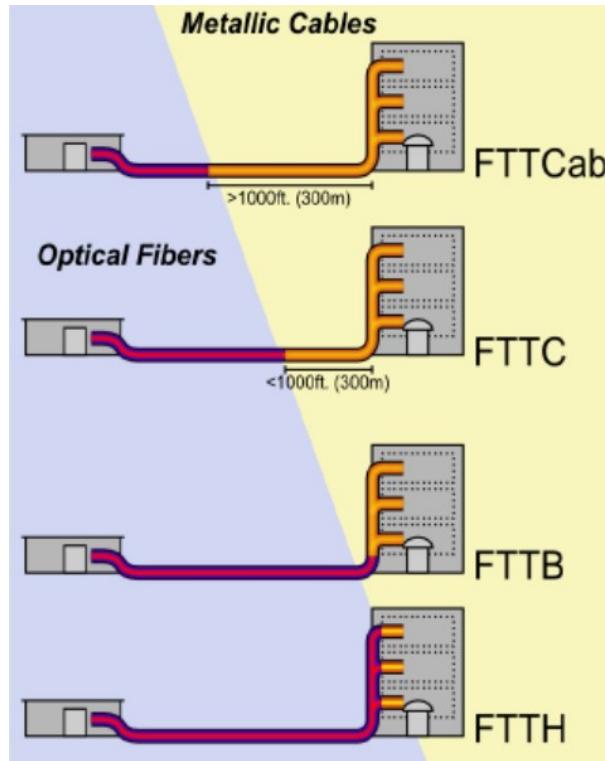


Figura 4.2: Esempio delle diverse strategie FTTx

4.2.1 Rete Ottica di Accesso: AON vs PON

Nel realizzare una rete ottica di accesso, si può optare tra due soluzioni principali: l'AON (Active Optical Network) o la PON (Passive Optical Network). La sostanziale differenza tra le due alternative consiste nella scelta dei dispositivi da utilizzare per fornire la connettività, che nel primo caso sono detti attivi, come amplificatori e dispositivi elettronici, mentre nel secondo sono detti passivi, come splitter ed accoppiatori [7]. Si intuisce da subito come questa decisione possa influire su alcuni parametri come lunghezza massima dei collegamenti e costi di installazione e manutenzione. Nella soluzione AON, la scelta dell'utilizzo di dispositivi attivi, comporta sia un costante consumo energetico, ma anche una maggiore spesa nel loro acquisto e nella loro manutenzione, poiché tipicamente più complessi e di conseguenza più costosi. Tuttavia, questa scelta ci consente di avere collegamenti più lunghi, grazie alla possibilità di amplificare il segnale che, nel propagarsi, subisce un'attenuazione proporzionale alla distanza che percorre. Tra le funzionalità più importanti che essi offrono c'è sicuramente quella di processare i dati. Al contrario, nell'alternativa PON, le funzionalità di cui i dispositivi passivi dispongono, sono molto limitate, come lo split o l'accoppiamento. Ma se da un lato le loro funzionalità risultano estremamente circoscritte, dall'altro si ha un notevole risparmio nel loro acquisto, nella loro manutenzione, e anche dal punto di vista energetico, in quanto non necessitano di alimentazione elettrica. Essendo inoltre dispositivi molto semplici, presentano guasti molto sporadici, offrendo grande affidabilità. Dopo queste considerazioni introduttive, la trattazione seguente verterà esclusivamente sulle PON, scenario dell'esempio conclusivo.

4.2.2 Rete Ottica Passiva (PON)

Struttura

Gli elementi costitutivi di una PON sono:

- Optical Line Termination (OLT): situato nel Central Office, è il dispositivo endpoint della PON
- Optical Distribution Network (ODN): costituita da cavi in fibra, può essere suddivisa in ODN primaria e secondaria. La prima rappresenta il tratto che collega l'OLT con lo splitter, in cui il mezzo è condiviso, mentre la seconda, indica il segmento finale che collega lo splitter alle varie ONT o ONU, in cui il mezzo è dedicato
- Optical Network Unit (ONU): rappresenta il punto di giunzione tra la porzione di rete in fibra e quella in rame
- Optical Network Termination (ONT): è il modo di chiamare una NIU³, quando essa è direttamente raggiunta in fibra
- Network Termination (NT): è il modo di chiamare una NIU, quando essa è raggiunta in rame
- Splitter e accoppiatori: giocano un ruolo chiave, nella PON. Mentre il primo replica il segnale presente sulla fibra in ingresso, su n fibre ad esso connesse in uscita, il secondo esegue l'operazione duale⁴. Esistono anche splitter bidirezionali, in grado di eseguire entrambi i compiti. Non essendo lo splitter un dispositivo alimentato, ogni segnale in uscita è una copia del segnale in ingresso, ma con un contributo energetico pari ad 1/n rispetto a quello del segnale in input. A tale proposito si definisce il cosiddetto “Splitting Ratio”, che per i comuni splitter a due porte è pari a 50:50, che indica una suddivisione equa del contributo energetico del segnale di ingresso, per entrambe le porte di uscita. Per questo motivo, è solitamente posto in prossimità delle ONU e delle ONT

La topologia classica, mostrata nella figura 4.3, è quella ad albero, in cui l'OLT, direttamente collegato allo splitter, è il nodo padre, mentre le ONU e le ONT sono i nodi figli. In downstream può essere vista come una rete punto-multipunto, viceversa in upstream come una multipunto-punto. È chiaro che devono esistere oltre a protocolli di condivisione del mezzo, anche politiche di sicurezza, come la crittografia basata su AES⁵. Esse risultano necessarie in quanto la trasmissione in downstream avviene solitamente in broadcast, e fanno sì che le ONU e le ONT non destinate di un traffico in ingresso, non riescano ad interpretarlo. La tecnologia PON è disponibile ed implementabile per qualsiasi scelta FTTX.

Strategie di trasmissione

Per gestire il traffico bidirezionale si può scegliere tra diverse soluzioni. La più semplice che si può immaginare è la Space Division Duplex (SDD), in cui si utilizzano due fibre distinte, una dedicata al traffico in upstream e una a quello in downstream. Nella trasmissione si utilizza la stessa lunghezza

³Network Interface Unit: indica il terminale dell'utente

⁴è evidente che è richiesto un protocollo che regoli la condivisione del mezzo per evitare collisioni

⁵Advanced Encryption Standard: algoritmo di cifratura a blocchi a chiave simmetrica

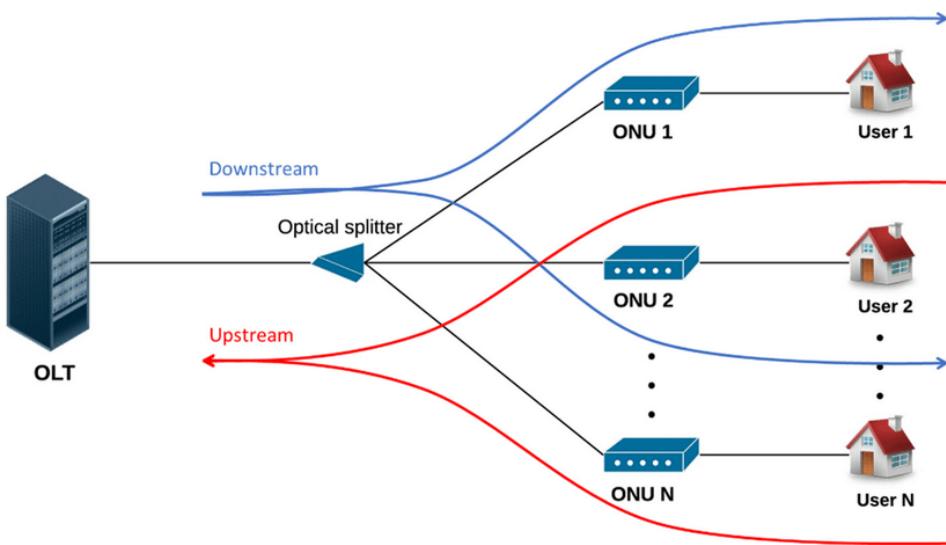


Figura 4.3: Struttura generale di una PON

d'onda su entrambe le fibre, pari a $1.3 \mu\text{m}$, sfruttando laser economici capaci di lavorare proprio a questa lunghezza d'onda. Questa soluzione prevede però l'utilizzo di un numero doppio di fibre, e splitter, con un conseguente aumento del cosiddetto CAPEX. Si è pensato allora alla strategia One Fiber Single Wavelength Full Duplex, in cui la stessa fibra trasporta il traffico in entrambe le direzioni. Per implementarla si ha bisogno di un accoppiatore direzionale all'interno dell'OLT e in ogni ONU/ONT, che però dà vita a fenomeni di cross talk locale, tra trasmettitore e ricevitore, molto vicini all'interno di uno stesso dispositivo. Un'altra possibile alternativa è la Time Division Duplex, che prevede la possibilità di trasmettere o ricevere il traffico su un'unica fibra, rispettando gli slot temporali decisi dall'OLT. Se da un lato essa elimina il fenomeno del cross-talk locale sopra descritto, dall'altro riduce il throughput complessivo di circa il 50%. Una soluzione concettualmente diversa è invece la Wavelength Division Duplex, in cui si scelgono due diverse lunghezze d'onda per il traffico in upstream e downstream, viaggianti sulla stessa fibra. Due segnali (o più) che viaggiano sulla stessa fibra, caratterizzati da lunghezze d'onda diverse, non si "disturbano". Grazie a questa proprietà, sullo stesso mezzo si possono realizzare più canali paralleli, tra loro indipendenti.

Tecniche di Multiplazione

Per consentire alle diverse ONU/ONT di condividere il mezzo trasmissivo, è necessaria una tecnica di multiplazione. Si può pensare di optare tra la Time Division Multiplexing (TDM), o la Wavelength Division Multiplexing (WDM), mostrate nelle figure 4.4 e 4.5. Nel primo caso, tutte le ONU/ONT utilizzano la stessa lunghezza d'onda per trasmettere, e possono farlo utilizzando un trasmettitore identico. Dovendo ricevere segnali, seppur da mittenti diversi, caratterizzati dalla stessa lunghezza d'onda, l'OLT ha bisogno di un unico ricevitore. È esattamente l'opposto nel caso WDM, in cui ogni ONU/ONT sceglie una precisa lunghezza d'onda per trasmettere e ricevere, e per questo motivo l'OLT necessita di un trasmettitore e di un ricevitore più complessi, o di stessa complessità ma in maggior numero. Per le operazioni di multiplazione e demultiplazione nel dominio ottico si ricorre ai cosiddetti Arrayed Waveguide Grating (AWG) multiplexer e demultiplexer. Si intuisce come questa soluzione, pur offrendo grandi potenzialità, risulta spesso molto dispendiosa dal punto di vista economico per essere implementata nella semplice rete di accesso, ed è per questo che la strategia

più comune risulta la TDM, con distribuzione Broadcast in downstream e tecniche Time Division Multiple Access (TDMA) in upstream. In alcuni standard più recenti ed avanzati, le due tecniche coesistono, con un conseguente aumento della complessità nei dispositivi, principalmente nell'OLT, ma con un notevole incremento delle capacità offerte. Questa strategia in cui si implementano entrambe le tecniche, prende il nome di Time Wavelength Division Multiplexing (TWDM). C'è da considerare anche che le diverse ONU/ONT non sono alla stessa distanza dallo splitter, fattore che rende ancora più delicata la sincronizzazione nella trasmissione. Infatti, nel momento della loro attivazione, ma con la possibilità di farlo anche quando già in servizio, l'OLT esegue la cosiddetta operazione di Ranging, per calcolare tale distanza. Per lo stesso motivo, ma percorrendo la fibra nel verso opposto, all'OLT arriveranno segnali con un contributo in termini di potenza, tanto maggiore quanto più la ONU/ONT è prossima allo splitter. Per lavorare in modalità full-duplex si scelgono solitamente due lunghezze d'onda distinte per i traffici in salita e in discesa, e se ne riservano altre per servizi specifici, come servizi video in downstream, o per impieghi futuri.

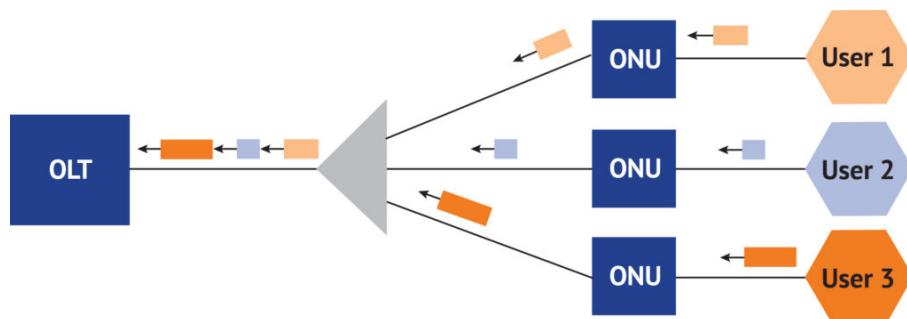


Figura 4.4: Esempio di Time Division Multiplexing

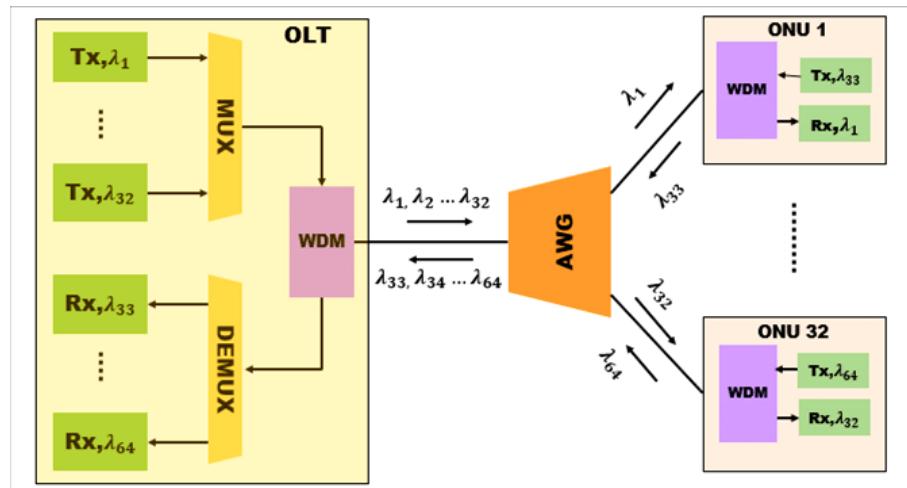


Figura 4.5: Esempio di Wavelength Division Multiplexing

GPON ed EPON

Le due versioni più popolari di PON, sono le cosiddette Ethernet PON (EPON) e le Gigabit-capable PON (GPON), e la principale differenza tra le due è il protocollo di livello 2 con cui si accede al mezzo. Vista la sua grande adozione nelle reti LAN e METRO, per le EPON si è pensato all'utilizzo del protocollo Ethernet. Il traffico all'interno di questa tipologia di PON è direttamente encapsulato

in trame Ethernet, e per questo motivo non sono richiesti ulteriori incapsulamenti e conversioni di protocollo. Analizzando il funzionamento dell'accoppiatore, che fa confluire il traffico proveniente dalle diverse ONU/ONT unicamente verso l'OLT, si capisce che la strategia CSMA-CD⁶ non può essere implementata, in quanto esse non sono in grado di fare Collision Detection. Sarà compito dell'OLT dirigere il traffico in upstream. Con la tecnologia GPON, il formato della trama non è limitato a quello di un particolare protocollo. I servizi ATM⁷, vengono distribuiti attraverso trame ATM, caratteristiche delle ATM PON (APON), ma allo stesso tempo, per altri servizi, si usano trame generiche. Grazie ad un livello detto GPON Transmission Convergence layer, le GPON Transmission Convergence frame possono accogliere al loro interno, sia trame ATM, che trame generiche. Oltre a questo importante compito, esso effettua la Dynamic Bandwidth Allocation, in modo da implementare QoS, introduce sicurezza, effettua correzione di errori, ed esegue Power Saving. Così come nelle GPON, anche nelle EPON è possibile fare Dynamic Bandwidth Allocation (DBA), soprattutto per il traffico in upstream. Dai loro nomi, si potrebbe erroneamente pensare che le EPON, a differenza delle GPON, non siano in grado di fornire data rate dell'ordine dei Gbps. Non sarebbe un'affermazione corretta, in quanto le bit rate offerte dalle prime, sono confrontabili con quelle delle seconde. Dunque, con la scelta del nome EPON, si vuole sottolineare l'impiego del protocollo Ethernet, peculiarità di questo standard, e non le capacità nella trasmissione dei dati.

Dynamic Bandwidth Allocation

Come alternativa alla Fixed Bandwidth Allocation, tecnica molto efficiente per traffico caratterizzato da bit rate costanti prodotto dalle singole ONU/ONT, è stata ideata la Dynamic Bandwidth Allocation (DBA). Nella prima strategia l'OLT assegna ad ognuna di esse slot temporali in cui poter trasmettere in upstream in maniera periodica, in base ai loro service level agreement (SLA). Dopo lo schedule iniziale di questi time slot, la gestione risulta essere molto semplice. Anche se l'OLT invia continuamente messaggi di controllo alle ONU/ONT, in linea di principio potrebbe anche farne a meno, in quanto ad esse è già stata fornita la regola da rispettare in fase di trasmissione. Per questo motivo i messaggi di controllo scambiati tra OLT e ONU/ONT risultano essere molto "leggeri", con un consumo di banda minimo, e con conseguente goodput⁸ massimo. Quanto più il traffico è intermittente, tanto minore sarà l'efficienza di questa tecnica. Una ONU/ONT che nel suo time slot ha la possibilità di trasmettere, ma che in quel momento non ha dati da inviare, non permette alle altre, che potrebbero presentare anche altissime quantità di dati da consegnare, di poter utilizzare il mezzo. Con la DBA, attraverso un continuo dialogo tra le ONU/ONT e l'OLT, si cerca di minimizzare il tempo in cui il mezzo è inutilizzato, attraverso l'assegnazione dinamica di transmission opportunity. Le ONU/ONT, che hanno bisogno di accedere al mezzo per trasmettere, lo comunicano all'OLT, che darà loro la possibilità di farlo, definendo le modalità con cui farlo. La necessità di questo continuo scambio di messaggi di controllo, riduce inevitabilmente il goodput, che è ulteriormente intaccato a causa delle collisioni che possono verificarsi nella fase di richiesta di trasmissione da parte delle ONU/ONT all'OLT (due o più di esse, potrebbero voler trasmettere, e quindi comunicarlo all'OLT, nello stesso momento). Si capisce come questa strategia non si sposi

⁶Carrier Sense Multiple Access with Collision Detection: è un protocollo di accesso ad un mezzo condiviso, che consente a chi sta trasmettendo, di rilevare eventuali collisioni

⁷Asynchronous Transfer Mode o ATM è un protocollo di rete di livello datalink che prevede l'incapsulamento dei dati in unità di lunghezza fissa, dette celle

⁸goodput: capacità di trasferimento di dati utili, percepita a livello applicativo dall'utente

con sorgenti caratterizzate da bit rate pressoché costanti. È importante osservare che l’assegnazione dinamica e quindi più efficiente della banda, fa sì che un maggior numero di ONU/ONT possano essere collegate allo stesso OLT, con buone prestazioni. Poiché la parte sperimentale del lavoro di tesi è stata effettuata su una GPON, ci concentreremo sulla descrizione della Time Division DBA, nel caso di traffico upstream, per reti GPON. Si precisa inoltre che il termine “bandwidth” è usato come termine equivalente di data rate, per indicare una porzione di capacità in uplink nelle GPON, che può essere assegnata in maniera specifica.

4.2.3 Dynamic Bandwidth Allocation nelle GPON

La Dynamic Bandwidth Allocacation (DBA) nelle reti GPON è il processo attraverso cui l’OLT alloca per il traffico in upstream le transmission opportunity alle entità che supportano il traffico, presenti nelle ONU [1]. Questa allocazione avviene in base all’indicazione dinamica del loro stato di attività e delle loro politiche di traffico. A queste entità, rappresentate dai Transmission Container (T-CONT), per essere identificate, l’OLT assegna un numero identificativo, detto allocation ID (Alloc-ID). Il principio di funzionamento è descritto nella figura 4.6. Poiché l’unico elemento in qualsiasi tipo di PON, che ha visione globale della rete è l’OLT, è a lui che spetterà il compito di organizzare il traffico in upstream (oltre a quello in downstream). Dal punto di vista delle funzionalità del Trasmission Convergence Layer, ONU ed ONT sono praticamente identiche, ed è per questo che nella descrizione sono state indicate con il solo termine ONU. Per spiegare nel dettaglio il principio di funzionamento della DBA nelle GPON, è necessario presentare formalmente i principali elementi:

- ONU Identifier (ONU-ID): è l’identificativo di 8 bit che l’OLT assegna ad ogni ONU presente nella PON, nel momento della sua attivazione. Esso è unico all’interno della PON e resterà tale fino a quando tale ONU non verrà spenta o disattivata dall’OLT. Nell’assegnazione si utilizzano messaggi di tipo PLOAM⁹
- Allocation Identifier (Alloc-ID): è un numero da 12 bit che l’OLT assegna ad ogni T-CONT. Ad ogni ONU ne è assegnato uno di default che è numericamente uguale al proprio ONU-ID, ma possono esserne associati altri
- Transmission Container (T-CONT): è un oggetto presente all’interno delle ONU, che rappresenta un gruppo di connessioni logiche, mascherate sotto un’unica entità, necessaria ai fini dell’operazione di bandwidth assignment del traffico in upstream all’interno della PON. Ogni ONU supporta un numero fisso di T-CONT, che istanzia autonomamente durante il processo di attivazione. Dopo una prima fase di scoperta da parte dell’OLT, esso li attiva, rendendoli pronti a supportare del traffico, attraverso un’operazione di mapping tra i T-CONT creati e gli Alloc-ID precedentemente assegnati. Ogni ONU deve averne almeno uno, ma la maggior parte ne ha di più, eterogenei tra loro per i diversi tipi di priorità e di traffico. Ad ognuno di essi saranno assegnati diversi Time Slot, in modo dinamico ed efficiente. Esistono cinque tipi di T-CONT:

⁹Physical Layer OAM Operations, Administrations and Maintenance: canale operativo e di gestione basato su messaggi tra OLT e ONU

1. Tipo 1: è caratterizzato solo dalla componente di Bandwidth fissa ¹⁰, e non è idoneo per la condivisione di quella in eccesso. Per questo motivo, se in un momento non produce traffico, quella porzione di Bandwidth sarà sprecata. È usato principalmente per servizi sensibili al ritardo e con alta priorità, come VoIP
2. Tipi 2 e 3: sono entrambi caratterizzati dalla componente di Bandwidth assicurata ¹¹, e solitamente usati per servizi video e dati con priorità maggiori. Nel tipo 3, oltre a quella assicurata, c'è anche quella massima ¹², e per questo è detta Bandwidth mista
3. Tipo 4: è caratterizzato dall'idoneità nella condivisione della Bandwidth cosiddetta Best-Effort ¹³, senza l'apporto né di quella fissa né di quella assicurata. È adatto per trasportare il traffico di servizi caratterizzati da data rate variabili e non sensibili al ritardo, come servizi dati a bassa priorità
4. Tipo 5: è un rafforzamento dei quattro tipi precedenti, comprendente tutti i tipi di Bandwidth e applicabile alla maggior parte dei diversi traffici dei servizi

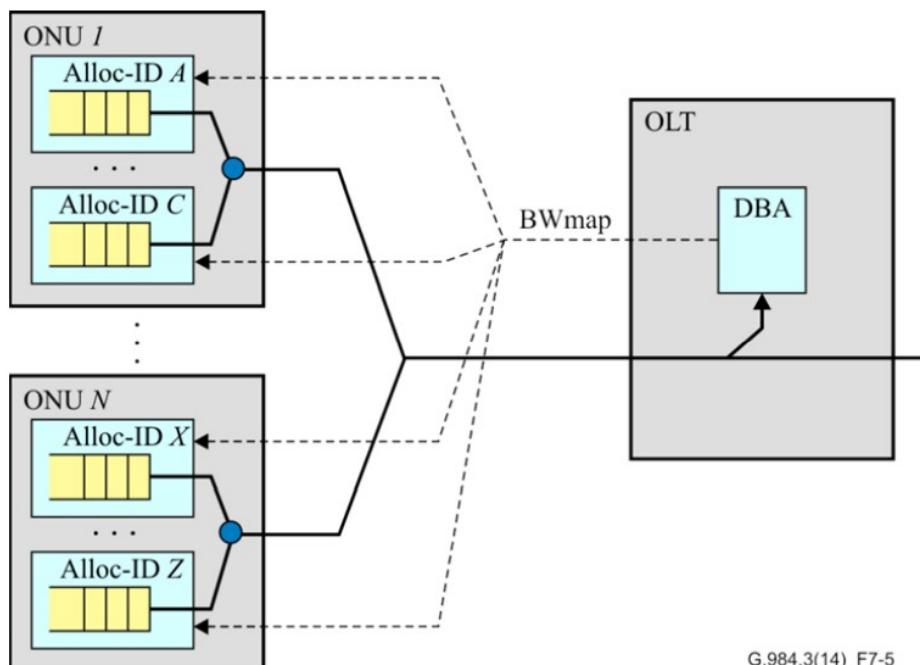


Figura 4.6: Scenario della DBA in una GPON

Dopo la prima fase di Ranging e la successiva di sincronizzazione in base alle informazioni raccolte in termini di distanza tra l'OLT e le diverse ONU, l'OLT concede ai diversi T-CONT

¹⁰Bandwidth fissa: porzione riservata della capacità in uplink che l'OLT alloca staticamente ad un preciso Alloc-ID, indipendentemente dalla sua domanda di traffico e dal carico complessivo del traffico sulla PON

¹¹Bandwidth assicurata: porzione della capacità in uplink che l'OLT dovrebbe assegnare agli Alloc-ID finché essi hanno una domanda di traffico insoddisfatta, indipendentemente dal carico complessivo del traffico sulla PON. Se la domanda di traffico viene soddisfatta, l'OLT può riassegnare completamente o parzialmente questa porzione di Bandwidth ad altri Alloc-ID

¹²Bandwidth massima: limite superiore della bandwidth totale che può essere assegnata al dato Alloc-ID in qualsiasi condizione di traffico

¹³Best-effort bandwidth: forma di bandwidth addizionale che l'OLT può dinamicamente assegnare ad un Alloc-ID, in proporzione alla porzione non garantita di massima bandwidth fornita a quell'Alloc-ID

delle Transmission Opportunity. Esse sono specificate nelle Bandwidth Map, generate e ricalcolate dinamicamente e continuamente dall'OLT, attraverso i cosiddetti algoritmi DBA.

Bandwidth Map

La Bandwidth Map, descritta in figura 4.7, è un array scalare composto da strutture di allocazione da 8 byte, ognuna delle quali rappresenta una singola allocazione di Bandwidth per uno specifico T-CONT. Il numero di queste strutture, e quindi la lunghezza complessiva, è specificato in un apposito campo detto PLend. Ogni elemento dell'array, è composto da 8 byte, ed è formato dai seguenti campi:

- Campo Allocation ID: composto da 12 bit, indica il destinatario della specifica Bandwidth Allocation, come un preciso T-CONT
- Campo Flag: composto da 12 bit, contiene indicazioni che controllano alcune funzioni della trasmissione in upstream associata
- Campo StartTime: composto da 16 bit, indica il tempo di inizio dell'allocation misurato in byte
- Campo StopTime: composto da 16 bit, indica il tempo di stop dell'allocation. Anch'esso è misurato in byte
- Campo CRC: composto da 8 bit, è usato per proteggere la struttura di allocazione da eventuali errori, utilizzando un CRC-8¹⁴, che il destinatario utilizzerà per implementare le funzioni di rilevamento, e quando possibile anche di correzione di eventuali errori. Nel caso di errore incorreggibile, quella struttura di allocazione sarà scartata

Rientrando nella categoria del traffico in downstream, esse sono distribuite in broadcast a tutte le ONU, che avranno le direttive in merito al tipo di traffico da trasmettere (in base al T-CONT specificato dall'Alloc-ID), e per quali intervalli di tempo.

Metodi di Dynamic Bandwidth Allocation

Capito come l'OLT calcola ed assegna le Transmission Opportunity, non resta che descrivere le modalità con cui le diverse ONU, o ancora più precisamente i diversi T-CONT, comunicano ad esso le proprie richieste. In base a come l'OLT recupera queste informazioni necessarie per organizzare il traffico in upstream, si possono descrivere due strategie di DBA, con rispettivi pro e contro. Esse sono:

- Status Reporting (SR) DBA: le ONU forniscono direttamente lo stato di occupazione dei buffer T-CONT, quando sollecitate dall'OLT. In questo report di risposta è indicata la quantità di dati in attesa di essere inviati per ogni buffer T-CONT. Una volta che l'OLT li ha ricevuti, ricalcola la BW Map e la inoltra alle ONU, che trasmetteranno nei time slot assegnati. Nel momento in cui una ONU non ha più dati da trasmettere, lo comunica all'OLT attraverso un messaggio vuoto, che indica che il buffer è stato liberato. In questo modo l'OLT può procedere con la riassegnazione del permesso di trasmissione ad altre ONU.

¹⁴CRC-8: Cyclic Redundancy Check 8, è un codice di rilevamento degli errori che produce una checksum di 8 bit

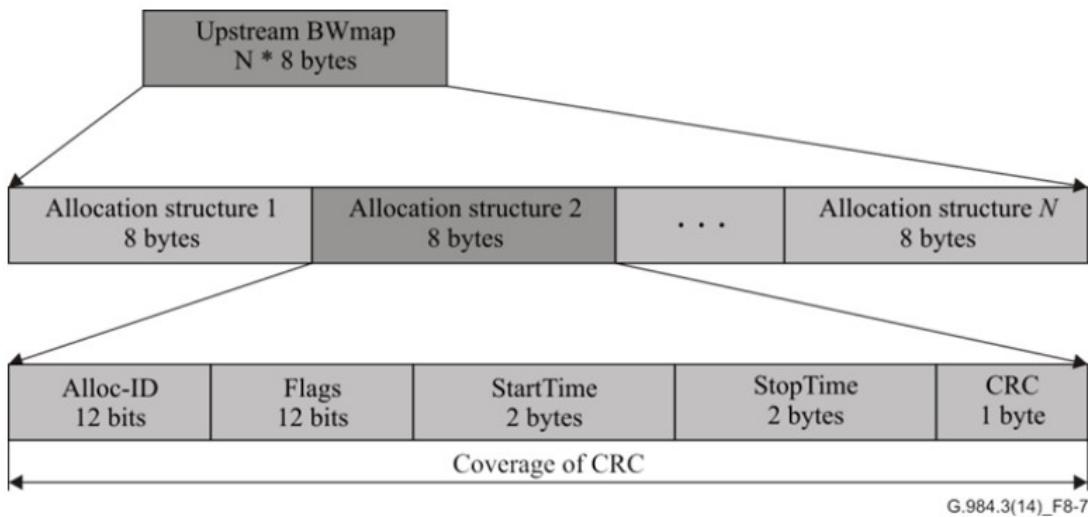


Figura 4.7: Struttura di una Bandwidth Map

- Non Status Reporting (NSR) DBA: è detta anche Traffic Monitoring, e come questo sinonimo suggerisce, è basata sulla capacità dell'OLT di osservare il traffico in upstream. In questa tecnica il report sullo stato dei buffer, non è fornito esplicitamente dalle ONU, ma è ricavato in modo implicito dall'OLT. Esso alloca continuamente una piccola porzione aggiuntiva di bandwidth per ogni ONU, le quali, se non hanno traffico in coda nei buffer, invieranno dei messaggi vuoti. In questo modo l'OLT capisce di poter ridurre o rimuovere la bandwidth a quelle ONU che non hanno dati da trasmettere, e al contrario, di poter aumentarla a quelle che non inviano messaggi vuoti, e che quindi ne avranno sicuramente.

In un primo confronto tra le due tecniche si evince che con la seconda, le ONU sono praticamente esentate dall'implementazione della DBA, in quanto non è richiesta una continua comunicazione dello stato dei loro buffer. Tuttavia, non si riesce ad ottenere l'allocazione migliore possibile, come con la prima.

Capitolo 5

Sezione Sperimentale

5.1 Introduzione

Nella prima parte di questa sezione conclusiva del lavoro di tesi, sono illustrate operazioni di monitoraggio del traffico in upstream all'interno di una PON, utilizzando il protocollo NETCONF, attraverso la libreria ncclient di Python. In seguito sono riportate operazioni di controllo e riconfigurazione dell'OLT, realizzate con gli stessi strumenti, che permettono di modificare il suo stato, al fine di ottenere il desiderato andamento del traffico all'interno della PON, al verificarsi di particolari eventi (come picchi di traffico). Queste operazioni avvengono in modo del tutto automatico, attraverso l'utilizzo di un controllore, implementato utilizzando il linguaggio Python, caratterizzato da tempi richiesti per riconfigurare dinamicamente il dispositivo, prossimi ai 0.45s. Per generare il traffico è stato utilizzato il tool da linea di comando iperf3 [14]. Grazie alle diverse funzionalità che esso offre, è stato possibile generare più di un flusso di traffico contemporaneamente, specificando alcuni parametri per modellarli a piacere. In appendice è possibile consultare tutti i codici Python implementati per il lavoro di tesi, i cui risultati sono stati descritti nelle apposite sezioni. Inoltre, sempre in appendice, è riportata un'ulteriore strategia utilizzata per controllare la validità dei dati acquisiti attraverso il codice Python di monitoraggio dei throughput, oltre al semplice confronto con quelli mostrati dai report di iperf3.

5.2 Architettura NG-PON2

Come già anticipato, nella trattazione teorica degli argomenti, dopo un'introduzione descrittiva e più generale, è stata approfondita con maggiore dettaglio l'architettura di una PON, con particolare riguardo per la Dynamic Bandwidth Allocation nelle GPON. Questa decisione è stata subordinata alla rete presente in laboratorio, su cui si è svolto il lavoro sperimentale oggetto della tesi. La GPON in questione, schematizzata nella figura 5.1, è precisamente una NG-PON2¹, standard successivo a G-PON e XG-PON1. Questa architettura, grazie alla tecnica del Time and Wavelength Division Multiplexing (TWDM), è capace di offrire un throughput di rete totale, pari a 40 Gbps, corrispondenti ad un limite massimo di 10 Gbps simmetrici in upstream e downstream, per ognuna

¹Next-Generation Passive Optical Network 2: architettura di una PON, standardizzata dall'ITU-T nel documento G.989.2

delle quattro wavelength [2]. In realtà, secondo lo standard, esistono tre diverse opzioni, in base ai data rate nominali di linea in downstream (DS) ed in upstream (US) che esse offrono:

- Basic rate: 9,95328 Gbps DS, 2,48832 Gbps US
- Rate option 1: 9,95328 Gbps DS, 9,95328 Gbps US
- Rate option 2: 2,48832 Gbps DS, 2,48832 Gbps US

Lo standard specifica le otto lunghezze d'onda, di cui quattro usate per il DS e altre quattro per l'US (se non riservate tutte all'US, usate per altri scopi), in termini di frequenze. Quelle dedicate al downstream sono: 187,8 THz, 187,7 THz, 187,6 THz e 187,5 THz; quelle per l'upstream sono: 187,4 THz, 187,3 THz, 187,2 THz e 187,1 THz. In termini di lunghezza d'onda ci troviamo nell'intervallo che va dai 1596,34 nm ai 1602,31 nm. Dovendo implementare la TWDM, è necessario che i dispositivi presenti nella rete ottica passiva, siano dotati di ricevitori e trasmettitori, cosiddetti "tunable", cioè regolabili, capaci di lavorare alle diverse wavelength. In downstream le quattro diverse wavelength sono combinate utilizzando uno Wavelength Division Multiplexer, e il segnale verrà poi filtrato da ogni ONU grazie a filtri regolabili, che recuperano il segnale di cui sono destinatarie, caratterizzato da una delle quattro lunghezze d'onda. In upstream, i laser presenti nelle ONU, anch'essi regolabili, vengono settati dinamicamente per trasmettere ad una precisa wavelength. Le fibre provenienti dalle diverse ONU vengono poi combinate con un Multiplexer/Accoppiatore passivo in un'unica fibra. Questo standard è stato inoltre progettato per garantire retrocompatibilità e coesistenza con quelli precedenti, facilitando l'operazione di deployment nelle reti ottiche di distribuzione già esistenti.

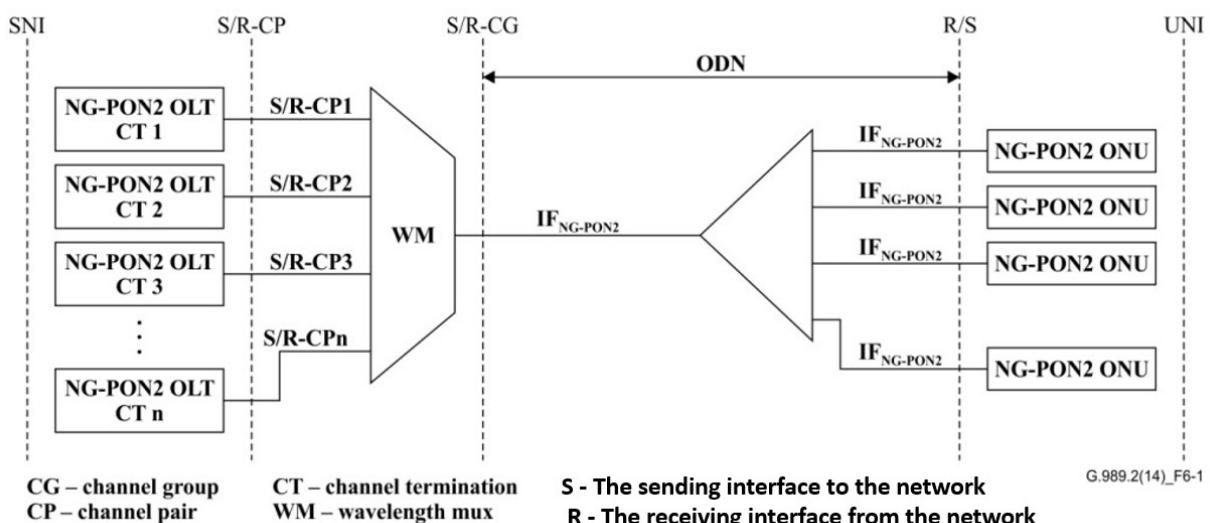


Figura 5.1: Architettura di una NG-PON2

5.2.1 Setup Laboratorio

La rete presente in laboratorio, schematizzata in figura 5.2, contenendo tutti gli elementi precedentemente descritti di una GPON, permette di simulare in maniera realistica gran parte delle dinamiche che si verificano realmente in una PON. Si possono osservare infatti tutti gli ingredienti

che compongono una rete ottica passiva, quali l'OLT, le due ONT, chiamate ONT1 e ONT2, lo splitter/accoppiatore, la ODN sia primaria che secondaria e le due NT, aventi come hostname "G1" e "precision". Dalla terminologia utilizzata dovrebbe essere chiaro che ONT ed NT sono collegate tra loro in rame, mentre le ONT all'OLT attraverso cavi in fibra ottica, dedicati fino allo splitter/accoppiatore, condiviso nel tratto di ODN primaria. Inoltre, collegato all'OLT c'è un server, indicato con "CORE1". Come detto la NG-PON2 consente di lavorare su quattro distinte lunghezze d'onda in downstream (DS) e quattro in upstream (US). Nella configurazione in laboratorio, si lavora su una lunghezza d'onda per l'US ed una per il DS, ottenendo a tutti gli effetti una XGS-PON. Essa fornisce una capacità di 10 Gbps sia in download che in upload, caratteristica rivelata dallo stesso nome attraverso la lettera "S", che sta per "Symmetric". Per la gestione remota dei dispositivi attraverso NETCONF, le NT, l'OLT ed il server sono all'interno della LAN con indirizzo privato 10.10.10.0/24, insieme al controllore di rete utilizzato per effettuare le diverse operazioni di configurazione e monitoraggio. Il gateway ha indirizzo 10.10.10.1/24 per poter andare fuori LAN, come di norma si usa. Inoltre, sono istanziate due VLAN, con identificativi 111 e 222, al cui interno sono presenti precision e CORE1 , e G1 e ancora CORE1, rispettivamente. La tecnologia della Virtual LAN (VLAN) permette di segmentare il dominio di broadcast di una LAN, suddividendola in più reti LAN Virtuali, isolate tra loro, ma che condividono la stessa infrastruttura fisica. Attraverso questa scelta si isola il traffico destinato ai due utenti precision e G1, collegati alle due diverse ONT.

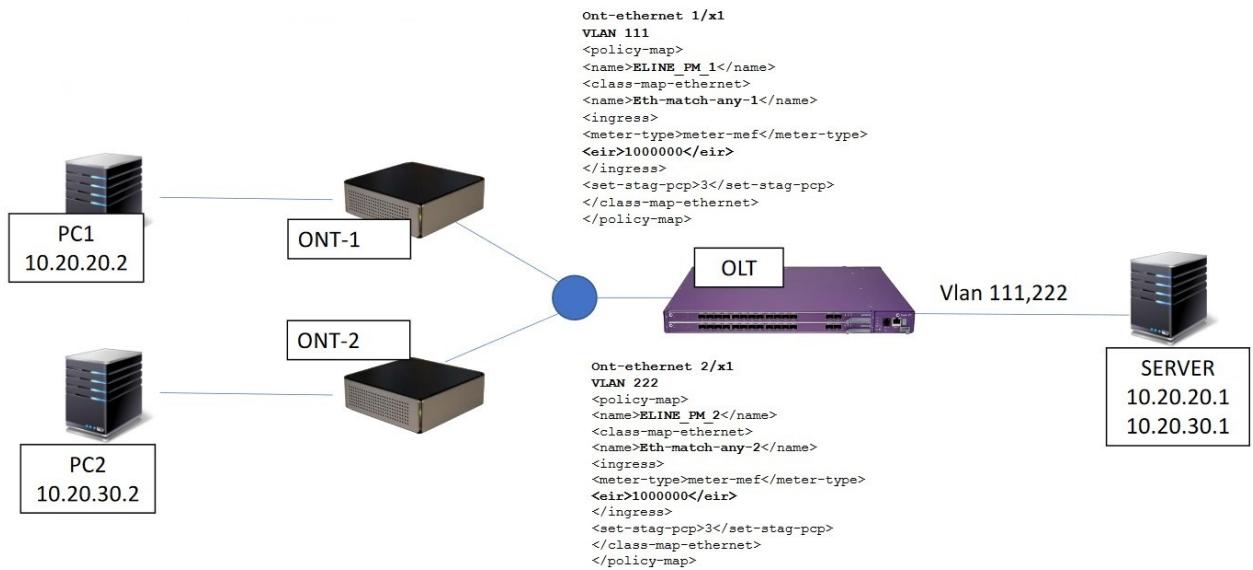


Figura 5.2: Setup presente in Laboratorio

5.3 Monitoraggio del traffico della PON senza politiche di controllo

In questa prima sezione, è mostrato l'andamento del traffico upstream all'interno della PON, attraverso uno script di monitoraggio, senza la riconfigurazione dinamica delle politiche di assegnazione di bandwidth. Come descritto, all'interno della PON sono presenti due ONT, ONT1 e ONT2, collegate rispettivamente a precision, e G1. precision e G1 generano traffico in upstream sulla PON, trasmettendo grandi quantità di dati verso CORE1. Si ipotizzi che precision stia utilizzando un servizio che richieda bandwidth minima pari a 5Gbps, mentre G1 genera traffico intermittente. Non

attuando misure di controllo si vede come non è garantito il rispetto della specifica sul traffico di precision quando G1 presenta picchi di traffico, inficiando sulla QoS che precision richiede. Nell'esempio presentato, il valore dell'eir² è settato di default al massimo, 10000000 (kbps, 10 Gbps), in entrambe le policy map³ "Eth-match-any-1", verso l'ONT1, e "Eth-match-any-2", verso l'ONT2, in modo da utilizzare tutte le risorse di bandwidth disponibili sulla PON. La figura 5.3 mostra l'andamento del throughput ottenuto attraverso l'osservazione dell'andamento del traffico, generato con iperf3, per una durata di 38 secondi. Quando il traffico prodotto da G1 si mantiene limitato, la condizione sulla bit rate di precision è rispettata. Viceversa, se esso cresce significativamente provando a trasmettere ad una bit rate superiore ai 3 Gbps circa, non avendo bandwidth garantita, si osserva che la bit rate di precision scende sotto i 5 Gbps, condizione che sarà evitata implementando una logica di controllo, riportata nell'esempio successivo.

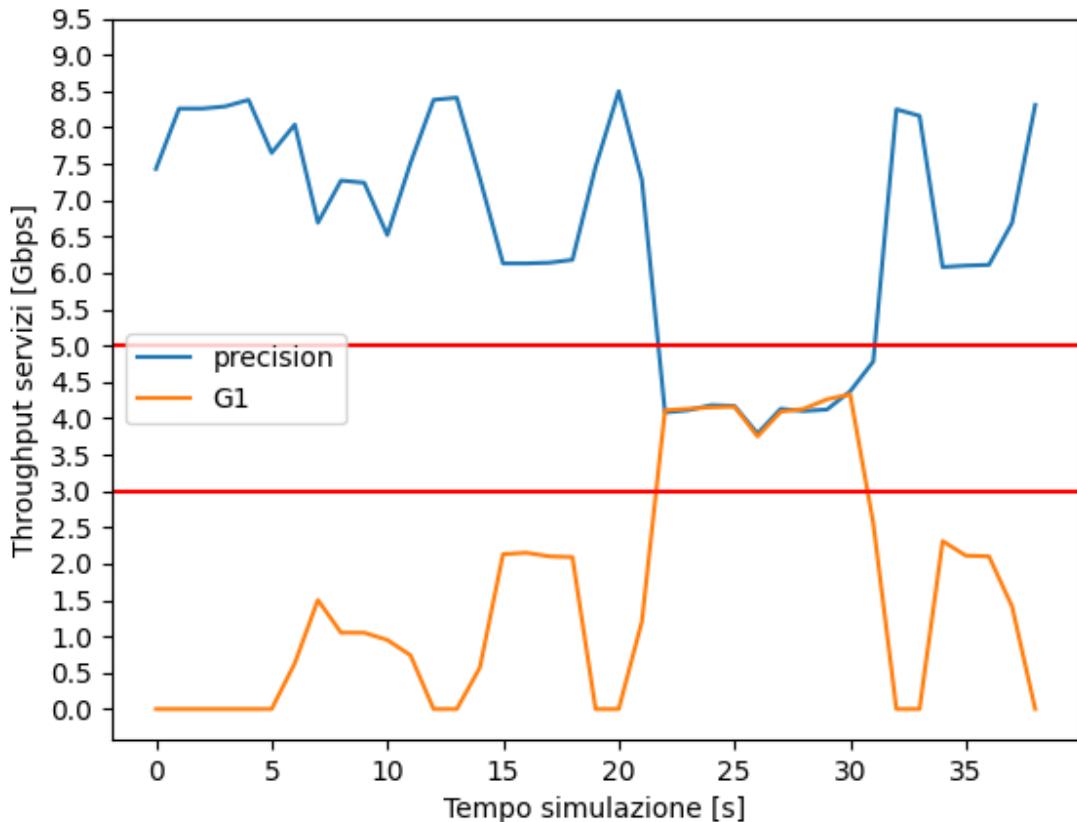


Figura 5.3: Plot dei throughput nel caso di semplice monitoraggio

In entrambi i codici, la bit rate di entrambi i flussi, riportata in Gbps, è calcolata attraverso la funzione `get_Gbps_and_Counters()`, interrogando ogni secondo l'OLT utilizzando l'operazione `<get>` di NETCONF, per conoscere il valore dei contatori dei byte in ingresso nella medesima interfaccia, ma associati alle due diverse ONT, che il dispositivo aggiorna in automatico, in tempo reale. Il numero di byte si riferisce al numero totale di byte che transitano in ingresso, inclusi quelli scartati [6]. Inoltre, per non dover instaurare per ogni `<rpc>` una nuova sessione NETCONF, per

²eir: Excess Information Rate, descritto formalmente in seguito

³Policy-Map: descritta formalmente in seguito

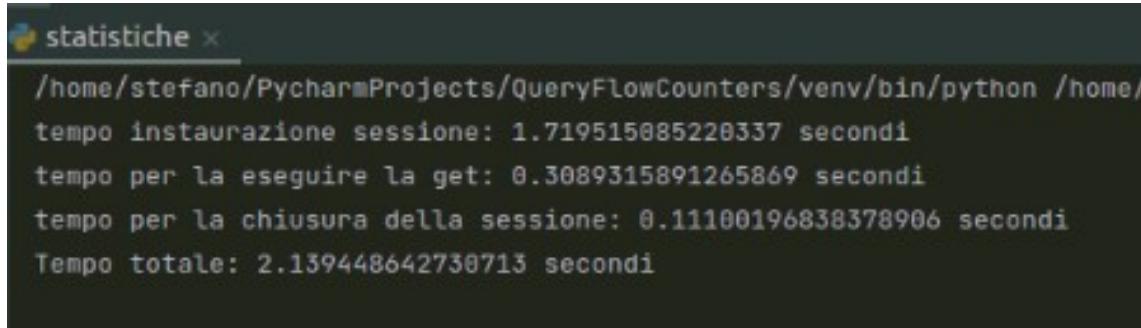
l'intera durata della simulazione si è lavorato all'interno della medesima sessione. Grazie a questa scelta si è minimizzata la latenza, e le operazioni di monitoraggio e controllo sono risultate molto più reattive. Ricordando che si sta utilizzando ssh come livello di trasporto sicuro per NETCONF, il quale si appoggia a sua volta sul protocollo di trasporto TCP, questa soluzione ci ha permesso di evitare di eseguire il Three Way Handshake per ogni <rpc>, così come il processo di autenticazione che ssh richiede. La figura 5.4 riporta la cattura eseguita attraverso Wireshark [21], di una sessione NETCONF, utilizzando la libreria ncclient di Python, in cui viene instaurata la sessione, eseguita una <get> (la stessa utilizzata anche nei codici successivi per richiedere i valori dei contatori dei byte in ingresso), e terminata la sessione. Attraverso essa, osservando l'ultimo campo sulla destra, in cui sono riportate le informazioni relative al pacchetto, possono essere visualizzati il processo del Three Way Handshake, e le operazioni che ssh richiede per instaurare una connessione sicura. Attraverso il tool, eseguendo la sottrazione tra l'ultimo ed il primo valore della colonna "time", è

No.	Time	Source	Destination	Protocol	Length	Info
145	1.749086485	10.10.10.30	10.10.10.217	TCP	76	22 → 54750 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SA..
148	1.750321835	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=1 Ack=25 Win=29056 Len=0 TSval=2025307 T..
151	1.783073380	10.10.10.30	10.10.10.217	SSHv2	89	Server: Protocol (SSH-2.0-OpenSSH_7.6)
154	1.786791607	10.10.10.30	10.10.10.217	SSHv2	572	Server: Key Exchange Init
158	1.808876615	10.10.10.30	10.10.10.217	SSHv2	380	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New..
161	1.851363689	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=838 Ack=929 Win=30592 Len=0 TSval=202540..
163	1.851716016	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=838 Ack=993 Win=30592 Len=0 TSval=202540..
164	1.851898133	10.10.10.30	10.10.10.217	SSHv2	132	Server: Encrypted packet (len=64)
166	1.893362959	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=902 Ack=1105 Win=30592 Len=0 TSval=20254..
167	1.966332584	10.10.10.30	10.10.10.217	SSHv2	116	Server: Encrypted packet (len=48)
169	1.967427246	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=950 Ack=1185 Win=30592 Len=0 TSval=20255..
170	2.000779965	10.10.10.30	10.10.10.217	SSHv2	1028	Server: Encrypted packet (len=960)
174	2.042617408	10.10.10.30	10.10.10.217	SSHv2	132	Server: Encrypted packet (len=64)
179	2.052947871	10.10.10.30	10.10.10.217	SSHv2	180	Server: Encrypted packet (len=112)
212	2.196372560	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=2088 Ack=2069 Win=33280 Len=0 TSval=2025..
261	3.457107351	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
263	3.457193279	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
265	3.457328133	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
267	3.457442330	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
269	3.457561436	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
271	3.457645915	10.10.10.30	10.10.10.217	SSHv2	1084	Server: Encrypted packet (len=1016)
273	3.457999972	10.10.10.30	10.10.10.217	SSHv2	1397	Server: Encrypted packet (len=1329)
275	3.458140685	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
277	3.458253834	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
279	3.458379117	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
281	3.458505622	10.10.10.30	10.10.10.217	SSHv2	1516	Server: Encrypted packet (len=1448)
283	3.458604664	10.10.10.30	10.10.10.217	SSHv2	1203	Server: Encrypted packet (len=1135)
285	3.460275381	10.10.10.30	10.10.10.217	SSHv2	1156	Server: Encrypted packet (len=1088)
288	3.566701559	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=19686 Ack=3393 Win=35968 Len=0 TSval=202..
296	3.773019886	10.10.10.30	10.10.10.217	SSHv2	996	Server: Encrypted packet (len=928)
301	3.875038227	10.10.10.30	10.10.10.217	TCP	68	22 → 54750 [ACK] Seq=20614 Ack=3649 Win=38656 Len=0 TSval=202..
302	3.884546618	10.10.10.30	10.10.10.217	SSHv2	356	Server: Encrypted packet (len=288)
304	3.887758129	10.10.10.30	10.10.10.217	SSHv2	244	Server: Encrypted packet (len=176)
144	1.988687014	10.10.10.217	10.10.10.30	TCP	76	76 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T..
146	1.749104371	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=282026709..
147	1.749969181	10.10.10.217	10.10.10.30	SSHv2	92	Client: Protocol (SSH-2.0-paramiko_2.8.0)
152	1.783095427	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=25 Ack=22 Win=64256 Len=0 TSval=28202670..
153	1.783767952	10.10.10.217	10.10.10.30	SSHv2	876	Client: Key Exchange Init
155	1.788615968	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=833 Ack=526 Win=64128 Len=0 TSval=282026..
156	1.787658252	10.10.10.217	10.10.10.30	SSHv2	148	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
159	1.808892381	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=913 Ack=838 Win=64128 Len=0 TSval=282026..
160	1.810392866	10.10.10.217	10.10.10.30	SSHv2	84	Client: New Keys
162	1.851388667	10.10.10.217	10.10.10.30	SSHv2	132	Client: Encrypted packet (len=64)
165	1.852265515	10.10.10.217	10.10.10.30	SSHv2	180	Client: Encrypted packet (len=112)
168	1.967068526	10.10.10.217	10.10.10.30	SSHv2	148	Client: Encrypted packet (len=80)
173	2.041638098	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=1185 Ack=1918 Win=64128 Len=0 TSval=2820..
175	2.042029662	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=1185 Ack=1974 Win=64128 Len=0 TSval=2820..
176	2.042564415	10.10.10.217	10.10.10.30	SSHv2	148	Client: Encrypted packet (len=80)
180	2.052967583	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=1265 Ack=2086 Win=64128 Len=0 TSval=2820..
198	2.154852828	10.10.10.217	10.10.10.30	SSHv2	1412	Client: Encrypted packet (len=1344)
262	3.457153095	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2069 Ack=3534 Win=64128 Len=0 TSval=2820..
264	3.457205054	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=4982 Win=63488 Len=0 TSval=2820..
266	3.457331937	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=6438 Win=64128 Len=0 TSval=2820..
268	3.457458815	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=7878 Win=64128 Len=0 TSval=2820..
270	3.45753556	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=9322 Win=64128 Len=0 TSval=2820..
272	3.457655286	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=10342 Win=64128 Len=0 TSval=2820..
274	3.458009978	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=11671 Win=64128 Len=0 TSval=2820..
276	3.458148594	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=13119 Win=63488 Len=0 TSval=2820..
278	3.458265610	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=14567 Win=62592 Len=0 TSval=2820..
280	3.458389646	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=16015 Win=64128 Len=0 TSval=2820..
282	3.458513036	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=17463 Win=63488 Len=0 TSval=2820..
284	3.458615124	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=18598 Win=62592 Len=0 TSval=2820..
286	3.460285932	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=2669 Ack=19686 Win=64128 Len=0 TSval=2820..
287	3.566239899	10.10.10.217	10.10.10.30	SSHv2	852	Client: Encrypted packet (len=784)
297	3.773053910	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=3393 Ack=20614 Win=64128 Len=0 TSval=2820..
300	3.874616882	10.10.10.217	10.10.10.30	SSHv2	324	Client: Encrypted packet (len=256)
303	3.884573124	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=3649 Ack=20902 Win=64128 Len=0 TSval=282..
305	3.887782775	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [ACK] Seq=3649 Ack=21078 Win=64128 Len=0 TSval=282..
306	3.887847883	10.10.10.217	10.10.10.30	TCP	68	54750 → 22 [RST, ACK] Seq=3649 Ack=21078 Win=64128 Len=0 TSva..

Figura 5.4: Cattura effettuata con Wireshark della sessione NETCONF descritta

stato possibile ricostruire la durata dell'intera sessione che ammonta a circa 2,13876 secondi, valore non accettabile per implementare sia un'efficiente strategia di monitoraggio, ma soprattutto una logica di controllo reattiva. Poiché non è possibile decrittare i pacchetti ssh catturati con Wireshark

per analizzarne il contenuto, per stimare i tempi richiesti da ognuna delle tre operazioni effettuate (instaurazione sessione, get, e chiusura sessione) è stato utilizzato un semplice programma Python, sfruttando la libreria “time”. Dalla cattura riportata in figura 5.5 si evince come il contributo in termini di tempo più significativo è dato proprio dalla fase di instaurazione della sessione. Occorre



```

statistiche x
/home/stefano/PycharmProjects/QueryFlowCounters/venv/bin/python /home/
tempo instaurazione sessione: 1.719515085228337 secondi
tempo per la eseguire la get: 0.3089315891265869 secondi
tempo per la chiusura della sessione: 0.11100196838378906 secondi
Tempo totale: 2.139448642730713 secondi

```

Figura 5.5: Output di statistiche.py

precisare che il tempo richiesto per l'esecuzione delle operazioni presenti nel codice, come l'accesso ad un elemento di una lista, la scrittura su file, o le istruzioni di print, non è confrontabile con quello che una NETCONF <rpc> richiede; pertanto, può essere trascurato con buona approssimazione. Come mostrato, e come si potrà osservare nei report successivi, il tempo di una <get> si aggira intorno ai 0,25s. In questo modo si risparmiano circa 2 secondi, e nel caso in cui oltre alla <get>, sia necessaria anche un'operazione di <edit-config>, essa verrà eseguita circa quattro secondi prima (2 per ciascuna operazione), ottenendo una maggiore prontezza. Come si vede nelle catture riportate in seguito, il tempo per effettuare un'operazione NETCONF, non può essere un valore esatto in quanto è un'informazione che transita sulla rete, e come tale, risentirà inevitabilmente delle condizioni della stessa: se molto congestionata richiederà più tempo, se invece le risorse sono disponibili, si avrà un Round Trip Time minore. Oltre al plot mostrato precedentemente, che fornisce informazioni quantitative sulla simulazione, per far riferimento a dati più concreti, fornendo informazioni qualitative, è riportata anche parte dell'output della medesima simulazione, copiata dal terminale Python.

```

/home/stefano/PycharmProjects/Tesi/venv/bin/python /home/stefano/PycharmProjects/Tesi/monitoraggio.py
.....
Orario inizio simulazione: 05:35:26 AM
Orario: 05:35:27 AM
precision:7.43 Gbits/sec — G1:0.0 Gbits/sec
Tempo impiegato: 0.22321367263793945 secondi
Orario: 05:35:28 AM
precision:8.26 Gbits/sec — G1:0.0 Gbits/sec
Tempo impiegato: 0.23621392250061035 secondi
Orario: 05:35:29 AM
precision:8.26 Gbits/sec — G1:0.0 Gbits/sec
Tempo impiegato: 0.23778700828552246 secondi
Orario: 05:35:30 AM
precision:8.29 Gbits/sec — G1:0.0 Gbits/sec
Tempo impiegato: 0.2729928493499756 secondi
.....
Orario: 05:35:47 AM
precision:8.5 Gbits/sec — G1:0.0 Gbits/sec
Tempo impiegato: 0.25260210037231445 secondi
Orario: 05:35:48 AM
precision:7.28 Gbits/sec — G1:1.2 Gbits/sec
Tempo impiegato: 0.25281476974487305 secondi
Orario: 05:35:49 AM
precision:4.08 Gbits/sec — G1:4.11 Gbits/sec
Tempo impiegato: 0.2570359706878662 secondi
Orario: 05:35:50 AM
precision:4.11 Gbits/sec — G1:4.13 Gbits/sec
Tempo impiegato: 0.2722196578979492 secondi
Orario: 05:35:51 AM

```

```

precision:4.18 Gbits/sec —— G1:4.15 Gbits/sec
Tempo impiegato: 0.2880973815917969 secondi
Orario: 05:35:52 AM
precision:4.17 Gbits/sec —— G1:4.16 Gbits/sec
Tempo impiegato: 0.309659481048584 secondi
Orario: 05:35:53 AM
precision:3.79 Gbits/sec —— G1:3.75 Gbits/sec
Tempo impiegato: 0.22171783447265625 secondi
Orario: 05:35:54 AM
precision:4.13 Gbits/sec —— G1:4.09 Gbits/sec
Tempo impiegato: 0.22557854652404785 secondi
Orario: 05:35:55 AM
precision:4.1 Gbits/sec —— G1:4.13 Gbits/sec
Tempo impiegato: 0.23114442825317383 secondi
Orario: 05:35:56 AM
precision:4.12 Gbits/sec —— G1:4.26 Gbits/sec
Tempo impiegato: 0.29855847358703613 secondi
Orario: 05:35:57 AM
precision:4.37 Gbits/sec —— G1:4.33 Gbits/sec
Tempo impiegato: 0.3500511646270752 secondi
Orario: 05:35:59 AM
precision:4.78 Gbits/sec —— G1:2.56 Gbits/sec
Tempo impiegato: 0.2541642189025879 secondi
Orario: 05:36:00 AM
precision:8.25 Gbits/sec —— G1:0.0 Gbits/sec
Tempo impiegato: 0.2591121196746826 secondi
Orario: 05:36:01 AM
precision:8.16 Gbits/sec —— G1:0.0 Gbits/sec
Tempo impiegato: 0.255568265914917 secondi
.....
Orario: 05:36:06 AM
precision:8.31 Gbits/sec —— G1:0.0 Gbits/sec
Tempo impiegato: 0.307905912399292 secondi

```

Listing 5.1: Output del terminale eseguendo il programma monitoraggio.py

Dall'output appena mostrato si osserva come dalle ore 05:35:49 AM, alle ore 05:35:59 AM, quindi per ben 10 secondi, la bit rate di precision, scende sotto i 5 Gbps, valore richiesto per rispettare la QoS. Si evince come senza logica di controllo, le prestazioni sono compromesse, e non è possibile rispettare la condizione di bandwidth minima superiore ad una certa soglia (nell'esempio 5Gbps) da garantire a precision. Per generare il traffico è stato utilizzato il tool iperf3. Nell'esempio riportato, il server Core1 lavora in modalità iperf3 -s (modalità server), ascoltando contemporaneamente su due porte, la 5001, su cui trasmette G1, e la 5201, su cui trasmette precision, che lavorano in modalità iperf3 -c (modalità client). Attraverso il comando -p, eseguito in modalità iperf3 -s, si specifica la porta su cui mettersi in ascolto, mentre in modalità iperf3 -c, quella del server su cui trasmettere, precedentemente abilitata all'ascolto. In modalità iperf3 -c è possibile settare alcuni parametri della sessione, come la bandwidth attraverso il comando -b, o la durata della trasmissione con il comando -t (può essere terminata prima digitando Ctrl+C). In figura 5.6 si riportano i comandi eseguiti lato client, mentre lato server è stato necessario eseguire precedentemente il comando "iperf3 -s -p", specificando il numero di porta. Per abilitare sullo stesso server due comunicazioni iperf3 distinte, ma simultanee, è stato effettuato un doppio accesso tramite ssh al server, eseguendo su un terminale il comando iperf3 -s -p 5001, e sull'altro iperf3 -s -p 5201. Mentre per precision la bandwidth fissata su iperf3 ammonta a 10 Gbps, e resta la stessa per l'intero svolgimento del programma, simulando un servizio che richiede costantemente una bit rate elevata e superiore ai 5 Gbps, per G1, volendo simulare traffico intermittente, viene settata a 1 Gbps e a 2 Gbps per descrivere lo scenario in cui G1 non abbia grandi quantità di dati da trasmettere, a 10 Gbps, per riprodurre picchi di traffico. Per modellare il traffico irregolare di G1, è stato utilizzato sempre lo stesso comando "iperf3 -c 10.20.30.1 -b X -p 5001 -t 20" lato client, mentre il server era in ascolto, variando il valore di X, assegnando 1, per trasmettere ad 1Gbps, 2, per 2Gbps, e 10 (Gbps), per generare picchi. Pur avendo provato a trasmettere a 10Gbps, G1 non è mai riuscita a superare la soglia di 4,2 Gbps,

circa, in quanto precision richiedeva per l'intera durata della simulazione, una bit rate pari a 10 Gbps. Infatti, nel caso in cui la somma di questi valori sfiora la bit rate massima raggiungibile in upstream nella GPON (in linea teorica 10 Gbps, ma come si evince dai report e dal plot, di circa 8/8.5 Gbps), essa viene spartita. I risultati ottenuti attraverso il codice Python, sopra riportati, sono perfettamente in linea con le statistiche che iperf3 riporta, mostrate in figura 5.6.

```

precision@precision-Precision-Tower-3620:~$ iperf3 -c 10.20.20.1 -b 10g -p 5201 -t 20
1 -t 150
Connecting to host 10.20.20.1, port 5201
[4] local 10.20.20.2 port 41984 connected to 10.20.20.1 port 5201
[4] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 785 MBytes 6.59 Gbits/sec 82 1.73 MBytes
[4] 1.00-2.00 sec 926 MBytes 7.77 Gbits/sec 11 1.62 MBytes
[4] 2.00-3.00 sec 926 MBytes 7.77 Gbits/sec 3 1.48 MBytes
[4] 3.00-4.00 sec 926 MBytes 7.76 Gbits/sec 9 1.36 MBytes
[4] 4.00-5.00 sec 926 MBytes 7.77 Gbits/sec 8 1.79 MBytes
[4] 5.00-6.00 sec 926 MBytes 7.76 Gbits/sec 2 1.66 MBytes
[4] 6.00-7.00 sec 926 MBytes 7.77 Gbits/sec 3 1.53 MBytes
[4] 7.00-8.00 sec 926 MBytes 7.76 Gbits/sec 1 1.40 MBytes
[4] 8.00-9.00 sec 926 MBytes 7.77 Gbits/sec 6 1.82 MBytes
[4] 9.00-10.00 sec 926 MBytes 7.77 Gbits/sec 2 1.70 MBytes
[4] 10.00-11.00 sec 926 MBytes 7.76 Gbits/sec 4 1.59 MBytes
[4] 11.00-12.00 sec 926 MBytes 7.77 Gbits/sec 4 1.44 MBytes
[4] 12.00-13.00 sec 925 MBytes 7.76 Gbits/sec 8 740 KBytes
[4] 13.00-14.00 sec 927 MBytes 7.77 Gbits/sec 0 1.48 MBytes
[4] 14.00-15.00 sec 926 MBytes 7.77 Gbits/sec 11 1.36 MBytes
[4] 15.00-16.00 sec 926 MBytes 7.77 Gbits/sec 8 1.78 MBytes
[4] 16.00-17.00 sec 926 MBytes 7.77 Gbits/sec 4 1.66 MBytes
[4] 17.00-18.00 sec 926 MBytes 7.77 Gbits/sec 6 1.52 MBytes
[4] 18.00-19.00 sec 926 MBytes 7.77 Gbits/sec 13 1.47 MBytes
[4] 19.00-20.00 sec 926 MBytes 7.77 Gbits/sec 4 1.30 MBytes
[4] 20.00-21.00 sec 926 MBytes 7.77 Gbits/sec 6 1.75 MBytes
[4] 21.00-22.00 sec 926 MBytes 7.77 Gbits/sec 1 1.64 MBytes
[4] 22.00-23.00 sec 926 MBytes 7.76 Gbits/sec 9 1.50 MBytes
[4] 23.00-24.00 sec 926 MBytes 7.77 Gbits/sec 1 1.36 MBytes
[4] 24.00-25.00 sec 903 MBytes 7.57 Gbits/sec 5 1.24 MBytes
[4] 25.00-26.00 sec 731 MBytes 6.13 Gbytes/sec 0 1.62 MBytes
[4] 26.00-27.00 sec 806 MBytes 6.76 Gbits/sec 12 1.48 MBytes
[4] 27.00-28.00 sec 805 MBytes 6.76 Gbits/sec 17 1.29 MBytes
[4] 28.00-29.00 sec 806 MBytes 6.76 Gbits/sec 8 1.69 MBytes
[4] 29.00-30.00 sec 829 MBytes 6.96 Gbits/sec 7 1.59 MBytes
[4] 30.00-31.00 sec 925 MBytes 7.76 Gbits/sec 5 1.46 MBytes
[4] 31.00-32.00 sec 924 MBytes 7.75 Gbits/sec 6 1.86 MBytes
[4] 32.00-33.00 sec 902 MBytes 7.56 Gbits/sec 19 1.74 MBytes
[4] 33.00-34.00 sec 686 MBytes 5.75 Gbits/sec 9 1.53 MBytes
[4] 34.00-35.00 sec 687 MBytes 5.76 Gbits/sec 17 1.35 MBytes
[4] 35.00-36.00 sec 686 MBytes 5.76 Gbits/sec 0 1.66 MBytes
[4] 36.00-37.00 sec 687 MBytes 5.76 Gbits/sec 4 1.49 MBytes
[4] 37.00-38.00 sec 878 MBytes 7.37 Gbits/sec 0 1.86 MBytes
[4] 38.00-39.00 sec 925 MBytes 7.76 Gbits/sec 6 1.74 MBytes
[4] 39.00-40.00 sec 847 MBytes 7.11 Gbits/sec 10 1.28 MBytes
[4] 40.00-41.00 sec 462 MBytes 3.87 Gbits/sec 0 1.53 MBytes
[4] 41.00-42.00 sec 462 MBytes 3.87 Gbits/sec 0 1.73 MBytes
[4] 42.00-43.00 sec 464 MBytes 3.98 Gbits/sec 1 1.44 MBytes
[4] 43.00-44.00 sec 464 MBytes 3.89 Gbits/sec 0 1.66 MBytes
[4] 44.00-45.00 sec 465 MBytes 3.98 Gbits/sec 3 1.35 MBytes
[4] 45.00-46.00 sec 464 MBytes 3.98 Gbits/sec 0 1.58 MBytes
[4] 46.00-47.00 sec 462 MBytes 3.87 Gbits/sec 2 1.25 MBytes
[4] 47.00-48.00 sec 462 MBytes 3.87 Gbits/sec 0 1.50 MBytes
[4] 48.00-49.00 sec 461 MBytes 3.87 Gbits/sec 0 1.71 MBytes
[4] 49.00-50.00 sec 541 MBytes 4.54 Gbytes/sec 1 1.46 MBytes
[4] 50.00-51.00 sec 924 MBytes 7.75 Gbits/sec 0 1.87 MBytes
[4] 51.00-52.00 sec 925 MBytes 7.76 Gbits/sec 1 1.75 MBytes
[4] 52.00-53.00 sec 710 MBytes 5.98 Gbits/sec 4 1.55 MBytes
[4] 53.00-54.00 sec 687 MBytes 5.76 Gbits/sec 26 1.32 MBytes
[4] 54.00-55.00 sec 687 MBytes 5.76 Gbits/sec 8 1.67 MBytes
[4] 55.00-56.00 sec 718 MBytes 0.02 Gbits/sec 16 1.48 MBytes
[4] 56.00-57.00 sec 924 MBytes 7.75 Gbits/sec 19 1.40 MBytes
g1@g1:~$ iperf3 -c 10.20.30.1 -b 1g -p 5001 -t 20
Connecting to host 10.20.30.1, port 5001
[4] local 10.20.30.2 port 48808 connected to 10.20.30.1 port 5001
[4] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 71.5 MBytes 6.00 Mbytes/sec 60 952 Kbytes
[4] 1.00-2.00 sec 158 MBytes 1.32 Gbits/sec 0 1.05 Mbytes
[4] 2.00-3.00 sec 119 MBytes 1.00 Gbits/sec 0 1.12 Mbytes
[4] 3.00-4.00 sec 119 MBytes 1.00 Gbits/sec 0 1.19 Mbytes
[4] 4.00-5.00 sec 119 MBytes 999 Mbytes/sec 0 1.26 Mbytes
[4] 5.00-5.65 sec 83.5 MBytes 1.07 Gbits/sec 0 1.30 Mbytes
[4] 6.00-7.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 7.00-8.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 8.00-9.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 9.00-10.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 10.00-11.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 11.00-12.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 12.00-13.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 13.00-14.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 14.00-15.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 15.00-16.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 16.00-17.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 17.00-18.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 18.00-19.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 19.00-20.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 20.00-21.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 21.00-22.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 22.00-23.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 23.00-24.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 24.00-25.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 25.00-26.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 26.00-27.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 27.00-28.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 28.00-29.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 29.00-30.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 30.00-31.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 31.00-32.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 32.00-33.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 33.00-34.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 34.00-35.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 35.00-36.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 36.00-37.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 37.00-38.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 38.00-39.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 39.00-40.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 40.00-41.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 41.00-42.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 42.00-43.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 43.00-44.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 44.00-45.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 45.00-46.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 46.00-47.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 47.00-48.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 48.00-49.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 49.00-50.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 50.00-51.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 51.00-52.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 52.00-53.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 53.00-54.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 54.00-55.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 55.00-56.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
[4] 56.00-57.00 sec 119 MBytes 1.00 Gbits/sec 0 1.30 Mbytes
g1@g1:~$ iperf3 -c 10.20.30.1 -b 2g -p 5001 -t 20
Connecting to host 10.20.30.1, port 5001
[4] local 10.20.30.2 port 48812 connected to 10.20.30.1 port 5001
[4] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 215 Mbytes 1.80 Gbits/sec 65 1.04 Mbytes
[4] 1.00-2.00 sec 238 Mbytes 2.00 Gbits/sec 0 1.20 Mbytes
[4] 2.00-3.00 sec 238 Mbytes 2.00 Gbits/sec 0 1.33 Mbytes
[4] 3.00-4.00 sec 238 Mbytes 2.00 Gbits/sec 0 1.45 Mbytes
[4] 4.00-4.30 sec 94.0 Mbytes 2.67 Gbits/sec 0 1.50 Mbytes
[4] 5.00-5.65 sec 670 Mbytes 994 Mbytes/sec 60 sender
[4] 6.00-7.00 sec 0.00 Bytes 0.00 bits/sec 0 receiver
iperf3: interrupt - the client has terminated
g1@g1:~$ iperf3 -c 10.20.30.1 -b 10g -p 5001 -t 20
Connecting to host 10.20.30.1, port 5001
[4] local 10.20.30.2 port 48816 connected to 10.20.30.1 port 5001
[4] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 400 Mbytes 3.35 Gbits/sec 59 1.50 Mbytes
[4] 1.00-2.00 sec 464 Mbytes 3.90 Gbits/sec 0 1.71 Mbytes
[4] 2.00-3.00 sec 462 Mbytes 3.88 Gbits/sec 1 1.41 Mbytes
[4] 3.00-4.00 sec 462 Mbytes 3.87 Gbits/sec 0 1.63 Mbytes
[4] 4.00-5.00 sec 461 Mbytes 3.87 Gbits/sec 2 1.31 Mbytes
[4] 5.00-6.00 sec 462 Mbytes 3.87 Gbits/sec 0 1.55 Mbytes
[4] 6.00-7.00 sec 463 Mbytes 3.88 Gbits/sec 0 1.76 Mbytes
[4] 7.00-8.00 sec 465 Mbytes 3.90 Gbits/sec 3 1.46 Mbytes
[4] 8.00-9.00 sec 464 Mbytes 3.89 Gbits/sec 0 1.68 Mbytes
[4] 9.00-10.00 sec 464 Mbytes 3.90 Gbits/sec 6 1.37 Mbytes
[4] 10.00-10.13 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 11.00-12.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 12.00-13.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 13.00-14.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 14.00-15.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 15.00-16.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 16.00-17.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 17.00-18.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 18.00-19.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 19.00-20.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 20.00-21.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 21.00-22.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 22.00-23.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 23.00-24.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 24.00-25.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 25.00-26.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 26.00-27.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 27.00-28.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 28.00-29.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 29.00-30.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 30.00-31.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 31.00-32.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 32.00-33.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 33.00-34.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 34.00-35.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 35.00-36.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 36.00-37.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 37.00-38.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 38.00-39.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 39.00-40.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 40.00-41.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 41.00-42.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 42.00-43.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 43.00-44.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 44.00-45.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 45.00-46.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 46.00-47.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 47.00-48.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 48.00-49.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 49.00-50.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 50.00-51.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 51.00-52.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 52.00-53.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 53.00-54.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 54.00-55.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 55.00-56.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
[4] 56.00-57.00 sec 62.1 Mbytes 3.89 Gbits/sec 0 1.41 Mbytes
g1@g1:~$ iperf3 -c 10.20.30.1 -b 2g -p 5001 -t 20
Connecting to host 10.20.30.1, port 5001
[4] local 10.20.30.2 port 48820 connected to 10.20.30.1 port 5001
[4] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 215 Mbytes 1.80 Gbits/sec 73 1.41 Mbytes
[4] 1.00-2.00 sec 238 Mbytes 2.00 Gbits/sec 0 1.53 Mbytes
[4] 2.00-3.00 sec 238 Mbytes 2.00 Gbits/sec 0 1.64 Mbytes
[4] 3.00-3.83 sec 208 Mbytes 2.09 Gbits/sec 0 1.72 Mbytes
[4] 4.00-4.38 sec 899 Mbytes 1.97 Gbits/sec 73 sender
[4] 5.00-5.38 sec 0.00 Bytes 0.00 bits/sec 0 receiver
iperf3: interrupt - the client has terminated

```

Figura 5.6: Generazione traffico tramite iperf3, nel caso di monitoraggio

5.4 Monitoraggio del traffico nella PON e configurazioni di controllo

lo

Per garantire a precision la QoS che esso richiede, è necessario introdurre un controllore che, in caso di eventi indesiderati, come picchi di traffico prodotti da G1, che tendono a sottrarre bandwidth a precision, risponda tempestivamente, attuando nuove politiche di allocazione di bandwidth, riconfigurando dinamicamente il dispositivo di rete e le politiche di traffico. Per comprendere l'idea del controllo, bisogna prima introdurre i concetti di Ethernet Policy Map e Class Map. Attraverso una Ethernet Policy Map si definiscono le politiche di traffico da applicare a tutti o solo ad alcuni flussi all'interno di una Ethernet Class Map. Una Policy Map si applica ad un'interfaccia Ethernet verso l'ONT [5] ed i meccanismi per implementare QoS sono descritti proprio da esse. Una Ethernet Class

Map definisce un insieme di criteri (regole di classe) per classificare il traffico in una classe, come ad esempio il traffico VoIP. La Policy Map definisce il livello di bandwidth o di priorità nelle code dei buffer, da applicare al traffico che rispetta i criteri specificati nella Class Map di riferimento. Intuitivamente, possiamo considerare una Class Map come un attributo di una Policy Map [4]. Esistono diversi parametri che possono essere configurati, e che permettono di modellare le politiche di traffico, sia in upstream che in downstream. Nella trattazione, ne sono stati considererati solo due: il Committed Information Rate (cir), e l'Excess Information Rate (eir). Il cir è il data rate medio massimo garantito per un traffico in upstream fornito per il particolare servizio Ethernet, che nel nostro caso di studio non può eccedere il valore di 8700000 (kbps, cioè 8,7 Gbps), e per default è settato a 0. L'OLT supporta di default al suo interno un algoritmo che controlla che questo valore non venga mai superato, sommando i contributi assegnati a ciascuna ONT, in modo da non generare errori che ne deriverebbero provando a superare la velocità fisica dell'interfaccia. L'eir rappresenta invece il data rate medio massimo non superabile, e a differenza del cir, non è traffico garantito, ma addizionale, assegnabile solo se disponibile. Anche esso è misurato in kbps, e non può eccedere il valore di 10000000 (10 Gbps), per default settato a 0. Attraverso questi due parametri, sfruttando principalmente il cir, è stata garantita a precisione, in caso di picchi di traffico da parte di G1, la sua porzione di bandwidth, con tempi richiesti per individuare l'evento indesiderato ed attuare le operazioni di riconfigurazione, prossimi a 0,45s. La strategia che è stata attuata in presenza di picchi generati da G1, per garantire la bandwidth a precisione, è stata l'assegnazione del valore del cir pari a 5000000 (5Gbps), nella class-map-ethernet "ELINE_PM_1", a sua volta nella policy-map "Eth-match-any-1". Così come per il processo di monitoraggio, anche per apportare la modifica nella policy-map, è stata utilizzata una <rpc> NETCONF, in questo caso una <edit-config>. Anche per questo esempio il valore dell'eir è settato di default al massimo, 10000000 (kbps, 10 Gbps), in entrambe le policy-map, per sfruttare l'intera bandwidth della PON. In questo secondo scenario, sempre utilizzando iperf3, è stato generato traffico in maniera del tutto analoga al caso precedentemente descritto. In più, attraverso l'opzione di iperf3 -T, che permette di stampare qualsiasi stringa prima del report, e l'implementazione di un ciclo for, direttamente su shell linux, è stato possibile visualizzare accanto alle statistiche riportate da iperf3, l'orario della generazione del primo picco, in modo da confrontarlo con l'orario printato dal codice Python. Nelle catture 5.7 e 5.8, sono riportati tutti i comandi iperf3 lato client utilizzati per generare il traffico per tutta la durata della simulazione. Lato server sono gli stessi descritti nell'esempio precedente. Per poter mostrare due diversi momenti di picco di traffico, tra loro distanziati nel tempo, ed il conseguente effetto delle politiche di controllo implementate, la durata di questa simulazione è stata leggermente maggiore alla precedente, per un totale di 55 secondi. Come nel caso precedente, sono riportati il plot, in figura 5.9, per descrivere quantitativamente i risultati ottenuti, e parte dell'output copiato dal terminale Python dopo l'esecuzione del programma MonitoraggioConControllo.py, per poter commentare alcuni aspetti interessanti.

```
/home/stefano/PycharmProjects/Tesi/venv/bin/python /home/stefano/PycharmProjects/Tesi/
MonitoraggioConControllo.py
.....
Orario inizio simulazione: 06:53:17 AM
Orario: 06:53:18 AM . Traffico di G1 nella norma
precision: 6.46 Gbits/sec — G1: 0.98 Gbits/sec
Tempo impiegato: 0.21541547775268555 secondi
.....
Orario: 06:53:26 AM . Traffico di G1 nella norma
precision: 6.56 Gbits/sec — G1: 2.13 Gbits/sec
Tempo impiegato: 0.28209662437438965 secondi
Orario: 06:53:27 AM, picco di traffico di G1
```

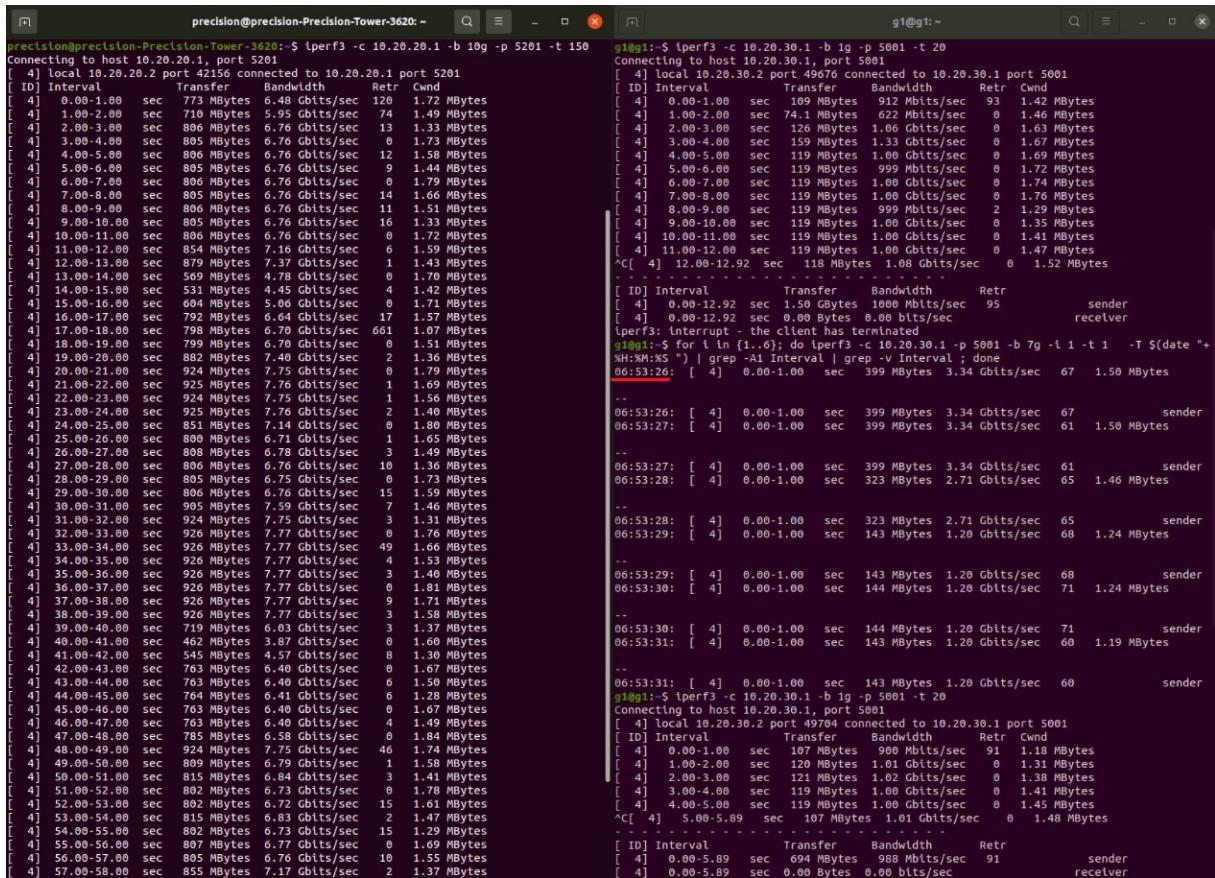


Figura 5.7: Generazione traffico tramite iperf3, nel caso di monitoraggio e controllo. Prima cattura

```

precision: 5.18 Gbits/sec —— G1: 3.15 Gbits/sec
RPCReply for CHANGING cir is <?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:2f938a4c-2acc-4d1f-8f67-28
d29cc3e61c" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><ok/></rpc-reply>
CHANGING cir successful
cir settato a 5.0 Gbits/sec
Tempo impiegato: 0.4594600200653076 secondi
Orario: 06:53:28 AM . Nuovo picco di traffico di G1
precision: 4.99 Gbits/sec —— G1: 3.12 Gbits/sec
Tempo impiegato: 0.2780120372772217 secondi
Orario: 06:53:29 AM . Nuovo picco di traffico di G1
precision: 6.29 Gbits/sec —— G1: 1.86 Gbits/sec
Tempo impiegato: 0.2999739646911621 secondi
.
.
.
Orario: 06:53:37 AM —— Il traffico di G1 sta rientrando nei parametri. Secondi al ripristino: 1 s
precision: 8.53 Gbits/sec —— G1: 0.02 Gbits/sec
Tempo impiegato: 0.2975337505340576 secondi
Orario: 06:53:38 AM —— Il traffico di G1 è rientrato nei parametri
RPCReply for CHANGING cir is <?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:f66fdfc4-d4bb-4b75-a149-1
f97e8e511b0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><ok/></rpc-reply>
CHANGING cir successful
cir settato a 0.0 Gbits/sec
Tempo impiegato: 0.47794628143310547 secondi
precision: 7.28 Gbits/sec —— G1: 1.16 Gbits/sec
.
.
.
Orario: 06:53:51 AM . Traffico di G1 nella norma
precision: 8.29 Gbits/sec —— G1: 0.0 Gbits/sec
Tempo impiegato: 0.26276493072509766 secondi
Orario: 06:53:52 AM . Traffico di G1 nella norma
precision: 8.47 Gbits/sec —— G1: 0.0 Gbits/sec
Tempo impiegato: 0.2847275733947754 secondi
Orario: 06:53:53 AM , picco di traffico di G1
precision: 4.99 Gbits/sec —— G1: 3.71 Gbits/sec
RPCReply for CHANGING cir is <?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:30dab2a4-e9eb-4e00-a25d-
eea8688c4547" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><ok/></rpc-reply>
CHANGING cir successful
cir settato a 5.0 Gbits/sec

```

5.4. Monitoraggio del traffico nella PON e configurazioni di controllo

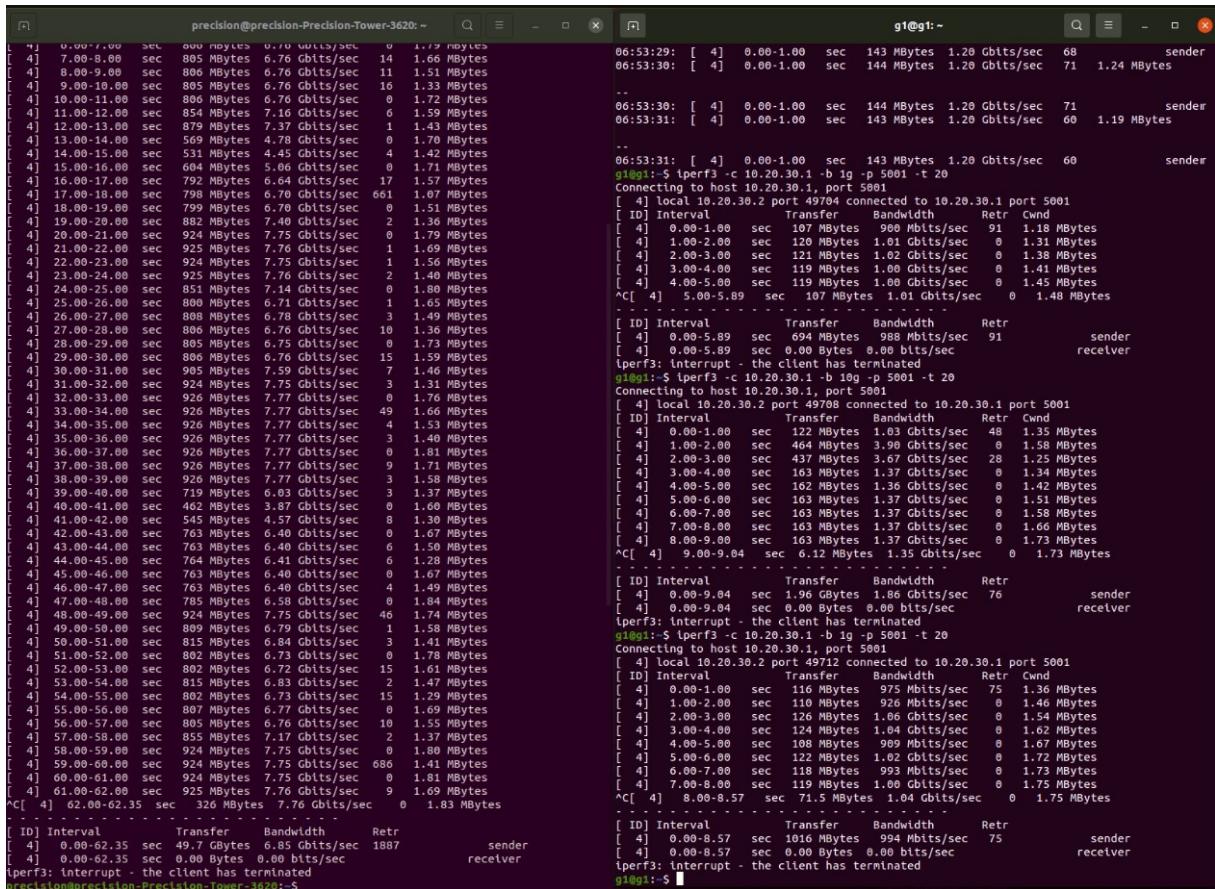


Figura 5.8: Generazione traffico tramite iperf3, nel caso di monitoraggio e controllo. Seconda cattura

```

Tempo impiegato: 0.48020458221435547 secondi
Orario: 06:53:54 AM . Nuovo picco di traffico di G1
precision: 4.07 Gbits/sec —— G1: 4.09 Gbits/sec
Tempo impiegato: 0.30437207221984863 secondi
Orario: 06:53:55 AM . Nuovo picco di traffico di G1
precision: 5.31 Gbits/sec —— G1: 2.03 Gbits/sec
Tempo impiegato: 0.22795414924621582 secondi
Orario: 06:53:56 AM . Nuovo picco di traffico di G1
precision: 6.96 Gbits/sec —— G1: 1.52 Gbits/sec
Tempo impiegato: 0.2856907844543457 secondi
Orario: 06:53:57 AM . Nuovo picco di traffico di G1
precision: 7.06 Gbits/sec —— G1: 1.48 Gbits/sec
Tempo impiegato: 0.3081777095794678 secondi
Orario: 06:53:58 AM . Nuovo picco di traffico di G1
precision: 6.24 Gbits/sec —— G1: 1.32 Gbits/sec
Tempo impiegato: 0.22061467170715332 secondi
Orario: 06:53:59 AM . Nuovo picco di traffico di G1
precision: 6.81 Gbits/sec —— G1: 1.47 Gbits/sec
Tempo impiegato: 0.2433152198791504 secondi
Orario: 06:54:00 AM . Nuovo picco di traffico di G1
precision: 6.9 Gbits/sec —— G1: 1.47 Gbits/sec
Tempo impiegato: 0.27735376358032227 secondi
Orario: 06:54:01 AM — Il traffico di G1 sta rientrando nei parametri. Secondi al ripristino: 5 s
precision: 7.69 Gbits/sec —— G1: 0.63 Gbits/sec
Tempo impiegato: 0.29358530044555664 secondi
Orario: 06:54:02 AM — Il traffico di G1 sta rientrando nei parametri. Secondi al ripristino: 4 s
precision: 7.94 Gbits/sec —— G1: 0.42 Gbits/sec
Tempo impiegato: 0.3233206272125244 secondi
Orario: 06:54:03 AM — Il traffico di G1 sta rientrando nei parametri. Secondi al ripristino: 3 s
precision: 6.64 Gbits/sec —— G1: 0.96 Gbits/sec
Tempo impiegato: 0.24065589904785156 secondi
Orario: 06:54:04 AM — Il traffico di G1 sta rientrando nei parametri. Secondi al ripristino: 2 s
precision: 7.26 Gbits/sec —— G1: 1.13 Gbits/sec
Tempo impiegato: 0.2541625499725342 secondi
Orario: 06:54:05 AM — Il traffico di G1 sta rientrando nei parametri. Secondi al ripristino: 1 s
precision: 7.18 Gbits/sec —— G1: 1.09 Gbits/sec
Tempo impiegato: 0.26658058166503906 secondi
Orario: 06:54:06 AM — Il traffico di G1 è rientrato nei parametri

```

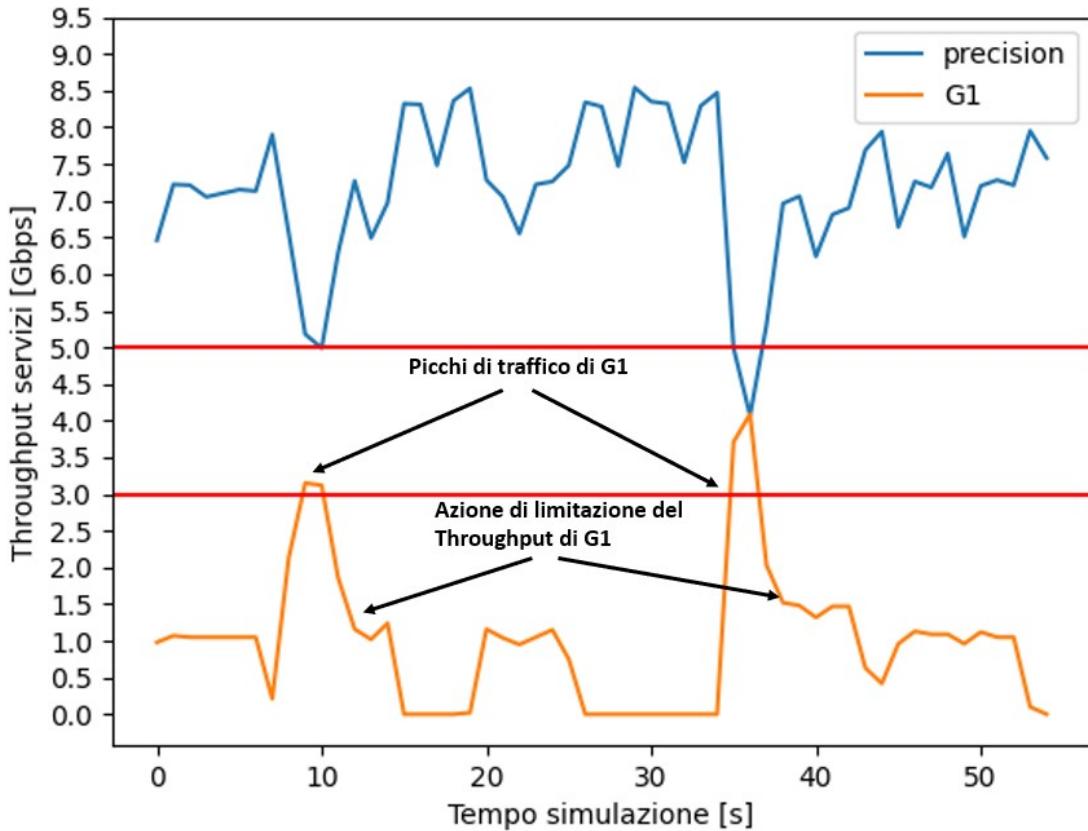


Figura 5.9: Plot dei throughput nel caso di monitoraggio e controllo

```
RPCReply for CHANGING cir is <?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:9ba51292-467b-4dc5-b93c
-475fe0584641" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><ok/></rpc-reply>
CHANGING cir successful
cir settato a 0.0 Gbits/sec
Tempo impiegato: 0.5301167964935303 secondi
precision: 7.64 Gbits/sec — G1: 1.09 Gbits/sec
.....
Orario: 06:54:12 AM . Traffico di G1 nella norma
precision: 7.58 Gbits/sec — G1: 0.0 Gbits/sec
Tempo impiegato: 0.22579503059387207 secondi
```

Listing 5.2: Parte dell'output del terminale Python eseguendo il programma MonitoraggioConControllo.py

Il leggero offset di circa 1 secondo tra i due orari di generazione, mostrato in 5.7 (06:53:26), e conseguente rilevazione, mostrato nell'output del programma Python (06:53:27 AM), del primo picco di traffico, è dovuto al fatto che, pur avendo il valore delle bit rate ad intervalli di un secondo esatto, sia con iperf3 che con il codice Python, le operazioni non vengono eseguite allo scoccare del secondo, dovuto al non perfetto sincronismo. In questo gap i contatori dell'OLT non sono aggiornati, in quanto non è ancora stato generato traffico con iperf3, mentre la `<get>` è già stata richiesta. Per contestualizzare il problema, ci serviamo di un esempio. Ipotizzando che la `<get>` parta alle ore 09:53:59.05 AM e la trasmissione iperf3 alle ore 09:53:59.37, c'è un evidente tempo morto, in cui la get è in esecuzione, ma i valori dei contatori non cambiano, dato che per 0,32 secondi non è presente alcun traffico. Eseguendo il prodotto 0,32s x 4,25Gbps si ottengono circa 1,36 Gbits non trasmessi, e quindi la bit rate sarà pari alla differenza tra quella massima disponibile 4,25 Gbps (metà della totale, in quanto entrambi i flussi spinti al massimo) e 1,36 Gbps appena ricavata. Questo valore ammonta a circa 2,89 Gbps, minore di 3 Gbps, soglia da superare per

cambiare le politiche di allocazione di bandwidth. Certamente nel secondo successivo, cioè dalle 09:53:59.05 AM alle 09:54:00.05 AM transiteranno tutti i 4,25 Gbits attesi, ottenendo la bit rate di 4,25 Gbps, sufficiente per attuare le strategie per la limitazione della bandwidth di G1. Per considerare il traffico di G1 un evento di picco, è sufficiente che la bit rate ad esso associata superi il valore di 3 Gbps. Dai plot precedenti riportati nelle figure 5.3 e 5.9, e dagli output copiati dal terminale Python, si è visto che la bit rate massima supportata per il traffico in upstream nella PON ammonta a circa 8/8,5 Gbps. Per questo motivo, evitare di far scendere il valore della bit rate di precision sotto la soglia dei 5 Gbps, equivale a non permettere al valore della bit rate di G1 di superare i 3/3,5 Gbps. In seguito ad entrambi i picchi, verificatisi nelle ore 06:53:27 AM e 06:53:53 AM, si vede dall'output ripreso dal terminale di Python come, pur continuando a voler trasmettere prima a 7Gbps e poi a 10Gbps (non interrompendo i comandi iperf3 utilizzati per richiedere da G1 una bandwidth di 7 e 10 Gbps, vedere nelle catture 5.7 e 5.8), non è consentito, grazie alle logiche di controllo implementate. Di conseguenza, non è possibile far scendere nuovamente la bandwidth di precision sotto i 5Gbps, ottenendo il comportamento desiderato. In questo lasso di tempo la bit rate di G1 è limitata a circa 1,5 Gbps. Ma il fatto che ora G1 trasmetta a 1,5 Gbps non vuol dire che non abbia più grandi quantità di dati da trasmettere. Proprio per questo, prima di ripristinare le politiche di traffico, è opportuno attendere un intervallo di tempo, nel codice 5 secondi, in cui G1 trasmette a meno di 1,2 Gbps, per essere sicuri che la diminuzione della bit rate non sia solo frutto delle configurazioni di limitazione eseguite, ma che la quantità di dati che G1 doveva inviare, sia stata smaltita. Infatti, la bandwidth in eccesso, circa 3 Gbps, viene spartita se precision eccede i 5Gbps garantiti. La spartizione, se la somma della bit rate che G1 richiede e della bit rate addizionale oltre ai 5Gbps a cui precision può attingere, supera i 3 Gbps, come nell'esempio, avverrà in maniera equa tra le due ONT. Ciò vuol dire circa 6,5 Gbps totali per precision e circa 1,5 Gbps per G1. Qualora non venisse considerata una soglia inferiore a 1,5 Gbps, non si starebbe verificando effettivamente il rientro nei parametri del traffico di G1, ma si starebbe aspettando solo un time out, dopo il quale, si potrebbe osservare il traffico di G1 viaggiare sotto i 3 Gbps, così come sopra i 3Gbps. Nel caso fosse rilevata una bit rate troppo elevata per G1, si dovrà immediatamente ritornare alla configurazione con limitazione di bandwidth, appena un secondo dopo aver ripristinato quella iniziale. Infatti è solo nel momento in cui il traffico si esaurisce di sua natura (nell'esempio in seguito alla conclusione del ciclo nel primo caso, o interrompendo la trasmissione nel secondo), e non solo per via delle nuove regole di assegnazione di bandwith, che inizia e termina il countdown per il ripristino. Questo problema si sarebbe potuto risolvere seguendo anche un'altra strada, ma in maniera meno efficiente, limitando l'eir all'interno della policy-map "Eth-match-any-2", scegliendo un valore inferiore ai 3 Gbps. Per ripristinare le politiche iniziali si sarebbe potuto utilizzare la stessa tecnica sopra descritta, scegliendo una soglia per effettuare il controllo, minore dell'eir settato. Tuttavia, questa strategia sarebbe risultata meno efficiente della precedente, in quanto, qualora precision avesse avuto in alcuni istanti di tempo del traffico da trasmettere a bit rate inferiori ai 5 Gbps, G1 non avrebbe potuto in alcun modo usufruire della bandwidth in eccesso, che sarebbe andata sprecata. Con la scelta invece di garantire cir sufficiente per precision, qualora esso avesse meno informazioni da trasmettere, avendo bisogno di bit rate minori, G1 potrà usufruire della bandwidth in eccesso, che in questo caso non andrà buttata. Come si osserva nell'output del terminale Python in seguito alle rpc-reply printate, il tempo necessario per individuare il picco o il rientro del traffico nei parametri, ed effettuare una <edit-config> per settare

il cir all'interno delle policy map, è prossimo a 0,45 secondi. Come anticipato precedentemente, si vede chiaramente come il tempo di queste operazioni non sia fissato, ma dipenda dalle condizioni della rete.

Capitolo 6

Conclusioni

Nel presente scritto, dopo aver presentato il concetto di Network Slicing, strumento fondamentale adottato nelle reti di nuova generazione, e le tecniche note come Software Defined Networking e Network Function Virtualization, di cui ci si serve per implementarlo, è stata descritta ed approfondita, la cosiddetta NETCONF-YANG solution. Supportata da un gran numero di vendor, si sta promuovendo come protocollo universale per la Southbound Interface, per configurare e gestire sia le VNF, che i dispositivi di rete fisici nello scenario SDN, riconoscendone le grandi potenzialità. Si è capito come la coesione del protocollo NETCONF e del linguaggio YANG, fornisca una valida soluzione per aggiornare programmaticamente e modificare le configurazioni di un dispositivo di rete. Nello specifico, YANG descrive le modifiche delle configurazioni che bisogna apportare, ed attraverso NETCONF, esse si applicano ad un preciso datastore presente all'interno del dispositivo. Il linguaggio XML è stato brevemente introdotto in quanto tutti i messaggi scambiati dal protocollo NETCONF, devono essere necessariamente codificati in XML, a differenza del protocollo REST-CONF, ad esempio, che supporta la codifica sia in XML che in JSON. Si è visto come l'utilizzo della libreria Python ncclient, ne semplifichi notevolmente l'utilizzo, fornendo un livello di astrazione molto alto, tipico di un linguaggio di programmazione, ricorrendo alla codifica in XML solo dove strettamente necessario. Proprio grazie a questa libreria, e a tutti gli strumenti tipici di un linguaggio di programmazione come contenitori, istruzioni condizionali, cicli, ecc, che Python offre, è stato sviluppato un controllore di rete, che permette di monitorare e riconfigurare le politiche di traffico al suo interno, in maniera del tutto automatizzata. La rete su cui è stato svolto il lavoro di tesi, come già detto, è una NG-PON2, una rete ottica passiva, che offre altissime prestazioni. In virtù di ciò, per poter comprenderne il funzionamento, e le dinamiche che si verificano al suo interno, nello scritto è stato dedicato un capitolo apposito alla descrizione dei concetti principali delle tecnologie ottiche, partendo dalle reti in fibra ottica, arrivando alla tecnica di Dynamic Bandwidth Allocation.

Bibliografia

- [1] G.984.3 : Gigabit-capable passive optical networks (g-pon): Transmission convergence layer specification. Available on line.
- [2] G.989.2 : 40-gigabit-capable passive optical networks 2 (ng-pon2): Physical media dependent (pmd) layer specification. Available on line.
- [3] Netconf message formats. Available on line.
- [4] Calix e-series (e7 os r2.6) engineering and planning guide, 2017.
- [5] Axos r21.x everypon services guide, 2021. Available on line.
- [6] Axos r21.x monitoring, maintenance, and troubleshooting guide, 2021. Available on line.
- [7] M. S. Ahsan, M. S. Lee, S. S. Newaz, and S. M. Asif. Migration to the next generation optical access networks using hybrid wdm/tdm-pon. *Journal of Networks*, 6(1):18, 2011.
- [8] S. Barth. Advanced netconf explorer: Graphical explorer for netconf / yang and gnm/i/grpc telemetry and java netconf 1.1 client library, 2020. Available on line.
- [9] R. Bassoli. Network function virtualization. In *Computing in Communication Networks*, pages 119–132. Elsevier, 2020.
- [10] M. Bjorklund. Rfc 7950: The yang 1.1 data modeling language [ol], 2016.
- [11] T. Bray, J. Paoli, C. Sperberg-McQueen, Y. Mailer, and F. Yergeau. Extensible markup language (xml) 1.0 5th edition, w3c recommendation, november 2008.
- [12] C. Centofanti et al. Improved dash video streaming performance by mec-enabled optical access. 2021.
- [13] J. CRAWSHAW and S. ANALYST. What's holding back adoption & how to accelerate it. 2017.
- [14] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu. iperf3, tool for active measurements of the maximum achievable bandwidth on ip networks. URL: <https://github.com/esnet/iperf>, 2014.
- [15] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. Network configuration protocol (netconf). 2011.

- [16] F. Fitzek, F. Granelli, and P. Seeling. *Computing in Communication Networks: From Theory to Practice*. Academic Press, 2020.
- [17] F. Granelli. Network slicing. In *Computing in Communication Networks*, pages 63–76. Elsevier, 2020.
- [18] t. L. C. Joe Kidder, Principal. Selecting netconf/yang management interface software, 2019. Available on line.
- [19] ONF. Software-defined networking: The new norm for networks, 2012. Available on line.
- [20] M. Patuano. Un nuovo modello di business per le telecomunicazioni. *Il Sole*, 24, 2015.
- [21] U. L. Richard Sharpe, Ed Warnicke. Wireshark user's guide, 2021. Available on line.
- [22] L. P. Shikhar Bhushan. ncclient project description, 2014. Available on line.
- [23] T. Soenen, R. Banerjee, W. Tavernier, D. Colle, and M. Pickavet. Demystifying network slicing: From theory to practice. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1115–1120. IEEE, 2017.
- [24] tail f. Yang overview. Available on line.

Appendice A

Appendix

A.1 Codici Python

```
#IMPORT
from ncclient import manager
import xml.etree.ElementTree as ET
#FUNZIONI UTILIZZATE

"""INSTAURA LA SESSIONE NETCONF UTILIZZANDO MANAGER DI NCCLIENT, PRENDENDO IN INGRESSO I PARAMETRI HOST,
PORT, USERNAME,
E PASSWORD, RESTITUENDO LA SESSIONE NETCONF INSTAURATA."""
def start_session(HOST, PORT, USERNAME, PASSWORD):
    m = manager.connect_ssh(
        host=HOST,
        port=PORT,
        username=USERNAME,
        password=PASSWORD)
    return m

"""RESTITUISCE IL VALORE DEI CONTATORI (BYTES FLOW-COUNTERS), E PRENDE IN INGRESSO LA SESSIONE NETCONF.
UTILE PER INIZIALIZZARE I VALORI DEI CONTATORI ALLA PRIMA CHIAMATA DI get_Bps_and_counters"""
def get_counters(m):
    bytesCounter = [0, 0]
    QUERY_Bytes = '''
        <filter>
        <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <ont-ethernet xmlns="http://www.calix.com/ns/exa/gpon-interface-std">
                    <flow-counters>
                        <per-flow-counter-index>
                            <bytes/>
                            <if-name></if-name>
                            <direction>ingress</direction>
                        </per-flow-counter-index>
                    </flow-counters>
                </ont-ethernet>
            </interface>
        </interfaces-state>
    </filter>
    '''

    netconf_reply = m.get(filter=QUERY_Bytes)

    root1 = ET.fromstring(netconf_reply.xml)

    i = 0
    for bytes in root1.findall('.//{*}bytes'):
        bytesCounter[i] = int(bytes.text)
        i += 1
    return bytesCounter

"""RESTITUISCE LA DIFFERENZA TRA I VALORI DEI CONTATORI (CALCOLATTI CON UNA GET, CAMPIONATI ALL'ISTANTE
DELLA CHIAMATA)
E I PARAMETRI (I VALORI DEI CONTATORI CALCOLATI ALL'ISTANTE PRECEDENTE) PASSATI ALLA FUNZIONE, ED I
VALORI DEI CONTATORI
AGGIORNATI AL MOMENTO DELLA CHIAMATA. RICHIENDE COME PARAMETRI APPUNTO I VALORI INIZIALI DEI CONTATORI,
ALL'INTERNO DI UNA LISTA,
E LA SESSIONE NETCONF"""

```

```

def get_Gbps_and_Counters(startValues , m):
    gigaBitsNumber = [0 , 0]
    endValues = [0 , 0]
    QUERY_Bytes = """
        <filter>
        <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <ont-ethernet xmlns="http://www.calix.com/ns/exa/gpon-interface-std">
                    <flow-counters>
                        <per-flow-counter-index>
                            <bytes/>
                            <if-name></if-name>
                            <direction>ingress</direction>
                        </per-flow-counter-index>
                    </flow-counters>
                </ont-ethernet>
            </interface>
        </interfaces-state>
        </filter>
        ,
    """

    netconf_reply = m.get(filter=QUERY_Bytes)

    root1 = ET.fromstring(netconf_reply.xml)
    i = 0
    for bytes in root1.findall('.//{*} bytes'):
        endValues[i] = int(bytes.text)
        i += 1

    for j in range(len(gigaBitsNumber)):
        gigaBitsNumber[j] = round(((endValues[j]-startValues[j])*8)/1000000000,2)

    return gigaBitsNumber , endValues

"""
Prende in ingresso l'oggetto RPC reply e una stringa che descrive l'operazione effettuata. Utile per
controllare che
un'operazione di edit-config sia stata eseguita correttamente"""
def _check_response(rpc_obj, snippet_name):
    check = False
    print("RPCReply_for_%s_is_%s" % (snippet_name, rpc_obj.xml))
    xml_str = rpc_obj.xml
    if "<ok/>" in xml_str:
        print("%s_successful" % snippet_name)
        check = True
    else:
        print("Cannot_successfully_execute:_%s" % snippet_name)
    return check

"""
Prende in ingresso la sessione NETCONF, il nome della Policy Map e della Class Map da riconfigurare, ed
il valore del
cir da settare"""
def set_cir(m, policyMapName, classMapEthernetName, cir):
    cirSTR= str(cir)
    CONFIGURE_cir1 = """
        <config>
            <config xmlns="http://www.calix.com/ns/exa/base">
                <profile>
                    <policy-map>
                        <name>"""
    CONFIGURE_cir2= """</name>
                    <class-map-ethernet>
                        <name>"""
    CONFIGURE_cir3= """</name>
                    <ingress>
                        <meter-type>meter-mef</meter-type>
                        <cir>"""
    CONFIGURE_cir4 = """</cir>
                    </ingress>
                    <class-map-ethernet>
                        </policy-map>
                    </profile>
                </config>
            </config>
        """

```

```

CONFIGURE_cir = CONFIGURE_cir1 + policyMapName + CONFIGURE_cir2 + classMapEthernetName +
    CONFIGURE_cir3 + cirSTR + CONFIGURE_cir4
rpc_response = m.edit_config(target="running", config=CONFIGURE_cir)
cirInGbits = round(cir/1000000, 2)
if _check_response(rpc_response, "CHANGING_cir") == True:
    print("cir_settato_a_%s_Gbits/sec" %cirInGbits)

"""Prende in ingresso la sessione NETCONF, il nome della Policy Map e della Class Map da riconfigurare, ed
il valore
dell'eir da settare"""
def set_eir(m, policyMapName, classMapEthernetName, eir):
    eirSTR= str(eir)
    CONFIGURE_eir1 = """
        <config>
            <config xmlns="http://www.calix.com/ns/exa/base">
                <profile>
                    <policy-map>
                        <name>"""

    CONFIGURE_eir2="""</name>
        <class-map-ethernet>
            <name>"""

    CONFIGURE_eir3="""</name>
        <ingress>
            <meter-type>meter-mef</meter-type>
            <eir>"""

    CONFIGURE_eir4 = """</eir>
        </ingress>
        </class-map-ethernet>
        </policy-map>
        </profile>
    </config>
    </config>
    """

    CONFIGURE_eir = CONFIGURE_eir1 + policyMapName + CONFIGURE_eir2 + classMapEthernetName +
        CONFIGURE_eir3 + eirSTR + CONFIGURE_eir4
    rpc_response = m.edit_config(target="running", config=CONFIGURE_eir)
    eirInGbits = round(eir / 1000000, 2)
    if _check_response(rpc_response, "CHANGING_eir") == True:
        print("eir_settato_a_%s_Gbits/sec" % eirInGbits)

```

Listing A.1: Contenuto del file functions.py

```

#IMPORT
from functions import *
import time

#Settings
HOST = "10.10.10.30"
PORT = 22
USERNAME = "sysadmin"
PASSWORD = "sysadmin"

starttime = time.time()
m = start_session(HOST, PORT, USERNAME, PASSWORD)
tempoInstaurazione = time.time() - starttime
print("tempo_instaurazione SESSIONE : %s secondi" %tempoInstaurazione)
starttime = time.time()
get_counters(m)
gettimal = time.time() - starttime
print("tempo_per_la_esecuzione_get : %s secondi" %gettimal)
starttime = time.time()
m.close_session()
closetime = time.time() - starttime
print("tempo_per_la_chiusura_della SESSIONE : %s secondi" %closetime)
print("Tempo totale : %s secondi" %(tempoInstaurazione + gettimal + closetime))

```

Listing A.2: Contenuto del file statistiche.py

```

# IMPORT
from matplotlib import pyplot
import csv
import numpy as np

t = []

```

```

yPrecision = []
yG1 = []
i = 0

with open('report2.csv', 'r') as csv_file: #usare report.csv o report2.csv a seconda dei risultato da
    plottare

    rows = csv.reader(csv_file)
    for row in rows:
        t.append(i)
        yPrecision.append(float(row[0]))
        yG1.append(float(row[1]))
        i += 1

with open('report2.csv', 'w') as csv_file: #usare report.csv o report2.csv a seconda del risultato da
    plottare
    write = csv.writer((csv_file))
    for c in range(len(t)):
        write.writerow([t[c], yPrecision[c], yG1[c]])

print(t)
print(yG1)
print(yPrecision)
# Plotting the data
pyplot.plot(t, yPrecision, label="precision")
pyplot.plot(t,yG1, label="G1")
pyplot.xlabel("tempo_in_secondi")
pyplot.ylabel("bit_rate_in_Gbps")
pyplot.yticks(np.arange(0, 10, 0.5))
pyplot.axhline(y = 5, color = 'r', linestyle = '-')
pyplot.legend()
pyplot.show()
csv_file.close()

```

Listing A.3: Contenuto del file plot.py utilizzato per eseguire i plot

```

#IMPORT
import time
from functions import *
import csv

#Settings
HOST = "10.10.10.30"
PORT = 22
USERNAME = "sysadmin"
PASSWORD = "sysadmin"
policyMapName1 = "Eth-match-any-1"
classMapEthernetName1 = "ELINE_PM_1"
policyMapName2 = "Eth-match-any-2"
classMapEthernetName2 = "ELINE_PM_2"
EIR = 10000000
counters = [0, 0]
gigaBitsPerSecond = [0, 0]
i = 0
file = open("report.csv", "w", newline='')
writer = csv.writer(file)

m = start_session(HOST, PORT, USERNAME, PASSWORD) #instauro la sessione NETCONF
set_eir(m, classMapEthernetName1, policyMapName1, EIR) #setto l'eir al massimo per Eth-match-any-1
set_eir(m, classMapEthernetName2, policyMapName2, EIR) #setto l'eir al massimo per Eth-match-any-2

start_time = time.time()
localtime = time.localtime()
result = time.strftime("%I:%M:%S.%p", localtime)
print("Orario_inizio_simulazione: %s" % result)
counters = get_counters(m) #prendo i valori iniziali
total_time = time.time() - start_time
time_sleep = 1 - total_time
time.sleep(time_sleep)

while True:
    start_time = time.time()
    localtime = time.localtime()
    result = time.strftime("%I:%M:%S.%p", localtime)
    gigaBitsPerSecond, counters = get_Gbps_and_Counters(counters, m)
    writer.writerow([gigaBitsPerSecond[0], gigaBitsPerSecond[1]])
    print("Orario: %s" % result)
    print("precision: " + str(gigaBitsPerSecond[0]) + " Gbits/sec" + "----G1: " + str(gigaBitsPerSecond[1])
          + " Gbits/sec")
    total_time = time.time() - start_time

```

```

print("Tempo impiegato: %s secondi" % total_time)
time.sleep = 1 - total_time
time.sleep(time.sleep)

```

Listing A.4: Contenuto del file monitoraggio.py utilizzato nel primo caso mostrato

```

#IMPORT
import time
from functions import *
import csv

#Settings
HOST = "10.10.10.30"
PORT = 22
USERNAME = "sysadmin"
PASSWORD = "sysadmin"
policyMapName1 = "Eth-match-any-1"
classMapEthernetName1 = "ELINE_PM_1"
policyMapName2 = "Eth-match-any-2"
classMapEthernetName2 = "ELINE_PM_2"
EIR = 10000000
counters = [0, 0]
gigaBitsPerSecond = [0, 0]
i = 0
timeout = 5
soglia1 = 3
soglia2 = 1.2 #la soglia e 1.4, che non puo superare, nel caso mostrato in cui PC1 richiede sempre bw massima
picco = False
file = open("report2.csv", "w", newline='')
writer = csv.writer(file)

m = start_session(HOST, PORT, USERNAME, PASSWORD) #instauro la sessione NETCONF
set_eir(m, classMapEthernetName1, policyMapName1, EIR) #setto l'eir al massimo per Eth-match-any-1
set_eir(m, classMapEthernetName2, policyMapName2, EIR) #setto l'eir al massimo per Eth-match-any-2

start_time = time.time()
localtime = time.localtime()
result = time.strftime("%I:%M:%S.%p", localtime)
print("Orario inizio simulazione: %s" % result)
counters = get_counters(m) #prendo i valori iniziali
total_time = time.time() - start_time
time.sleep = 1 - total_time
time.sleep(time.sleep)

while True:
    start_time = time.time()
    localtime = time.localtime()
    result = time.strftime("%I:%M:%S.%p", localtime)
    gigaBitsPerSecond, counters = get_Gbps_and_Counters(counters, m)
    writer.writerow([gigaBitsPerSecond[0], gigaBitsPerSecond[1]])
    if gigaBitsPerSecond[1] > soglia1:
        picco = True
        print("Orario: %s, picco di traffico di G1" % result)
        print("precision: %s Gbits/sec -- G1: %s Gbits/sec" % (str(gigaBitsPerSecond[0]), str(gigaBitsPerSecond[1])))
    set_cir(m, classMapEthernetName1, policyMapName1, 5000000)
    while picco == True and i <= timeout + 1: #qui decido il tempo da aspettare prima di ripristinare le politiche iniziali
        total_time = time.time() - start_time
        print("Tempo impiegato: %s secondi" % total_time)
        time.sleep = 1 - total_time
        time.sleep(time.sleep)
        start_time = time.time()
        localtime = time.localtime()
        result = time.strftime("%I:%M:%S.%p", localtime)
        gigaBitsPerSecond, counters = get_Gbps_and_Counters(counters, m)
        writer.writerow([gigaBitsPerSecond[0], gigaBitsPerSecond[1]])
        if gigaBitsPerSecond[1] <= soglia2: #COMMENTO
            if i == timeout:
                picco = False
                print("Orario: %s -- Il traffico di G1 è rientrato nei parametri" % result)
                set_cir(m, classMapEthernetName1, policyMapName1, 0)
                i = -1
                total_time = time.time() - start_time
                print("Tempo impiegato: %s secondi" % total_time)
                time.sleep = 1 - total_time
                time.sleep(time.sleep)

        else:
            tempoStabile = timeout - i

```

```

        print("Orario: %s Il traffico di G1 sta rientrando nei parametri. Secondi al-
              ripristino: %s"
              % (result, tempoStabile))
    i += 1
else:
    print("Orario: Nuovo picco di traffico di G1" % result)
    i = 0
print("precision: %s Gbits/sec --- G1: %s Gbits/sec" % (str(gigaBitsPerSecond[0]), str(
    gigaBitsPerSecond[1])))
else:
    print("Orario: Traffico di G1 nella norma" % result)
    print("precision: %s Gbits/sec --- G1: %s Gbits/sec" % (str(gigaBitsPerSecond[0]), str(
        gigaBitsPerSecond[1])))
total_time = time.time() - start_time
print("Tempo impiegato: %s secondi" % total_time)
time_sleep = 1 - total_time
time.sleep(time_sleep)

```

Listing A.5: Contenuto del file MonitoraggioConControllo.py utilizzato nel secondo caso mostrato

A.2 I tool tshark e pyshark

A.2.1 tshark

Un tool molto utile per effettuare catture di pacchetti è tshark. Rispetto a Wireshark offre sostanzialmente le stesse funzioni, come la possibilità di selezionare l'interfaccia su cui catturare, filtrare in base all'indirizzo ip, al numero di porta, al protocollo, ecc, ma il tutto da linea di comando. Nel lavoro di tesi, è stato utilizzato per effettuare delle catture su G1, come ulteriore strumento di controllo per verificare il corretto funzionamento del controllore di rete implementato in Python, nel calcolare il numero dei byte trasmessi da G1 e precision. Non potendo catturare con Wireshark, come mostrato in figura A.1 in quanto G1 configurato e controllato solo tramite ssh durante tutto il lavoro di tesi, è stato utilizzato questo valido tool. Per catturare sull'interfaccia "enp1s0f1", scrivere l'output della cattura in "flow.pcap" e implementare un'azione di filtraggio, catturando solo i pacchetti caratterizzati da indirizzo ip di destinazione "10.20.30.1", è stato utilizzato il comando riportato in figura A.2. Nella stessa figura è mostrato il comando iperf3 per generare il traffico

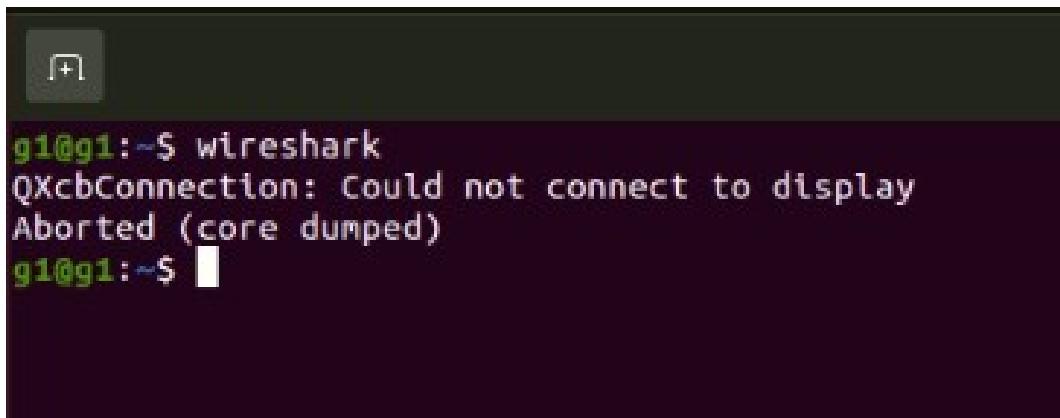


Figura A.1: Risposta al comando Wireshark da pc remoto

catturato che, come indicato dall'output, è contenuto all'interno dei 214 pacchetti collezionati. Attraverso il comando scp di linux, mostrato nella cattura A.3, il file "flow.pcap" è stato copiato sul pc utilizzato per effettuare tutte le operazioni di configurazione riportate nel presente scritto, e tramite il comando ls, è stata verificata la sua presenza nella cartella selezionata in fase di trasferimento. A questo punto "flow.pcap" può essere ispezionato attraverso la libreria Python pyshark o aperto

The figure consists of two vertically stacked terminal windows. The top window shows the command `sudo tshark -i enp1s0f1 -w flow.pcap -f "dst host 10.20.30.1"` being run, followed by a password prompt and the message "Running as user "root" and group "root". This could be dangerous. Capturing on 'enp1s0f1'.

The bottom window shows the command `iperf3 -c 10.20.30.1 -b 10M -p 5201 -t 5` being run, followed by output showing a local connection to port 52784 and a bandwidth test between 10.20.30.2 and 10.20.30.1. The output includes columns for ID, Interval, Transfer, Bandwidth, Retr, Cwnd, and sender/receiver roles.

Figura A.2: Comandi cattura tshark e traffico iperf3

The figure shows a terminal window with the command `scp g1@10.10.10.147:flow.pcap ~/tSharkAnalysis` being run. It prompts for the password of the remote host. The file is transferred at 103.5MB/s. After the transfer, the command `ls tSharkAnalysis/` is run, listing the files `flow.pcap`, `numeroPorta.py`, `pacchettiCodice.py`, and `scriptPythonPYshark`.

Figura A.3: Comandi linux scp e ls

tramite Wireshark per usufruire dell’interfaccia grafica che questo tool offre, come fatto in figura A.4.

A.2.2 pyshark

La libreria Python pyshark permette, tra le diverse opzioni, di analizzare e manipolare i dati contenuti all’interno di un file .pcap. Grazie agli strumenti che offre, è possibile ispezionare tutti i pacchetti collezionati nella cattura scritti su tale file, con la possibilità di filtrare, anche in questo caso, fornendo un indirizzo ip, una porta, o un protocollo. Attraverso i metodi implementati, si può accedere al contenuto del pacchetto, stampandolo su terminale, in una forma simile a quella fornita da Wireshark, mostrata nella cattura A.5. Essendo una libreria Python, i metodi che fornisce possono essere utilizzati all’interno di istruzioni condizionali, cicli, ecc, potendo analizzare i pacchetti catturati in maniera “intelligente”, senza dover ispezionarli uno alla volta come ad esempio accade con Wireshark. Il codice Python riportato, mostra come a partire da un file .pcap, descrittivo di una cattura, si possano calcolare il payload totale misurato in byte, il numero totale di byte inviati,

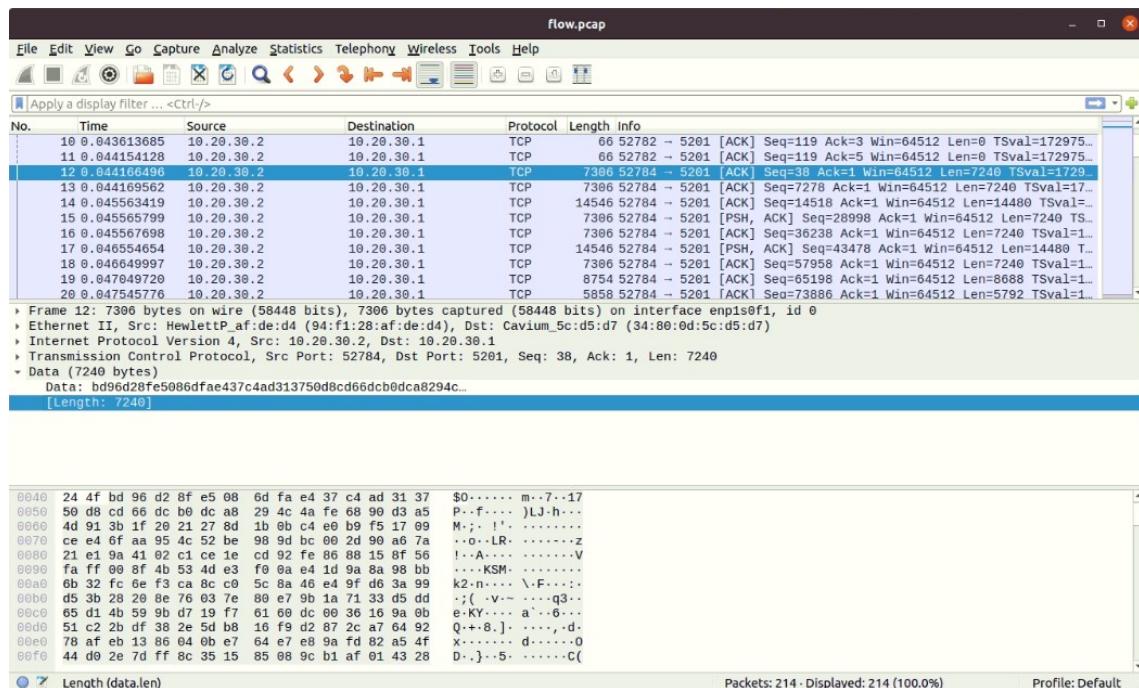


Figura A.4: File "flow.pcap" visualizzato con Wireshark

il numero di byte complessivo utilizzati per intestazioni e per l'invio di messaggi di controllo (per esempio per instaurare la sessione TCP, attraverso il Three Way Handshake), della cattura fornita (nell'esempio descritta in flow.pcap). Parte dell'output restituito eseguendo "pacchettiCodice.py", è riportato nelle catture A.6 e A.7. Nel programma Python, sono stati considerati pacchetti di traffico iperf3 verso 10.20.30.1, tutti quelli con porta sorgente 52784, e lunghezza del payload TCP maggiore di 0 byte. Questo numero di porta utilizzato per classificare i pacchetti, tra pacchetti di traffico o di controllo iperf3, è stato ricavato ispezionandone uno, scelto in modo casuale (nell'esempio il 101-esimo), controllandone la dimensione del payload (23168 byte, non può essere un semplice pacchetto di controllo, trasporta dati). Tutto ciò in virtù del fatto che una sessione iperf3 utilizza una sola porta lato server, specificata con -p (nell'esempio la 5201), e due porte lato client, una per il traffico di controllo iperf3, ed una per i dati veri e propri. Alcune versioni permettono di specificare attraverso il comando -cport il numero di questa seconda porta, comando che nell'esempio riportato non era supportato (altrimenti si sarebbe inserito direttamente quel numero come numero di source port nel codice Python). Un altro modo per individuare tale numero, qualora non sia disponibile il comando -cport, è ricorrere alla visualizzazione grafica della cattura attraverso Wireshark. Nella cattura A.4 sono infatti individuabili due numeri di porte lato client, la 52784, già descritta, e la 52782, per i pacchetti di controllo iperf3. Nella figura A.7 oltre al numero totale dei diversi tipi di byte, è possibile visualizzare il numero di pacchetti totali, pari a 214, come riportato nella figura A.2, e non 213, in quanto nella loro numerazione si è partiti da 0, e non da 1. Di seguito sono riportati i due codici Python eseguiti per mostrare gli output precedentemente descritti.

```
#Capisco la source port da inserire come filtro
#in codicePacchetti.py
```

```
import pyshark
cap = pyshark.FileCapture('flow.pcap')
pkt = cap[100]
```

```

stefano@stefano:~/tSharkAnalysis$ python3 numeroPorta.py
Packet (Length: 23234)
Layer ETH:
  Destination: 34:80:0d:5c:d5:d7
  Address: 34:80:0d:5c:d5:d7
    .... ..0. .... .... .... = LG bit: Globally unique address (factory default)
    .... ..0. .... .... .... = IG bit: Individual address (unicast)
  Source: 94:f1:28:af:de:d4
    .... ..0. .... .... .... = LG bit: Globally unique address (factory default)
    .... ..0. .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
  Address: 94:f1:28:af:de:d4
Layer IP:
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  0000 00.. = Differentiated Services Codepoint: Default (0)
  .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 23220
  Identification: 0x0c7e (3198)
  Flags: 0x4000, Don't fragment
  0.... .... .... = Reserved bit: Not set
  .1... .... .... = Don't fragment: Set
  ..0. .... .... .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x839b [validation disabled]
  Header checksum status: Unverified
  Source: 10.20.30.2
  Destination: 10.20.30.1
Layer TCP:
  Source Port: 52784
  Destination Port: 5201
  Stream index: 1
  TCP Segment Len: 23168
  Sequence number: 1777070 (relative sequence number)
  Sequence number (raw): 910683534
  Next sequence number: 1800238 (relative sequence number)
  Acknowledgment number: 1 (relative ack number)
  Acknowledgment number (raw): 1291819854
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x010 (ACK)

```

Figura A.5: Accesso tramite pyshark alle informazioni di un pacchetto catturato

```

stefano@stefano:~/tSharkAnalysis$ python3 pacchettiCodice.py
Il pacchetto numero 0 è di controllo iperf3. 74 0
Il pacchetto numero 1 è di controllo iperf3. 66 0
Il pacchetto numero 2 è di controllo iperf3. 103 37
Il pacchetto numero 3 è di controllo iperf3. 66 0
Il pacchetto numero 4 è di controllo iperf3. 70 4
Il pacchetto numero 5 è di controllo iperf3. 143 77
Il pacchetto numero 6 è di controllo iperf3. 74 0
Il pacchetto numero 7 è di controllo iperf3. 66 0
Il pacchetto numero 8 è di traffico iperf3. 103 37
Il pacchetto numero 9 è di controllo iperf3. 66 0
Il pacchetto numero 10 è di controllo iperf3. 66 0
Il pacchetto numero 11 è di traffico iperf3. 7306 7240
Il pacchetto numero 12 è di traffico iperf3. 7306 7240
Il pacchetto numero 13 è di traffico iperf3. 14546 14480
Il pacchetto numero 14 è di traffico iperf3. 7306 7240
Il pacchetto numero 15 è di traffico iperf3. 7306 7240

```

Figura A.6: Comando per eseguirlo e output iniziale di pacchettiCodice.py

```

Il pacchetto numero 205 è di controllo iperf3. 70 4
Il pacchetto numero 206 è di controllo iperf3. 66 0
Il pacchetto numero 207 è di controllo iperf3. 258 192
Il pacchetto numero 208 è di controllo iperf3. 66 0
Il pacchetto numero 209 è di controllo iperf3. 66 0
Il pacchetto numero 210 è di controllo iperf3. 66 0
Il pacchetto numero 211 è di controllo iperf3. 67 1
Il pacchetto numero 212 è di controllo iperf3. 66 0
Il pacchetto numero 213 è di controllo iperf3. 66 0
bytes total: 6145165 bytes
payload totale: 6130709 bytes
overhead: 14456 bytes
stefano@stefano:~/tSharkAnalysis$ █

```

Figura A.7: Output finale di pacchettiCodice.py

```
print(pkt)
```

Listing A.6: Contenuto del file numeroPorta.py per visualizzare il contenuto di un pacchetto ed individuare la source port

```

import pyshark
cap = pyshark.FileCapture('flow.pcap')
i=0
bytes=0
payload=0
overhead=0
for pkt in cap:
    if int(pkt.tcp.dstport) == 5201 and int(pkt.tcp.srcport) == 52784 and int(pkt.tcp.len) > 0:
        print("Il pacchetto numero %s è di traffico iperf3." %i, pkt.length, pkt.tcp.len)
        payload = payload + int(pkt.tcp.len)
        overhead = overhead + int(pkt.length) - int(pkt.tcp.len)
    else:
        print("Il pacchetto numero %s è di controllo iperf3." %i, pkt.length, pkt.tcp.len)
        overhead = overhead + int(pkt.length)
    bytes = bytes + int(pkt.length)
    i=i+1
print("bytes_total:%s_bytes" %bytes)
print("payload_totale:%s_bytes" %payload)
print("overhead:%s_bytes" %overhead)

```

Listing A.7: Contenuto del file pacchettiCodice.py utilizzato per visualizzare il numero totale dei diversi tipi di byte ed il numero dei pacchetti