ALMA MATER STUDIORUM
UNIVERSITÁ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

SDN - SOFTWARE DEFINED NETWORKING

Tesi di laurea in programmazione di reti

Relatore:
Franco Callegati

Presentata da:
Antonio Pitzus

Sessione I
Anno accademico 2016/2017

Indice

1 Architettura SDN **1**

1.1 Terminologia . . . 1
1.1.1 Astrazione . . . 1
1.1.2 Associazione . . . 1
1.1.3 Client . . . 1
1.1.4 Client context . . . 1
1.1.5 Dominio . . . 1
1.1.6 Gestione-controllo . . . 2
1.1.7 Orchestrazione . . . 2
1.1.8 Polizza . . . 2
1.1.9 Ricorsione . . . 2
1.1.10 Risorsa . . . 2
1.1.11 Server . . . 3
1.1.12 Server context . . . 3
1.1.13 Service . . . 3
1.1.14 Service context . . . 3
1.1.15 Virtualizzazione . . . 3
1.2 Introduzione . . . 3
1.3 Definizione . . . 4
1.4 Sommario esecutivo . . . 4
1.4.1 Relazioni del controller SDN . . . 5
1.5 Concetti . . . 6
1.5.1 Scopo e requisiti . . . 6
1.5.2 Principi . . . 7
1.5.3 Obiettivi dell' architettura . . . 8
1.5.4 Ruoli . . . 9
1.5.5 Modelli di servizi e risorse . . . 10
1.5.6 Primitive . . . 12
1.6 Controller SDN . . . 13
1.6.1 Controller SDN come nodo di feedback . . . 14
1.6.2 Orchestrazione . . . 14
1.6.3 Virtualizzazione . . . 15
1.6.4 Condivisione di risorse . . . 16
1.6.5 Delegazione . . . 16
1.6.6 Client context . . . 17
1.6.7 Service context . . . 18
1.6.8 Server context . . . 19
1.7 Applicazioni . . . 19
1.8 Architettura integrata . . . 20
1.8.1 Interfacce . . . 20
1.9 Prospettive dell' architettura . . . 22
1.10 Considerazioni operative . . . 22
1.10.1 Affidabilità e disponibilità . . . 22
1.10.2 Identificatori . . . 23
1.10.3 Considerazioni di realizzazione . . . 23
1.10.4 Inizializzazione . . . 23
1.10.5 Complessità . . . 24
1.10.6 Persistenza . . . 24

2 Architettura SDN per le reti di trasporto **25**
2.1 Introduzione e scopo 25
2.2 Funzionamento 25
2.2.1 Applicazione dell' architettura al modello d' informazione 25
2.2.2 Utilizzo dell' astrazione e della virtualizzazione nella rete di trasporto 26
2.2.3 Modellazione di un dominio multiplo e di una rete di trasporto multi-livello 28
2.3 Modellazione dei servizi di connettività di trasporto 30
2.3.1 Flusso contro connessione 30
2.3.2 Ciclo di vita della connessione di trasporto 31
2.3.3 Configurazione dell' inoltramento del flusso per diversi tipi di connessione 32
2.3.4 Comportamento del flusso in una connessione protetta . . 32
2.4 Lavoro futuro 32

3 Paradigma NBI **33**
3.1 Introduzione 33
3.2 Sommario esecutivo 33
3.3 Principi di NBI 34
3.4 Proprietà e struttura di NBI 35
3.4.1 Implementazione 36
3.4.2 Esempio operativo 37
3.5 Benefici di Intent NBI 38

4 NFV : Network Function Virtualization **39**
4.1 Terminologia 39
4.1.1 Virtualizzazione 39
4.1.2 Orchestrazione 40
4.1.3 Gestione-controllo 40
4.1.4 Domini 40
4.2 Introduzione 41
4.3 Componenti 41
4.4 Relazioni e differenze tra SDN e NFV 41
4.4.1 Concetti chiave 41
4.4.2 Operazioni 42
4.4.3 Sviluppo combinato 42
4.5 Lavoro ulteriore 43

5 Riferimenti bibliografici **44**

1 Architettura SDN

1.1 Terminologia

Durante l' esposizione della struttura e del funzionamento dell' architettura SDN si presenteranno spesso i seguenti termini. È bene quindi farvi conoscenza per una corretta comprensione del documento. [1]

1.1.1 Astrazione

L' astrazione è la rappresentazione di una o di un gruppo di entità accomunate da un certo insieme di criteri, ignorando tutti gli altri aspetti che non ne fanno parte.

1.1.2 Associazione

L' associazione è l' informazione di cui necessitano due entità come condizione necessaria per stabilire una sessione tra loro.
Una tipica associazione potrebbe includere credenziali di sicurezza e la corrispondente polizza locale che riguarda le caratteristiche della sessione cifrata.
Tale associazione può essere stabilita dalla negoziazione e dall' installazione manuale delle entità associate da parte dell' amministratore.
Un' associazione consente di stabilire una sessione per ogni coppia di dispositivi che soddisfano i criteri di sicurezza, i quali possono essere ristretti per dispositivi specifici.
La differenza tra un' associazione e una sessione è che l' associazione rappresenta il potenziale per la comunicazione, mentre la sessione rappresenta un canale di comunicazione aperto tra due entità che ne fanno parte.

1.1.3 Client

Un client è un' entità che riceve servizi da un server.

1.1.4 Client context

Un client context è il componente concettuale di un server che rappresenta tutte le informazioni riguardanti un dato client ed è responsabile della partecipazione delle operazioni di gestione-controllo tra client e server.

1.1.5 Dominio

Un dominio è un raggruppamento di entità che rispettano un insieme di criteri. I domini d' interesse comune includono:

- i domini amministrativi, il completo gruppo di risorse che vi appartengono o che sono controllate da una data entità;
- i domini geografici;
- i domini di controllo e l' insieme delle risorse controllate direttamente da un dato controller;
- la tecnologia di sviluppo dei domini.

1.1.6 Gestione-controllo

Si tratta del principio per cui le funzioni di gestione-controllo sono largamente, se non totalmente, le stesse.
In SDN, entrambi i termini possono essere usati quando le loro funzioni si adeguano all' uso comune, anche se il termine "controllo" viene preferito in quanto corrisponde meglio all' idea di feedback in tempo reale.

1.1.7 Orchestrazione

Rappresenta l' attuale selezione e uso di risorse da parte di un server per soddisfare i bisogni del client in accordo con i criteri di ottimizzazione.

1.1.8 Polizza

La polizza è la regola o il gruppo di regole che specificano le azioni che possono essere intraprese quando si verificano una o più condizioni.
Il termine polizza è ampiamente usato ma raramente definito. In alcuni contesti, ogni regola può essere una polizza.
In quest' architettura una polizza è concepita per focalizzarsi sulla gestione dei propositi del cliente.

1.1.9 Ricorsione

Rappresenta l' applicazione ripetitiva di uno schema nel quale l' input di ogni iterazione è derivato dall' output dell' iterazione precedente.
Nel contesto della rete di trasporto suddivisa in livelli, il principio di ricorsione è stato la chiave nell' assicurare che gli schemi e le strutture di rete non vengano oscurate da relazioni complesse. Il partizionamento e il livellamento rappresentano le dimensioni topologiche e funzionali della ricorsione.
La gerarchia della rete di trasporto è descritta attraverso il paradigma ricorsivo client-server.
L' architettura identifica due tipi di ricorsioni ritenute significative:

- la ricorsione gerarchica è uno schema in cui i controller di alto livello orchestrano un ampio insieme di risorse e servizi attraverso uno o piu controller di basso livello;
- la ricorsione dei vicini è uno schema nel quale i controllers "pari" consegnano risorse attraverso i domini di controllo.

1.1.10 Risorsa

Qualsiasi elemento utilizzato per la consegna di un servizio è incluso nella definizione di risorsa.
Per esempio possiamo considerare "risorse" tutti i seguenti elementi:

- punti di terminazione
- macchine virtuali
- Virtual Network Functions (VNFs)
- commutatori

- firewalls
- sottoreti
- cataloghi di servizi offerti

Una risorsa è modellata come un' istanza di una classe di oggetti. Le risorse possono essere suddivise e combinate con altre risorse in ogni modo supportato dal modello d' informazione.
Quando le risorse sottostanti sono virtualizzate da un controller, il risultato è una nuova risorsa che può essere offerta al client.

1.1.11 Server

Un server è un' entità che fornisce servizi ad un client.

1.1.12 Server context

Un server context è il componente concettuale di un client che rappresenta tutta l' informazione di un dato server ed è responsabile della partecipazione delle operazioni di controllo tra client e server.

1.1.13 Service

Un servizio è la consegna di un valore per un certo intervallo di tempo da parte di un server verso un client. Un servizio continua ad esistere durante l' intervallo di vita della sessione tra client e server.

1.1.14 Service context

Il service context è il componente concettuale di un dato client context che rappresenta tutta l' informazione riguardo un dato servizio.

1.1.15 Virtualizzazione

La virtualizzazione rappresenta l' astrazione delle risorse sottostanti, il quale criterio di selezione è rappresentato dall' allocazione delle risorse verso un particolare client, applicazione o servizio.

1.2 Introduzione

In un periodo in cui tutto si evolve rapidamente, il settore delle telecomunicazioni sta assistendo alla crescita esponenziale del numero di dispositivi mobili costantemente connessi alla rete, all' utilizzo sistematico da parte di aziende e privati del cloud-computing e al cambiamento del modo in cui gli host comunicano tra loro, passando dal modello client-server al modello peer-to-peer; ciò richiede sempre con maggior insistenza la necessità di un nuovo modo di gestire le reti.
La nuova visione che sta maturando in questi ultimi tempi è quella di adottare un modello di rete dinamico, flessibile e soprattutto affidabile, in grado di adattarsi ai cambiamenti del futuro senza richiedere grossi sforzi di manutenzione o l'installazione di ulteriori hardware da parte degli operatori.

Una rete con queste caratteristiche può essere sviluppata grazie ad un modello architetturale innovativo come il Software Defined Networking (SDN) e ad un nuovo modo di sfruttare le funzionalità degli apparati di rete come la Network Function Virtualization (NFV), la quale è a sua volta un processo di virtualizzazione delle funzionalità di rete svolte da apparati di telecomunicazione fisici.
Questi due concetti sono strettamente legati tra loro e possono comportare particolari vantaggi se applicati contemporaneamente, ma sono di per sè indipendenti.
Passiamo ora ad analizzare il primo modello, ovvero SDN. [2]

1.3 Definizione

Software Defined Networking (SDN) è un' architettura utilizzata per la realizzazione di reti di telecomunicazione nelle quali il piano di controllo della rete è logicamente separato da quello del trasporto dei dati.
Questa separazione logica permette di:

- gestire via software tutta la rete da un unico controller;
- utilizzare indifferentemente apparati prodotti dalle diverse aziende che non conterranno più al loro interno le funzioni di gestione.

Questa separazione garantisce da una parte una maggiore scalabilità e migliori standard di affidabilità e sicurezza, e dall' altra favorisce la nascita di una rete indipendente dai protocolli utilizzati. [1]

1.4 Sommario esecutivo

Un' architettura è una collezione di prospettive riguardanti un gruppo di idee e serve a facilitare lo sviluppo di concetti in realtà.
L' architettura descritta in questo documento mira ad essere consistente, aperta e utile.

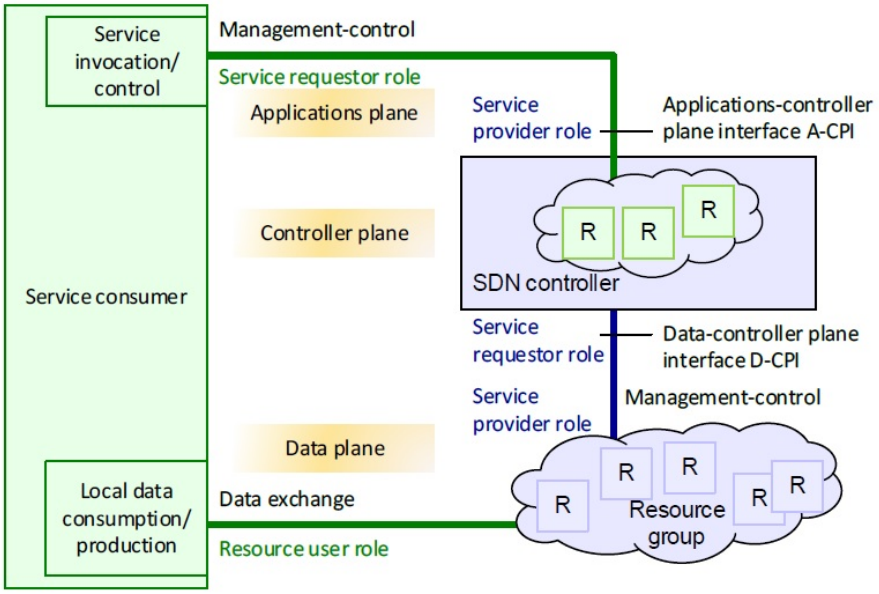

Figura 1: Modello base di SDN

La figura 1 illustra il modello base di SDN, dove un consumatore di servizi (client, user, customer) raffigurato in verde, scambia sia dati che operazioni di controllo con qualche server SDN o fornitore raffigurato in blu.
Il consumatore controlla i suoi servizi attraverso una sessione, nella quale le risorse (R) vengono processate e inoltrate.
È compito di SDN virtualizzare e orchestrare la vista dei servizi e delle risorse sottostanti.
La scelta di rappresentare client e server con due colori differenti serve ad enfatizzare la necessità di isolare il traffico, mascherare l' infomazione e garantire sicurezza alle interfacce.
L' architettura estende il modello base, esposto in precedenza, aggiungendo ulteriori funzionalità, come per esempio la condivisione di risorse tra più clients in modo dinamico e ottimale. [1]

1.4.1 Relazioni del controller SDN

L' entità centrale dell' architettura è il controller. L' architettura è modellata come un gruppo di relazioni di tipo client-server tra i controllers e le altre entità, le quali a loro volta possono essere altri controllers. Quando opera come server, il controller può offrire servizi ad un certo numero di clients, mentre quando opera come client invoca servizi da un certo numero di servers.

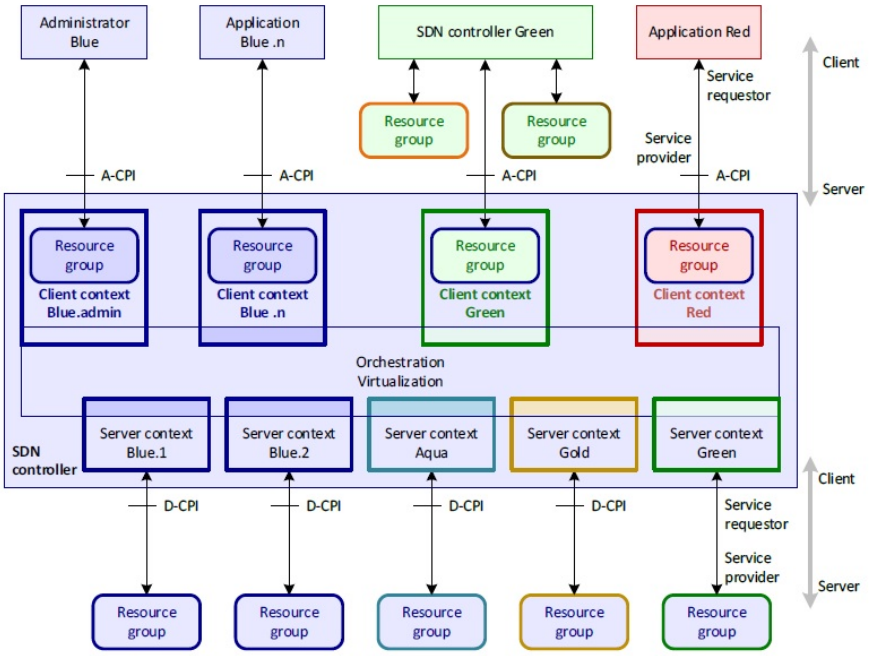

Figura 2: Cuore dell' architettura

L' architettura riconosce la doppia prospettiva dell' interfaccia client-server, come mostrato dalla figura 2.
La prospettiva dei servizi è raffigurata in maniera top-down, mentre quella delle risorse in maniera bottom-up. Queste due prospettive sono complementari ed enfatizzano i due diversi aspetti.
Il controller soddisfa le richieste del client attraverso la virtualizzazione e l' orchestrazione delle risorse sottostanti. Il controller è inoltre responsabile del continuo aggiornamento della rete, in base al cambiamento delle richieste del client.
Le risorse e i servizi vengono esposti ai vari clients attraverso le A-CPIs (Application-Controller Plane Interfaces) e vengono consumati attraverso le D-CPIs (Data-Controller Plane Interfaces). Ognuna di esse è un punto di riferimento per il mascheramento del servizio e per l' isolamento del traffico. [1]

1.5 Concetti

1.5.1 Scopo e requisiti

L' architettura SDN comprende i seguenti requisiti:

- supporto all' interoperabilità basata su A-CPIs e D-CPIs;
- indipendenza dalla distribuzione del controller;
- scalabilità e supporto alla ricorsione per includere tutte le architetture;
- applicabilità verso una semplificata e unificata configurazione delle risorse del piano dati;

- limiti alla polizza e alla sicurezza riguardanti l' attendibilità e la condivisione dell' informazione;
- supporto alla gestione delle interfacce, attraverso cui viene stabilita la polizza sulle risorse;
- coesistenza con altri sistemi di supporto e altri domini di tecnologie.

Per evitare una specificazione troppo ampia, l' architettura descrive soltanto le funzioni che sono richieste, ma ciò non ne preclude di nuove. [3]

1.5.2 Principi

SDN si basa su tre principi cardine e un importante aspetto.

1. **Disaccoppiamento dell' inoltramento e del processamento del traffico dal controllo:** la proposta di questo principio è quella di permettere lo sviluppo del controllo, dell' inoltramento e del processamento del traffico come entità.
   Il disaccoppiamento è una precondizione necessaria del controllo centralizzato e della ricorsione. Questo principio è dato attraverso un' entità chiamata controller, la quale è responsabile delle operazioni di gestione-controllo di un gruppo di risorse. Questi ultimi sono considerati come all' interno del piano dati, così chiamato poichè la maggior parte di esse sono direttamente o indirettamente associate al processamento del traffico del client.

2. **Controllo logicamente centralizzato:** il termine logico significa che il controllo si comporta come una singola entità, indipendentemente dalla sua possibile implementazione in forme distribuite.
   Il disaccoppiamento del processamento del traffico dal controllo è una precondizione per un controllo centralizzato, il quale afferma che le risorse possono essere usate più efficientemente quando viste da una prospettiva più ampia.
   Un controller centralizzato può orchestrare le risorse che ricoprono un gran numero di entità subordinate.
   Il miglior esempio di ciò è l' esposizione di un singolo e monolitico dominio d' inoltramento, costruito sopra una rete sottostante grande e complessa.

3. **Programmazione dei servizi di rete:** questo principio permette al client di scambiare informazioni con il controller, sia per la negoziazione dei servizi richiesti che durante il tempo di vita del servizio, in accordo ai cambiamenti dei bisogni del client o allo stato delle sue risorse virtuali.
   Il principio di programmazione deriva dal beneficio attraverso cui le risorse traggono a partire dalla collaborazione e dalla negoziazione dettagliata dei servizi.
   D' altra parte, l' interfaccia si aspetta che il client esprima i propri bisogni dopo una considerevole negoziazione, ma che lasci al controller la loro realizzazione e ottimizzazione real-time.

4. **Interfacce aperte:** questo aspetto concerne l' implementazione e lo sviluppo dell' architettura.
   L' implementazione presuppone la partizione ben definita delle funzioni e

delle interfacce, specificando che queste devono essere aperte e pubbliche. Questo concetto è una raccomandazione per lo sviluppo di funzioni standard in maniera standard, che supportino la flessibilità e l' estensione di interfacce per funzioni proprietarie.
Il merito dell' interfaccia è quello di districare le operazioni di risorsa dalle intenzioni del client, permettendogli di essere più indipendente all' interno dell' architettura.

Attraverso questi quattro elementi, l' architettura può processare e inoltrare il traffico, sia come parte di un valore aggiunto al servizio, sia per assicurare una corretta manutenzione di rete.
I componenti principali sono le risorse e i controllers. Questi ultimi sono posti tra le risorse e i clients per spedire servizi. [3]

1.5.3 Obiettivi dell' architettura

L' obiettivo globale dell' architettura è quello di assistere al meglio i fornitori di servizio e di ridurre al minimo il costo d' invio dei servizi richiesti.
Per adempiere al meglio a questo obiettivo, sono necessari i seguenti requisiti:

- un ambiente che riduca al minimo il tempo e il costo di sviluppo dei servizi;
- definire in modo flessibile le risorse, includendo le VNFs (Virtual Network Functions);
- assemblamento delle risorse in servizi su richiesta;
- caricamento più efficiente delle risorse, con un' ottimizzazione continua e real-time;
- un modello d' informazione comune che faciliti la semantica globale;
- fusione delle funzioni di supporto di sistema e di business (BSS/OSS) con il controllo.

Poichè l' obiettivo globale è molto ampio, non tutti i requisiti possono essere soddisfatti immediatamente.
Siccome l' architettura si propone di creare un linguaggio di comprensione universale, il requisito che dev' essere soddisfatto per primo è sicuramente quello della realizzazione di un modello d' informazione comune.
Uno dei problemi più radicati nell' industria sono stati i "silos", aree separate con separate competenze, servizi e interessi. Alcuni esempi sono:

- trasporto contro pacchetto
- wireline contro wireless
- data center, cloud contro elementi dedicati e dispersi
- NFV contro SDN

L' architettura si propone inoltre di aprire nuove possibilità e specialmente di trovare un territorio comune dove poter far collassare i silos e mantenere la situazione sensata. [1]

1.5.4 Ruoli

I ruoli primari di SDN sono quelli di amministratore e di fornitore e consumatore di servizi e risorse. Vediamoli più nel dettaglio.

1. **Ruolo d' amministratore:** un amministratore possiede visibilità e privilegi maggiori rispetto ad un client ordinario. La figura 3 mostra come l'

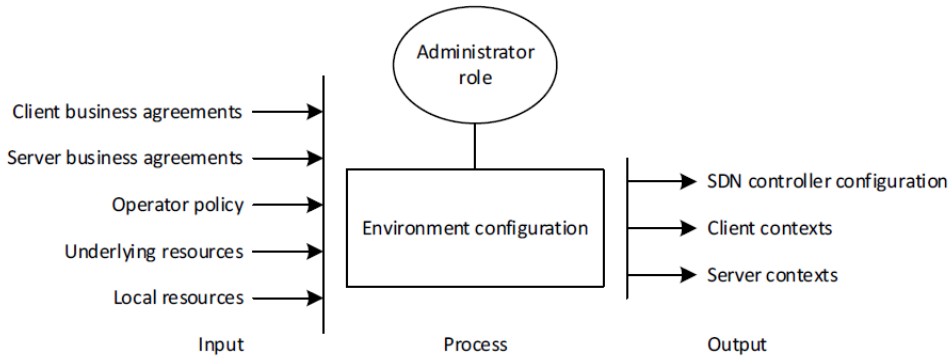

Figura 3: Ruolo d' amministratore

amministratore configura i controllers e tutto il contesto necessario.
Il controller è creato di default da un amministratore insieme ad un client context, il quale ha visibilità e autorità ristretta per eseguire tutte le altre operazioni. L' amministratore configura poi il controller assieme al server context per accedere alle risorse sottostanti e aggiornarle in base alle necessità. Anche le risorse sottostanti sono configurate dagli amministratori. L' amministratore crea poi un client context per ognuno dei suoi clients, inclusa l' allocazione delle risorse sottostanti verso il client attraverso un processo chiamato virtualizzazione, così come la configurazione supplementare. L' amministratore configura ogni client context con la polizza che definisce le azioni e i confini permessi al client. Un amministratore può modificare il client context durante il suo tempo di vita, e può distruggerlo se la relazione col client termina.
Poichè il controller SDN è responsabile per la continua ottimizzazione delle sue risorse in accordo con la polizza di ottimizzazione globale, l' amministratore installa e modifica tale polizza in base alle necessità.

2. **Ruolo di fornitore e di consumatore:** i clients o i consumatori di servizio soddisfano le loro necessità richiedendo i servizi al controller e ottenendoli come utenti delle risorse corrispondenti.

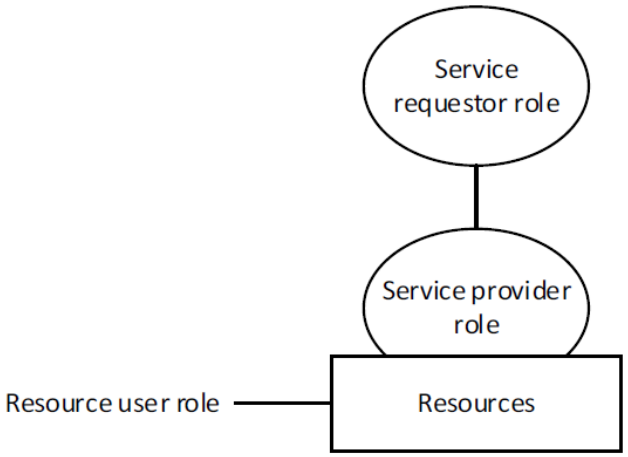

Figura 4: Ruoli di consumatore e fornitore

Come illustrato dalla figura 4, le risorse sono implicitamente o esplicitamente disponibili al client in veste di service provider, il quale offre servizi al service requestor per la configurazione.
Il service requestor rappresenta l' aspetto di gestione-controllo del client per la configurazione del servizio desiderato. Mentre il service requestor rappresenta la configurazione del servizio da parte del client, il resourse user rappresenta l' uso che il client fa delle risorse per soddisfare le proprie richieste. Di solito questo è accompagnato da scambi all' interno del piano dati. [1]

1.5.5 Modelli di servizi e risorse

1. **Modello orientato ai servizi:** le operazioni base delle interfacce sono l' invocazione e la gestione del servizio. In accordo con questo modello, una richiesta del client può riguardare la creazione, la lettura, l' aggiornamento e l' eliminazione (CRUD operations) di un oggetto del client context.
   Il server corrisponde al controller, il quale si aspetta di validare la richiesta in base alla polizza attuale e alle risorse disponibili, per poi soddisfare la richiesta o lanciare un' appropriata eccezione. Le risorse legate al servizio vengono rilasciate quando il service context viene eliminato.
   Una richiesta di servizio viene necessariamente espressa in termini di entità, azioni e nomi/indirizzi noti al client.
   Il server conserva necessariamente il service context, il quale può includere l' invocazione del servizio come espresso dal client.
   La richiesta del client è ciò che guida il server nell' orchestrazione e nella virtualizzazione delle risorse sottostanti. In generale, nè il client, nè il server, sono nella posizione di fornire una mappatura completa dell' informazione: questa deve quindi essere una responsabilità condivisa.
   Ci sono un buon numero di modi per popolare la funzione di mappatura. Alcuni di questi sono:

- accessi ai punti di presenza (PoPs), con successiva fornitura di un' esplicita mappatura equivalente alla mappatura del database, accessibile sia dal client che dal server;
- pubblicazione da parte del server di un catalogo di servizio, o equivalentemente di uno schema che può essere usato per soddisfare il servizio desiderato;
- negoziazione di un accordo come protocollo. Un server può offire servizi a un buon numero di clients. Tutte le informazioni del server per un particolare client sono contenute in un componente concettuale chiamato client context, il quale può contenere a sua volta un numero qualsiasi di service contexts.

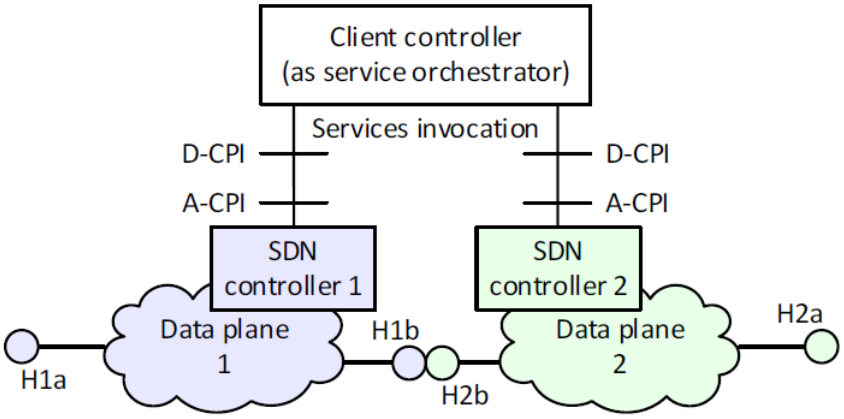

Figura 5: Client come orchestratore del servizio

Come illustrato dalla figura 5, il client, che agisce anch' esso da controller, può orchestrare i servizi, i quali possono a loro volta coprirne altri.
L' orchestrazione di servizi composti richiede che il client sia responsabile della scelta e della fornitura intermedia di punti di interfaccia, così come del comportamento del servizio specifico in ogni nodo.
La successiva partizione è un modo di realizzare il servizio, dove un singolo controller può eseguire i processi della sottorete all' interno del suo stesso dominio.
I controllers "pari" possono invocare la ricorsione dei vicini per conseguire l' obiettivo stabilito. In ogni caso, i punti estremi devono essere coordinati ed essere ricorsivamente visibili.

2. **Modello orientato alle risorse:** le risorse sono le "cose" che possono essere utilizzate da SDN. Una risorsa è rappresentata da un' interfaccia attraverso un' istanza (attuale o potenziale) di qualche classe di oggetti in un modello d' informazione. Un controller espone le risorse al cliente,

il quale è libero di usarle nelle sue queries d' informazione e nella sua richiesta di servizio.
Le risorse del client esplicite sono tipicamente stabilite tramite un accordo e sono fornite attraverso la funzione di virtualizzazione del controller da parte dell' amministratore. Queste risorse includono una funzione di mappatura a partire dal nome/indirizzo del client. Inoltre, non è scontato che le risorse sottostanti dedicate al client possano cambiare durante il servizio. Queste infatti col tempo possono subire variazioni, dovute per esempio a fallimenti, funzioni di ripristino e azioni amministrative di costruzione di rete. [1]

1.5.6 Primitive

1. **Modello d' informazione:** una delle affermazioni di SDN sostiene che l' ambiente delle telecomunicazioni includerà in futuro più fornitori e un numero maggiore di prodotti offerti. Proprio perchè l' invocazione dei servizi passa attraverso le APIs, possono insorgere delle discrepanze semantiche nelle informazioni richieste, causando quindi confusione. La chiave è stabilire un modello d' informazione comune.
   I data models specifici e le viste personalizzate possono quindi essere derivati da un modello d' informazione comune, il quale può essere facilmente adattato fin quando la semantica verrà preservata.

2. **Risorse e gruppi di risorse:** ogni servizio SDN è costruito sopra alcuni gruppi di risorse, le quali funzioni e interfacce sono configurate in base ai bisogni specifici. Le risorse possono essere fisiche o virtuali, attive o passive, e in molti casi possono essere create, scalate o distrutte dal client e/o dal server.
   Le Virtual Network Functions (VNFs) sono anch' esse considerate come risorse.
   Una risorsa è modellata da un' interfaccia come un' istanza di una classe di oggetti in un modello di informazione e possono essere suddivise o combinate in risorse più grandi o più piccole in base alle esigenze.
   Le risorse sono esposte al client attraverso una vista. Quest' ultima è espressa necessariamente attraverso l' utilizzo d' identificatori e concetti noti al client.
   Una vista può restringere l' informazione disponibile al client, così come le azioni che possono essere invocate e le notifiche che possono venir pubblicate.
   I gruppi di risorse non sono fondamentali, ma sono delle astrazioni convenienti. Queste possono presentare alcune delle seguenti proprietà, oppure tutte:

   - destino condiviso: la probabilità che tutte le risorse nel gruppo perdano o riacquisiscano la connettività con il controller, che falliscano o che vengano ripristinate, è sempre la stessa;
   - tutte le risorse all' interno del gruppo sono soggette alle stesse associazioni di gestione-controllo;
   - l' abilità di utilizzare identificatori diversi all' interno del gruppo piuttosto che identificatori globali;

- una pubblicazione comune delle notifiche all' interno dello stesso gruppo;
- un significativo punto topologico all' interno del grafo di rete, utile per l' ottimizzazione del percorso o per altre computazioni.

3. **Database delle risorse:** il database delle risorse (RDB) è il contenitore concettuale dell' informazione che necessita di esser contenuta all' interno del controller.
   Un importante criterio per la conservazione dei frammenti dei dati nel database è il bisogno che sia disponibile dopo la reinizializzazione del controller.
   Un altro importante criterio della conservazione del database afferma che alcune informazioni che sono create da un' entità vengono successivamente usate come base per le altre.

4. **Controller e piani:** l' architettura comprende tre piani.

   (a) **Il piano dati:** composto da elementi di rete, che espone le proprie funzioni al livello di controllo (piano di controllo) attraverso la D-CPI.

   (b) **Il piano di controllo:** che traduce i requisiti delle applicazioni ed esercita un controllo granulare verso gli elementi di rete.
   I servizi vengono offerti alle applicazioni attraverso l' A-CPI, anche detto NBI. Un controller può orchestrare le richieste concorrenti dell' applicazione in caso di risorse di rete limitate.

   (c) **Le applicazioni SDN:** che risiedono nel piano d' applicazione, e comunicano i loro requisiti di rete attraverso l' A-CPI.

   Nonostante molte funzioni tradizionali di gestione vengano aggirate grazie all' A-CPI, altre rimangono essenziali.
   Nel piano dati, le funzioni di gestione sono inizialmente richieste per la configurazione degli elementi di rete e per l' assegnamento delle risorse verso il rispettivo controller.
   Nel piano di controllo, le funzioni di gestione sono richieste per la configurazione del controller e per il monitoraggio delle performance del sistema.
   Nel piano di controllo, le funzioni di gestione sono richieste per la configurazione del Service Level Agreement (SLA), applicato poi al piano di controllo.
   In tutti i piani, le funzioni di gestione configurano le funzioni di sicurezza che permettono alle funzioni distribuite d' intercomunicare in maniera sicura. [1]

1.6 Controller SDN

Il controller è un' entità intelligente che controlla le risorse per consegnare servizi e rappresenta il cuore dell' architettura.
La sua funzione primaria è quella di cambiare in modo real-time e in modo multi-dimensionale l' ambiente delle risorse e dei servizi in base ai criteri di ottimizzazione, i quali sono a loro volta soggetti a cambiamento nel tempo.
Come esposto dall' infrastruttura sottostante, una risorsa è controllata da un solo controller. Un controller esercita funzioni di gestione-controllo sopra un

gruppo di risorse.
Un controller offre servizi ai suoi clients attraverso l' esposizione di un gruppo di risorse. Esso può venire implementato tramite la tecnologia di computazione distribuita o centralizzata, con o senza ridondanza. [1]

1.6.1 Controller SDN come nodo di feedback

Abbiamo già affermato in precedenza che la funzione chiave del controller è quella di adattare continuamente l' ambiente dei servizi e delle risorse in base alla polizza di ottimizzazione stabilita anche con il client. Il controller partecipa infatti come un' entità attiva in un ciclo di feedback, come illustrato dalla figura sottostante. Il controllo è il processo che stabilisce e mantiene lo stato

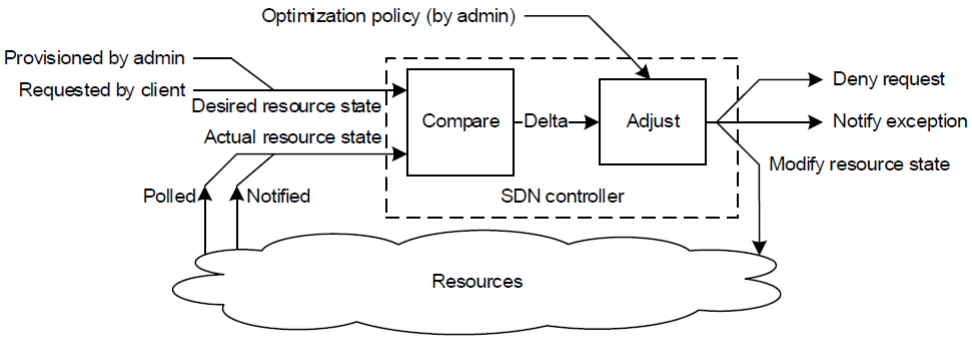

Figura 6: Controllo come feedback

desiderato. La figura 6 illustra che l'amministratore delle risorse configura lo stato desiderato, il quale può cambiare di volta in volta. I vari clients invocano il servizio richiesto, che si traduce con lo stato desiderato. Il controller valuta la differenza tra stato desiderato e attuale in base alla polizza di ottimizzazione attuale e prova a modificare lo stato delle risorse.
La polizza di ottimizzazione include inoltre l' abilità di creare nuove risorse. Se lo stato attuale e quello desiderato non possono essere riconciliati all' interno dei confini della polizza, il controller lancia un' eccezione rifiutando la richiesta del client oppure attraverso una notifica durante l' operazione in corso. [1]

1.6.2 Orchestrazione

L' orchestrazione è la caratteristica che definisce il controller. L' orchestrazione è la selezione delle risorse in grado di soddisfare i servizi richiesti in maniera ottimale, dove le risorse disponibili, i servizi richiesti e il criterio di ottimizzazione sono tutti soggetti a cambiamento.
Quando la funzione di ottimizzazione indica che uno stato migliore può essere raggiunto, la funzione di orchestrazione aggiusta lo stato delle risorse sotto il suo controllo per spostarsi verso lo stato desiderato.

In base all' ambiente e ai criteri di ottimizzazione, la funzione d' orchestrazione può essere semplice o complessa. Gli algoritmi candidati all' orchestrazione valutano inoltre la complessità e il tempo necessario al cambiamento verso lo stato desiderato.
L' orchestrazione include le seguenti funzioni:

- validazione dei servizi richiesti da parte del client tenendo conto della sua polizza specifica, negando quindi le richieste che non rientrano in tale polizza;
- configurazione delle risorse e dei servizi subordinati necessari per soddisfare la richiesta del client tenendo conto della polizza SLA associata a esso:
  - questo include la configurazione delle entità del piano dati con l' intento di rinforzare la polizza stessa;
  - questo include la creazione e la configurazione dei meccanismi del piano dati per una condivisione dinamica;
  - questo include la configurazione e l' incapsulamento, la traduzione degli indirizzi, o altre misure che possono garantire l' isolamento del traffico del client;
  - questo include la richiesta d' istanziamento, di scalamento, di migrazione o di eliminazione delle risorse, ovviamente se permesso dalla polizza del client;
  - questo include la configurazione delle operazioni di supporto alla rete come, la Connettivity Fault Management (CFM), gli Spanning Tree Protocols (STP) e il monitoraggio delle performance come specificato dalla polizza SLA del client;
  - queste operazioni hanno bisogno di essere invocate ricorsivamente da parte del controller.
- coordinazione dei servizi richiesti e del cambiamento dei servizi con i domini vicini;
- pubblicazione di notifiche d' interesse verso un client particolare, all' interno del client context pertinente.

La soddisfazione del servizio richiesto dal client può richiedere la funzione di orchestrazione per la selezione addizionale di risorse dal server. La selezione di tali risorse è ovviamente legata alla polizza. [1]

1.6.3 Virtualizzazione

La virtualizzazione è il processo complementare all' orchestrazione, la quale popola e mantiene le risorse e gli identificatori che collegano l' orchestrazione con i clients.
Nonostante vengano descritte come funzioni separate, l' orchestrazione e la virtualizzazione lavorano in modo coeso e inestricabile.
Un controller possiede un RDB che contiene concettualmente le risorse sottostanti, le quali inizialmente non sono allocate per uno scopo preciso.
Un amministratore può creare una funzione di virtualizzazione personalizzata

per ogni client context, dedicata a uno specifico client.
Non tutte le risorse virtuali sono necessariamente visibili al client. Di quelle che gli vengono esposte è richiesto l' identificatore e può succedere che queste risorse siano visibili anche solo parzialmente al client (magari gli viene mostrata solo una parte dei suoi attributi). Una virtualizzazione può essere creata e popolata in ogni maniera adatta alle circostanze.
Uno dei metodi può essere l' esplicita negoziazione tra fornitore e consumatore, per esempio un accordo sul numero di User Network Interfaces (UNIs) per un particolare client, oppure la specificazione dell' intera topologia della sottorete che il fornitore può affittare. Le risorse prenegoziate permettono al client di specificare dettagliatamente le sue richieste di servizio, che corrispondono a un numero ridotto di scelte disponibili all' orchestratore per il suo soddisfacimento.
Un altro metodo è la virtualizzazione dinamica.
Mentre provvede a soddisfare la richiesta di servizio, la funzione d' orchestrazione può suddividere, combinare o astrarre le risorse sottostanti in base alle necessità. Il servizio risultante verrà poi registrato negli appropriati client e service context.
I cambiamenti all' ambiente delle risorse, agli accordi di commercio o altri fattori, necessitano dell' abilità di aggiornare in modo incrementale la funzione di virtualizzazione coordinata con la funzione di orchestrazione. [1]

1.6.4 Condivisione di risorse

Il principio del controllo logicamente centralizzato implica che ogni risorsa non venga controllata da più di un' identità esterna. Se diversi clients si contendono la risorsa, devono contendersela come clients di un controller, il quale è responsabile dell' arbitraggio delle loro richieste.
Le risorse vengono permanentemente allocate al dato client interamente, senza possibilità di contesa. Le risorse possono anche essere allocate al momento del bisogno, sia su richiesta che tramite tecniche di pianificazione.
Le risorse possono essere anche condivise dinamicamente, pacchetto per pacchetto per esempio. Questo può però richiedere caratteristiche specializzate all' interno dell' infrastruttura, ad esempio che il controller SDN supporti almeno alcuni aspetti dell' accodamento basato sulla priorità equa e ponderata.
La condivisione dinamica e prioritaria delle risorse di rete è complicata dalla necessità d' identificare le risorse sottostanti in un miscuglio di topologie di client arbitrari. Questa complessità incoraggia l' allocazione di un grande flusso che aggiusti la capacità del percorso attraverso la rete. [1]

1.6.5 Delegazione

L' architettura consente al controller di delegare funzioni all' interno del sottostante piano dati per varie ragioni.
Il controller agisce come amministratore per installare la polizza che governa il comportamento di queste funzioni. Attraverso la certificazione, il testing e altre operazioni, l' operatore umano si aspetta di conformare l' intervallo possibile delle polizze, e si aspetta che il controller non riscontri dei conflitti quando delegherà i dettagli della funzione. [1]

1.6.6 Client context

La relazione tra client e server è rappresentata da un' associazione, la quale definise i parametri di inter-operabilità tra le entità del client e del server. Per rappresentare la relazione reciproca, i client sono configurati come server contexts, mentre i server come client contexts.
Un client context modella tutto ciò che esiste in un controller per supportare un dato client. Rispetto ad un controller, un client esiste solo durante il tempo di vita del suo client context. Se la relazione con il client termina, tutte le sue tracce verranno rimosse attraverso l' eliminazione del suo client context.
Quando un nuovo client viene riconosciuto da un dato controller, l' amministratore del server crea un client context che contiene gli attributi minimi dell' associazione, per esempio per stabilire l'identità e la sicurezza che permette di eseguire la sessione di gestione-controllo tra client e server. L' amministratore può anche popolare il client context con risorse virtuali e una polizza adatta al client.
Dopo la creazione del client context, il client e il server possono stabilire una o più sessioni di gestione-controllo (per essere brevi le chiameremo direttamente sessioni). Una sessione è il meccanismo che supporta la comunicazione tra una specifica istanza di un client e uno specifico controller nel contesto dell' associazione. Durante la sessione, il client può invocare i servizi e modificare lo stato delle sue risorse.
Simile all' operazione di login, generalmente una sessione comincerà con uno scambio di credenziali d' identificazione e di sicurezza, seguite da un accordo sullo stato iniziale e continuerà con uno scambio d' informazioni. Ognuna di queste operazioni può essere attribuita ad una sessione. Una sessione può continuare in modo indefinito, o terminare al momento dell' operazione di logout, di un fallimento o di un timeout. In molti casi, il client rigenererà automaticamente la sessione perduta e ci si aspetta che il controller sia capace di ristabilirla.
È richiesta la persistenza dell' associazione del client e dei suoi servizi anche in assenza della sessione tra client e server. Quando il controller e il client stabiliscono la sessione, le risorse e le viste devono essere ristabilite. I file di log possono essere necessari, in parte per permettere al client di vedere gli eventi d' interesse che occorrono durante la disconnessione. Una grande parte del client context deve invece essere persistente. La figura 7 illustra che, logicamente, un client ha visibilità e controllo solo all' interno del suo gruppo di risorse. Semanticamente, l' A-CPI tra il client e il controller sta al confine del gruppo delle risorse. Il client context contiene concettualmente anche le informazioni di supporto e le funzionalità che sono richieste per supportare il client.
Andiamo ad analizzare ora i suoi contenuti funzionali.

- **Il gruppo di risorse:** il quale definisce le interfacce semantiche (A-CPI) esposte al client. Questi due sottogruppi sono identificati come:
  - le risorse virtuali, che rappresentano l' infrastruttura delle risorse create a partire dal controller attraverso il processo di virtualizzazione e che sono esposte al client tramite la funzione di mappatura. Il più classico degli esempi è il Client' s Point of Presence (CPoP);
  - le risorse di supporto, che rappresentano funzioni situate nel controller stesso. Le loro proposte sono quelle di abilitare o facilitare l'

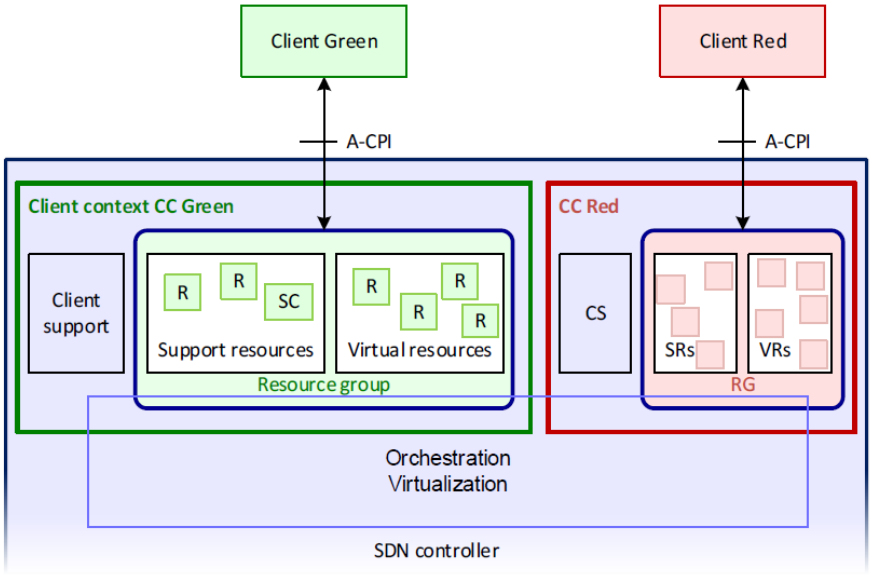

Figura 7: Client context

interazione con il client. Esempi di questo includono profili di login, notifiche d' iscrizione, file di logs.

- **Client support:** questa funzione supporta il client in diversi modi in base alle necessità, ad esempio includendo codici, dati effimeri e dati persistenti. Finchè le risorse del gruppo del client contengono risorse che sono visibili a esso, il blocco di supporto del client può contenere risorse addizionali che non gli sono esposte totalmente. Il blocco di supporto del client può includere le informazioni per mappare gli identificatori e le azioni tra client, server e polizze riguardo ciò che il client può e non può vedere/fare.

La decomposizione in figura 7 viene intesa unicamente per chiarire le funzioni del client context.
Quando un controller viene creato, è richiesto un client context di default, con privilegi e viste ristrette. Questo permette poi all' amministratore di creare e popolare i client contexts per clients addizionali. [1]

1.6.7 Service context

I servizi coinvolgono lo scambio d' informazioni del piano dati tra i domini del client e del server. Questi scambi del piano dati tipicamente continuano anche quando non esiste alcuna sessione tra il controller e il client.
Un service context contiene concettualmente almeno gli attributi del servizio richiesti dal client e le informazioni specifiche del server necessarie a mappare gli attributi e per realizzare il servizio richiesto.
Un service context è popolato dal server e guidato da eventi che possono includere una o tutte le seguenti operazioni:

- creazione del client context da parte dell' amministratore del server per fornire i servizi prenegoziati o di default;
- creazione del servizio da parte del client, specificando il tipo di servizio e i suoi attributi;
- aggiornamento del servizio da parte del client;
- cambiamenti nello stato della rete o nelle polizze che alterano gli attributi statici dell' orchestrazione e della virtualizzazione necessari per realizzare il servizio.

Quando viene cancellato un service context, tutte le risorse usate da quel servizio vengono rilasciate e all' interno del server non rimane alcuna traccia. [1]

1.6.8 Server context

Un server context contiene tutte le informazioni necessarie e sufficienti ad un controller per gestire un gruppo di risorse sottostanti. Un amministratore crea un server context per ogni gruppo di risorse sottostanti ed è parzialmente persistente.
Un controller deve contenere un server context per ogni gruppo di risorse con cui può connettersi. Questo include attributi di associazione, informazioni di login e di sicurezza ritenute accettabili per il gruppo delle risorse sottostanti.
Possono essere inclusi anche software personalizzati, per esempio per adattare le APIs al modello d' informazione del gruppo delle risorse subordinato.
La vista fornita dal gruppo delle risorse sottostanti è disponibile al server del controller.
Il server context può anche contenere informazioni come i file di log. [1]

1.7 Applicazioni

Il modello classico dell' architettura descrive un piano di applicazione, popolato da istanze di applicazioni. Un' applicazione è un entità client che può richiedere servizi di vario tipo ad un server. Un controller offre servizi, mentre invece un entità del piano di applicazione invoca i servizi dalla sua A-CPI.
Le interazioni di controllo e gestione tra le applicazioni e il controller sono contenute all' interno delle sessioni, che consistono in associazioni di tipo client-server. Ricordiamo adesso alcune caratteristiche utili a chiarire la situazione:

- dal punto di vista del controller, ogni cosa che può invocare un servizio da una A-CPI è un' applicazione;
- se un' entità orchestra più di un gruppo di risorse tramite la sua D-CPI allora è un controller;
- se un' entità espone le sue funzionalità all' utente umano, probabilmente è un' applicazione.

Un' applicazione conforme contiene quindi un server context per ogni controller con cui stabilire una sessione. [1]

1.8 Architettura integrata

La figura 8 riprende la figura 2. Questa mostra il controller come elemento centrale all' interno dell' architettura, insieme ai componenti concettuali di orchestrazione e di virtualizzazione. In questa figura, il controller in blu offre servizi al client in verde. Un amministratore, in blu, possiede vista e privilegi ristretti rispetto al controller. [1]

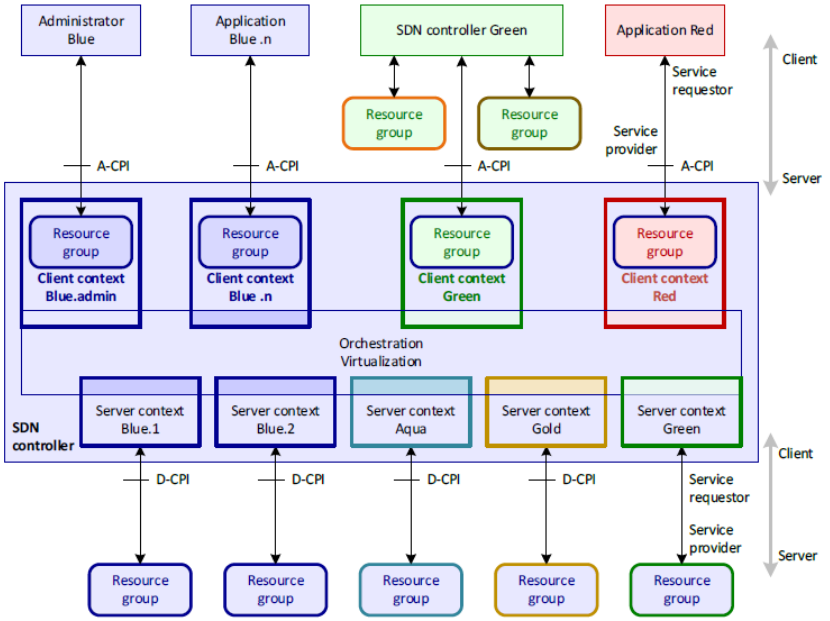

Figura 8: Contenuti del controller SDN

1.8.1 Interfacce

1. **Interfacce del piano di controllo:** il D-CPI è l' interfaccia tra un controller e il gruppo di risorse sotto il suo diretto controllo.
   L' A-CPI è l' interfaccia tra l' applicazione e il controller, attraverso la quale un' applicazione invoca i servizi da un controller.
   Viste dal punto di vista del server e del client, A-CPI e D-CPI sono rispettivamente istanze di un' interfaccia comune.

2. **Funzioni supplementari:** un controller può usare altre interfacce e altre funzioni, in particolare può invocare delle funzioni supplementari come esposto dalla figura 9. Se una funzione supplementare modifica lo stato delle risorse o dei servizi di rete, lo stato modificato deve essere notificato al controller che ha invocato la funzione, per esempio tramite una notifica. Le funzioni supplementari possono anche essere interne al controller. L' architettura non fa distinzioni tra le funzioni interne ed esterne.
   Un controller può eseguire una data funzione attraverso l' esecuzione del suo codice, incluso il codice di libreria, o attraverso l' invocazione di una

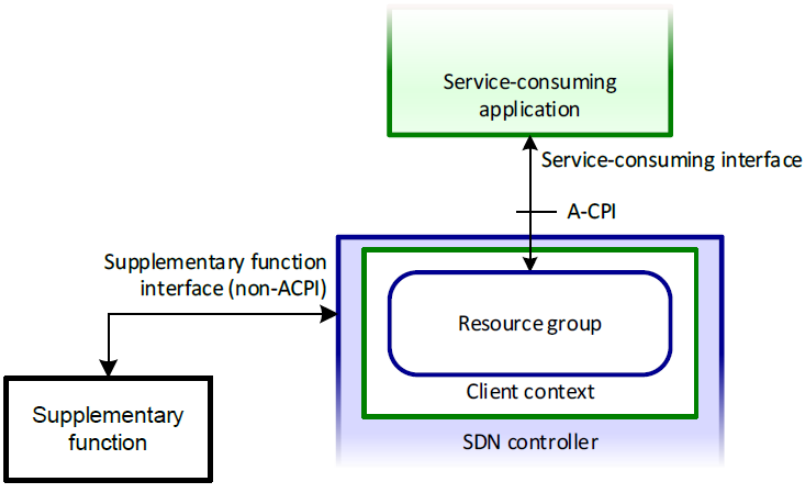

Figura 9: Contenuti del controller SDN

funzione supplementare che risiede esternamente. Un controller è in grado d' istanziare, distruggere e scalare le risorse.

3. **Cartelle e databases:** un controller richiede informazioni considerevoli per le sue normali operazioni, le quali sono modellate come esseri disponibili in un data base concettuale di risorse. Può essere desiderabile che alcune di queste informazioni risiedano altrove, accessibili al controller quando richieste.
   Le informazioni che supportano la mappatura tra client e server possono risiedere esternamente al controller. I dati di backup sono esempi addizionali di informazioni esterne.

4. **Notifiche:** le notifiche si riferiscono ai messaggi autonomi che forniscono informazioni riguardo eventi, per esempio allarmi, monitoraggio delle performance, cambiamento dei valori degli attributi, cambi di stato, eccetera. Le notifiche sono un importante collegamento nel meccanismo di feedback. Ci si aspetta che tutte le entità seguano un modello di notifiche di sottoscrizione-pubblicazione. Le sottoscrizioni e i profili sono associati con un login particolare di un client e sono persistenti.
   Un client context include una definizione delle notifiche disponibili ad un dato client, e questo è parte della virtualizzazione delle risorse sottostanti configurate dall' amministratore del server.
   Le notifiche disponibili ad ogni dato client devono essere scoperte da quest' ultimo, e la capacità di gestione delle notifiche dev' essere supportata. Questo può includere la gestione delle sottoscrizioni, la lista attiva degli allarmi e il controllo dei loro report.
   Le notifiche a partire da un livello sottostante devono essere interesse del client, il quale deve virtualizzarle e mapparle all' interno del suo client context. Questo può richiedere più di una traduzione dell' identificatore.
   Poichè un client non ha bisogno di mantenere una sessione continua di

gestione-controllo con il controller, le notifiche visibili al client possono essere richieste. [1]

1.9 Prospettive dell' architettura

Così come SDN o altri paradigmi di controllo, anche il piano dati deve fronteggiare gli stessi problemi riguardanti la sicurezza, come ad esempio:

- lo sviluppo del software eseguibile;
- la sicurezza e il controllo della rete di comunicazione;
- l' integrità delle cartelle;
- la possibilità che si verifichino errori o attacchi interni.

Un problema importante dell' architettura è quello di assicurare un comportamento proprio alle interfacce, le quali possono fornire una vasta gamma di servizi, in quanto queste ultime sono ampiamente esposte a terzi, aumentando quindi la possibilità del manifestarsi di errori o attacchi. Indirizzare le interfacce verso un modello d' informazione comune può essere un buon metodo per garantire l' integrità delle API fin dall' inizio.
L' architettura afferma che la sessione instaurata tra un client e un controller è sicura e autenticata. Una volta stabilita la sessione verrà creato il client context che conterrà come elementi essenziali:

- le credenziali di accesso sicuro
- le risorse virtuali
- la polizza del client specifico

Tuttavia possono essere presenti alcuni difetti all' interno del client context, come per esempio errori di configurazione, di mappatura o di rete. [1]

1.10 Considerazioni operative

1.10.1 Affidabilità e disponibilità

Nel piano dati, l' affidabilità e la disponibilità sono largamente invariate quando il controllo è logicamemente separato dall' infrastruttura di processamento del traffico.
Dal punto di vista dell' affidabilità e della disponibilità, una rete implementata in un software richiede una riesaminazione, in quanto:

- le funzioni del piano dati possono essere più finemente decomposte e richiedere più risorse di rete;
- la dinamica della scoperta del fallimento può cambiare;
- poichè il fallimento può portare alla perdita di stato, lo stato di ripristino a partire da dei checkpoints risulta essere una soluzione parecchio redditizia.

Come il controllo, la possibile separazione del controller dal processamento del traffico implica una riduzione della disponibilità. Questa separazione può essere eseguita secondo le seguenti ipotesi:

- i motori di processamento del traffico possono richiedere che le loro funzioni rimangano invariate anche in assenza di una connessione di un controller attivo;
- i controllers possono essere implementati attraverso configurazioni ridondanti e con una sincronizzazione real-time.

La nozione di ricorsione implica che una richiesta di servizio richieda di essere propagata attraverso un numero di controllers vicini prima di venir soddisfatta. Se un controller fallisce sarà necessario rilasciare le risorse committate. [1]

1.10.2 Identificatori

Gli identificatori sono essenziali nel determinare cosa è sotto considerazione e dove può essere trovato. Questi sono importanti perchè la vista delle risorse e dei servizi del client dev' essere mappata in uno spazio d' identificazione del server, così come la possibilità di traduzione in diverse classi di oggetti.
Non è specificato come gli identificatori debbano essere stabiliti inizialmente, in quanto:

- questi sono configurati e conosciuti attraverso una scoperta manuale;
- questi sono proposti da un lato e accettati dall' altro durante la negoziazione dei servizi.

Una risorsa virtuale può anche essere basata su una visione complessa di diverse risorse sottostanti. [1]

1.10.3 Considerazioni di realizzazione

L' amministratore crea e modifica il client e il server context e i criteri di ottimizzazione del controller. Ognuno di essi può essere eseguito attraverso degli strumenti offline.
Data una definizione adatta alle istanze dell' applicazione client alle quali è permesso connettersi alle istanze del controller, almeno una parte del client context dev' essere accessibile dai depositi esterni, i quali possono essere scaricati da diversi controllers.
Un controller richiede significativi quantitativi d' informazioni persistenti, in particolare nel client e nel server context. I backup possono essere particolarmente appropriati in questi contesti.
Un controller può essere implementato in una forma distribuita di computazione. Mantenere lo stato delle operazioni sincronizzato durante le operazioni di recupero è molto importante. Come entità di rete, ci si può aspettare che un controller venga spesso implementato come una Virtual Network Function (VNF). [1]

1.10.4 Inizializzazione

La preparazione delle risorse di rete per i servizi richiede operazioni come l' installazione e la configurazione dei parametri iniziali.
L' architettura presuppone un controller iniziale che includa un software in funzione con un amministratore associato al client context, il quale permette

la creazione di un server context e di ulteriori client contexts. A questo punto viene stabilita una sessione con il server, e le risorse esposte sono disponibili nel RDB, nel quale queste vengono allocate staticamente dall' amministratore verso un particolare client context o consumate dinamicamente dalla funzione d' orchestrazione. Fornito il materiale fisico o virtuale, un controller può istanziare nuove risorse. [1]

1.10.5 Complessità

La complessità del controller è una preoccupazione legittima. Diversi approcci sono disponibili, alcuni dei quali sono esposti nell' elenco sottostante:

- i continui miglioramenti nella tecnologia di computazione assistono il controller durante il mantenimento di carichi di lavoro sempre maggiori;
- il problema di ottimizzazione può essere semplificato se i criteri di feedback del controller definiscono un intervallo di risultati ritenuti accettabili;
- un controller, nonostante le sue ridotte funzionalità, è semplificato dall' allocazione delle sue risorse pre-esistenti;
- l' architettura assume che alcune parti delle risorse sottostanti siano efficientemente virtualizzate in un insieme disgiunto che soddisfa i bisogni dei vari clients.

L' orchestrazione, funzione chiave di un controller, rappresenta un problema multi-dimensionale e real-time sopra uno spazio complesso e potenzialmente ampio. Inoltre, questa lavora simultaneamente attraverso le risorse sottostanti e la loro virtualizzazione. [1]

1.10.6 Persistenza

L' informazione ha uno scopo utile in un tempo di vita utile.
La polizza per l' immagazzinamento d' informazioni ridondanti e persistenti dovrebbe essere riconciliata con il costo di derivazione. I fattori che dovrebbero essere considerati includono sicurezza, sincronizzazione, ampiezza di banda, amministrazione, eccetera.
Ci si aspetta che la maggior parte del contenuto del client context, del server contexts e dei criteri di ottimizzazione della polizza giustifichino l' immagazzinamento ridondante e possibilmente la ridondanza locale per ridurre la probabilità di un fallimento da parte di un controller. [1]

2 Architettura SDN per le reti di trasporto

2.1 Introduzione e scopo

Questo capitolo descrive l' applicazione dell' architettura alle reti di trasporto. Le reti di trasporto necessitano di soddisfare la crescente richiesta di banda dei data centers e di fornire dei responsi real-time ai cambiamenti della Quality of Service (QoS). Gli obiettivi specifici che questo capitolo si impegna ad adempiere sono:

- l' applicazione dell' architettura alle reti di trasporto;
- l' uso dell' architettura per supportare la virtualizzazione delle risorse di trasporto di rete;
- l' uso dell' architettura per supportare il controllo integrato di diversi multi-domini e di reti di trasporto multi-livello;
- i servizi di connettività di trasporto relativi ad un modello di flusso unidirezionale e come questi sono legati all' architettura;
- i comportamenti comuni delle reti di trasporto come monitoraggio e protezione e come questi possono essere accolti all' interno dell' architettura.

L' architettura e il modello d' informazione applicano alle reti di trasporto degli elementi chiave, come l' orchestrazione/virtualizzazione e la relazione tra flusso e connettività. [4]

2.2 Funzionamento

2.2.1 Applicazione dell' architettura al modello d' informazione

In questo paragrafo si specifica come SDN controlla il trasporto delle risorse di rete, e di come queste vengono mostrate al controller. Ciò viene fatto attraverso l' elaborazione delle risorse di rete ad un alto livello di rappresentazione attraverso l' utilizzo della rete di trasporto e grazie al modello architetturale degli elementi di rete.
Essendo la rete di trasporto molto vasta e composta da numerosi componenti, è appropriato definire un modello di rete ben definito. La rete dei trasporti può essere descritta attraverso la definizione di associazioni tra punti nella rete. Il risultato logico permette di separare le connessioni, i percorsi fisici e le risorse usate.
Per semplificare la descrizione, la rete di trasporto utilizza i concetti di livellamento e di partizionamento all' interno di ogni livello di rete, in modo da consentire un gran livello di ricorsione del piano dati.

- Il livellamento consente la decomposizione di una rete di trasporto multilivello in un una serie di livelli indipendenti, all' interno dei quali viene descritta la generazione, il trasporto e la terminazione di particolari informazioni.

- Il partizionamento è la divisione di una grande rete in sottoreti disgiunte che sono interconnesse da collegamenti. Il componente delle reti di trasporto può essere caratterizzato da componenti topologici, funzioni di processamento ed entità di trasporto.
- Le funzioni di processamento di trasporto sono usate per modellare i processi, i quali manipolano l' informazione che viene trasferita attraverso la rete di trasporto.
- Le entità di trasporto trasferiscono l' informazione attraverso la rete di trasporto, attraverso dei punti di connessione. Le entità di trasporto sono configurate all' interno dei componenti topologici.

Il componente topologico fornisce la descrizione più astratta di una rete di trasporto in termine di relazioni tra gruppi di punti di connessione all' interno di un livello di rete. [4]

2.2.2 Utilizzo dell' astrazione e della virtualizzazione nella rete di trasporto

L' architettura comprende il piano applicativo, il piano di controllo e il piano dati dove il controller gestisce le risorse del piano dati e le applicazioni che ricevono servizi dal controller. Come descritto dall' architettura, il piano di controllo può coinvolgere più controllers.
Al livello più basso della ricorsione, le risorse del piano dati sono esposte da entità fisiche della rete. In ogni caso il client context fornisce una visione astratta delle risorse. L' interfaccia del piano di controllo è l' interfaccia generica attraverso la quale un' istanza del modello d' informazione è gestita.
La figura 10 mostra in maniera semplificata come le operazioni risiedano in una visione astratta delle risorse fisiche. Nello specifico, ci sono client context che rappresentano le entità controllate (Network Elements in verde e giallo) mentre i controllers in verde e giallo agiscono su questi client context, rispettivamente.

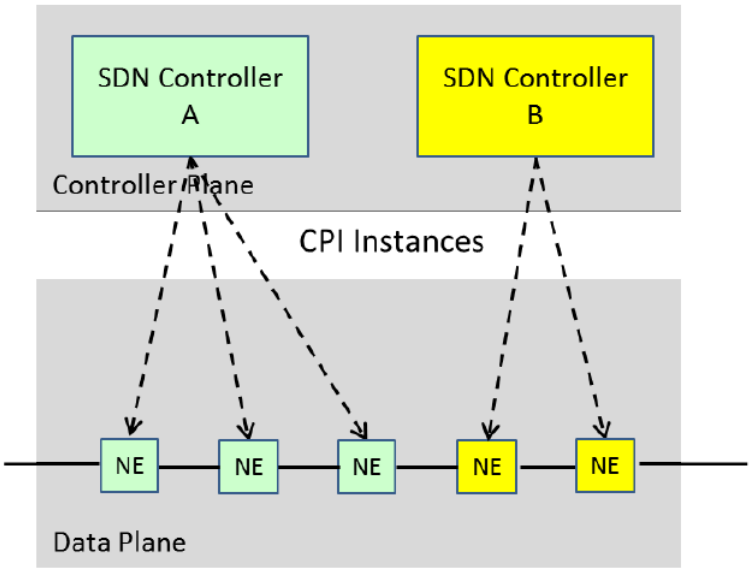

Figura 10: Esempio della gestione dei propri rispettivi elementi di rete da parte dei due controllers

In figura 11 viene invece illustrata l' associazione tra client e server context. In questa situazione è presente invece una gerarchia con il controller in blu che offre risorse al controller in verde. Dalla prospettiva del controller in verde, tutto ciò che sta sotto di lui è all' interno del piano dati. Il controller B offre un' API di trasporto al controller G e l' abilità di creare e controllare i servizi di connettività attraverso un gruppo comune di risorse esposto al controller G. Il controller B possiede le conoscenze delle relazioni tra la rete virtuale esportata al controller G, il modello sottostante delle risorse allocate da B a G e la mappatura delle risorse di B. [4]

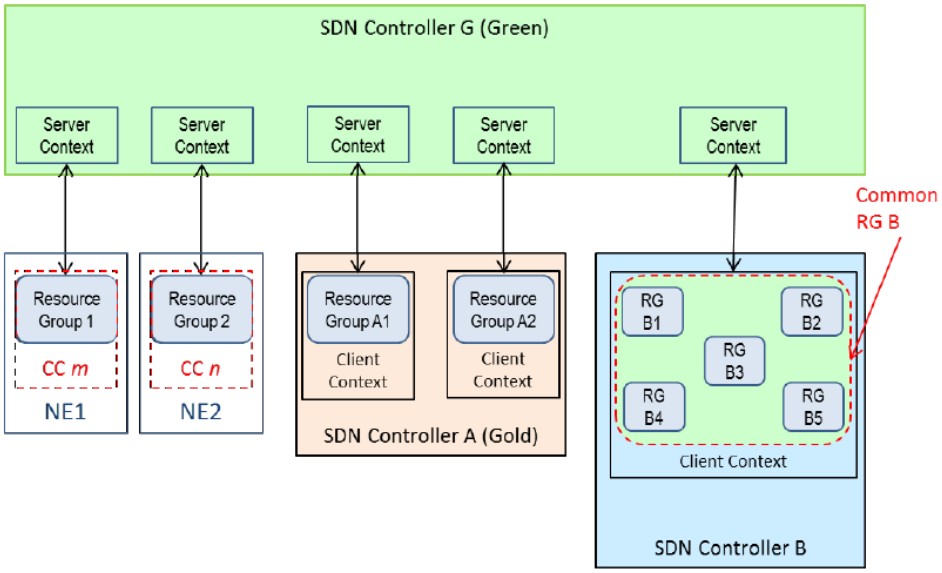

Figura 11: Esempio del controllo del piano dati sia di risorse fisiche che virtuali

2.2.3 Modellazione di un dominio multiplo e di una rete di trasporto multi-livello

1. **Multi-dominio e multi-livello di reti:**
   il fornitore di servizi della rete di trasporto è comunemente partizionato in più domini amministrativi. L' architettura supporta un modello di controllo integrato di domini multipli.
   La relazione tra i controllers in questi domini separati non è necessariamente gerarchica. La collaborazione tra i controllers può esssere richiesta per rinforzare i servizi attraverso i domini, come mostra la figura 12.
   La rete suddivisa in più livelli può anche essere implementata con l' integrazione del livello di controllo dei domini che sono amministrativamente separati e che operano internamente a livelli diversi. Questa integrazione di una rete in più livelli può essere modellata usando un controllo gerarchico dei domini amministrativi multipli, dove controllers "figli" sono coordinati attraverso un comune controller "padre", come mostrato dalla figura sottostante.

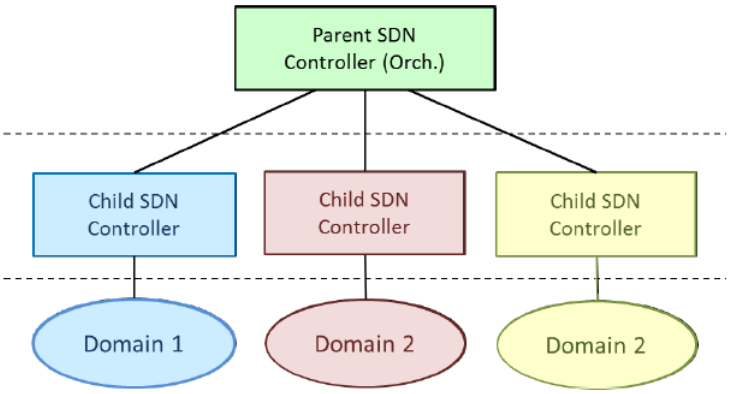

Figura 12: Controllers gerarchici all' interno di una rete multi-dominio

Il controller "padre" ha accesso alla topologia e all' informazione di stato per ogni dominio server e può coordinare le azioni attraverso entrambi i domini.

2. **Multi-livellamento all' interno di un singolo dominio:**
   L' adattamento multi-livello occorre anche in singoli domini amministrativi. Quando due domini usano diversi livelli di tecnologia, un dominio o un' altro devono supportare l' adattamento dal suo livello di cambiamento interno all' altro.
   Il supporto di questo adattamento interno è un' altro scenario di rete multilivello. Due cause sono possibili:

   - il dominio di trasporto di rete supporta un accesso ad un' interfaccia che differisce dal suo livello interno di cambiamento;
   - il dominio di rete di trasporto supporta l' adattamento tra più segnali all' interno della stessa tecnologia.

   L' adattamento permette alle informazioni caratteristiche del client di essere poste in una forma utile a essere trasportate su un livello server.
   Il supporto all' adattamento tra livelli di trasporto di rete richiede l' abilità d' identificare il livello client e controllare il livello specifico che viene usato. La figura 13 mostra questo adattamento all' interno del dominio [4]

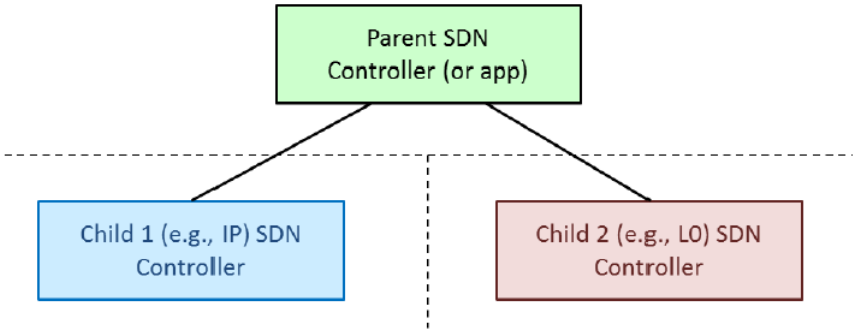

Figura 13: Controllo e adattamento all' interno del dominio

2.3 Modellazione dei servizi di connettività di trasporto

2.3.1 Flusso contro connessione

Le reti di trasporto sono generalmente progettate per metttersi d' accordo tramite le connessioni, entità che comprendono molteplici flussi correlati.
Una connessione di una sottorete è un' entità di trasporto che trasferisce l' informazione attraverso una sottorete formata da un' associazione di porte.
Un flusso è definito come uno stream di dati identificabili con alcuni criteri come dei valori dell' intestazione del pacchetto, il quale è inoltrato in base ad un comune gruppo di regole all' interno dei confini della connessione.
I flussi possono esistere in ogni livello usando dei campi associati al livello corrispondente. I flussi possiedono inoltre un identificativo che può essere guidato per indirizzare l' inoltramento dei flussi all' interno di una determinata connessione.
Il protocollo OpenFlow Switch (OFS) può essere usato per controllare l' inoltramento dei flussi all'interno di una connessione che supporta l' inoltramento dei flussi senza connessione.
Nel livello di trasporto una connessione rappresenta un trasporto di entità nel quale zero o più flussi vengono trasportati. I casi più complessi riguardano connessioni punto-multipunto e multipunto-multipunto. Lo stato e le performance di una connessione vengono monitorate attraverso l' uso di flussi OAM (Operations, Administration, Maintenance), i quali sono associati al flusso dati e processati all' interno degli elementi di rete. L' inoltramento di un singolo flusso dati può inoltre venir modificato secondo le informazioni OAM.
I flussi OAM possono essere trasferiti in un canale diverso dal canale del flusso dati client, oppure nello stesso. Nell' ultimo caso possono verificarsi delle interferenze l' uno con l' altro; il problema può essere risolto suddividendo il canale OAM in canali più piccoli sempre di tipo OAM, ognuno con un' ampiezza di

banda specifica. Inoltre i flussi dati e quelli OAM sono distinti tra loro attraverso l'uso di etichette (implicite o esplicite).
Le funzioni di adattamento atomiche all' interno della connessione possono generare e inserire uno o più flussi OAM e terminare uno o più flussi OAM. Le funzioni di adattamento possono anche fornire una gestione di alto livello e controllare le funzioni di accesso a un sottocanale OAM in modo da consentire una comunicazione con le altre funzioni di controllo. [4]

2.3.2 Ciclo di vita della connessione di trasporto

Le connessioni create in una rete di trasporto seguono tipicamente un ciclo di vita pensato per validare la connettività e assicurare che la connessione acceda ai servizi in accordo con le richieste del cliente. Molte funzioni sono compiute oltre lo stabilimento dell' inoltramento del flusso, come ad esempio:

- il blocco e lo sblocco amministrativo;
- la configurazione di PM (Performance Monitoring) dal monitor verso le connessioni end-to-end;
- la configurazione di punti TCM (Tandem Connection Monitoring) alle performance del monitor attraverso segmenti di connessione.

Consideriamo ora le seguenti azioni.

1. Creazione ed eliminazione, così come le conferme associate:
    - configurazione/distruzione di una connessione;
    - creazione/rimozione di monitors e contatori PM.
2. Modifica e conferma degli attributi:
    - modifica dello stato amministrativo (blocco/sblocco);
    - attivazione/disattivazione TCM;
    - report degli allarmi (attivazione/disattivazione);
    - attivazione, disattivazione, aggiornamento, reset, registrazione PM;
    - attivazione, disattivazione, aggiornamento, reset registrazione della misura del ritardo;
    - comandi di protezione.
3. Notifiche del cambio di attributi:
    - notifica di cambio di stato delle operazioni;
    - cambio di stato della protezione.
4. Altre funzioni:
    - aggiunta/rimozione di una connessione esistente;
    - aggiunta/rimozione della misurazione del ritardo su una connessione esitente;
    - test di attivazione/disattivazione.

Queste azioni e notifiche prendono posto tipicamente per supportare la configurazione e le operazioni di una connessione. [4]

2.3.3 Configurazione dell' inoltramento del flusso per diversi tipi di connessione

Le connessioni di diverso tipo possono essere descritte in termini d' inoltramento di flussi unidirezionali a partire da input in output dello switch.
Il numero di flussi e d' associazioni dipendono dal tipo di connessione.
Esempi di un comune inoltramento del traffico includono diversi comportamenti a seconda del contesto di connessione:

- porta non connessa
- connessione punto-punto
- connessione punto-multipunto
- connessione multipunto-punto
- connessione multipunto-multipunto

L' inoltramento dei flussi associati ad una particolare connessione possono essere affetti da OAM o da un monitoraggio del segnale che può coinvolgere specialmente i segnali generati come una connessione aperta o un rimpiazzamento di segnale inserito in un flusso. I cambiamenti nella configurazione di un flusso possono essere fatti sotto la direzione di un controller. A causa di questi svantaggi, il monitoraggio del segnale e il controllo della configurazione dei flussi associati cambia in base alla generazione dei segnali speciali consentiti. [4]

2.3.4 Comportamento del flusso in una connessione protetta

L' inoltramento del traffico può anch' essere guidato dalla protezione della connessione in caso di fallimento. In SDN si assume che il gruppo di protezione sia configurato, modificato ed eliminato sotto la gestione di un controller, sia durante che dopo la configurazione della connessione. In caso di fallimento, l' inoltramento del traffico dev' essere modificato rapidamente per tenere le informazioni all' interno del percorso protetto.
Ci sono diversi modi per proteggere le disposizioni usate nella rete dei trasporti. Ci sono diversi tipi di protezioni e di triggers, inclusa la protezione che può essere innescata a partire da un evento o da una differente ma correlata connessione. Cambiamenti all' inoltramento del traffico possono essere eseguiti teoricamente sia grazie al controller che grazie ad un processo locale. In ogni caso al controller dev' essere prima notificato l' evento di fallimento da parte dello switch e poi dev' essere inizializzata una nuova configurazione del flusso che compensi il fallimento. In accordo con i requisiti di servizio, la riconfigurazione del flusso dev' essere in molti casi eseguita da alcune funzioni autonome che abbiano precedentemente scaricato istruzioni per evitare latenze. [4]

2.4 Lavoro futuro

Questo capitolo ha descritto gli aspetti generali dell' architettura SDN per le reti di trasporto. Ci sono altri aspetti da tenere a mente e che si potrebbero implementare in futuro, come per esempio le operazioni di amministrazione e di fornitura della connessione, le interazione tra domini multipli di amministrazione, la connessione protetta e i dettagli aggiuntivi sul ciclo di vita della connessione. [4]

3 Paradigma NBI

3.1 Introduzione

Questo capitolo descrive i principi operazionali, architetturali e strutturali che definiscono Intent NBI (North Bound Interface).
Intent NBI è un paradigma dichiarativo usato per rendere più semplice possibile l' inoltramento delle richieste del consumatore da parte del fornitore (i quali possono essere operatori umani oppure sistemi macchina).
La richiesta NBI non specifica nè i dettagli d' implementazione del servizio, nè le risorse necessarie per soddisfarlo. [5]

3.2 Sommario esecutivo

Come abbiamo affermato precedentemente, Intent NBI è un paradigma dichiarativo in quanto non specifica nè i dettagli dell' implementazione del servizio richiesto, nè le risorse da utilizzare per soddisfarlo.
Dal punto di vista del server possiamo immaginare Intent NBI come un' A-CPI e come una D-CPI dal punto di vista del client.
Intent NBI può essere implementato sia come architettura client-server che come peer-to-peer. L' applicazione del paradigma risulta fortemente appropriata in due diversi contesti:

- nell' interazione diretta tra sistemi (umani/macchina) di consumatori-fornitori;
- nelle interazioni tra le applicazioni di alto livello e tra i componenti di basso livello.

Elenchiamo ora gli strumenti con cui Intent NBI realizza i suoi servizi.

- **La mappatura:** rappresenta un meccanismo d' intermediazione che permette al consumatore e al fornitore di comunicare in maniera "naturale" tra loro. I termini che appaiono nelle richieste da parte del consumatore vengono infatti tradotti con termini direttamente rilevanti e comprensibili al fornitore.
- **Il ciclo di confronto:** tra le richieste nuove e già presenti e le risorse controllate e gli stati, i quali possono evolversi nel tempo per garantire un corretto funzionamento del servizio.
- **Le notifiche:** sono dei messaggi autonomi che forniscono informazioni riguardanti determinati eventi. In termini dichiarativi possiamo considerare le richieste NBI come un servizio che viene prodotto e mantenuto dal fornitore.
  Nel caso in cui questo non possa più essere mantenuto, il fornitore provvederà allora ad avvisare il consumatore tramite una notifica.

La figura sottostante illustra l' interazione tra i sistemi consumatore-fornitore grazie alla mappatura. [5]

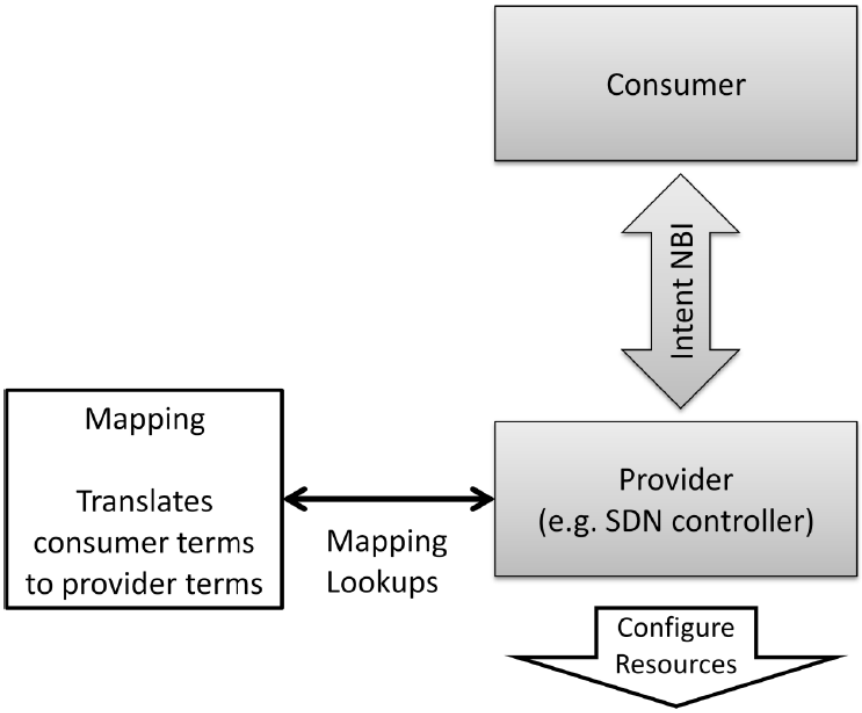

Figura 14: Rappresentazione architetturale dell' Intent NBI e della mappatura

3.3 Principi di NBI

NBI semplifica le richieste di servizio tra i sistemi consumatore e fornitore. Questo accade grazie alle mappature, le quali permettono al consumatore di richiedere i servizi utilizzando termini conosciuti e rilevanti per lui, mentre la scelta del meccanismo di adempimento del servizio viene lasciata interamente al fornitore. Ciò aiuta a separare, architetturalmente e operativamente, la gestione e la presentazione dei servizi dal loro meccanismo di consegna.
Un' analogia utile per definire il modo di operare di NBI è simile a quello del trasporto dei taxi: il consumatore ha bisogno, partendo da un punto A, di raggiungere un punto B. Al consumatore generalmente interessa solo raggiungere il punto B, non importa quale percorso verrà scelto. Questo tipo di richiesta coinvolge solo informazioni rilevanti e note al consumatore. Il modo in cui il trasporto viene effettuato è comprensibile al trasportatore, attraverso la consultazione di alcune informazioni esterne come mappe o databases.
La richiesta "portami da A a B" può essere offerta ad ogni trasportatore: questa informazione infatti non contiene alcuna preferenza verso un particolare trasportatore. Quest' ultimo è libero di usare le proprie risorse disponibili, in accordo con la sua polizza, per fornire il servizio richiesto. Intent NBI può includere dei "modificatori" i quali possono aggiungere dettagli alla richiesta esistente. Per esempio, sempre prendendo in considerazione il trasporto in taxi, le informazioni aggiuntive potrebbero includere anche "entro un tempo C" oppure "con un prezzo non superiore a D".

La figura sottostante mostra l' interazione tra consumatore e fornitore attraverso Intent NBI. [5]

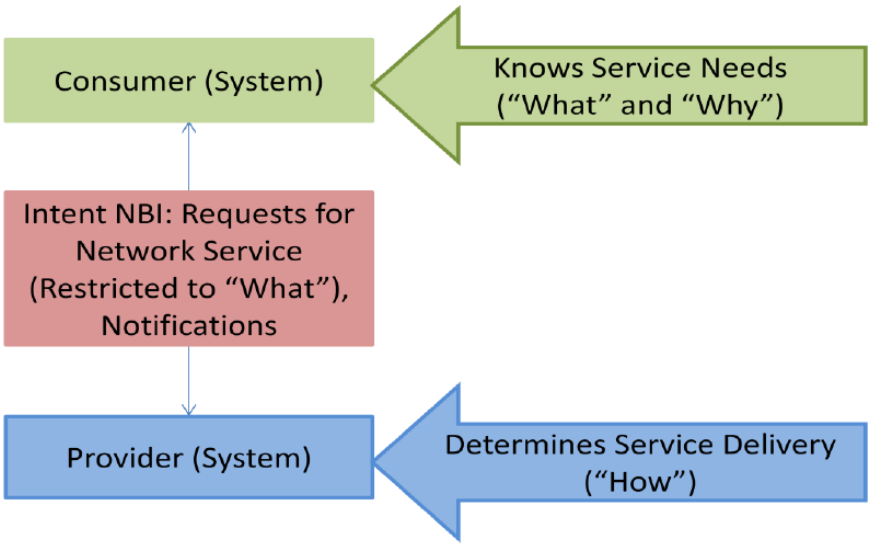

Figura 15: Interazione tra fornitore e consumatore attraverso NBI

3.4 Proprietà e struttura di NBI

NBI è caratterizzato dalle seguenti proprietà.

1. **Non prescrizione:** rappresenta i modi in cui una richiesta NBI può venir formulata da parte del consumatore. Le richieste specificano "cosa" (servizi richiesti) ma non "come" devono essere consegnati. Le decisioni riguardanti l' uso delle risorse sono lasciate interamente al fornitore, così come le scelte d' implementazione riguardanti la tecnologia, i media, i nodi, le porte, i collegamenti, i percorsi e i server utilizzati, e pertanto alcuni parametri non vengono specificati nelle richieste.

2. **Indipendenza del fornitore:** è una proprietà della richiesta che deriva dalla non-prescrizione. La stessa richiesta può essere presentata, senza variazioni, ad ogni fornitore al quale è permesso interoperare con il consumatore, il quale a sua volta provvederà a soddisfarla attraverso le mappature. I termini apparsi nella richiesta possono essere tradotti, sempre attraverso le mappature, in termini direttamente rilevanti ai sistemi di basso livello.

3. **Dichiarazione:** il consumatore dichiara cosa vuole ottenere dal fornitore. Non è necessario specificare nessun' altra informazione aggiuntiva.

Queste proprietà rendono possibile la separazione dei sistemi fornitore-consumatore e rendono la loro interazione più semplice possibile. [5]

3.4.1 Implementazione

I sistemi NBI possono essere efficacemente implementati e rappresentati come dei "motori" posti sopra i sistemi di fornitura dei servizi, come evidenziato dalla figura sottostante. Il motore Intent è, in effetti, un client del controller e

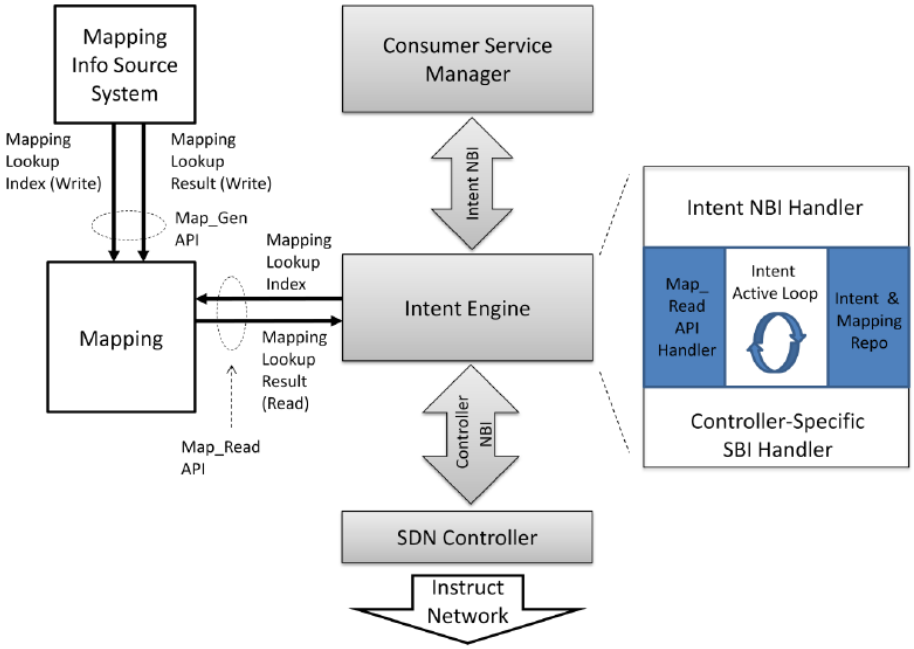

Figura 16: Rappresentazione schematica del motore Intent NBI

un server del consumatore (in figura indicato come Consumer Service Manager). Il motore (che in figura è rappresentato dal macro blocco sulla destra) può essere decomposto in cinque componenti principali.

1. Un deposito d' informazioni che contengono i servizi attivi e i valori di mappatura.
2. Un API Handler di lettura della mappa. Questa è responsabile del mantenimento dei valori aggiornati delle mappature.
3. Un Intent NBI Handler, il quale è il responsabile:
   - della ricezione degli Intent da parte del consumatore e della loro consegna verso il deposito d' informazioni;
   - della consegna di eventuali notifiche al consumatore.
4. Un ciclo attivo di Intent, il quale provvede a:
   - valutare costantemente il servizio di Intent e di mappatura;
   - istanziare un nuovo servizio oppure modificarne un' altro già presente.

Queste due azioni vengono eseguite attraverso la computazione di particolari comandi che vengono poi inoltrati all Handler SBI, seguendo regole ben precise.
Infatti tali comandi devono aderire ad un "modello base": per esempio il comando deve specificare uno o più punti di interconnessione, o magari devono essere specificati dei vincoli a questi ultimi.
In ogni caso nel comando dev' essere specificato ogni appropriato dettaglio che soddisfi il "modello base" dell Handler SBI.

5. Un handler SBI (South-Bound Interface), il quale si preoccupa di:
    - ricevere input dal ciclo di Intent e di restituirli al controller in una forma appropriatamente modificata;
    - riceve informazioni dal controller e inoltrarle al ciclo di Intent in una forma appropriatamente modificata.

Il motore NBI non è solo un utile veicolo concettuale ma può, in alcune circostanze, essere un vantaggioso metodo d' implementazione. Attraverso la separazione degli Intent dai controllers, è possibile utilizzare dei controllers con specifici handler del motore NBI. [5]

3.4.2 Esempio operativo

In questo esempio, il client Intent NBI è rappresentato da un' azienda di sicurezza. Un manager di rete (un operatore umano) è responsabile della creazione di gruppi utente, di punti d' accesso internet e dell' indicazione dell' indirizzo IP corrente applicabile per ognuno di questi servizi. Un' altro manager VNF (un operatore umano) è responsabile dell' istanziamento di diverse classi di funzioni VNFs e dell' indicazione del corrente indirizzo IP per ognuno di questi servizi.
Diciamo ora che il manager dei servizi dell' azienda vorrebbe permettere a tutti i suoi utenti di connettersi a internet, filtrando solo le connessioni che superino un firewall. Questa operazione può essere eseguita grazie all' uso del motore Intent. Infatti attraverso le funzioni di mappatura il motore Intent ottiene gli indirizzi IP che corrispondono agli utenti e genera una serie di coppie di endpoint di connessione che vengono passate al controller sotto forma di indirizzi IP, con le quali vengono create delle connessioni punto-punto.
Il ciclo attivo del motore si incarica invece di:

- tenere aggiornata la lista degli indirizzi IP degli utenti del gruppo;
- monitorare costantemente la connessione resa disponibile dal controller;
- modificare all' occorrenza alcune istruzioni per mantenere attiva la connessione degli utenti a internet mediante il firewall.

In caso di problemi o malfunzionamenti alla rete, il sistema di notifiche provvederà ad avvisare l' azienda.
Il comportamento descritto in precedenza può anche venir rappresentato dalla figura sottostante. [5]

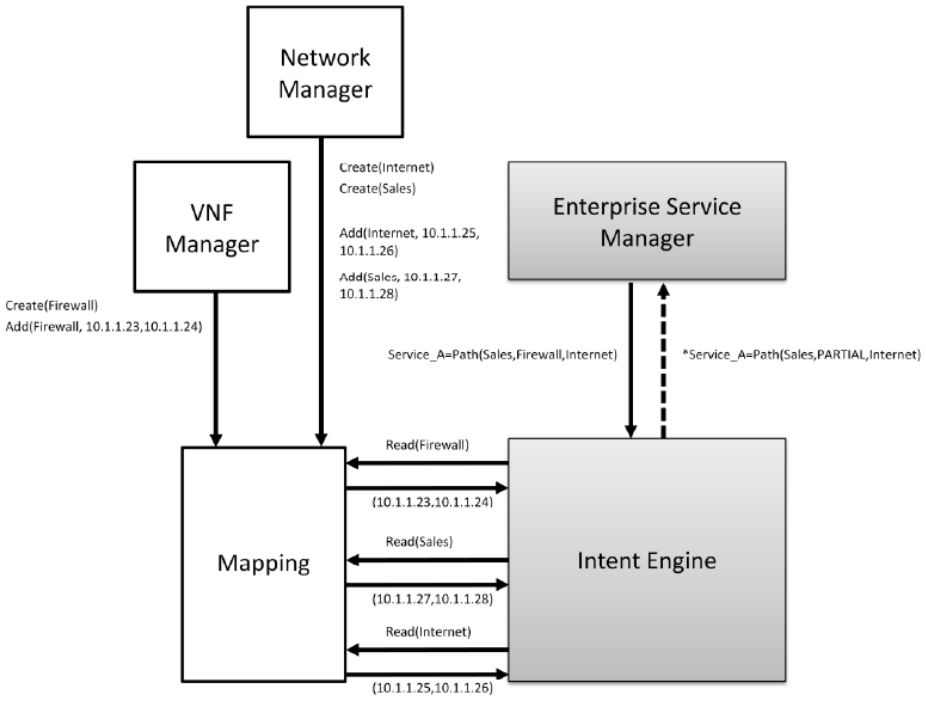

Figura 17: Un esempio delle operazioni di sistema di NBI per un semplice servizio

3.5 Benefici di Intent NBI

Riesiaminiamo alcune proprietà descritte in precedenza e i relativi benefici.
Una richiesta NBI è **non-prescrittiva** rispetto alle implementazioni della richiesta. Le scelte che riguardano la tecnologia, i media, i percorsi, eccetera sono lasciati al fornitore e generalmente molti parametri non appaiono mai nelle richieste NBI.
Una richiesta NBI è **componibile**, nel senso che le richieste possono essere effettivamente costituite da un insieme di input specifici.
Il paradigma NBI è **universale**, nel senso che è sempre possibile per un consumatore esprimere i propri requisiti di servizio con termini compatibili con il paradigmi NBI.
Intent NBI può mitigare i **conflitti d' allocazione delle risorse** che possono insorgere in casi di richieste di servizio concorrenti verso un dato fornitore. Questa proprietà deriva dalla non-prescrizione: da quando le richieste dell Intent NBI non specificano quale fornitore debba allocare le risorse specifiche, i fornitori guadagnano una relativa libertà di assegnamento delle risorse in servizi in modo da minimizzare la possibilità potenziale della contesa su di esse.
I sistemi basati su Intent NBI possono essere più **sicuri** dei sistemi basati su prescrizione. In un sistema NBI, esiste un piccolo margine di possibilità di attacco alla sicurezza, poichè NBI non possiede alcuna abilità di controllo degli aspetti della gestione della risorsa. [5]

4 NFV : Network Function Virtualization

4.1 Terminologia

Durante l' esposizione di alcuni concetti di questo capitolo, si presenteranno spesso i seguenti temini. È bene quindi farvi conoscenza per una corretta comprensione del documento. [6]

4.1.1 Virtualizzazione

La virtualizzazione possiede differenti significati nelle due architetture SDN e NFV.
SDN considera la virtualizzazione come "un astrazione che rappresenta le entità in termini di determinate caratteristiche comuni, ignorando e nascondendo quelle irrilevanti ai criteri di selezione".
Diverse conseguenze seguono questo concetto:

1. tutte le viste semplici sono astratte, solo quella selezionata è virtuale;
2. non c' è un' unità atomica di granularità. La virtualizzazione modella le entità al livello appropriato allo scopo;
3. la ricorsione è naturale: una virtualizzazione SDN può essere eseguita in molti modi, e nel livello di dettaglio desiderato;
4. le risorse possono essere partizionate o combinate arbitrariamente durante la virtualizzazione. Le proprietà dell' entità virtuale SDN possono differire completamente dalla prospettiva di ogni componente;
5. le risorse che contribuiscono alla virtualizzazione SDN possono esistere individualmente ad un diverso livello di virtualizzazione.

Nella documentazione NFV, il virtuale si riferisce sempre ad un' entità software all' interno di un contenitore, inteso tipicamente come una virtual machine sopra un supervisore o un server COTS (Commercial Off-The-Shelf).
Per NFV, "la virtualizzazione mira a trasformare il modo con cui gli operatori architettano la rete attraverso l' evoluzione dell' esistente virtualizzazione IT e facendo uso di tecniche di computazione cloud".
La rete virtuale è concepita come "il componente topologico usato per influenzare l' inoltramento d' informazioni con caratteristiche specifiche". All' interno della rete NFVI (Network Function Virtualization Infrastructure), una rete virtuale inoltra l' informazione alle istanze della virtual machine e alle interfacce della rete fisica, fornendo la connettività necessaria e assicurando un isolamento del traffico sicuro dalle diverse reti virtuali.
La rete NFVI si focalizza sulla fornitura di servizi di connettività per le VNFs. In SDN, una rete include nodi che possono processare trafico così come altri che possono soltanto inoltrarlo.
La virtualizzazione SDN è primariamente diretta alla rappresentazione di risorse, mentre quella NFV è diretta a separare le funzionalità dall' infrastruttura.

4.1.2 Orchestrazione

Sebbene SDN non definisca formalmente l' orchestrazione, il significato del concetto viene definito dalle seguenti proposizioni.

- Un controller coordina un numero di risorse intercorrelate, spesso distribuite su un numero di piattaforme subordinate, e alcune volte per assicurare un' integrità transazionale come parte del processo. Questo viene comunemente chiamato orchestrazione.
- Un applicazione può invocare altri servizi esterni e può orchestrare ogni numero di controllers SDN per raggiungere i propri obiettivi.
- Il continuo processo di allocazione delle risorse per soddisfare i bisogni richiesti in maniera ottimale.

Per quanto riguarda la NFV, una chiara definizione non è stata fornita. Il termine può essere tratto dalla definizione di NFVO (Network Function Virtualization Orchestrator), il quale è "un blocco funzionale che gestisce i servizi di rete e coordina la gestione del ciclo di vita dei servizi di rete". Il ciclo di vita della NFV assicura un' allocazione ottimizzata delle risorse necessarie e della connettività. Infine, la gestione del ciclo di vita è un gruppo di funzioni richieste per gestire l' istanziamento, il mantenimento e la terminazione di una VNF o di un NS (Network Service).

4.1.3 Gestione-controllo

NFV utilizza il termine gestione per estendere il termine controllo. In SDN, controllo è il termine operativo, mentre gestione è un termine secondario.
Per introdurre l' argomento nel contesto SDN, bisogna ricordare che per eseguire il proprio compito, le funzioni di orchestrazione e di virtualizzazione devono essere configurate. Le entità che eseguono questo tipo di funzioni devono avere un' ampia visione del piano dati.
Il concetto emergente di gestione-controllo afferma che non esistono criteri di selezione per distinguere la gestione dal controllo, se non per una questione di prospettiva.
Per usi futuri, il termine controllo è raccomandato, perchè meglio incarna l' idea del continuo responso di feedback e di ottimizzazione in tempo reale. Entrambi i termini rimangono comuni e usati l' uno a discapito dell' altro a seconda del contesto in cui sono posti.

4.1.4 Domini

Un dominio è costituito da un gruppo di entità accomunate da un certo insieme di caratteristiche. I domini possono diventare più piccoli o più grandi nel tempo, in base alle entità candidate a farne parte in base ai correnti criteri di selezione. Per SDN il dominio si riferisce al gruppo di risorse controllate da un controller, mentre per NFV il dominio viene spesso usato per includere sia il piano dati che le funzioni di gestione e controllo.

4.2 Introduzione

La Network Function Virtualization (NFV) è il processo di virtualizzazione delle funzionalità di rete svolte da apparati di telecomunicazione fisici. I maggiori vantaggi che l'operatore può trarre dall'utilizzo della NFV derivano sostanzialmente dal fatto di non essere più vincolati all hardware (switch, server, apparati di storage, eccetera) necessario per introdurre nuovi servizi i rete.
Questo, naturalmente, comporta tutta una serie di benefici in termini economici (connessi all' acquisto di nuovi apparati, al consumo di energia elettrica e alla loro manutenzione specializzata), di riduzione del time-to-market grazie al processo di virtualizzazione e dello spazio fisico necessario per l' alloggiamento dei dispositivi.
Grazie alla NFV è possibile assicurare una rete più flessibile e affidabile, potendo spostare le Virtual Functions (VF) da un server fisico ad un altro, oltre a fornire servizi personalizzati agli utenti semplicemente rimodulando il software di gestione degli apparati. [2]

4.3 Componenti

Il framework NFV consiste di tre macro componenti.

1. VNFs (Virtualized Network Functions) che sono le implementazioni delle funzioni di rete che sviluppano una NFVI (Network Functions Virtualization Infrastructure) sopra una funzione di rete.

2. NFVI (NFV Infrastructure) rappresenta la totalità dei componenti hardware e software che construiscono l' ambiente su cui vengono sviluppate le VNFs. L' infrastruttura può consistere di piu locazioni. La connettività di rete tra queste locazioni viene considerata come parte di essa.

3. NFV-MANO (NFV Management and Orchestration), una collezione di blocchi funzionali, depositi dati, punti di referenza e interfacce attraverso le quali questi blocchi funzionali possono scambiarsi informazioni e orchestrare le NFVI e le VNFs.

Il blocco funzionale sia di NFVI che di NFV-MANO è la piattaforma NFV, la quale implementa degli elementi usati per gestire e monitorare i componenti della piattaforma, eseguire ripristini in caso di errori e fornire sicurezza all' interno della rete. [7]

4.4 Relazioni e differenze tra SDN e NFV

4.4.1 Concetti chiave

SDN è un concetto legato a NFV, la differenza è che le due architetture si riferiscono a due domini differenti. La proposta comune è quella di trasmettere informazioni tra entità arbitrarie. Il traffico può essere scartato, conservato, sostituito o modificato. SDN si focalizza specialmente sull' uso delle risorse per fornire servizi al consumatore. NFV si focalizza sulla creazione e sul ciclo di vita della gestione delle risorse del software, e sulla sua configurazione per poi venir usato oltre il dominio di VNF. Per SDN, l' idea di risorsa è generica: principalmente, qualunque cosa che possa costruire un qualsiasi tipo di servizio

può essere inclusa nello scopo. Alcuni sottogruppi di risorse utilizzate da SDN possono essere fornite da NFV, il cui ruolo è basato sulla riduzione del tempo e del costo per fornire una certa classe di servizi di risorse. Rispetto ad un controller SDN, una VNF è solo un' altra risorsa, una funzione nodo capace di risiedere in ogni punto adeguato dell' infrastruttura. L' agilità del server SDN è rafforzata dall' abilità di NFV di creare, scalare o ricollocare una risorsa virtuale.Le due parti operano in modo coordinato, in quanto SDN non può usare risorse che non conosce, e NFV non dovrebbe distruggere le risorse che sono in uso. [6]

4.4.2 Operazioni

La sinergia delle soluzioni SDN ed NFV, permette alla rete di raggiungere le migliori performance. Infatti, SDN fornisce a NFV i vantaggi di una connessione programmabile tra le funzioni di rete virtualizzate; NFV, invece, mette a disposizione di SDN la possibilità di implementare le funzioni di rete tramite software su server COTS (Commercial Off-The-Shelf). Si ha, così, la possibilità di virtualizzare il controller implementandolo su un cloud che può essere facilmente fatto emigrare in qualsiasi altra posizione in base alle esigenze della rete. Il controller orchestra le risorse di rete attraverso risorse non NFV e con le VNFs, o opzionalmente con i VNF Network Services (VNF-NSs). Il controller costruisce le risorse selezionate nel servizio end-to-end.
Mentre un controller deve eseguire e configurare parametri operativi specifici, NFV si concentra sul mantenimento del ciclo di vita dell' inizializzazione delle risorse con valori di attributi globalmente validi. Un controller ha bisogno normalmente di interagire con altri domini, garantendo un servizio di monitoraggio appropriato. Questo significa che gli attributi di scambio devono essere conosciuti, per esempio tramite un accordo. [6]

4.4.3 Sviluppo combinato

Se esistono molteplici e distinti domini SDN e/o NFV, sarà importante assicurare che le loro attività non lavorino a scopi incrociati. Per ogni evento è richiesto un grande accordo d' informazioni tra i due domini, e per questo sarà necessaria la creazione di un framework che permetterà ai due domini di scambiarsi informazioni d' interesse comune tramite delle notifiche. Per la gestione del ciclo di vita delle VNF, NFV ha bisogno di informazioni preventive riguardo l' istanziamento, presumibilmente includendo informazioni di inizializzazione delle porte. SDN ha bisogno anche di conoscere le funzionalità delle VNFs disponibili o potenziali, di come connetterle a servizi del piano dati e di come accedervi per controlarle. Devono essere inoltre coordinati gli aggiornamenti alla connettività delle VNF. I domini NFV e SDN devono coordinare le loro richieste verso le risorse condivise, anche dinamicamente. La figura 18 mostra come i controllers possano essere sia client che server di un dominio NFV (mostrato in color oro). In cima alla figura ci sono le richieste di alto livello del client verso il controller di ordine n+2, il quale provvede a fornire le risorse richieste dal client attraverso le proprie risorse disponibili, le quali possono essere a loro volta delle VNFs.
Supponiamo che però il controller n+2 non possegga le risorse necessarie per soddisfare la richiesta del client. In questo caso effetuerà una richiesta a catena al controller n+1, e così via in maniera ricorsiva, fino al reperimento delle risorse

necessarie a soddisfare il servizio richiesto.
A qualsiasi livello di astrazione un controller può invocare operazioni da NFV per creare o scalare le risorse necessarie. L' interconnessione delle risorse implica che il manager NFV possa reciprocamente invocare il controllo SDN ad un livello meno astratto. [6]

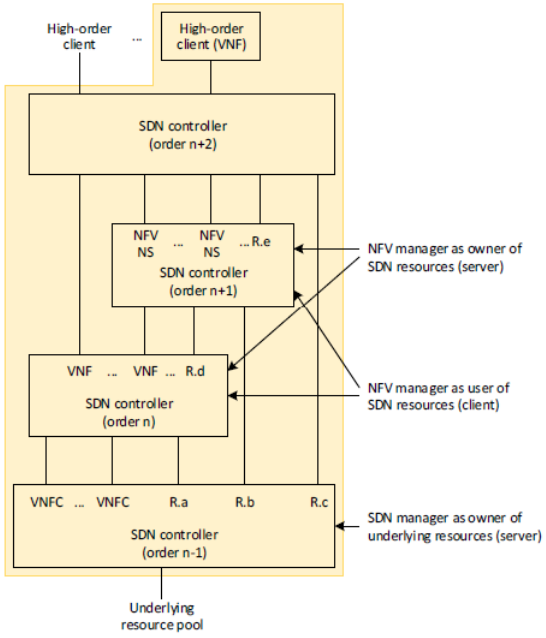

Figura 18: Relazioni client-server complesse

4.5 Lavoro ulteriore

Questo paragrafo identifica un numero di aree nelle quali SDN e NFV eseguono le stesse funzioni o sono nella posizione di offrire servizi l' uno verso l' altro. Essi devono inevitabilmente condividere un gran numero d' informazioni comuni, sia statiche che dinamiche. Sia per quanto riguarda l' allineamento dei modelli, è importante essere d' accordo su un certo tipo di argomenti, come la sicurezza, i diritti condivisi di scrittura e se necessario anche i meccanismi di pubblicazione-sottoscrizione.
Questi obiettivi possono essere raggiunti in modo collaborativo. Può essere possibile estendere alcune aree come la gestione del ciclo di vita di SDN.
NFV si focalizza sull' installazione del software eseguibile, e ha senso che SDN utilizzi i suoi strumenti in questo contesto. Comunque gli stessi strumenti possono essere utilizzati per instanziare risorse generali.
Se i concetti e gli strumenti di NFV venissero direttamente applicati al controller o ad altre parti del suo dominio, questo potrebbe aggiungere ulteriore valore.
Se NFV e SDN fossero implementati come domini separati ma collaborativi, sarebbe necessario formalizzare delle interfacce attraverso le quali poter invocare

l' uno i servizi dell' altro e viceversa.
Tuttavia sfruttando i concetti di SDN per implementare e gestire un infrastruttura NFV si ottengono grossi benefici, e proprio per questo che molte piattaforme fanno un uso incorporato di entrambi i sistemi. [6]

5 Riferimenti bibliografici

[1] Open Networking Foundation, SDN Architecture, TR-521, 2016, https://www.opennetworking.org/sdn-resources/technical-library#tech-spec.

[2] TransTec Services, SDN e NFV: il nuovo concetto di architettura di rete, http://www.transtecservices.com/index.php/news/28-news/focus/363-sdn-e-nfv-il-nuovoconcetto-di-architettura-di-rete

[3] Open Networking Foundation, SDN Architecture Overview, TR-5034, November 2014, https://www.opennetworking.org/sdn-resources/technical-library#tech-spec

[4] Open Networking Foundation, SDN Architecture for Transport Networks, TR-522, March 2016, https://www.opennetworking.org/sdn-resources/technical-library#tech-spec

[5] Open Networking Foundation, Intent NBI: Definition and principles, TR-523, October 2016, https://www.opennetworking.org/sdn-resources/technical-library#tech-spec

[6] Open Networking Foundation, Relationship of SDN and NFV, TR-518, October 2015, https://www.opennetworking.org/sdn-resources/technical-library#tech-spec

[7] Wikipedia, NFV: Network Function Virtualization, https://en.wikipedia.org/wiki/Network_function_virtualization