ALMA MATER STUDIORUM - UNIVERSITÁ DI BOLOGNA
SEDE DI CESENA

SECONDA FACOLTÁ DI INGEGNERIA CON SEDE A CESENA
Corso di Laurea in Ingegneria Informatica, Elettronica e Telecomunicazioni

GESTIONE INTEGRATA DELLE RISORSE APPLICATIVE E DI RETE IN DATA CENTER DISTRIBUITI

Elaborato in
RETI DI CALCOLATORI

Relatore:
Prof. Ing.
FRANCO CALLEGATI

Correlatore:
Ing.
WALTER CERRONI

Presentata da:
FABIO PAOLUCCI

Sessione II
Anno Accademico 2011-2012

Ai miei genitori
la mia forza...
Ai miei amici
il mio sorriso...

Indice

Elenco delle figure

Introduzione

In un mondo in cui comunicare e scambiare informazioni risulta la base di ogni forma di processo umano o disciplina sviluppabile e in cui la tecnologia è giunta a livelli evolutivi fino a pochi anni fa inimmaginabili, continuando attualmente nella sua crescita esponenziale, l'integrazione tra questi ambiti risulta ormai da tempo teorizzata e sintetizzata nel concetto di Information and Communication Tecnology (ICT).

L'uso della tecnologia nella gestione e nel trattamento delle informazioni oggigiorno è qualcosa di oramai scontato, parte di un bisogno di cui è difficile fare a meno: la possibilità di comunicare e condividere dati in ogni istante, con chiunque e in qualunque parte del mondo esso sia situato.

Qualsiasi utente di livello medio, è in grado di usufruire facilmente e costantemente di tutto ciò, tramite i mezzi sempre più avanzati che vengono messi a disposizione ogni anno, di cui esempi recenti in tablet e smartphone. La trasmissione di informazioni tra calcolatori connessi in rete rappresenta, quindi, un aspetto di un fenomeno più generale, di grande portata pratica e concettuale: la progressiva convergenza tra informatica e telecomunicazioni.

Se un tempo, infatti, la descrizione schematica delle reti di calcolatori poteva essere descritta tramite una netta separazione tra due concetti, da una parte terminali e servizi applicativi collegati a ciò che vien definito il *cloud* e rappresentato dai collegamenti e i nodi di comunicazione Internet, attualmente la tendenza che sta avendo larga crescita consiste in un mutamento dello schema classico, con lo spostamento dei servizi stessi da una posizione di terminale ad un loro assorbimento all'interno della nuvola stessa. Il cloud-computing rappresenta un significativo esempio in questo senso, un nuovo approccio nella fornitura di risorse, spazio di archiviazione e capacità computazionali sotto forma di servizi di rete.

In questo contesto, si avverte sempre più l'esigenza di poter integrare e gestire

servizi e risorse all'interno di scenari distribuiti in maniera sempre più dinamica ed eterogenea, cercando di abbattere le barriere che fino ad oggi hanno stabilito una separazione concettuale tra mondo applicativo e infrastrutture di rete.

Tutto questo, si riassume nella necessità di creare e studiare nuovi elementi, verso un concetto di rete che si occupa non solo di fornire un servizio di trasposto e instradamento di pacchetti, ma in grado di localizzare gestire e fornire all'utente finale ciò di cui ha bisogno, scaricando sulla rete stessa il compito di scegliere la risorsa disponibile più adatta, in termini di performance, stabilità e sicurezza del contesto di utilizzo.

I servizi di elaborazione, archiviazione e comunicazione del futuro saranno, quindi, fortemente pervasivi: persone e macchine conviveranno all'interno di un cyber-ambiente fortemente decentralizzato, composto da risorse connesse tramite reti cognitive altamente dinamiche.

L'idea e l'obiettivo di questa tesi è quello di portar avanti un approccio basato sul modello Transport Service Layer, già da tempo oggetto di studio e di ricerca all'interno del più ampio ambito legato alle architetture di rete di nuova generazione (NGN), e di definire ed analizzare le possibilità di interfacciamento e utilizzo di quest'ultimo in ambiente Openstack, una architettura di cloud computing open-source dalle prospettive future all'avanguardia, fonte di interesse di molte industrie del settore e scelta come riferimento all'interno del progetto universitario del Centro Interdipartimentale di Ricerca Industriale sul'ICT (CIRI ICT) dell'Università di Bologna.

Nel primo capitolo verranno descritte le basi concettuali con cui è possibile affrontare un approccio al problema della gestione automatica di risorse ai diversi livelli logici di rete, con riferimento a scenari di cognitive network in reti di prossima generazione. Nel secondo capitolo verrà discussa l'architettura di cloud-computing di riferimento all'interno del progetto CIRI ICT, concentrandosi sulle caratteristiche e proprietà di quest'ultima, che permettono di mappare a livello configurativo gli elementi presentati nel primo capitolo. Infine, nel terzo ed ultimo capitolo, verrà presentata una documentazione del lavoro svolto nel configurare il cloud, in modo tale da rendere possibile un'interazione ed un interfacciamento con esso da parte dei moduli del Cognitive Transport Service, aprendo, quindi, la strade per l'impostazione di una successiva fase di testbed.

Capitolo 1

Cognitive Network di Prossima Generazione

L' attuale modello Internet si fonda su un' idea di base molto semplice che ne ha determinato il grande successo, soppiantando il modello più completo e articolato ISO/OSI: definire in modo chiaro e sintetico tutte le funzionalità e le primitive di connessione fino al layer di trasporto e lasciare tutto il resto alla gestione applicativa.

Il compito della rete essenzialmente si riduce alla gestione di tutto ciò che riguarda il traffico, l'affidabilità e l'instradamento di pacchetti dati, ignorando completamente la semantica di questi ultimi. Tutto questo, si adatta in maniera naturale ai principi di scalabilità e trasparenza tutt'oggi alla base dei sistemi distribuiti, scaricando però un gran carico di complessità a livello degli end-point.

Lo sviluppo tecnologico e l'impegno di ricerca portato avanti fino ad oggi hanno portato a sistemi applicativi sempre più complessi, la cui flessibilità può essere maggiormente garantita proponendo un' evoluzione della rete da un modello essenzialmente statico a un modello sempre più pervasivo, ovvero una rete in grado di fornire dei servizi di accesso alle risorse e di interagire e rispondere alle richieste del livello applicativo, riconfigurandosi al meglio in base al compito da soddisfare.

Nelle prossime sezioni verranno brevemente descritti gli elementi concettuali e l'approccio principale di riferimento su cui è possibile sviluppare un modello di Cognitive-Network alla base del lavoro di tesi, procedendo con una breve descrizione dei moduli e protocolli necessari allo sviluppo di un' architettura di rete service-aware[2].

1.1 Next-Generation Network (NGN)

Una modello di riferimento verso la gestione integrata di trasporto e risorse applicative può essere facilmente ritrovato nelle specifiche di una architettura di Next-Generation-Network (NGN), una cui sintetica e chiara descrizione è data dalla definizione formale fornita dalla ITU-T [8]:

> *A Next Generation Networks (NGN) is a packet-based network able to provide Telecommunication Services to users and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent of the underlying transport-related technologies. It enables unfettered access for users to networks and to competing service providers and services of their choice. It supports generalised mobility which will allow consistent and ubiquitous provision of services to users.*

Una NGN propone quindi un'evoluzione verso una rete integrata, in cui vi è una netta separazione tra il concetto di trasporto e quello di servizio stesso, che si mappa in due livelli logici separati:

- **Transport Stratum**: ha il compito di fornire e garantire la connettività IP e il forwarding tra gli apparati che partecipano alla rete.
- **Service Stratum**: offre i servizi di rete necessari all'accesso delle risorse multimediali e al controllo di sessione.

Questo significa che nel momento in cui un provider vuole definire un nuovo servizio, ciò è possibile in maniera completamente indipendente dal gestore della rete e dai dettagli legati al trasporto [9].

1.2 Verso una Cognitive Network

Mantenendo il riferimento alla suddivisione concettuale proposta nella sezione precedente è possibile fare un ulteriore analisi sui nuovi compiti che la rete deve essere in grado di assorbire per ottimizzare l'uso delle risorse applicative e di comunicazione. L' Internet del futuro si propone di integrare una forma di *knowledge plane*

[1] all'interno della rete, che quindi non si limita più alla semplice funzione di forwarding dei dati, ma è in grado di svolgere compiti attivi e collaborativi con le applicazioni stesse. Ciò sottintende una rete in grado di autogestirsi e autoconfigurarsi nel migliore dei modi per supportare la creazione, archiviazione e propagazione di informazioni e allo stesso tempo in grado di garantire una QoS (Quality of Service) nei confronti dell'utente finale.

La chiave di tutto ciò risiede in tre concetti principali:

- **Osservazioni**: capacità di interagire con i layer sottostanti ed ottenere una conoscenza dello stato attuale della configurazione di rete.
- **Asserzioni**: capacità di catturare gli obiettivi e le richieste delle applicazioni di alto livello.
- **Spiegazioni**: unisce osservazioni e asserzioni, comunica e riconfigura i vari nodi della rete nella maniera più appropiata, localizza e contatta la migliore risorsa disponibile in quel momento, costruendo un canale di accesso stabile e sicuro verso l'applicazione.

É opportuno, fare riferimento alla seguente chiara ed esaustiva definizione di cognitive network [3]:

> *A cognitive network is a network composed of elements that, through learning and reasoning, dynamically adapt to varying dynamical conditions in order to optimize end-to-end performances.*

Se un nodo di una cognitive network possiede un certo livello di conoscenza dello stato complessivo della rete, la capacità di far fronte a un servizio di qualità risulta superiore, rispetto ad uno scenario contrario.

Per le applicazioni, si apre, quindi, la possibilità di ottenere un semplice controllo e utilizzo globale di qualsiasi risorsa messa a disposizione, tramite un *setup on demand* dei servizi di prossima generazione, conforme alle specifiche dell'attività che intendono svolgere.

Tutto ciò, apre la strada allo sviluppo di nuovi ambienti assai dinamici, fino al momento non realizzabili senza dover scendere direttamente a riconfigurazioni manuali di basso livello.

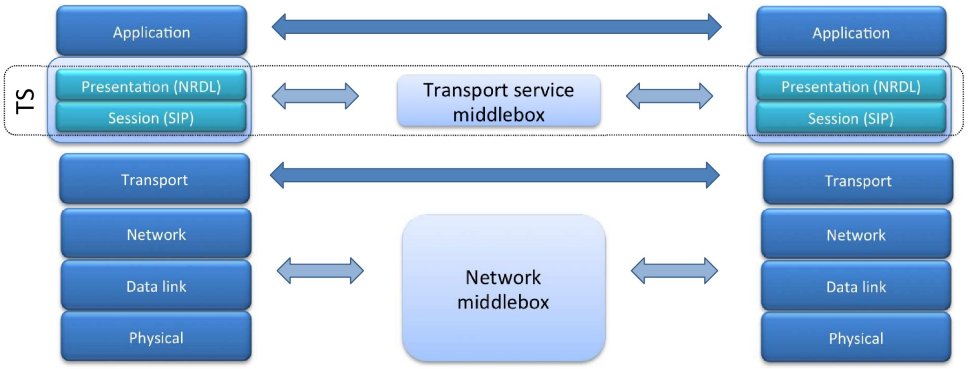

Figura 1.1: Modello di riferimento per il *TS Layer* [5]

1.3 Un approccio di riferimento: TS Layer

Rimanendo nell'ottica precedente, l'approccio di riferimento verso una Cognitive Next-Generation Internet su cui intende basarsi questa tesi e che da alcuni anni è stato intrapreso a livello di ricerca, consiste nello sviluppo di un di un Cognitive Transport Service [4].

L' obiettivo di base che ci si propone è di agire sull'infrastruttura di rete in maniera tale che sia possibile fornire un supporto alle applicazioni, nella ricerca e allocazione di risorse applicative, con opportuno setup dei servizi di connettività e trasporto, il tutto fornendo un servizio ad alto livello che permetta alle applicazioni stesse di potersi esprimere in maniera diretta, tramite i parametri da esse percepiti.

Per rendere possibile uno scenario di questo genere, è necessario integrare l'attuale layer di trasporto TCP/IP con la definizione di un nuovo Transport Service Layer (vedi Figura 1.1), in quanto l'attuale servizio di trasporto non ha capacità di interagire direttamente con i servizi di rete sottostanti e quindi di instradare i pacchetti applicativi sulla base di ciò che Internet offre e le applicazioni richiedono in un determinato momento. Occorre, quindi, sviluppare una infrastruttura di segnalazione in grado garantire un' autoconfigurazione e autogestione dei livelli di rete e che sia in grado di:

1. Dare alle applicazioni gli strumenti necessari per scambiare messaggi semanticamente ricchi con la rete, per una negoziazione e rilevazione automatica delle risorse.

2. Definire un linguaggio sufficientemente completo e indipendente dalla tecnologia e topologia di rete che permetta di esprimere i requisiti delle applicazioni.

3. Definire una metodologia a sua volta indipendente dalla tecnologia che permetta di mappare le richieste di servizi in direttive di controllo della rete.

Dagli studi eseguiti fino ad ora è risultata una scelta vincente definire una organizzazione implementativa del TS Layer [5] cercando di disaccoppiare l'utilizzo di un protocollo di segnalazione, necessario per definire la sintassi utilizzata per la negoziazione dei servizi, dall' utilizzo di un protocollo in grado di descrivere la semantica delle caratteristiche delle risorse applicative e di rete. Per questo motivo il TS Layer può essere ulteriormente suddiviso a livello logico in due substrati:

- **Session Layer**: è strettamente incentrato sull'implementazione di meccanismi di segnalazione session-oriented, dove per sessione si intende un insieme di attività e flussi di dati logicamente correlati che prevedono diversi scambi di informazioni, connessioni che mantengono lo stato della comunicazione end-to-end.

- **Presentation Layer**: è legato alle funzioni cognitive del servizio che si vuol ottenere, ovvero garantisce il supporto necessario affinché l'applicazione possa aver una conoscenza di ciò di cui ha bisogno per ottenere un determinato servizio, quale genere di comunicazione o connessione fornisce la rete sottostante in quell'istante.

In riferimento alle tecnologie affermate, viene proposto l'uso del protocollo di segnalazione SIP(Session Initiated Protocol) per il Session Layer e l'uso del Network Resource Description Language (NRDL) per il Presentation Layer, le cui caratteristiche implementative verranno successivamente brevemente esposte, in quanto non è obiettivo principale di questa tesi soffermarsi sull'approfondimento di questi temi.

1.3.1 Session Initiation Protocol (SIP)

SIP è un protocollo session-oriented sviluppato a partire dal 1999 per iniziativa di IETF (Internet Engineering Task Force) e standardizzato attualmente nel RFC

3261 [6]. Il SIP non è in se un fornitore di servizi, ma piuttosto specifica un insieme di primitive standard tramite cui è possibile organizzare un flusso di messaggi tra end-point (User Agent) che consente di dare inizio, modificare o terminare una sessione, quindi utilizzabile per l' instaurazione di un dialogo interattivo tra i servizi sovrastanti. Si tratta di un protocollo basato sul modello richiesta-risposta e si fonda sui concetti cardine di transazione,dialogo e sessione. SIP è, inoltre, in grado di lavorare in maniera completamente indipendente dalla tipologia di sessione che gestisce e dal particolare protocollo di trasposto utilizzato (TCP,UDP etc.), quindi è adattabile all'interno di scenari assai dinamici. La scelta del SIP è stata guidata dal bisogno di ottenere un protocollo solido anche dal punto di vista della sicurezza in riferimento alle specifiche AAA (authentication, authorization and accounting) e in grado di incapsulare e trasportare del payload nel corpo dei propri messaggi di segnalazione, caratteristica che lo rende facilmente estendibile semanticamente tramite il linguaggio NRDL.

1.3.2 Network Resource Description Language (NRDL)

Il Network Resource Description Language (NRDL) [7] è un linguaggio descrittivo, estensione del Network Description Language (NDL) e si pone l'obiettivo di descrivere informazioni e risorse di rete nella maniera più flessibile possibile, in quanto deve far fronte alla loro riservazione tramite lo scambio applicativo di richieste-risposte, neutrale rispetto al protocollo di sessione che ne incapsula e trasporta i documenti. NRDL è in grado, quindi, di rendere le applicazioni capaci di descrivere e i requisiti caratteristici di un certo servizio, in un formato che la rete poi è in grado di interpretare e tradurre in primitive di segnalazione. Si propone, inoltre, come un linguaggio con grandi margini di espressività, che presenta una sintassi generica, sulla quale è possibile costruire messaggi con alto valore semantico. Strutturalmente si rifà al modello XML, da cui trae le basi, estendendolo con il concetto di vocabolario RDF [10].

1.3.3 Application-Oriented Module (AO-M)

Il TS Layer, grazie all'integrazione tra SIP e NRDL, fornisce le funzionalità necessarie a sostenere un concetto di Internet conoscitivo in termini prettamente generali. A livello pratico, viene implementato tramite l'introduzione di un nuovo

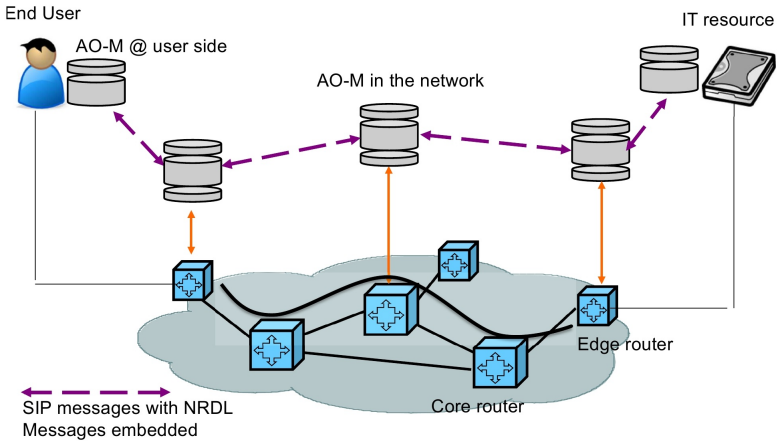

Figura 1.2: Implementazione del TS Layer tramite *modulo AO-M* [5]

modulo applicativo che prende il nome di Application-Oriented Module. Quest' ultimo deve essere distribuito tra gli end-users e i nodi della rete(vedi Figura 1.2), integrando tutti i concetti esposti fino ad ora. Il AO-M costituisce l'entità software che conferisce effettivamente alla rete la capacità di essere application-aware e può essere suddiviso, a sua volta, in tre moduli distinti:

- **Modulo Applicativo (APP-M)**: è rappresentato da un parser NRDL, in grado di leggere e capire i contenuti di presentazione incorporati all'interno dei messaggi SIP e capace di interagire con un database, in cui vengono archiviate le informazioni relative allo stato attuale delle risorse applicative e delle funzionalità di rete.

- **Modulo SIP (SIP-M)**: è un proxy SIP che fornisce le interfacce di interazione con i moduli sovrastanti e in cui risiedono le logiche di segnalazione SIP, responsabili della gestione delle sessioni e dell'instradamento dei messaggi verso la loro destinazione finale.

- **Modulo di rete (NET-M)**: è un modulo strettamente *network-dependent* in grado di configurare la rete fisica sottostante, in accordo con le specifiche richieste dai servizi.

1.4 Principi di funzionamento

Il processo di invio e ricezione dei dati in una infrastruttura che implementa il CTS layer risulta essenzialmente lo stesso del modello Internet attuale, estendendo però quest'ultimo con la possibilità di richiedere l'insieme delle risorse e dei servizi di trasporto, in accordo con le preferenze dell'utente finale.

Grazie a NRDL gli utenti applicativi possono generare proposte e domande in formato user-oriented, generando un' astrazione di alto livello di ciò che la rete rappresenta in termini di risorse e, allo stesso tempo, rimanendo all'oscuro delle caratteristiche topologiche e tecniche della rete stessa.

Grazie al SIP viene sempre garantita una gestione consistente della comunicazione e della consegna delle informazioni avendo, inoltre, un forte supporto in termini di pubblicazione, scoperta e riservazione delle risorse sparse per la rete.

Un'applicazione utente può facilmente registrarsi su un server SIP per essere notificata in caso di pubblicazione di determinate risorse o può richiederne la riservazione direttamente. La richiesta di un servizio prevede le seguenti operazioni (vedi Figura 1.3) da parte del TS layer e in particolare del modulo AO-M:

- Preparazione di un documento NRDL che descrive le informazioni necessarie all' attivazione di un servizio, in accordo con i bisogni applicativi.
- Avvio di una sessione SIP da parte di AO-M mittente verso un destinatario (tramite il SIP-M), e invio del contenuto NRDL incorporato insieme ai messaggi SIP di segnalazione. Il messaggio NRDL attraversa vari nodi della rete cognitiva, i cui AO-M sono in grado di aggiungere o modificare dinamicamente le informazioni riguardanti lo stato e la capacità attuale della rete.
- Conferma dell'avvio di una sessione da parte di AO-M destinatario e preparazione di una risposta sempre come documento NRDL, che ha scopo sia di acknowledge sia di descrivere il servizio di rete dal punto di vista del destinatario.
- Inizio di un dialogo che continua fino a quando il servizio richiesto è completamente instaurato o abortito, con la negoziazione delle risorse e delle funzionalità richieste dal mittente, facendo un attento riferimento anche alle informazioni di configurazione della rete, raccolte durante il tragitto.

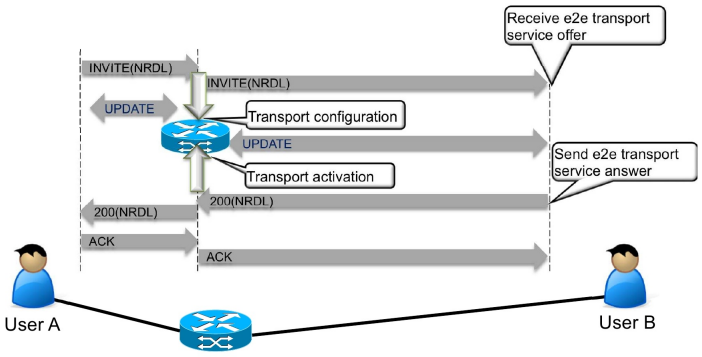

Figura 1.3: Schema logico di funzionamento del *Cognitive Transport Service* [5]

1.5 Scenario applicativo

Servendosi dell'aiuto di un esempio applicativo, si presenterà ora una breve descrizione di un possibile scenario di utilizzo di un servizio fondato su Cognitive Transport Service [4], riferendosi in particolare ad un ambiente cloud-oriented, in quanto l'obiettivo su cui intende svilupparsi questa tesi è analizzare le possibilità di interfacciamento del TS Layer all'interno di un architettura di cloud. L'esempio comune che possiamo prendere in considerazione fa riferimento a una situazione classica, in cui un utente si appresta alla visione di un video streaming (e.g. Youtube) . Supponiamo di porci in uno scenario ormai comune, in cui il server sorgente sia virtualizzato all'interno di un cluster sparso nella rete. Questa situazione, nonostante si presenti a livello utente molto banale, in realtà nasconde molte considerazioni che possono essere effettuate nei confronti dell' allocazione, riservazione e gestione ottimale delle risorse applicative e di rete, che possiamo porre in analisi nei punti successivi:

- Conoscenza dell'indirizzo del server a cui connettersi, ovvero quello che al momento attuale può garantire una maggiore efficienza e qualità di accesso alle risorse.
- Configurazione della rete in maniera tale che la qualità video sia garantita all'utente, anche in caso di congestione della rete stessa.
- Gestione delle politiche di bilanciamento del carico, che potrebbero portare a una migrazione della risorsa nel momento del suo utilizzo, garantendo il più possibile la trasparenza nei confronti dell'utente.

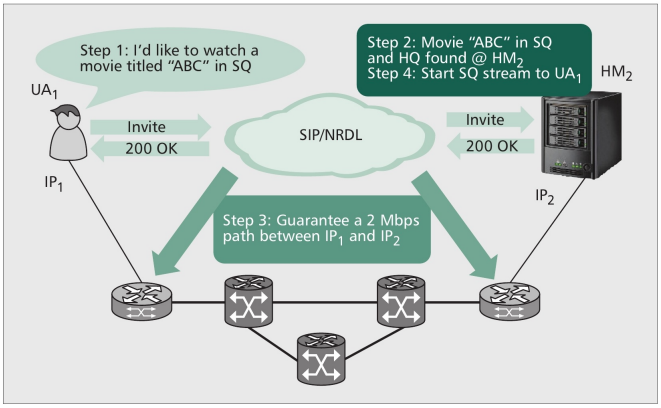

Figura 1.4: Scenario di allocazione risorsa tramite *Cognitive Transport Service* [4]

Tutto ciò, fa emergere facilmente i limiti che l' attuale modello Internet trova difficoltà a superare, in quanto risulta altamente complicato garantire i requisiti del servizio di comunicazione descritto, senza un intervento di configurazione manuale, o con grande appesantimento della parte applicativa. L'inserimento del TS layer, può risultare, invece, un valido strumento di supporto, capace di risolvere agilmente buona parte di questi problemi.

Nel momento in cui l'applicativo lato utente esprime, attraverso un linguaggio user-oriented, la propria necessità di voler ottenere una risorsa, uno streaming audio-video, l'infrastruttura di CTS fondata sull'integrazione di SIP e NRDL e, quindi, la rete stessa, è in grado di localizzare in modo completamente automatico il miglior server di host (HM) disponibile su cui si trova una virtual machine (VM) sorgente di video. L' utente non ha più bisogno di specificare il server a cui intende connettersi, in quanto è la rete cognitiva che assicura l' accesso alla risorsa, configurando automaticamente la rete fisica per fornire la banda di trasferimento minima a garantire la qualità di visualizzazione richiesta (vedi Figura 1.4).

A questo punto, supponiamo che mentre l'utente stia procedendo alla visione del video, il gestore di risorse automatico del sistema (ARM[1]), si accorga, secondo una opportuna politica di bilanciamento del carico e delle connessioni, che sia più conveniente migrare la VM con la sorgente video in una locazione fisica differente all'interno del data center. Anche in questo caso, l'ARM può comunicare

[1]Automatic Resource Manager: amministratore automatico e intelligente che si occupa della gestione, allocazione e ottimizzazione dinamica delle risorse (VM) del sistema in base alle disponibilità fisiche di quest'ultimo.

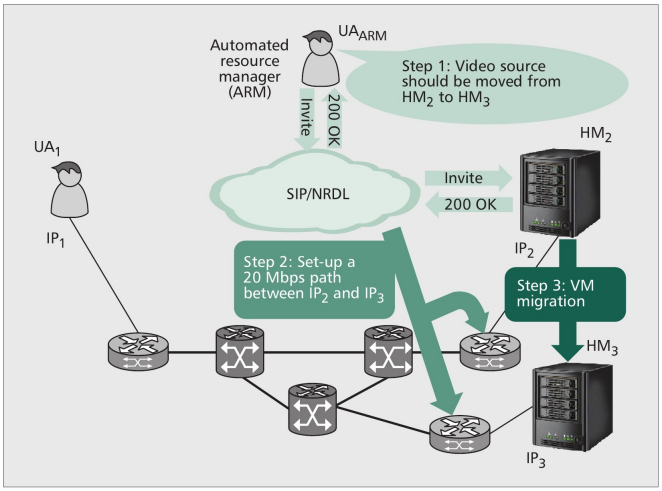

Figura 1.5: Scenario di migrazione tramite *Cognitive Transport Service* [4]

le proprie intenzioni interagendo con il TS layer (vedi Figura 1.5), che procederà nuovamente ad attuare le configurazioni di rete necessarie affinché la migrazione della sorgente abbia il minimo impatto sulla visione del video da parte dell'utente (i.e. configurando un canale con una banda garantita, su cui instradare il traffico di migrazione).

La generalizzazione del concetto di risorsa proposta dall' approccio cognitivo della rete permette, quindi, come descritto, di adattare facilmente il modulo Cognitive Transport Service a diversi scenari di utilizzo, determinandone una grande flessibilità.

1.6 Prospettive

Affinchè il modello di ricerca, qui brevemente descritto, possa affermarsi con successo nello sviluppo di un Internet del futuro, è necessario affrontare e impostare un'attenta attività di testbed, il cui obiettivo è mirato a sperimentare il nuovo modulo TS Layer all'interno degli svariati scenari dinamici ed innovativi che oggi la tecnologia ci propone, verificandone i punti di forza e i limiti, atti ad ottenere una conferma di stabilità e sostenibilità pratica del sistema ampiamente elaborato. Ciò che ci si propone di analizzare ed approfondire nella pagine successive consiste nello stringere il *focus* di riferimento, da i primi due punti architetturali su cui si fonda il TS Layer, presentati nelle precedenti sezioni (vedi Sezione 1.3 a pagina 4) e attualmente già implementati, al terzo punto considerato.

In particolare, ci si focalizzerà sulle possibilità di interfacciare e quindi integrare il Transoport Service Layer all'interno di un'architettura di cloud e capire come sia possibile da parte di quest'ultimo utilizzare le API messe a disposizione per la creazione, la modifica e la migrazione dell'infrastruttura di rete virtuale all'interno del data center.

Capitolo 2

Architettura e Tecnologia in Analisi

Considerando l' odierna tendenza all'abbandono di una tecnologia e di modelli essenzialmente statici, verso un concetto di richiesta-fornitura di servizi sempre più on-demand, risulta interessante effettuare una fase di Testbed del modello fondato su Cognitive Transport Service (vedi Sezione 1.3 a pagina 4), ponendoci all' interno di una architettura di cloud-computing. Quest'ultima sposa, infatti a sua volta, una nozione di allocazione di servizi su richiesta, che può essere unita al dinamismo garantito dalla virtualizzazione, a livello di risorse applicative e di rete.

In tale panorama, per essere in grado di studiare e capire come sviluppare una possibile integrazione tra questi modelli, risulta necessario concentrarsi su una tecnologia di riferimento. Il progetto CIRI ICT [15] dell'università di Bologna, ha scelto di prendere in analisi la piattaforma Openstack, la quale sposa il concetto di software open source, ben adattabile e personalizzabile a una fase sperimentale, oltre a presentare una serie di caratteristiche che ne permettono una sua facile integrazione al caso di studio del TS-Layer.

Verranno, quindi, introdotti i concetti principali dell'infrastruttura Openstack, a cui un maggior dettaglio è rimandato alla documentazione ufficiale [18], concentrandosi in particolare sul modulo Quantum, il progetto per la gestione dinamica di rete all'interno del cloud. Quest'ultimo è fonte di particolare interesse di questa tesi, in quanto risulta fondamentale capirne e approfondirne le caratteristiche di estendibilità e configurazione, necessarie ad introdurre una fase di testbed di Cognive Transport Network all'interno di data center distribuiti.

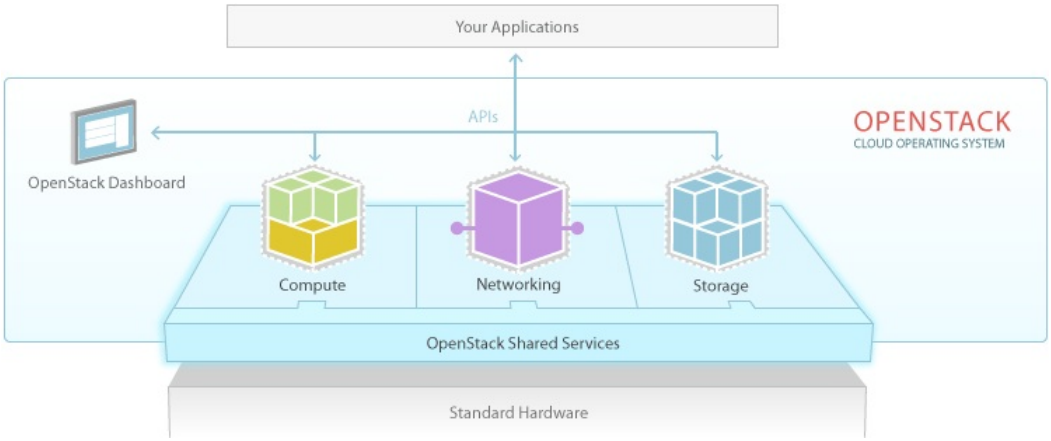

Figura 2.1: Diagramma dell' infrastruttura *Openstack* [17]

2.1 Infrastruttura di cloud: Openstack

Openstack [17] nasce ufficialmente il 19 luglio 2010 e la paternità è attribuibile sia alla Nasa, che ne ha fornito il cuore pulsante nel progetto Nova, sia dell'azienda di hosting Rackspace, con lo scopo di ottenere un servizio della stessa portata di Amazon Elastic Compute Cloud (EC2) ma rilasciato sotto licenza Apache, quindi con nessuna restrizione di utilizzo e implementazione. Si tratta di un' infrastruttura sofware distribuita all'interno di un data center, in grado di gestire tutte le funzionalità necessarie alla nascita di un ambiente di cloud-computing[1] completamente open source. É in grado di gestire grandi insiemi di risorse computazionali, di storage e di rete presenti nel datacenter, offrendo una facile intefaccia web utente (la dashboard) tramite cui si ha il controllo amministrativo dell'intero sistema e si ha la possibilità di allocare le risorse e i servizi necessari agli utenti. Openstack raccoglie, inoltre, una grande comunità di sviluppatori a livello globale, che collaborano per la stessa missione: un sistema operativo di cloud, semplice da implementare e allo stesso tempo altamente scalabile.

[1] Attualmente nei datacenter, molti computer soffrono di forte sottoutilizzo di potenza computazionale e di banda di comunicazione, dovuta ad una gestione non ottimizzata delle risorse. Il concetto di cloud-computing, racchiude in se la volontà di poter usufruire di un servizio on-demand direttamente dal browser, con una flessibilità tale da poter allocare e rilasciare le risorse necessarie (potenza di calcolo, storage, rete etc) in maniera completamente automizzata.

2.1.1 Componenti Principali

Tradizionalmente, Openstack può essere pensato come l' insieme dei componenti (vedi Figura 2.2 a pagina 16) distribuiti che verranno elencati successivamente, ognuno dei quali è specializzato nella gestione di particolari compiti all'interno del cloud, utilizzando come mezzo di interazione e scambio di conoscenza un database comune (MySql) e un servizio per la gestione della messaggistica (RabbitMQ).

Openstack Compute (Nova)

É responsabile della gestione degli host di Hypervisor, ovvero della configurazione, creazione, vita e distruzione delle macchine e risorse virtuali e della loro distribuzione all'interno del cluster fisico. Openstack è pensato per essere flessibile e altamente scalabile, di conseguenza questo modulo gode della possibilità di lavorare in maniera performante, con un ampio insieme di tecnologie di virtualizzazione disponibili (e.g. KVM, Xen, VMWare, vSphere e Hyper-V).

Openstack Image Service (Glance)

Si occupa della gestione del Virtual Machine Image Repository, ovvero di fornire tutti gli elementi per salvare, archiviare e accedere alle immagini delle macchine virtuali istanziabili all'interno del cloud. Più macchine virtuali possono essere istanziate come copie in esecuzione della stessa immagine salvata.

Openstack Object Storage (Swift)

É in grado di organizzare il cluster distribuito per l'immagazzinamento, l'aggiornamento e l'accesso di multi-petabyte di file di informazioni, tutto come se si trattasse di un unico sistema di storage. Deve, quindi, gestire particolari politiche di replicazione e consistenza per mantenere l'integrità, la sicurezza e la protezione dei dati distribuiti nel cloud. É utilizzato per esempio, a differenza di Glance per lo storage a lungo termine delle immagini delle VM.

Openstack Identity Service (Keystone)

Gestisce tutte le politiche di sicurezza e di autorizzazione per l'accesso degli utenti alla piattaforma tramite account, con diversi privilegi. Inoltre, è in grado di fornire

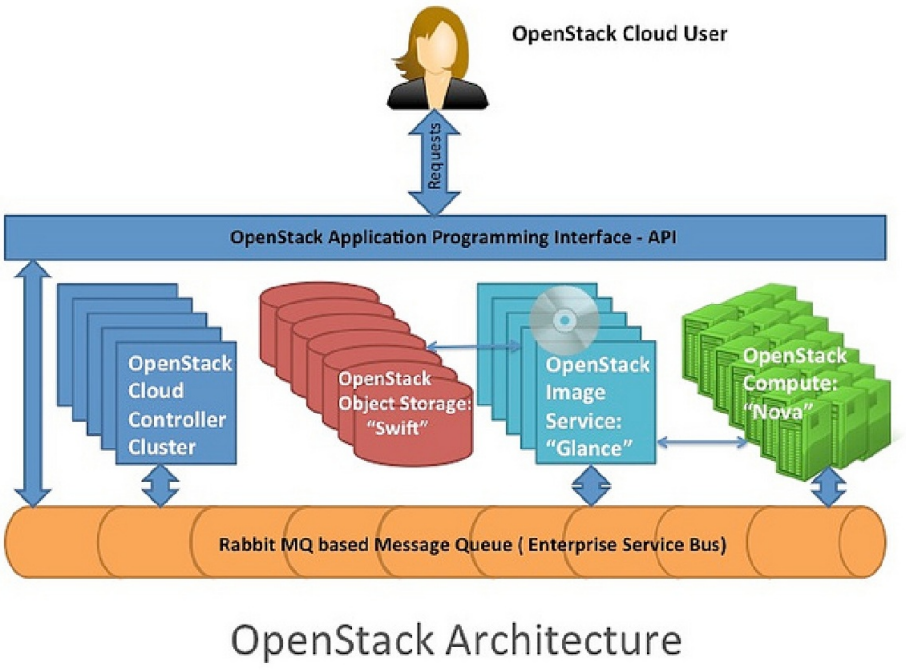

Figura 2.2: Diagramma dei servizi di *Openstack* [25]

i riferimenti a un particolare servizio all'interno del cloud tramite l'accesso via API, (modello pagine gialle).

Openstack Dashboard (Horizon)

É una semplice interfaccia grafica accessibile via web e che gira su un Apache web server di default, per una facile e intuitiva gestione delle risorse di cloud da parte degli utenti.

2.1.2 Architettura Logica

Openstack, come definito precedentemente, è un'architettura basata sullo scambio di messaggi (*RabbitMQ*) tra servizi distribuiti all' interno di un data center. Questo garantisce una grande flessibilità del sistema, in quanto è possibile installare ogni servizio su macchine tra loro indipendenti, aprendo molte scenari possibili di configurazione.

L'organizzazione logica del data center più adatta alla creazione di un ambiente di sviluppo e sperimentazione, consiste nel suddividere il sistema in due mansioni principali:

- **Cloud Controller**: generalmente un server con più interfacce di rete che fa da ponte tra la rete privata del cluster e la rete pubblica. Si occupa della funzione di *master*, ovvero della gestione dello stato globale del sistema, raccoglie, quindi, tutti i servizi di Openstack, eccetto opzionalmente nova-compute, e il database principale a cui questi fanno riferimento.
- **Compute Nodes**: più macchine all'interno del cluster con funzione di nodi computazionali e di storage, su cui installare nova-compute. Svolgono, essenzialmente, la funzione di *workers*, agenti che gestiscono le risorse disponibili localmente all'interno di ogni singolo nodo.

Openstack è, inoltre, pensato come un sistema altamente scalabile, utilizzabile da un grande numero di utenti ed amministratori con mansioni e risorse accessibili differenti. É possibile, quindi, suddividere i singoli partecipanti al sistema (*user*) in gruppi amministrativi (*tenants*), a ognuno dei quali vengono assegnati dei ruoli (*roles*) abilitandone o meno l'accesso ai servizi e alle risorse. I ruoli controllano le azioni che un uno user, identificato da *username* e *password*, può eseguire all'interno del sistema e all'interno del particolare gruppo di appartenenza.

2.1.3 Integrazione con Linux

Essendo una tecnologia completamente aderente al mondo open-source, Openstack è facilmente integrabile all'interno di Linux, un altro forte esponente di questa mentalità nel mondo dei sistemi operativi. In particolare, è installabile su Linux, nelle sue diverse distribuzioni (e.g. Ubuntu, Debian, Fedora, CentOS etc), sia tramite pacchetti sia da sorgenti. A questo punto, è fondamentale capire come vengono mappati i servizi di Openstack all'interno del sistema operativo. Nel mondo Linux, un servizio (detto anche demone) rappresenta un singolo programma eseguito in background che tipicamente rimane in eterno ascolto su una porta, in attesa di eseguire dei compiti e rispondere alle richieste che gli vengono fatte. Un servizio di Openstack, viene implementato come un insieme di demoni che lavorano e interagiscono tra loro, per rappresentare al meglio le caratteristiche che il servizio

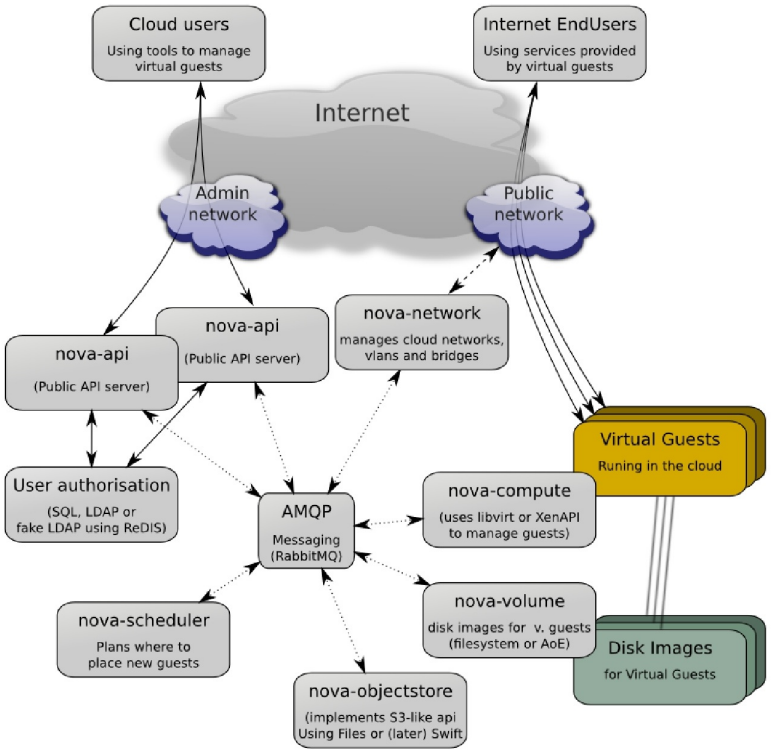

Figura 2.3: Diagramma riassuntivo dei servizi di *Openstack* e di *Linux* [22]

richiede. Quindi, possiamo raggruppare a livello logico un servizio di Openstack in più servizi di Linux, come viene riassunto in Tabella 2.1 a pagina 20. Infine, un buono schema integrativo di come si concretizza il sistema nel suo complesso all'interno del cluster di macchine Linux è visibile in Figura 2.3 a pagina 18.

2.2 Perchè Openstack?

Il progetto Openstack, per le caratteristiche illustrate, rispetta perfettamente i requisiti di un sistema distribuito moderno. La missione, comune a tutti i membri della community globale, consiste nel portare avanti un modello *open* di cloud-computing altamente scalabile e facilmente estendibile, a tal punto da suscitare forti interessi verso i principali colossi del settore (tra cui *Cisco*) mostrandosi come una possibile e valida alternativa su cui investire per contrapporsi al monopolio stabilito da *VMware* e *Amazon*.

Il punto di forza principale di Openstack risulta, senza dubbio, la pubblicazione di API (Application Programming Interfaces) disponibili per ogni servizio della piattaforma [18], che vengono a loro volta utilizzate per la comunicazione e l'interazione interna dei servizi stessi, ma soprattutto potenziano la caratteristica di apertura del sistema, permettendo tramite una mediazione degli accessi effettuata da Keystone, una facile integrazione e sperimentazione del cloud in scenari applicativi di grande varietà. E' proprio quest' ultima caratteristica, che sicuramente apre la possibilità di portare avanti un interessante fase di testbed all'interno del sistema tramite la sua integrazione con il modulo applicativo AO-M e più in generale con il modello di Transport Service Layer descritto nel Capitolo 1.

Un ulteriore passo avanti verso questa direzione, si ha grazie alla possibilità di studiare le caratteristiche di un nuovo modulo sperimentale (*Quantum*) per la gestione delle risorse di rete virtuali all'interno del cloud, che gli sviluppatori hanno aggiunto a partire dalle ultime release di Openstack (Essex e Folsom)[19]. Nelle prossime sezioni ci si focalizzerà sull' analisi di quest'ultimo, in quanto rappresenta l'elemento cuore di questa tesi.

Servizi Openstack	**Servizi Linux**	**Caratteristiche**
Nova	nova-api	accetta e risponde alle chiamate delle API di Nova
	nova-compute	gestisce la creazione e la distruzione delle VM istanziate tramite le API dell' hypervisor
	nova-volume	gestisce i dischi virtuali agganciabili alle VM come filesystem permanente anche dopo la distruzione di un istanza
	nova-network	gestisce le specifiche di rete tra le VM
	nova-schedule	gestisce la distribuzione delle VM all'interno del cluster fisico
Glance	glance-api	accetta e risponde alle chiamate della API di Glance
	glance-registry	gestisce l'aechiviazione dei metadati delle immagini salvate (dimenzione, tipo etc.)
Swift	swift-proxy-server	accetta le richiesta via API di Swift e gestisce l'upload e il dowload di risorse di storage
Keystone	keystone-all	gestisce gli account di accesso al sistema e il registro dei servizi contattabili nel sistema tramite API
Horizon	django	gestisce l'interfaccia web su server Apache

Tabella 2.1: Servizi di Linux per Openstack

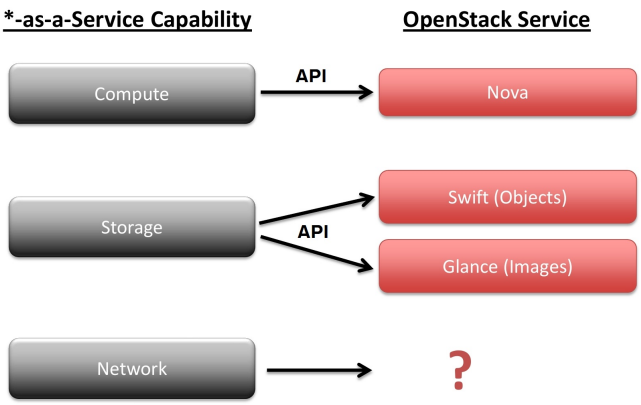

Figura 2.4: Diagramma dei problemi-servizi *Openstack*

2.3 Il progetto Quantum

Facendo una attenta analisi di ciò che Openstack cerca di proporre come idea guida, è possibile riassumere il tutto nel concetto di servizio, il cui compito consiste nel concentrarsi sulla gestione e funzionamento di uno degli aspetti (e.g. computazione, storage) che concorrono alla creazione dell'ambiente di cloud (il classico approccio *divide et impera* fondamentale nella costruzione di sistemi complessi).

Un servizio è a sua volta mappabile in due concetti chiave:

- Un **front-end** definito in modo standard (tramite API), che rappresenta le astrazioni logiche con cui è possibile accedere al servizio in maniera flessibile rispetto alle esigenze attuali e future del mondo applicativo.
- Un **back-end** legato alla tecnologia di riferimento (attualmente Linux) che ha il compito di mappare e implementare le astrazioni logiche sullo strato fisico disponibile.

Tale suddivisione è facilmente rintracciabile all'interno dei servizi standard di cui il cloud ha bisogno, come si può osservare in Figura 2.4.

Ciò che, a questo punto, risulta evidente, analizzando la lista dei principali sottocomponenti del sistema, è la completa assenza di un servizio per la gestione della rete. Questo, essenzialmente, per una ragione storica che vedeva gli aspetti di rete, originariamente, come parte integrata del servizio Nova stesso.

A partire dalle ultime release gli sviluppatori, però, si sono effettivamente resi conto come anche la rete, all'interno di un sistema di cloud, necessiti al giorno

d'oggi della stessa flessibilità di cui godono tutti gli altri servizi, affinchè possa essere in grado di soddisfare facilmente i numerosi vincoli di sistema, variabili da scenario a scenario (e.g scalabità e multiutenza, integrazione con differenti data center, mobilità delle VM).

Il 22 settembre del 2011 è nato, così, il progetto Quantum [27], la cui missione, ancora in fase sperimentale, si pone l'obiettivo di sviluppare una nuova astrazione di servizio, in grado di modellare le topologie e le configurazioni di rete in Openstack.

2.3.1 I limiti di Nova-Network

La gestione degli aspetti di rete, fino a questo punto, era stata sempre affidata al sottocomponente di Nova denominato Nova-Network, in grado di sopportare essenzialmente solo due modelli di networking:

- **Flat Networking**: una singola rete virtuale globale, su cui inserire le istanze delle VM, definita una volta per tutte a tempo di configurazione dell'intera infrastruttura.
- **VLAN Based Networking**: un meccanismo di segmentazione opzionale della rete, che sfrutta direttamente la tecnologia fornita da Linux per la creazione di VLAN, limitate ad unica interfaccia su cui gestire le VM.

Questo rappresenta un vero e proprio collo di bottiglia alla flessibilità della rete, in quanto questi semplici meccanismi non sono in grado di coprire totalmente l'ampio spazio di possibili casi d'uso, promosso dalle tecnologie di rete emergenti (e.g. QoS, VXLAN, OpenFlow). Il modello proposto risulta alquanto semplicistico, in quanto strettamente legato ad una limitata e particolare tecnologia di back-end (i.e Linux bridge) , quindi, difficilmente estendibile, per esempio, verso l' integrazione con aspetti avanzati (e.g. Firewall, NAT) o l'abilitazione di collegamenti tra reti private su data center remoti.

E' opportuno, inoltre, sottolineare la completa assenza di un front-end dinamico, in grado di garantire agli utenti applicativi un controllo personalizzabile della topologia di rete del proprio ambiente di cloud, basandosi su una assegnazione non prefissata dello spazio IP e sulla creazione di più reti private distinte a seconda del dominio di accesso.

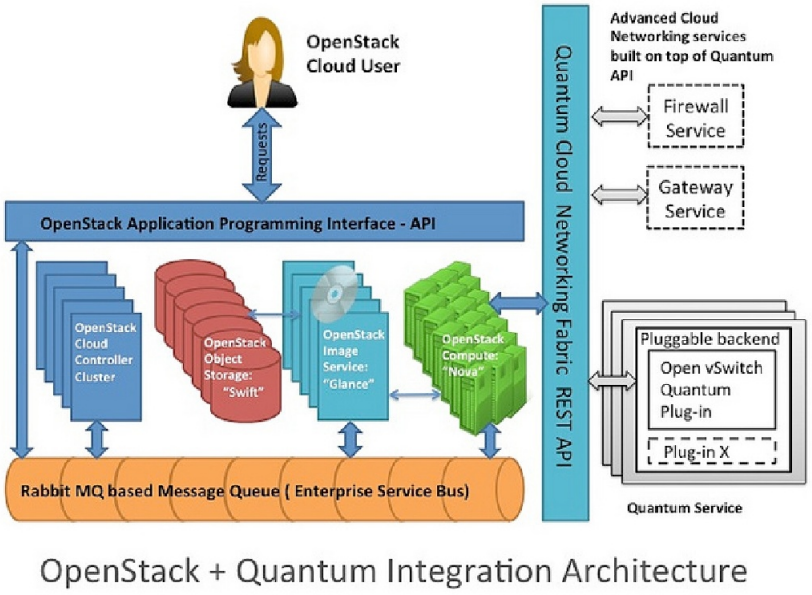

Figura 2.5: Integrazione di *Quantum* in *Openstack* [25]

All'interno di questo contesto, Quantum si pone come soluzione ottimale all' evidente *gap* [25] di Nova-Network, fondandosi sul concetto di *plugin*, che permette un interfacciamento *ad hoc* alle tecnologie esistenti e a quelle future, mantenendo, allo stesso tempo, un approccio standard e di alto livello, garantito dalle *Quantum API*.

2.3.2 Architettura Concettuale

Il servizio Quantum è in grado di astrarre dall'implementazione di rete fisica di basso livello, la cui gestione è affidata ad uno specifico *plugin* della tecnologia in uso, consentendo così di richiedere una particolare configurazione di rete IP, tramite un semplice meccanismo di richiesta e risposta (vedi Figura 2.5). Quantum controlla la connettività di rete configurando gli switch virtuali e fisici all'interno del data center, in accordo con i comportamenti esprimibili a livello logico tramite l'uso di opportune API. Le astrazioni logiche fondamentali che definiscono l'espressività del modello ad alto livello verranno proposte successivamente.

Quantum Network

Una *Quantum Network* non è altro che un dominio di rete L-2 virtuale, con caratteristiche strettamente analoghe a uno spazio di rete fisico. É identificata da un ID specifico e dal ID del gruppo di utenti (*tenants*) per cui è creata. Può essere pubblica, in questo caso ogni istanza di una VM attiva all'interno del cloud ne farà parte, oppure riservata a singoli gruppi. Un *tenant* può a sua volta creare delle porte logiche su una Network e collegare l'interfaccia virtuale di una VM a singole Network specifiche.

Quantum Port

Una *Quantum Port* abilita la connessione ad una specifica Quantum Network, per cui è stata definita. Può essere vista come una porta di uno switch virtuale che da accesso alla Network, e a cui va collegata l'interfaccia logica di una VM per scambiare traffico. Possiede uno stato che può essere DOWN o ACTIVE: se si trova in stato DOWN blocca il flusso del traffico in entrata/uscita.

Quantum Attachment

Un *Quantum Attachment* rappresenta un identificatore per la connessione di una interfaccia di rete logica[2] ad una porta. É possibile collegare nello stesso tempo una sola interfaccia alla stessa porta, quindi, tipicamente un Quantum Attachment identifica all'interno della Network l'interfaccia logica e la VM corrispondente.

Quadro di funzionamento

Tramite la creazione e la configurazione delle Quantum Network, è possibile definire topologie di rete ricche e flessibili, su cui gli altri servizi di Openstack come Nova possono connettere le proprie istanze (vedi Figura 2.6). In particolare, qualsiasi istanza (VM) del data center può accedere a una Quantum Network, purché essa si provvista di un'appropriata Virtual Interface (VIF) con cui creare un Quantum Attachment verso una Quantum Port. La definizione delle VIF (a livello logico analoghe alle interfaccia di rete fisiche) non fa parte delle responsabilità di Quantum, ma deve essere gestita dai servizi esterni che si occupano di istanziare una

[2]Una interfaccia di rete logica, in questo caso, può essere vista come qualsiasi cosa che può essere sorgente di traffico, come una vNIC di una VM o un' interfaccia di un Load Balancer.

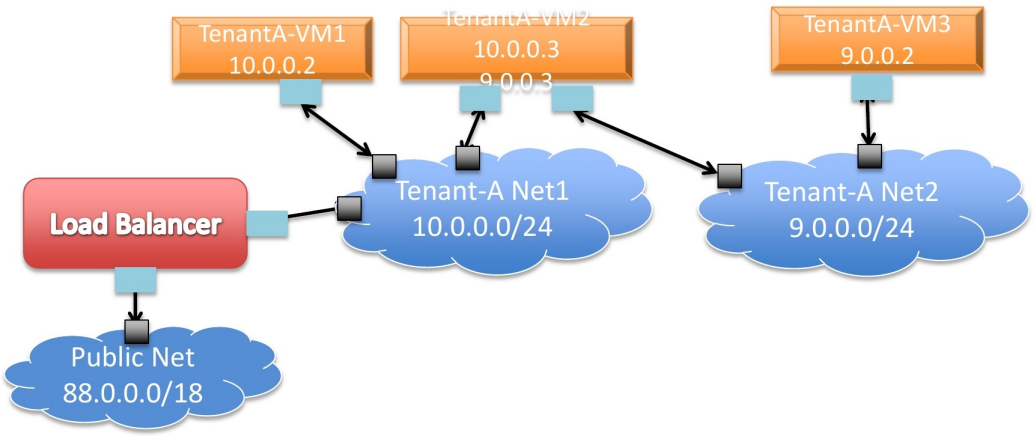

Figura 2.6: Flessibilità delle *Quantum Network*

VM (i.e. Nova). Quando una VIF è agganciata ad una Quantum Port, il Quantum stesso provvede alla creazione di un oggetto Attachment identificato univocamente all'interno delle Network, e interagisce con il *plugin* responsabile di implementare le politiche di instradamento e di segmentazione della rete fisica esistente, per garantire la connettività descritta ad alto livello tramite le astrazioni Network, Port e Attachment.

Quantum consente, inoltre, ad ogni dominio amministrativo di definire più reti private e scegliere il proprio schema di indirizzamento IP, anche se questo coincide con quello di un altro *tenant* presente nel sistema. In questo modo, si abilita la possibilità di valutare casi d'uso e scenari molto avanzati, in cui un utente applicativo può scegliere direttamente a quale rete connettere una VM (vedi Figura 2.6), o addirittura migrare una VM, senza cambiare IP, da una rete a un'altra tra data center remoti.

2.4 Il concetto di plugin

Tradizionalmente quando si parla di *networking*, non si pensa mai a un modello di rete scalabile rispetto alle dimensioni del cloud e in grado di autoconfigurarsi in relazione al modello di rete pensato ad alto livello. Quantum introduce il concetto di *Plugin*, che rappresenta il vero e proprio back-end del servizio, interponendosi tra

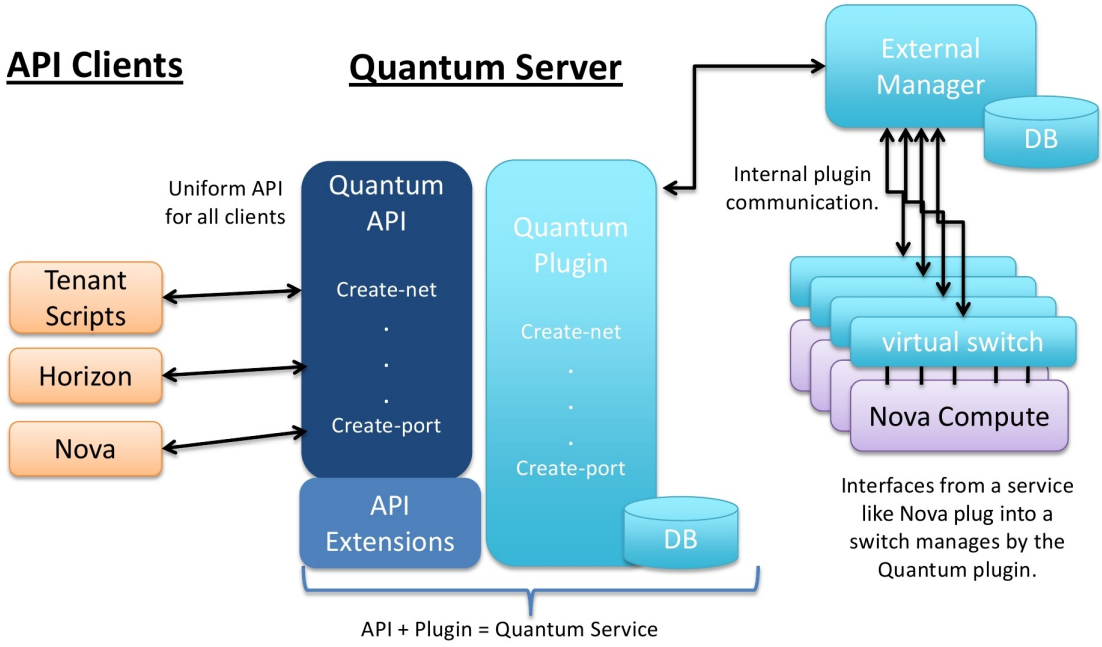

Figura 2.7: Schema di interazione con *Quantum Plugin* [27]

le astrazioni esprimibili tramite le API e la tecnologia specifica di cui si dispone in un determinato momento. Un plugin è qualcosa di indipendente dal mondo Openstack, sviluppabile privatamente dalle aziende o in forma open source affinché sia in grado di adattarsi e sfruttare al meglio gli apparati fisici di rete (e.g. switch) con cui ha il compito di interagire.

I compiti di un plugin (vedi Figura 2.7) possono essere riassunti in:

- **Elaborare le richieste delle API**: il plugin deve essere in grado di mappare le Quantum Network, Port richieste tramite API, nelle funzionalità che la tecnologia di basso livello mette a disposizione (e.g. mappare un Network ID in un VLAN ID)

- **Configurare gli switch virtuali**: il plugin deve interagire con Nova-Compute e fare da ponte di collegamento tra un Attachment (vedi Sottosezione 2.3.2 a pagina 24) e la riconfigurazione dinamica necessaria degli apparati sottostanti (e.g. assegnare una vswitch port di una VM a una particolare VLAN configurata)

Attualmente sono disponibili molte ditribuzioni di plugin, tutti in grado di interagire con le stesse API di Quantum e quindi integrabili o sostituibili all'interno dello stesso sistema:

- **OpenvSwitch**: permette l'integrazione con la tecnologia OpenvSwitch ed è compatibile con hypervisor KVM e XenServer (plugin di default per la fase sperimentale del progetto Quantum)
- **Cisco**: consente l'integrazione con gli switch Cisco UCS e Nexus per hypervisor KVM
- **LinuxBridge**: consente l'integrazione con Linux Bridge per hypervisor KVM
- **Nicira Nvp**: permette l'utilizzo di Nicira Networking Virtualization Platform (NVP) per hypervisor KVM e XenServer
- **Ryu**: permette l'integrazione con Ryu OpenFlow Controller per hypervisor KVM

2.4.1 Struttura implementativa

A livello implementativo, l'architettura di un semplice Quantum plugin (vedi Figura 2.8) fa riferimento ai seguenti componenti:

Controller

Il *Controller* è un server responsabile di offrire il servizio Quantum agli utenti dell'ambiente di coud-computing, ovvero è disponibile e contattabile tramite l'interazione via API ad un certo indirizzo IP e porta (default 9696).

Data Model

Il *Data Model* è la rappresentazione del modello di connettività, espresso dalle astrazioni di Quantum, ovvero la schematizzazione delle risorse di rete che garantiscono la connessione tra le VM presenti nell'ambiente di cloud. Un meccanisco per conservare i vari Data Model consiste nell'archiviare ed organizzare le informazioni all'interno di un database interno, parte integrante del plugin stesso. Tale

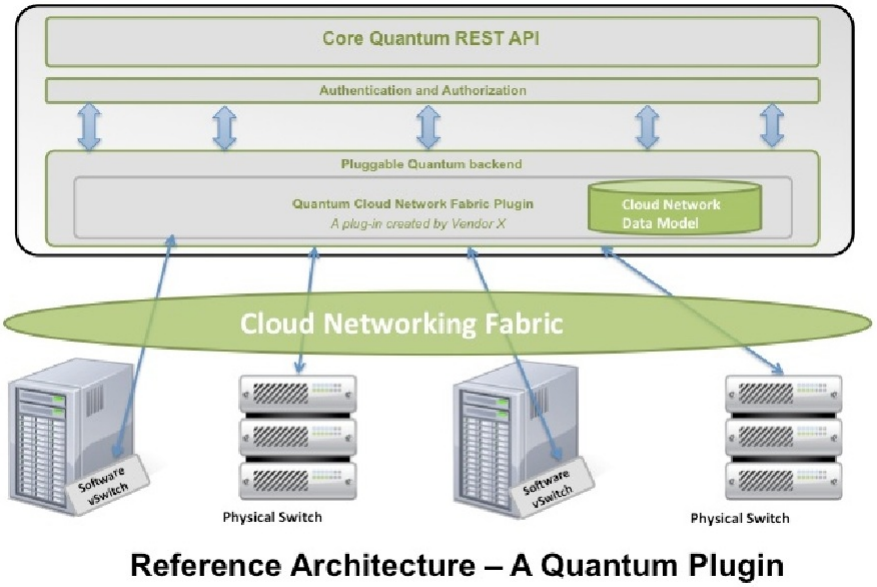

Figura 2.8: Elementi strutturali di un *Quantum Plugin* [25]

database, inoltre, ha lo scopo di rappresentare una vista completa della configurazione di rete attuale, a cui fan riferimento sia le API, sia i vari switch fisici o virtuali.

Communication Channel

Il *Communication Channel* è infine il meccanismo con cui il Controller può comunicare con gli switch fisici presenti nel sistema, per implementare la connettività di rete richiesta. In questo caso, ci possono essere vari approcci a seconda del sistema fisico di riferimento, uno di questi consiste nell'installare un Quantum Plugin Agent rappresentante il nostro Communication Channel su ogni Compute-node del cluster. L'agente ha il compito di contattare il database per ricevere gli aggiornamenti apportati dal controller e di comunicare agli switch fisici le nuove configurazioni da eseguire. L'agente si occupa anche di scegliere i meccanismi più opportuni a seconda della tecnologia, per creare l'isolamento tra i pacchetti di differenti Quantum Network.

2.5 Le API di Quantum

Le API (Application Programming Interface) rappresentano uno strumento base per poter interagire con una macchina di basso livello, sfruttando le astrazioni di alto livello necessarie per lo sviluppo di sistemi complessi. Nel progetto Quantum, rappresentano la vera e propria chiave di volta, nello sviluppo di un servizio di rete in grado di separe la logica dall' implementazione.

I benefici e la potenza che questa scelta garantisce, possono essere brevemente riassunti in:

- Fornire un protocollo di alto livello con cui è possibile interagire con gli apparati di rete in termini di risorse da allocare, tramite un linguaggio semanticamente ricco.
- Aumentare la portabilità del sistema di rete, consentendo la descrizione di reti virtuali in forma standard (ReSTful) e indipendente dalla tecnologia (i.e. switch fisici o virtuali, presenti in forma eterogenea nel sistema).
- Definire un modello estendibile, tramite le API Extensions e aprirne le porte a una possibile evoluzione indipendente dalle altre API dei servizi Openstack.
- Abilitare la piattaforma di cloud ad una facile integrazione con aspetti avanzati di rete(e.g. Firewall, Load Balancer, MPLS) e con una gestione flessibile delle differenti topologie di rete.

E' proprio sulla configurazione di Quantum all'interno di un data center in ambiente Openstack e sulla sperimentazione delle Quantum API, che si concentrerà la fase di Testbed successiva.

2.5.1 Il Modello ReSTful HTTP

Le Openstack Quantum API [26] vengono definite seguendo il modello ReSTful HTTP Service [12]. Le astrazioni su cui si fondano sono estremamente semplici e traggono vantaggio da gran parte degli aspetti del protocollo HTTP come caching, compressione dei contenuti, connessioni e molti altri.

L'utilizzo delle API si basa sul classico meccanismo di richiesta-risposta.

Formato	Header	Estensione	Default
XML	application/xml	.xml	si
JSON	application/json	.json	no

Tabella 2.2: Formati di utilizzo *Quantum API*

Ogni richiesta deve specificare l' identificatore del *tenant* mittente e la versione delle API a cui fa riferimento, in quanto essendo Quantum ancora un progetto in corso di sperimentazione possono essere rialasciate versioni con differenti compatibilità. Queste informazioni vengono rappresentate sfruttando un URI (RFC 2396)[11] così composto:

```
/{Quantum-API-version}/tenants/{tenant-id}/networks/
```

Attualmente le versioni API v1.0/v1.1 non richiedono la gestione delle credenziali di accesso legate al servizio Keystone di Openstack, come avviene per tutti gli altri servizi, ma tale integrazione verrà attuata a partire già dalla release v2.0.

Per ciò che riguarda i formati di serializzazione dei dati, le Quantum API vengono proposte sia nello standard XML sia JSON (vedi Tabella 2.2).

In Appendice vengono riportati i tipici *API template* per l'interazione HTTP con il Quantum Controller.

Comportamente Asincrono

É importante sottolineare come l'interazione tra API e plugin venga mediata dal Quantum Controller in maniera completamente asincrona. Questa scelta raccoglie i vantaggi e gli svantaggi tipici di un' interazione asincrona. In particolare, ciò significa che il ritorno di una richiesta HTTP di vario tipo (i.e. POST, GET, PUT o DELETE) non da alcuna garanzia che le modifiche apportate alle risorse di rete siano state effettivamente ed immediatamente mappate nelle configurazioni che il plugin deve effettuare sugli apparati di rete fisica. L'unica garanzia che il client API possiede al ritorno della richiesta HTTP, consiste solo nella effettiva modifica del modello logico di rete presa in considerazione.

API Extensions

Quantum garantisce un meccanismo di estendibilità che abilita chiunque a personalizzare ed estendere il Core API di default. Questa caratteristica è importante, in quando abilita una libera estensione delle API con il pieno utilizzo della potenza che ogni tecnologia di rete mette a disposizione, a seconda dello scenario di riferimento. Un classico esempio, consiste nell'estensione delle API per l'abilitazione dei parametri di configurazione di QoS.

Capitolo 3

Lavoro Svolto

Il lavoro svolto in questa tesi rappresenta un primo passo verso l'avvio di una interessante fase di Testbed che coinvolgerà il modulo TS-Layer descritto nel primo capitolo e le sue possibili interazioni con una moderna architettura di cloud-computing, quale quella proposta da Openstack, installata e configurata come parte del progetto CIRI ICT nella sede di Ingegneria presso la città di Cesena.

In particolare, ci si concentrerà sullo studio pratico e sulle possibili configurazioni del modulo Quantum, descritto nel secondo capitolo, cercando di fornire gli strumenti adatti per una gestione integrata delle risorse applicative e di rete all'interno del sistema di cloud.

L'obiettivo consiste nel cercare di capire come Quantum possa effettivamente rappresentare un possibile ponte di collegamento e personalizzazione verso le sperimentazioni successive, nella direzione di definire un modello funzionante dell'architettura di segnalazione Network Transport Service all'interno di un ambiente di rete virtuale, come quello ricreabile in Openstack.

Successivamente verranno, quindi, descritti i passi configurativi principali di Quantum all'interno del cluster di macchine fisiche a disposizione e verranno discusse le scelte effettuate, per capire come sfruttare al meglio la tecnologia e la potenza a disposizione di questo nuovo progetto sperimentale.

3.1 Dettagli Architetturali

Prima di procedere alla configurazione vera e propria del modulo Quantum occorre effettuare una fase di analisi dell' ambiente in cui ci troviamo e dei componenti di Openstack già installati in seguito a lavori sviluppati precedentemente.

Il cluster di riferimento è costituito da un totale di 7 macchine fisiche, organizzate come segue:

- **1 server**: è una macchina che svolge il compito di *master* del sistema. In particolare si dedica a tutte quelle funzioni tipiche di un Openstack Cloud Controller (vedi Sottosezione 2.1.2 a pagina 16) e rappresenta il nodo di uscita sulla rete pubblica.

- **5 host**: sono macchine con le stesse caratteristiche hardware, che svolgono la funzione di Compute Nodes (vedi Sottosezione 2.1.2 a pagina 16), ovvero sono pronte ad ospitare le VM a seconda dello scheduler deciso dal Cloud Controller.

- **1 host di test**: è una macchina introdotta nel cluster, ma momentaneamente non facente parte del cloud. La sua funzione è essenzialmente quella di possedere un ambiente di test, con le stesse caratteristiche delle macchine fisiche, su cui provare le configurazioni e le installazioni sperimentali, prima di renderle effettive in Openstack. La cosa risulta utile, in quanto si evitano eventuali conflitti e problemi che potrebbero rendere inagibile il sistema precedentemente stabile. Conclusa la fase di Testbed, tale macchina verrà aggiunta ai Compute Nodes già esistenti.

- **1 switch fisico**: è l' apparato di rete che si occupa di rendere possibili i collegamenti tra le varie macchine, ovvero della definizione della LAN fisica del cluster.

Di seguito verranno riportati i dettagli hardware, software e di rete del sistema.

3.1.1 Hardware

Server

- Processore: *Intel Xeon E31230 3.20 GHz quadCore (4 CPU fisiche + 4 CPU virtualizzate)*
- RAM: *8GB*
- HD: *500GB*
- Interfaccia di rete:
  - *eth0*, collegata allo switch condivide Management e Data
  - *eth1*, inutilizzata
  - *eth2*, collegata alla rete pubblica
  - *eth3*, inutilizzata

Host

- Processore: *Intel Pentium G620*
- RAM: *4GB*
- HD: *250 GB*
- Interfaccia di rete:
  - *eth0*, collegata allo switch condivide Management e Data

3.1.2 Software

Server

- SO: *Linux CentOS 6.3* [16]
- Database: *MySQL*
- Network Time Protocol: *NTP*
- Openstack: *Essex*

- Componenti:
  - *Keystone* (*keystone-all*)
  - *Nova* (*nova-api, nova-compute, nova-network, nova-scheduler*)
  - *Glance* (*glance-api, glance-registry*)
  - *Horizon* (interfaccia web *Django* su server Apache raggiungibile da rete pubblica)
- Altro: *Python, pip, rabbitMQ, Apache, PHPmyAdmin*

Host

- SO: *Linux CentOS 6.3* [16]
- Network Time Protocol: *NTP*
- Openstack: *Essex*
- Componenti:
  - *Nova* (*nova-compute, nova-volume*)
- Altro: *Python, pip, rabbitMQ, Apache*

3.1.3 Rete

- Switch fisico: *Switch Ethernet 3Com.*
- 10.10.0.0/24 : rete di Network Management utilizzata fino ad ora.
- 10.20.0.0/24: rete di Network Data virtuale utilizzata dalle VM, basata unicamente sull'integrazione tra Linux Bridge e Nova-Network.
- 137.204.191.15: IP pubblico di accesso al server.

3.2 Preparazione dell'interfaccia di Testbed

La configurazione e l'integrazione del progetto Quantum (vedi Sezione 2.3 a pagina 21) all'interno del sistema Openstack attualmente installato nel data center richiede una serie di considerazioni iniziali, che stabiliscono le linee guida e i vincoli per le successive fasi di lavoro.

3.2.1 Quantum release: Essex vs Folsom

Il modulo Quantum è ancora in fase sperimentale, quindi, periodicamente vengono rilasciate nuove release, che arricchiscono il progetto con funzionalità e caratteristiche in grado di abilitare una sempre maggiore personalizzazione di rete nel cloud, rimanendo aderenti alle idee base proposte nel Capitolo 2.

Attualmente, nel sistema è installato Openstack versione *Essex*, per la quale è disponibile una versione di Quantum in stato embrionale e contenente le sole funzionalità di base. Questo rappresenta, comunque, un buon punto di partenza per la sperimentazione delle sue potenzialità.

Recentemente è stata rilasciata dalla community una nuova release di Openstack, versione *Folsom*, a cui sono dedicate intere sezioni di documentazione, relative alle innovazioni introdotte rispetto al prototipo precedente. Risulta fondamentale capirne le differenze chiave, per impostare un buon approccio alla preparazione del lavoro.

- **Essex**: Propone una integrazione tra Nova e Quantum tramite *QuantumManager*, ovvero un servizio che fa da proxy di collegamento tra Nova-Network e il nuovo modulo Quantum, dedicato alla gestione di rete. In questo caso, requisito e limite per il funzionamento del sistema è che vi sia un solo nodo in cui venga abilitato Nova-Network, che nel nostro caso sarà rappresentato dal server. Per la creazione di nuove Quantum Network sfruttando l'interazione da CLI (Command Line Interface), occorre utilizzare il comando *nova-manage*. Una alternativa a questa opzione, consiste nell'installare un Quantum CLI, applicativo in grado di tradurre i comandi direttamente in richieste API spedite al Controller del Plugin (vedi Sottosezione 2.4.1 a pagina 27). Di default, sfruttando il *QuantumManager* l'interazione con il plugin tramite API può avvenire anche direttamente dalla rete pubblica

senza aver bisogno di definire le credenziali tipiche del servizio Keystone, tuttavia questa opzione può essere abilitata opzionalmente. Infine, per la gestione degli aspetti avanzati come DHCP, connessione di reti private tramite router virtuali e floating IP ci si affida ai supporti e driver di default proposti dall' integrazione con Nova-Network.

- **Folsom**: Diversamente dai settaggi classici di Nova, in questa recente versione tutta la gestione della rete viene destinata al servizio Quantum, eliminando definitivamente l'integrazione con Nova-Network (il servizio non viene proprio più installato), avvicinandosi ai concetti teorici che vedono Quantum come un vero e proprio servizio stand-alone, posizionabile su qualsiasi macchina fisica all'interno del data center. In questo nuovo scenario, tutti i comandi legati al *QuantumManager*, proposti per la versione Essex, non sono più utilizzabili e l'interazione con Quantum può avvenire solo tramite le API. Risulta, quindi, fondamentale configurare Keystone a tal scopo. Infine, le funzioni DHCP e L3 management (router virtuali, floating IP) vengono forniti tramite specifici *Agent* installabili in qualsiasi nodo del sistema.

In questa fase si procederà alla configurazione di Quantum *Essex* per riuscire a mantenere la compatibilità con il sistema già configurato, provvedendo a testare e prendere confidenza con le API 1.0/1.1, che troviamo a disposizione. In futuro, si auspica l'upgrade di tutto il sistema alle innovazioni che ha introdotto la versione *Folsom*, in modo da poter impostare più agevolmente le fasi di Testbed che coinvolgeranno il Network Transport Service e la sua integrazione nell'architettura di cloud Openstack.

3.2.2 Il plugin di riferimeno: tecnologia OpenVSwitch

Come si può notare nei dettagli architetturali(vedi Sezione 3.1) si ha a disposizione uno switch *3Com*, che non rientra all'interno dei plugin di Quantum disponibili e già sviluppati (vedi Sezione 2.4 a pagina 25), situazione abbastanza comune, in quanto la lista è attualmente ridotta. Per questo motivo, occorre impostare il lavoro cercando di separare la gestione degli aspetti logici di rete dal particolare switch fisico a disposizione, che manterrà solo le caratteristiche di un semplice *hub* di collegamento fra le macchine del sistema. Risulta, quindi, opportuno e conveniente affidarsi alla tecnologia di virtualizzazione di rete proposta da *OpenVSwitch* [23]. In

particolare, la soluzione consiste nell' installare e configurare degli switch virtuali su ogni macchina fisica del data center, in modo tale che sia possibile utilizzare il Quantum Plugin che questa tecnologia mette a disposizione e che si consiglia in generale di adottare, in quanto utilizzato e testato dalla community di sviluppatori Openstack durante le fasi di sperimentazione. Si rimanda alla documentazione ufficiale [24], lo studio delle specifiche di rete che OpenVSwitch è in grado di abilitare, limitandosi a sottolineare che si tratta di un software molto potente con cui è possibile configurare anche aspetti avanzati di rete come l' OpenFlow.

3.2.3 Scelte Configurative

L'utilizzo della tecnologia OpenVSwitch all'interno del cluster di laboratorio, in realtà mette in luce un problema su cui è utile focalizzarsi, in quanto influente nelle procedure di configurazione successive. In particolare, analizzando la struttura del data center a cui facciamo riferimento, è stato scelto di affidarsi a un' unica interfaccia di rete fisica *eth0*, su cui viene instradato in maniera promiscua sia il traffico di Management (rete 10.10.0.0/24) tra i vari host, sia il traffico dati tra le VM (rete virtuale 10.20.0.0/24) instanziate nel sistema.

Introducendo la virtualizzazione dello switch, tramite l'uso di OVS Bridge, è fortemente consigliato disporre di due interfacce di rete fisiche su ogni macchina, per mantenere separato il traffico tra Management e Data. Questo, in particolare, poiché una volta assegnata un'interfaccia a un OVS Bridge, quest' ultima perde il suo ip fisico, mantenendo solo l'ip del bridge virtuale. In questa fase di sperimentazione, è stato scelto di posticipare l'introduzione di nuove interfacce di rete fisiche sugli host e di adottare il seguente approccio.

- **Server**:
  - *eth0*: rete di Management 10.10.0.0/24
  - *eth1*: integrazione con OVS Bridge 10.30.0.0/24
  - *eth2*: rete pubblica su IP statico 137.204.191.15
  - *eth3*: inutilizzata
- **Host**:
  - *eth0*: cambiamento dinamico tramite script tra

* rete di Management 10.10.0.0/24 all'avvio standard della macchina
* integrazione con OVS Bridge 10.30.0.0/24 all'avvio di OpenVSwitch

I punti di forza di questo approccio risiedono nel poter comunque gestire uno switch virtuale tramite OpenVSwitch, sfruttando un' unica interfaccia di rete fisica, che a seconda delle necessità è configurabile sia come bridge, sia come classica interfaccia di Management.

3.3 Installazione del modulo Quantum

La configurazione di Quantum, coinvolge l'interazione di molti servizi Linux (vedi Sottosezione 2.1.3 a pagina 17) dislocati all'interno dei vari host del data center.

Facendo riferimento alla Figura 3.1 a pagina 41, possiamo definire la struttura organizzativa con cui verranno eseguiti i passi di configurazione. L'installazione del servizio Quantum in ambiente *Linux CentOS* si suddivide in:

- **Quantum-Server** (*quantum-server*): è il processo principale, un demone Python in attesa di richieste effettuate tramite Quantum API, che si occupa di mapparle in configurazioni del plugin di riferimento, ovvero OpensV-Switch.
- **Plugin-*-Agent** (*quantum-openvswitch-agent*): agente che si occupa dell'interazione con l' hypervisor (*KVM*) per eseguire le configurazioni sui vswitch locali.

Gli agenti OVS sono distribuiti su ogni Compute Node del data center e interagiscono con il Quantum-Server e il data-base di riferimento, posizionati sul Controller Node, nel seguente modo:

- Attraverso RPC, sfruttando il servizio RabbitMQ.
- Attraverso le Quantum API standard.

Al momento dell' istanziazione di una VM, Nova-Compute appoggiandosi al *QuantumManager* comunica l'intenzione di voler collegare ogni interfaccia virtuale (*vNIC*) in una particolare Quantum Network. A questo punto, l'agente del nodo effettua in automatico tutte le configurazioni necessarie con cui è possibile mappare

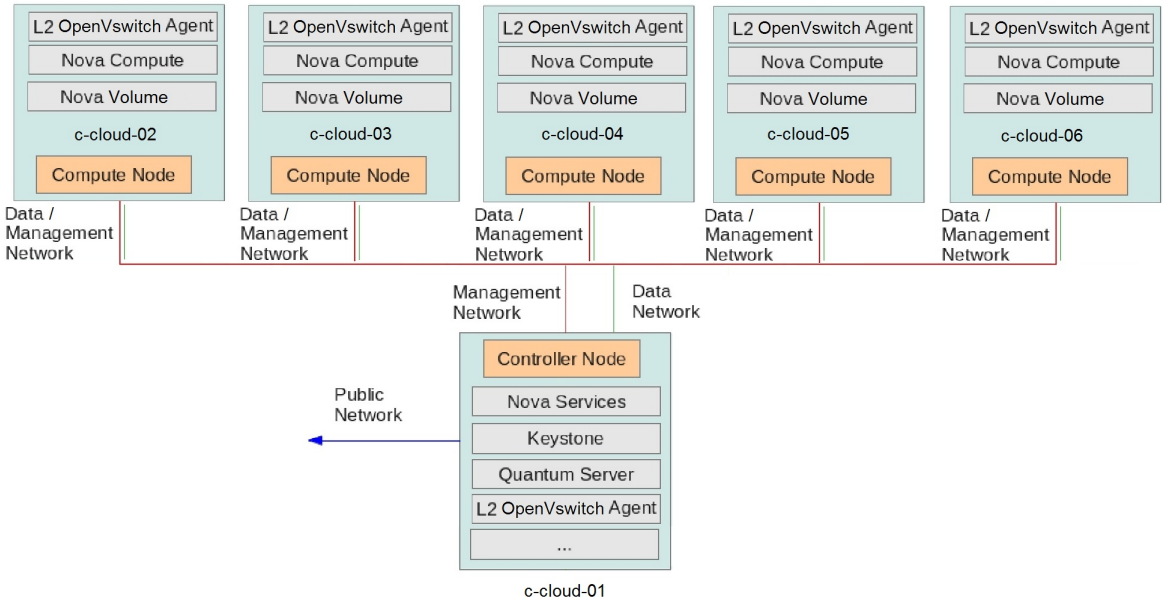

Figura 3.1: Configurazione dei moduli *Openstack* nel data center

questa richiesta sulla tecnologia a disposizione. Nel caso di OpensVSwitch, l'effetto consisterà nella creazione di un bridge virtuale (*tap*) con cui si ha accesso alla particolare VLAN assegnata ad ogni Quantum Network e che permette di tenere separato il traffico tra reti differenti.

3.3.1 Configurazione OpenVSwitch

Prima di configurare OpenVSwitch occorre portare il sistema corrente, in uno stato di inattività, su cui è possibile effettuare le nuove integrazioni.

Innanzi tutto, occorre terminare le eventuali istanze di VM attive in questo momento.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nova list
+----------+----------+--------+------------------+
|  ID      |   Name   | Status |      Networks    |
+----------+----------+--------+------------------+
| 10bac83a | ubuntu02 | ACTIVE | public=10.20.0.20 |
| c7a3bc4f | cirrus01 | ACTIVE | public=10.20.0.19 |
```

```
| fe6e0bc6 | cirros03 | ACTIVE | public=10.20.0.21 |
+----------+----------+--------+-------------------+
[root@c-cloud-01 ~]#nova delete ID_ISTANCES
[root@c-cloud-01 ~]#
```

Cancellare la rete virtuale[1] creata su `br100`.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nova-manage network list
    | id | IPv4 | IPv6 | start address | ... ...
[root@c-cloud-01 ~]#nova-manage network delete
                    --uuid= ID_NETWORK
[root@c-cloud-01 ~]#
```

Controllare nel database tramite *PHPMyAdmin* che nella tabella `nova/networks` non ci sia più la *net* precedente.

A questo punto, si è pronti per l'installazione effettiva di OpenVSwitch. Esistono due modalità in cui procedere:

- Installazione da sorgenti.
- Installazione da pacchetto RPM.

Nel nostro caso, si è scelto di procedere con la modalità pacchetto[2], per mantenere conformità con le installazioni precedenti di Openstack e per avere una buona modalità di ripristino in caso si incontrino problemi critici nelle prossime fasi.

Oltre ai normali processi eseguibili in user-mode, OpenVSwitch prevede la configurazione di un modulo in Linux kernel-space, che andrà a sostituire l'attuale componente Bridge.

Prima di caricare il modulo OVS, occorre quindi verificare che non sia già presente il modulo Bridge, o nel caso disabilitarlo.

[1] Se compaiono alcuni WARNING è normale

[2] La creazione dei pacchetti è riportata in Appendice

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#lsmod | grep bridge
                    ...
[root@c-cloud-01 ~]#rmmod bridge
[root@c-cloud-01 ~]#
```

É possibile ora procedere con l'installazione effettiva dei pacchetti dalla directory in cui sono stati posizionati, rispettando l'ordine seguente.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#rpm -ivh
            kmod-openvswitch-1.9.90-1.el6.x86_64.rpm
[root@c-cloud-01 ~]#rpm -ivh
            openvswitch-1.9.90-1.x86_64.rpm
[root@c-cloud-01 ~]#rpm -ivh
            openvswitch-debuginfo-1.9.90-1.x86_64.rpm
[root@c-cloud-01 ~]#
```

É opportuno verificare che una volta installati gli rpm, questi vengano riconosciuti tra i moduli del kernel. In particolare, ricercare `weak-updates/openvswitch/brcompat.ko` e `weak-updates/openvswitch/openvswitch.ko` dopo aver eseguito il comando `modprobe -l`.

Procediamo con il caricamento del modulo OpenVSwitch.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#insmod lib/modules
            /"KERNEL_VERSION"/weak-updates
                /openvswitch/openvswitch.ko
[root@c-cloud-01 ~]#
```

Per verificare che tutto sia andato a buon fine[3] si può utilizzare `dmesg | tail` .

A questo punto, occorre definire il database di riferimento per la configurazione di OpenVSwitch.

[3]In quanto le operazioni sul kernel sono abbastanza delicate per eventuali errori si rimanda al sito ufficiale [24].

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#mkdir -p /etc/openvswitch
[root@c-cloud-01 ~]#mkdir -p /var/run/openvswitch
[root@c-cloud-01 ~]#ovsdb-tool create
            /etc/openvswitch/conf.db
             /usr/share/openvswitch/vswitch.ovsschema
[root@c-cloud-01 ~]#
```

Avviamo il database[4] con *bind* su una Unix Socket e senza protocollo di sicurezza SSL.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#ovsdb-server
       --remote=punix:/var/run/openvswitch/db.sock \
       --remote=db:Open_vSwitch,manager_options \
       --pidfile --detach
[root@c-cloud-01 ~]#
                     ...
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#ovs-vsctl --no-wait init
[root@c-cloud-01 ~]#
```

Per poter continuare ad utilizzare il comando `brctl` con le relative opzioni, in modo tale che vengano tradotte direttamente sul nuovo kernel-module `openvswitch.ko` occorre caricare un altro utile componente (`brcompat.ko`).

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#insmod lib/modules
                /"KERNEL_VERSION"/weak-updates
                /openvswitch/brcompat.ko
[root@c-cloud-01 ~]#
```

Per avviare OpenVSwitch procediamo come segue.

[4]Ogni nodo in cui viene installato OVS dovrà possederne uno proprio.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#ovs-vswitchd --pidfile --detach
[root@c-cloud-01 ~]#
                    ...
[root@c-cloud-01 ~]#ovs-brcompatd --pidfile --detach
[root@c-cloud-01 ~]#
```

Volendo controllare che tutto venga eseguito correttamente, è possibile utilizzare i comandi seguenti.

```
ps aux | grep ovs-vswitchd
ps aux | grep ovs-brcompatd
```

Per visualizzare la lista dei bridge disponibili, si può utilizzare indifferentemente

```
brctl show
ovs-vsctl list-br
```

Di seguito viene riportata una possibile alternativa di avvio veloce, che però non abilita il modulo `brcompat`, sempre da avviare manualmente.

```
/etc/init.d/openvswitch start
```

Note Particolari

I passi di configurazione successivi si differenziano tra Host e Server, in quanto gli Host, a differenza del Server, possiedono un' unica interfaccia di rete (*eth0*). In particolare, occorre creare un nuovo OVS Bridge che identificheremo con `br-int`, a cui associare un' interfaccia fisica della macchina corrente. L'elemento importante da prendere in considerazione consiste nel fatto che occorre assegnare un IP al bridge stesso e non più all'interfaccia di uscita nella rete fisica.

Per mantenere una buona organizzazione del traffico di rete, utile per le fasi di debug, conviene definire una nuova rete che si differenzi da quella standard di Management (10.10.0.0/24) a cui assegneremo l'intervallo 10.30.0.0/24.

Server

Una volta avviato correttamente OpenVSwitch è necessario creare un nuovo bridge `br-int`, a cui assegnamo IP `10.30.0.1/24` e che associamo all'interfaccia `eth1` .

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]# ovs-vsctl add-br br-int
[root@c-cloud-01 ~]#
                    ...
[root@c-cloud-01 ~]#ifconfig br-int 10.30.0.1/24
[root@c-cloud-01 ~]#
                    ...
[root@c-cloud-01 ~]#ovs-vsctl add-port br-int eth1
[root@c-cloud-01 ~]#
```

In questo modo, il server sarà comunque raggiungibile sulla `eth0` di Management (`10.10.0.0/24`) dai servizi di Openstack e allo stesso tempo sarà collegato allo switch virtuale di OpensVSwitch sulla rete `10.30.0.0/24` .

Affinché i vari host possano continuare ad uscire sulla rete publica, è opportuno aggiungere la seguente regola di NAT ad `iptables`.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#iptables -t nat -I POSTROUTING 1
                    -s 10.30.0.0/24 -j MASQUERADE
[root@c-cloud-01 ~]#
```

Host

Per tutti gli host occorre creare il bridge `br-int`, assegnargli un IP della rete `10.30.0.X/24` e procedere col collegamento di `eth0` alla porta del vswitch.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]# ovs-vsctl add-br br-int
[root@c-cloud-0X ~]#
                    ...
[root@c-cloud-0X ~]#ifconfig br-int 10.30.0.X/24
[root@c-cloud-0X ~]#
```

```
                    ...
[root@c-cloud-0X ~]#ovs-vsctl add-port br-int eth0
[root@c-cloud-0X ~]#
```

A questo punto, gli host vengono isolati dalla rete di Management (`10.10.0.0/24`), mentre sono raggiungibili sul vswitch (10.30.0.0/24).

Affinché tutti i servizi di Openstack possano continuare a interagire correttamente con il server, è importante risistemare la tabella di routing per ogni singolo host, eliminando la rotta `10.10.0.0/24` , che altrimenti girerebbe il traffico sull'interfaccia `eth0` e impostando come default gateway il server sulla rete virtuale (`10.30.0.1/24`).

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#route del -net 10.10.0.0/24
[root@c-cloud-0X ~]#route del default
[root@c-cloud-0X ~]#route add default gw 10.30.0.1
[root@c-cloud-0X ~]#
```

3.3.2 Configurazione Quantum-Server e Quantum-Agent

Installazione pacchetti

Prima di tutto, installare gli utility scripts per CentOS, che ci forniranno degli utili script configurativi.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#yum install openstack-utils
[root@c-cloud-01 ~]#
```

Installare, poi, i pacchetti RPM di Openstack Quantum relativi al plugin per OpenVSwitch.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#yum install openstack-quantum
[root@c-cloud-01 ~]#
                    ...
```

```
[root@c-cloud-01 ~]#yum install
                    openstack-quantum-openvswitch
[root@c-cloud-01 ~]#
```

I passi configurativi successivi[5], si differenziano nuovamente tra Server, su cui verrà ablitato *Quantum-Server* e *Quantum-OpenVSwitch-Agent* e Host, dove invece verrà avviato solo *Quantum-OpenVSwitch-Agent*.

Server

É necessario configurare un database *MySQL*[6] per la gestione delle risorse di rete del servizio Quantum (vedi Sottosezione 2.4.1 a pagina 27).

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#mysql -u root -p
[root@c-cloud-01 ~]#CREATE DATABASE ovsquantum;
[root@c-cloud-01 ~]#
```

Creare un nuovo utente che gestisca il database con tutti i privilegi.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#grant all on ovsquantum.* to
            'quantumAdmin'@'%'
            identified by 'quantumSecret';
[root@c-cloud-01 ~]#
```

Configurare Quantum[7] in modo che il plugin installato faccia riferimento al nuovo database.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano /etc/quantum/plugins/
```

[5]In Appendice vengono riportati in forma completa i file di configurazione principali.

[6]MySQL non accetta caratteri speciali, quindi, per evitare problemi chiamiamo il database semplicemente ovsquantum

[7]In caso di reboot Quantum-Server dovrebbe sempre partire dopo MySQL se no si rischiano errori

```
                    openvswitch/ovs_quantum_plugin.ini
[root@c-cloud-01 ~]#
                    ...

sql_connection = mysql://quantumAdmin:quantumSecret
                    @10.10.0.1/ovsquantum
```

Abilitare Quantum-Server all'uso di OpenVSwitch plugin.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano /etc/quantum/plugins.ini
[root@c-cloud-01 ~]#
                    ...

#provider = quantum.plugins.sample.
                SamplePlugin.FakePlugin
provider=quantum.plugins.openvswitch.
            ovs_quantum_plugin.OVSQuantumPlugin


                    ...
```

Impostare la connessione tramite RPC (*RabbitMQ*) di Quantum-Server e definire OpenVSwitch plugin come *core* plugin di default.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano /etc/quantum/quantum.conf
[root@c-cloud-01 ~]#
                    ...
[DEFAULT]
                    ...

#RABBITMQ
rabbit_host = 10.10.0.1
rabbit_port = 5673
                    ...
```

```
#CORE PLUGIN
core_plugin=quantum.plugins.openvswitch.
            ovs_quantum_plugin.OVSQuantumPlugin

                    ...
```

A questo punto, procediamo all'aggiunta di un *fix* fondamentale per risolvere un *bug* riconosciuto ufficialmente e legato all'integrazione di OpenVSwitch plugin con Hypervisor di virtualizzazione KVM presente nel data center.

Occorre, quindi, comportarsi nel modo seguente:

- Disabilitare SELinux.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano etc/selinux/config
                    ...

seSELINUX=disabled

                    ...
```

- Portarsi nel file di configurazione del hypervisor *libvirt KVM* e abilitare le seguenti opzioni.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano /etc/libvirt/qemu.conf
                    ...

user = "root"
                    ...
group = "root"
                    ...
cgroup_device_acl = [
        "/dev/null", "/dev/full", "/dev/zero",
        "/dev/random", "/dev/urandom",
        "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
```

```
        "/dev/rtc", "/dev/hpet", "/dev/net/tun",
]
                        ...
clear_emulator_capabilities = 0

                        ...
```

- Riavviare hypervisor *libvirt*.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#/etc/init.d/libvirtd restart
[root@c-cloud-01 ~]#
```

- Disabilitare la richiesta di una shell interattiva per il comando `sudo`, utilizzando `visudo` per editare il file `/etc/sudoers` e commentando la linea seguente.

```
                ...
#Defaults       requiretty
                ...
```

Infine, occorre modificare la configurazione standard di Nova affinché, al momento dell'istanziazione di una VM, possa interagire con il Quantum-OpenVSwitch-Agent per creare un Attachment(vedi Sottosezione 2.3.2 a pagina 24) tra vNic virtuale e Quantum Port, all'interno di una specifica Quantum Network.

Per fare ciò, è necessario portarsi nel file `nova.conf` e abilitare l'interazione con il *QuantumManager*(vedi Sottosezione 3.2.1) e i vari driver necessari all'integrazione con OpenVSwitch. Prima di tutto disabilitare, tramite commento, le configurazioni standard legate a Nova-Network, mantenendo solo il flag relativo al servizio DHCP, poi configurare il tutto come segue.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano /etc/nova/nova.conf
                ...

#network_manager=nova.network.
```

```
                    manager.FlatDHCPManager
#force_dhcp_release=True
#dhcpbridge_flagfile=/etc/nova/nova.conf
#firewall_driver=nova.virt.libvirt.
                    firewall.IptablesFirewallDriver
## Change my_ip to match each host
#my_ip=10.10.0.1
##public_interface=virbr0
##vlan_interface=br100
#flat_network_bridge=br100
#flat_interface=eth0
#fixed_range=10.20.0.0/24

dhcpbridge=/usr/bin/nova-dhcpbridge
                    ...

network_manager=nova.network.
                quantum.manager.QuantumManager

# QUANTUM

quantum_connection_host=10.10.0.1
quantum_connection_port=9696
quantum_use_dhcp=True

# OPENVSWITCH
libvirt_vif_driver=nova.virt.libvirt.
                vif.LibvirtOpenVswitchDriver
libvirt_vif_type=ethernet
linuxnet_interface_driver=nova.network.
                linux_net.LinuxOVSInterfaceDriver
linuxnet_ovs_integration_bridge=br-int


                    ...
```

É importante sottolineare come nel caso in cui il bridge OpenVSwitch creato in precedenza (vedi Sottosezione 3.3.1) abbia nome differente da `br-int`, occorre aggiustare la configurazione di `linuxnet-ovs-integration-bridge` con il nome identificativo selezionato.

Come ultima cosa, ricordarsi di riavviare i servizi di Openstack Nova-Compute, Nova-Network e Nova-Scheduler prima di procedere all'avvio effettivo del Quantum-Server e Quantum-OpenVSwitch-Agent.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nova-compute &>/dev/null &
[root@c-cloud-01 ~]#nova-network &>/dev/null &
[root@c-cloud-01 ~]#nova-scheduler &>/dev/null &
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#/etc/init.d/quantum-server start
[root@c-cloud-01 ~]#/etc/init.d/quantum-
                        openvswitch-agent start
[root@c-cloud-01 ~]#
```

Host

Configurare Quantum in modo che il plugin installato faccia riferimento al database del Cloud-Controller.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nano /etc/quantum/plugins/
                      openvswitch/ovs_quantum_plugin.ini
[root@c-cloud-0X ~]#
                      ...

sql_connection = mysql://quantumAdmin:quantumSecret
                      @10.10.0.1/ovsquantum
```

Abilitare Quantum all'uso di OpenVSwitch plugin.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nano /etc/quantum/plugins.ini
```

```
[root@c-cloud-0X ~]#
                    ...

#provider = quantum.plugins.sample.
               SamplePlugin.FakePlugin
provider=quantum.plugins.openvswitch.
            ovs_quantum_plugin.OVSQuantumPlugin

                    ...
```

Impostare la connessione tramite RPC (*RabbitMQ*) verso Quantum-Server e definire OpenVSwitch plugin come *core* plugin di default.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nano /etc/quantum/quantum.conf
[root@c-cloud-0X ~]#
                    ...
[DEFAULT]
                    ...

#RABBITMQ
rabbit_host = 10.10.0.1
rabbit_port = 5673
                    ...

#CORE PLUGIN
core_plugin=quantum.plugins.openvswitch.
            ovs_quantum_plugin.OVSQuantumPlugin

                    ...
```

Procediamo all'aggiunta del *fix* per risolvere il *bug* legato all'integrazione di OpenVSwitch plugin con Hypervisor di virtualizzazione KVM, presente sugli host nel data center.

- Disabilitare SELinux.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nano etc/selinux/config
                    ...

seSELINUX=disabled

                    ...
```

- Portarsi nel file di configurazione del hypervisor *libvirt KVM* e abilitare le seguenti opzioni.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nano /etc/libvirt/qemu.conf
                    ...

user = "root"
                    ...
group = "root"
                    ...
cgroup_device_acl = [
        "/dev/null", "/dev/full", "/dev/zero",
        "/dev/random", "/dev/urandom",
        "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
        "/dev/rtc", "/dev/hpet", "/dev/net/tun",
]
                    ...
clear_emulator_capabilities = 0

                    ...
```

- Riavviare hypervisor *libvirt*.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#/etc/init.d/libvirtd restart
[root@c-cloud-0X ~]#
```

- Disabilitare la richiesta di una shell interattiva per il comando `sudo` utilizzando `visudo` per editare il file `/etc/sudoers` e commentando la linea seguente.

```
                    ...
#Defaults          requiretty
                    ...
```

Infine, portarsi nel file `nova.conf` e abilitare l'interazione con il *QuantumManager* e i vari driver di intergazione, disabilitando, tramite commento, le configurazioni di Nova-Network e mantenendo solo il flag relativo al servizio DHCP.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nano /etc/nova/nova.conf
                    ...

#network_manager=nova.network.
                    manager.FlatDHCPManager
#force_dhcp_release=True
#dhcpbridge_flagfile=/etc/nova/nova.conf
#firewall_driver=nova.virt.libvirt.
                    firewall.IptablesFirewallDriver
## Change my_ip to match each host
#my_ip=10.10.0.1
##public_interface=virbr0
##vlan_interface=br100
#flat_network_bridge=br100
#flat_interface=eth0
#fixed_range=10.20.0.0/24

dhcpbridge=/usr/bin/nova-dhcpbridge
                    ...

network_manager=nova.network.
                quantum.manager.QuantumManager
```

```
# QUANTUM

quantum_connection_host=10.10.0.1
quantum_connection_port=9696
quantum_use_dhcp=True

# OPENVSWITCH
libvirt_vif_driver=nova.virt.libvirt.
                vif.LibvirtOpenVswitchDriver
libvirt_vif_type=ethernet
linuxnet_interface_driver=nova.network.
                linux_net.LinuxOVSInterfaceDriver
linuxnet_ovs_integration_bridge=br-int

                    ...
```

Se il bridge OpenVSwitch creato in precedenza (vedi Sottosezione 3.3.1) presenta un nome differente da `br-int`, occorre aggiustare la configurazione di *linuxnet-ovs-integration-bridge* con il nome identificativo selezionato.

Ricordarsi, ancora una volta, di riavviare i servizi di Openstack presenti sugli host, ovvero Nova-Compute, prima di procedere all'avvio effettivo di Quantum-OpenVSwitch-Agent.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nova-compute &>/dev/null &
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#/etc/init.d/quantum-
                        openvswitch-agent start
[root@c-cloud-01 ~]#
```

Log Utili

Di seguito vengono riportati alcuni log utili in fase di configurazione e debug.

- Servizi di rete generici:

`/var/log/nova/network.log`
`/var/log/nova/nova-dhcpbridge.log`

- Quantum-Server:

  `/var/log/quantum/quantum-server.log`

- Quantum-OpenVSwitch-Agent:

  `/var/log/messages`

- Hypervisor Libvirt Qemu KVM:

  `/var/log/libvirt/libvirtd.log`

3.3.3 Abilitazione e Riavvio rapido di sistema

In caso di riavvio delle macchine presenti nel data center, viene elencata la procedura manuale[8], per riportare ad uno stato stabile il sistema e i servizi Openstack.

Server

Avviare i servizi di utili all'architettura Openstack.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#/etc/init.d/httpd start
[root@c-cloud-01 ~]#/etc/init.d/dhcpd start
[root@c-cloud-01 ~]#/etc/init.d/ntpd start
[root@c-cloud-01 ~]#/etc/init.d/rabbitmq-server start
[root@c-cloud-01 ~]#/etc/init.d/mysqld start
[root@c-cloud-01 ~]#/etc/init.d/iscsi start
[root@c-cloud-01 ~]#/etc/init.d/netbsd-iscsi start
[root@c-cloud-01 ~]#/etc/init.d/tgtd start
[root@c-cloud-01 ~]#
```

Poiché di default, al riavvio, il bridge `br-int` perde l'IP assegnato, occorre ristabilire la connettività sulla rete `10.30.0.0/24` con la modalità seguente.

[8]Eventualmente ottimizzabile tramite opportuini script di avvio

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#ifconfig br-int 10.30.0.1/24
                    ...

[root@c-cloud-01 ~]#iptables -t nat -I POSTROUTING 1
                    -s 10.30.0.0/24 -j MASQUERADE
[root@c-cloud-01 ~]#
```

Abilitare Openvswitch e il modulo `brcombat` .

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#/etc/init.d/openvswitch restart
                    ...

[root@c-cloud-01 ~]#insmod lib/modules
                /"KERNEL_VERSION"/weak-updates
                /openvswitch/brcompat.ko
                    ...

[root@c-cloud-01 ~]#ovs-brcompatd --pidfile --detach
[root@c-cloud-01 ~]#
```

Avviare i servizi Openstack.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#keystone-all &>/dev/null &
[root@c-cloud-01 ~]#glance-api &>/dev/null &
[root@c-cloud-01 ~]#glance-registry &>/dev/null &
[root@c-cloud-01 ~]#nova-api &>/dev/null &
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nova-compute &>/dev/null &
[root@c-cloud-01 ~]#nova-network &>/dev/null &
[root@c-cloud-01 ~]#nova-scheduler &>/dev/null &
[root@c-cloud-01 ~]#
```

Abilitare Quantum-Server e Quantum-OpensVSwitch-Agent

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#/etc/init.d/quantum-server start
[root@c-cloud-01 ~]#/etc/init.d/quantum-
                        openvswitch-agent start
[root@c-cloud-01 ~]#
```

Host

Avviare i servizi di utili all'architettura Openstack.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#/etc/init.d/httpd start
[root@c-cloud-0X ~]#/etc/init.d/ntpd start
[root@c-cloud-0X ~]#/etc/init.d/rabbitmq-server start
[root@c-cloud-0X ~]#/etc/init.d/mysqld start
[root@c-cloud-0X ~]#/etc/init.d/iscsi start
[root@c-cloud-0X ~]#/etc/init.d/netbsd-iscsi start
[root@c-cloud-0X ~]#/etc/init.d/tgtd start
[root@c-cloud-0X ~]#
```

Avendo a disposizione un' unica interfaccia di rete *eth0* (vedi Sottosezione 3.3.1 a pagina 45), considerare che al riavvio di default, viene abilitato l'IP di Management(`10.10.0.0/24`), ma l'interfaccia rimane collegata alla porta del bridge `br-int` (`10.30.0.X/24`) che perde l'IP virtuale. Questo significa, che il riavvio dell'host deve avvenire direttamente dalla macchina e non da remoto, se non si vuole perdere la raggiungibilità di rete.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#ifconfig br-int 10.30.0.1/24
                      ...

[root@c-cloud-0X ~]#route del -net 10.10.0.0/24
[root@c-cloud-0X ~]#route del default
[root@c-cloud-0X ~]#route add default gw 10.30.0.1
[root@c-cloud-0X ~]#
```

Abilitare Openvswitch e il modulo `brcombat` .

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#/etc/init.d/openvswitch restart
                    ...

[root@c-cloud-0X ~]#insmod lib/modules
                /"KERNEL_VERSION"/weak-updates
                /openvswitch/brcompat.ko
                    ...

[root@c-cloud-0X ~]#ovs-brcompatd --pidfile --detach
[root@c-cloud-0X ~]#
```

Avviare i servizi Openstack.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#nova-compute &>/dev/null &
[root@c-cloud-0X ~]#nova-volume &>/dev/null &
[root@c-cloud-0X ~]#
```

Abilitare Quantum-OpensVSwitch-Agent.

```
[root@c-cloud-0X ~]#
[root@c-cloud-0X ~]#/etc/init.d/quantum-
                        openvswitch-agent start
[root@c-cloud-0X ~]#
```

3.4 Test API Quantum

Risulta interessante analizzare, infine, alcuni esempi applicativi di utilizzo del nuovo *Quantum Service*, utili per una primo approccio alle funzionalità introdotte nel sistema e su cui sviluppare successivamente ulteriori fasi di Testbed avanzato.

3.4.1 Interazione tramite CLI

Per la creazione di una Quantum Network è possibile utilizzare sia il Quantum CLI[9] sia il comando di gestione `nova-manage`[10]. Con riferimento a quest' ultimo, per la visualizzazione dell'elenco di reti disponibili digitare

```
nova-manage network quantum_list
nova-manage network list
```

oppure[11]

```
quantum list_nets <tenant-id>
```

É opportuno creare una rete pubblica(`10.50.0.0/24`) su cui tutte le VM lanciate nel data center definiranno un Quantum Attachment e quindi saranno raggiungibili.

```
nova-manage network create
                    --label=public
                    --fixed_range_v4=10.50.0.0/24
```

Per la creazione reti private legate ai singoli *tenants* di accesso, procedere nel modo seguente.

```
nova-manage network create
            --label= <network-name>
            --fixed_range_v4= <network-cidr>
            --project_id= <tenant-UUID-from-keysonte>
```

[9]Per approfondire digitare il comando `quantum` e leggersi le varie opzioni di *help*.
[10]Per opzioni avanzate fare riferimento a [29]
[11]Per visualizzare le reti pubbliche impostare `<tenant-id> = default`

Per lanciare una VM su una particolare rete privata

```
nova boot --image <name-image-from-nova-image-list>
          --flavor <id-from-nova-flavor-list>
          --nic net-id= <network-id-from-network-list>
          INSTANCE_NAME
```

Per eliminare una rete definitivamente

```
nova-manage network delete --uuid= <network-id>
```

3.4.2 Interazione tramite HTTP

L'accesso e l'utilizzo delle API di Quantum Service versione *Essex* non necessita dell'integrazione con Keystone, abilitabile solo opzionalmente[12]. Questo significa che, a differenza degli altri servizi in cui l' interazione via API richiede l'invio di un token temporaneo di identificazione, per Quantum è sufficiente fare semplici richieste sulla porta 9696.

Per comunicare con il sistema utilizzando le API, occorre quindi, inviare richieste HTTP verso il Quantum-Server, impiegando un qualunque client HTTP disponibile in rete, tra cui sono consigliabili:

- **CURL**: client Linux a linea di comando[30].
- **RESTClient**: un estensione di Firefox con una intuitiva interfaccia grafica[31].

Facendo riferimento al client *CURL*, per verificare che il Quantum-Server sia abilitato e visualizzare la versione delle API disponibile utilizzare il seguente comando[13].

```
curl -k -X 'GET' -v 10.10.0.1:9696/
     -H 'Content-type: application/xml'
```

Per un utilizzo avanzato delle API fare riferimento ai template principali riportati in Appendice e alla guida ufficiale [26].

[12] La procedura di abilitazione è disponibile in appendice

[13] Per ottenere una risposta in formato JSON sostituire `-H application/json`

Capitolo 4

Conclusioni

Giunti al termine di questa esperienza di tesi, è possibile fare alcune riflessioni sui concetti approfonditi e la tecnologia coinvolta durante le fasi di attività. Il mondo sta presumibilmente andando nella direzione di sistemi distribuiti intelligenti e situati all' interno dell' ambiente circostante, in cui gli aspetti di rete risultano sicuramente fondamentali.

In questa direzione, la fase di sperimentazione verso un concetto di Cognitive Network Service e verso una gestione ad alto livello di ciò che Internet mette a disposizione rappresenta, sicuramente, una valida strada percorribbile e sviluppabile in vari scenari di riferimento. Openstack fornisce un' idea nuova ed estremamente flessibile di cloud-computing, che unito al TS-Layer potrebbe portare a delle soluzioni innovative all'interno di questo campo di Information and Communication Technology.

Con questa tesi, si è cercato di dare una libera e sentita contribuzione all' attività di ricerca in questa direzione, gettando le basi per le successive fasi di Testbed che verranno portate avanti nei prossimi tempi. In particolare, si è cercato di documentare in maniera chiara ed esplicita i concetti logici e le configurazioni necessarie per abilitare l'utilizzo della piattaforma Openstack e la sua integrazione al nuovo progetto Quantum proposto dalla community.

Le possibilità future auspicabili, consistono in un upgrade della versione corrente Openstack *Essex* verso la recente *Folsom*, rilasciata negli ultimi mesi, in cui il modulo Quantum trova sicuramente la sua massima esspressività, in termini di poternza e stabilità. Questa scelta, potrebbe portare alla risoluzione di molti di quei problemi configurativi che si sono incontrati durante questo lavoro di tesi e

che, effettivamente, fanno ben notare la differenza tra sistemi commerciali e sistemi sperimentali. L'approccio *open source* di Openstack, comunque, rappresenta anche sotto questo punto di vista un grande vantaggio, in quanto la libera informazione e la presenza di una community globale che partecipa attivamente al progetto, fa si che non si è mai gli unici ad incontrare un particolare problema in un determinato momento e permette di porre all'attenzione di tutti, i vari *bug* che si possono incontrare durante gli stati configurativi.

Attualmente, il sistema è pronto per le successive fasi di sperimentazione e migrazione di risorse applicative e di rete all'interno di un ambiente di cloud, a cui sta lavorando il progetto *CIRI ICT* [15] dell'Università di Bologna e a cui, a sua volta, si spera di aver donato un piccolo e personale contributo.

Appendice A

Dettagli di Configurazione

A.1 /etc/nova/nova.conf

```
[DEFAULT]

# LOGS/STATE
verbose=True
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova

# AUTHENTICATION
auth_strategy=keystone

# SCHEDULER
compute_scheduler_driver=nova.scheduler.
                filter_scheduler.FilterScheduler

# VOLUMES
volume_group=nova-volumes
volume_name_template=volume-%08x
iscsi_helper=tgtadm

# DATABASE
sql_connection=mysql://novaAdmin:novaSecret@10.10.0.1/nova
```

```
# COMPUTE
libvirt_type=kvm
connection_type=libvirt
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.
                contrib.standard_extensions
ec2_dmz_host=10.10.0.1
s3_host=10.10.0.1

# RABBITMQ
rabbit_host=10.10.0.1
rabbit_port = 5673

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=10.10.0.1:9292

# NETWORK
#network_manager=nova.network.manager.FlatDHCPManager

network_manager=nova.network.quantum.manager.QuantumManager

#force_dhcp_release=True
#dhcpbridge_flagfile=/etc/nova/nova.conf
#firewall_driver=nova.virt.libvirt.
                firewall.IptablesFirewallDriver
## Change my_ip to match each host
#my_ip=10.10.0.1
##public_interface=virbr0
##vlan_interface=br100
#flat_network_bridge=br100
```

```
#flat_interface=eth0
#fixed_range=10.20.0.0/24
dhcpbridge=/usr/bin/nova-dhcpbridge

# QUANTUM
quantum_connection_host=10.10.0.1
quantum_connection_port=9696
quantum_use_dhcp=True
# OPENVSWITCH
libvirt_vif_driver=nova.virt.libvirt.
                vif.LibvirtOpenVswitchDriver
libvirt_vif_type=ethernet
linuxnet_interface_driver=nova.network.
                linux_net.LinuxOVSInterfaceDriver
linuxnet_ovs_integration_bridge=br-int

# NOVNC CONSOLE
novncproxy_base_url=http://10.10.0.1:6080/vnc_auto.html
# Change vncserver_proxyclient_address and
#               vncserver_listen to match each compute host
vncserver_proxyclient_address=10.10.0.1
vncserver_listen=10.10.0.1
```

A.2 /etc/quantum/quantum.conf

```
[DEFAULT]
#RABBITMQ
rabbit_host = 10.10.0.1
rabbit_port = 5673

#CORE PLUGIN
core_plugin = quantum.plugins.openvswitch.
                ovs_quantum_plugin.OVSQuantumPlugin

# Show more verbose log output (sets INFO log level output)
```

```
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = True

# Address to bind the API server
bind_host = 0.0.0.0

# Port the bind the API server to
bind_port = 9696

# Path to the extensions.  Note that this can be a
# colon-separated list of
# paths.  For example:
# api_extensions_path =
# extensions:/path/to/more/extensions:/even/more/extensions
# The __path__ of quantum.extensions is appended to this,
# so if your extensions are in there you don't
# need to specify them here
api_extensions_path =

[composite:quantum]
use = egg:Paste#urlmap
/: quantumversions
/v1.0: quantumapi_v1_0
/v1.1: quantumapi_v1_1

[pipeline:quantumapi_v1_0]
# By default, authentication is disabled.
# To enable Keystone integration uncomment the
# following line and comment the next one
pipeline = extensions quantumapiapp_v1_0
#pipeline = authN extensions quantumapiapp_v1_0

[pipeline:quantumapi_v1_1]
# By default, authentication is disabled.
```

```
# To enable Keystone integration uncomment the
# following line and comment the next one
pipeline = extensions quantumapiapp_v1_1
#pipeline = authN extensions quantumapiapp_v1_1

[filter:authN]
paste.filter_factory = keystone.middleware.
                quantum_auth_token:filter_factory
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
auth_version = 2.0

auth_admin_user = <user-value>
auth_admin_password = <user-pass>
#auth_admin_tenant_name = <tenant-name>

#auth_admin_token = <token-value>

[filter:extensions]
paste.filter_factory = quantum.extensions.extensions:
                plugin_aware_extension_middleware_factory

[app:quantumversions]
paste.app_factory = quantum.api.versions:Versions.factory

[app:quantumapiapp_v1_0]
paste.app_factory = quantum.api:APIRouterV10.factory

[app:quantumapiapp_v1_1]
paste.app_factory = quantum.api:APIRouterV11.factory
```

A.3 Creazione Pacchetti RPM per OpenVSwitch

Installare i pacchetti necessari a creare gli RPM[1].

```
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#yum install rpm-build
[root@c-cloud-07 ~]#yum install redhat-rpm-config
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#yum install make gcc pkgconfig
    openssl openssl-devel glibc-devel iproute
    kernel-devel git python python-zope-interface
    python-twisted-conch PyQt4 wge
[root@c-cloud-07 ~]#
```

In *CentOS 6.3* è necessario aggiornare `autoconf` ad una versione superiore la `2.63` .

```
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#wget http://ftp.gnu.org/gnu
        /autoconf/autoconf-latest.tar.gz
[root@c-cloud-07 ~]#./configure
[root@c-cloud-07 ~]#make
[root@c-cloud-07 ~]#make install
```

Per proteggere le librerie di sistema da eventuali corruzioni, conviene definire, per i passsi successivi, uno *user* che non possegga i privilegi di *root* .

```
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#adduser ovswitch
[root@c-cloud-07 ~]#su - ovswitch
[root@c-cloud-07 ~]$
```

Aggiungere la directoy per la creazione dei pacchetti.

```
[root@c-cloud-07 ~]$
[root@c-cloud-07 ~]$mkdir -p ~/rpmbuild/
```

[1]Procedura effettuata su host `c-cloud-07`

```
        {BUILD,RPMS,SOURCES,SPECS,SRPMS}
[root@c-cloud-07 ~]$echo '%_topdir %(echo $HOME)/
        rpmbuild' > ~/.rpmmacros
[root@c-cloud-07 ~]$
```

Sacricare i sorgenti di OpenVSwitch tramite *git*.

```
[root@c-cloud-07 ~]$
[root@c-cloud-07 ~]$git clone
        git://openvswitch.org/openvswitch
                    ...

[root@c-cloud-07 ~]$./boot.sh
[root@c-cloud-07 ~]$./configure
        --with-linux=/lib/modules/`uname -r`/build
[root@c-cloud-07 ~]$make
[root@c-cloud-07 ~]$
```

Creare un archivio della configurazione e copiarlo nella directory `SOURCES` .

```
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#make dist
                    ...
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#cp openvswitch-1.9.90.tar.gz
        /home/ovswitch/rpmbuild/SOURCES/
                    ...
[root@c-cloud-07 ~]#tar xzvf openvswitch-1.9.90.tar.gz
[root@c-cloud-07 ~]#
```

Inserire un *fix* per la creazione dei pacchetti in *Linux CentOS 6.3*.

```
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#cat << EOF >>
                    rhel/openvswitch.spec
[root@c-cloud-07 ~]#/usr/bin/ovsdbmonitor
[root@c-cloud-07 ~]#/usr/share/applications/
```

```
                    ovsdbmonitor.desktop
[root@c-cloud-07 ~]#/usr/share/man/man1/
                    ovsdbmonitor.1.gz
[root@c-cloud-07 ~]#/usr/share/ovsdbmonitor/*
[root@c-cloud-07 ~]#EOF
[root@c-cloud-07 ~]#
```

Procedere alla creazione degli RPM nel modo seguente.

```
[root@c-cloud-07 ~]#
[root@c-cloud-07 ~]#rpmbuild -bb rhel/openvswitch.spec
                    ...
[root@c-cloud-07 ~]#rpmbuild -bb rhel/
                    openvswitch-kmod-rhel6.spec
[root@c-cloud-07 ~]#
```

Nella directory `RPMS` dovrebbero ora trovarsi 3 pacchetti:

```
openvswitch- kernel vers
openvswitch-debuginfo- kernel vers
kmod-openvswitch
```

A.4 Integrazione di Quantum con il servizio Keystone

Le richieste tramite l'uso delle API, verso il Quantum-Service, posso essere autenticate da Openstack Keystone Identity Service, tramite un protocollo basato sull' invio di un token temporaneo di identificazione. Il progetto Quantum, si trova ancora in fase sperimentale, quindi di default non viene proposta tale integrazione e il servizio è contattabile senza protezione sulla propria porta.

Per abilitare le specifiche di integrazione tra Quantum e Keystone procedere come segue[2].

Prima di tutto è necessario creare una Quantum Service Entry.

[2]Questa opzione non è attualmente abilitata nel data center.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#keystone service-create
            --name quantum
            --type network
            --description=
            "Openstack Networking Service"
                    ...

[root@c-cloud-01 ~]#keystone endpoint-create
            --region myregion
            --service_id <service-id>
            --publicurl="http://10.10.0.1:9696/"
            --internalurl="http://10.10.0.1:9696/"
            --adminurl="http://10.10.0.1:9696/"
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#
```

Definire un Quantum Service User.

```
[root@c-cloud-01 ~]#keystone user-create
            --name quantum
            --pass quantum
            --tenant_id <tenant-id>
            ...

[root@c-cloud-01 ~]#keystone user-role-add
            --token <token-value>
            --endpoint http://10.10.0.1:35357/v2.0
            user-role-add
            --user <quantum-user-id>
            --tenant_id <service-tenant-id>
            --role <admin-role-id>
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#
```

Di seguito vengono riportati alcuni comandi utili di controllo.

```
keystone role-list
keystone user-list
keystone tenant-list
keystone endpoint-list
```

Occorre configurare `quantum.conf` per l'interazione tramite Keystone.

```
[root@c-cloud-01 ~]#
[root@c-cloud-01 ~]#nano /etc/quantum/quantum.conf
                    ...

[pipeline:quantumapi_v1_0]
#pipeline = extensions quantumapiapp_v1_0
pipeline = authN extensions quantumapiapp_v1_0

[pipeline:quantumapi_v1_1]
#pipeline = extensions quantumapiapp_v1_1
pipeline = authN extensions quantumapiapp_v1_1
                    ...
```

Per l'identificazione del servizio Quantum, si può utilizzare sia *user* e *password* sia un *token*[3].

```
[filter:authN]
paste.filter_factory = keystone.middleware.
            quantum_auth_token:filter_factory
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
auth_version = 2.0
auth_admin_user = <user-value>
auth_admin_password = <pass-value>
#auth_admin_tenant_name = <service-name>
#auth_admin_token = <token-value>
```

Per ricevere il *token* di riconoscimento tramite HTTP, seguire il template seguente:

[3]Se li abilito entrambi, il token è prioritario.

- *Destinazione*:

```
http://137.204.X.X:5000/v2.0/tokens
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
{
    "auth":{
        "passwordCredentials":{
            "username":"admin",
            "password":"admin"
        },
        "tenantName":"admin"
    }
}
```

Il *token* di ritorno va inserito nel `Header` delle richieste HTTP successive nel formato:

```
X-Auth-Token: <token-value>
```

Appendice B

Template interazione API Quantum

B.1 Quantum Networks

Lista Networks

- *Destinazione*:

  ```
  GET http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/detail
  ```

- *Header*:

  ```
  Content-Type: application/xml
  ```

- *Body*:

  ```
  NONE
  ```

Dettagli Network

- *Destinazione*:

  ```
  GET http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                                          <network-id>/detail
  ```

- *Header*:

  ```
  Content-Type: application/xml
  ```

- *Body*:

  ```
  NONE
  ```

Creazione Network

- *Destinazione*:

```
POST http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
<network
    name="<network-name>"/>
```

Eliminazione Network

- *Destinazione*:

```
DELETE http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                                                    <network-id>
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
NONE
```

B.2 Quantum Ports

Lista Ports

- *Destinazione*:

```
GET http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                                  <network-id>/ports/detail
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
NONE
```

Dettagli Port

- *Destinazione*:

```
GET http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                          <network-id>/ports/<port-id>/detail
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
NONE
```

Creazione Port

- *Destinazione*:

```
POST http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                                          <network-id>/ports
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
<port
    state="ACTIVE"/>
```

Eliminazione Port

- *Destinazione*:

```
DELETE http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                                  <network-id>/ports/<port-id>
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
NONE
```

B.3 Quantum Attachment

Lista Attachments per una Port

- *Destinazione*:

```
GET http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                    <network-id>/ports/<port-id>/attachment
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
NONE
```

Creazione Attachment

- *Destinazione*:

```
PUT http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                    <network-id>/ports/<port-id>/attachment
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
<attachment
    id="test_interface_identifier"/>
```

Eliminazione Attachment

- *Destinazione*:

```
DELETE http://<IP_ADDRESS>:9696/v1.0/tenants/<tenant-id>/networks/
                    <network-id>/ports/<port-id>/attachment
```

- *Header*:

```
Content-Type: application/xml
```

- *Body*:

```
NONE
```

Bibliografia

Articoli di riferimento

[1] D.D.Clark et al., *A Knowledge Plane for the Internet*, Proc. SIGCOMM'03, Karlsruhe, Germania, agosto 2011

[2] A.Galis, H.Abramowicz et al., *Management and Service-Aware Networking Architectures for Future Internet*, Mana Future Internet group, maggio 2009

[3] A.Manzalini et al., *Self-Optimized Cognitive Network of Networks*, Computer J., Oxford Univ., febbraio 2011

[4] F.Callegati, W.Cerroni, A.Campi, *Application Scenarios for Cognitive Transport Service in Next-Generation Networks*, DEIS, Bologna Univ., marzo 2012

[5] F.Callegati, W.Cerroni, A.Campi, *Automated Transport Service Management in the Future Internet: Concepts and Operations*, DEIS, Bologna Univ., gennaio 2011

[6] J.Rosenberg et al., *SIP: Session Initiation Protocol*, IETF, RFC 3261, giugno 2002

[7] F.Callegati, A.Campi, *Network Resource Description Language*, IEEE GLOBECOM 2009, Honolulu, HI, dicembre 2009

[8] ITU-T Y.2001, *General Overview of NGN*, dicembre 2004 `http://www.itu.int/en/ITU-T/gsi/ngn/Pages/definition.aspx`

[9] K.Knighson et al., *NGN Architecture: Generic Principles Functional Architecture, and Implementation*, IEEE, Communications Magazine, ottobre 2005

[10] O.Lassila, *Resource Description Framework (RDF) Model and Syntax Specification* W3C Recommendation, febbraio 1999 `http://www.w3.org/standards/techs/rdf#w3c_all`

[11] T.Berners-Lee, R.Fielding, and L.Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*, IETF, RFC 2396, agosto 1998 `http://www.ietf.org/rfc/rfc2396.txt`

[12] L.Richardson, and S.Ruby, *RESTful Web Services*, O'Reilly, 2007

Siti di riferimento

[13] Wikipedia - `http://www.wikipedia.org`.

[14] IEEE Explore - `http://ieeexplore.ieee.org`

[15] CIRI ICT - `http://www.ciri-ict.unibo.it`

[16] CentOS - `http://www.centos.org/`

[17] Openstack - `http://http://www.openstack.org/`.

[18] Documentazione Openstack - `http://docs.openstack.org/`.

[19] Openstack wiki - `http://wiki.openstack.org/`.

[20] Configurazione Openstack - `http://docs.openstack.org/essex/openstack-compute/install/yum/content/`.

[21] Openstack Compute - `http://docs.openstack.org/trunk/openstack-compute/admin/content/`.

[22] Openstack Service Architecture - `http://docs.openstack.org/trunk/openstack-compute/admin/content/service-architecture.html`.

[23] Openvswitch wiki - `http://openvswitch.org/`.

[24] Documentazione Openvswitch - `http://openvswitch.org/support/`.

[25] Quantum e Opensvswitch - `http://openvswitch.org/openstack/openstack-blog/`.

[26] Quantum API (1.0) - http://docs.openstack.org/api/openstack-network/1.0/content/Show_Attachment.html.

[27] Quantum wiki - http://wiki.openstack.org/Quantum.

[28] Configurazione Quantum - http://docs.openstack.org/trunk/openstack-network/admin/content/.

[29] Configurazione Quantum Essex - http://docs.openstack.org/trunk/openstack-network/admin/content/Net-Create-dle455.html

[30] Client HTTP Curl - http://curl.haxx.se/

[31] Client HTTP RESTClient - https://addons.mozilla.org/en-US/firefox/addon/restclient/

Ringraziamenti

Stay hungry, stay foolish...

Steve Jobs

Concluso questo elaborato di tesi, è giunto il momento di fare alcuni ringraziamenti.

Un mio primo pensiero lo rivolgo ai miei genitori, Anna e Marco, che mi sento di ringraziare per l'opportunità che mi hanno dato fino a questo momento nel costruirmi un percorso di istruzione che probabilmente sarà fondamentale per il resto della mia vita e che intendo proseguire nei prossimi due anni. Grazie, per esserci sempre stati, per avermi sempre sostenuto anche nei momenti più difficili che posso aver incontrato fino ad ora e che probabilmente incotrerò in futuro.

Ringrazio, inoltre, il professor Franco Callegati per la sua disponibilità nell'avermi fornito sempre gli agganci giusti, sia in fase di tirocinio, sia poi nell'impostazione e nella scelta dell'argomento di ricerca che ho avuto modo di approfondire in questa tesi. Allo stesso modo, ringrazio in professor Walter Cerroni, vero e proprio punto di riferimento nei momenti di difficoltà che ho incontrato durante le fasi di configurazione e grazie a cui son riuscito a risolvere la maggior parte dei problemi a cui sono andato incontro.

Un sentito ringraziamento, infine, a tutti i miei amici, nessuno escluso, con cui ho passato e continuo a passare le serate e i momenti migliori, al di fuori di tutto ciò che riguarda l' ambito scolastico e professionale.

Grazie a tutti.

Fabio Paolucci,
Gatteo, 11 dicembre 2012