UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Facoltà di Ingegneria

Corso di Dottorato di Ricerca in Ingegneria Informatica ed Automatica
XXI Ciclo
Dipartimento di Informatica e Sistemistica

UN PARADIGMA INNOVATIVO PER MODERNI SERVIZI DI RETE, IL SERVICE SWITCHING

VITTORIO MANETTI

Tesi di Dottorato

TUTOR
Prof. Giorgio Ventre

COORDINATORE
Prof. Luigi Pietro Cordella

COTUTOR
Prof. Ioannis Stavrakakis

Dicembre 2008

Ringraziamenti

Desidero ringraziare tutte le persone che sono state per me un supporto durante il triennio di Dottorato, quindi colleghi, amici, e naturalmente la mia famiglia.

Un grazie infinito al Prof. Giorgio Ventre, per aver creduto nelle mie potenzialità e per avermi offerto la possibilità di vivere un percorso così ricco di esperienze importanti e impegnative, ma molto spesso anche piacevoli e divertenti.

Ringrazio il Prof. Roberto Canonico, il quale mi ha fornito numerose e importanti indicazioni, ed è stato un fondamentale riferimento per il lavoro di tesi, ed il Prof. Simon Pietro Romano, i cui suggerimenti sono sempre preziosissimi.

Ringrazio l'intero gruppo di ricerca COMICS, composto da amici prima ancora che colleghi, ed in particolare Alessandro, Claudio e Tobia, agli "amici dell'aperitivo", i quali in un momento un po' particolare della mia vita, mi hanno aiutato a ritrovare il sorriso.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La rete Internet è stata fondata su un modello molto semplice che prevede da un lato l'uso di nodi localizzati all'interno della rete e responsabili dell'instradamento dei pacchetti da sorgente a destinazione, e dall'altro quello di programmi applicativi in esecuzione su sistemi di elaborazione dislocati invece ai margini. Tuttavia, negli ultimi anni la netta separazione tra processi applicativi e meccanismi per l'instradamento dei pacchetti, quindi tra l'infrastruttura di calcolo e quella di rete, è andata sfumando. Esiste infatti una folta categoria di applicazioni distribuite, in grado di prendere decisioni relative all'instradamento in maniera completamente autonoma, e allo stesso tempo, esistono elementi di rete in grado di eseguire specifiche operazioni a livello applicativo in funzione del contenuto dei pacchetti processati. Questo è in effetti il risultato della convergenza di due comunità di ricerca storicamente separate tra loro, la *Distributed Systems Community* (comunità di ricerca sui sistemi distribuiti) e la *Network Community* (comunità di ricerca sulle reti di calcolatori). Tradizionalmente, la prima vede la rete come un semplice insieme di condutture per i bit, utile esclusivamente alla trasmissione di informazioni tra macchine localizzate in posizioni geografiche distinte, mentre l'altra si preoccupa dell'instradamento dei pacchetti senza alcun riguardo alla semantica delle applicazioni. Consapevoli, però, della sinergia che è possibile ottenere se le risorse disponibili all'interno della rete si muovono in maniera coerente rispetto ai requisiti delle applicazioni, entrambe le comunità lavorano al potenziamento ed all'incremento di sistemi dislocati nella rete, all'interno dei quali siano cablate delle funzionalità di alto livello [33].

In relazione a tale contesto, definiamo un paradigma innovativo per moderni servizi di rete, il *Service Switching*, un modello di comunicazione basato sul concetto di mobilità dei servizi, e proponiamo una architettura che supporta tale modello, il nodo di rete *Service Switch*.

La mobilità dei servizi ricopre un ruolo di rilevante importanza nelle reti di nuova generazione; nel contesto dell'*ubiquitous computing*, ad esempio, in cui utenti nomadi si spostano all'interno della rete muovendosi tra ambienti eterogenei, il supporto alla mobilità aiuta ad incrementare la qualità percepita nella fruizione del generico servizio. In piattaforme per l'hosting di servizi, invece, è di fondamentale importanza curare aspetti legati alla gestione efficiente delle risorse disponibili, ed implementare meccanismi finalizzati al bilanciamento del carico computazionale a beneficio di risorse particolarmente sovraccariche; poter migrare servizi su nodi differenti di una infrastruttura o in generale della rete, consente di raggiungere tali obiettivi. In sistemi di rete per la distribuzione di servizi, è inoltre indispensabile poter fare affidamento su meccanismi che si occupino di rilevare eventuali condizioni critiche, e reagire a tali condizioni in modo da garantire continuità nella fornitura dei servizi, quindi un adeguato livello di resilienza. Molto spesso, la migliore soluzione per far fronte a condizioni di questo tipo, consiste proprio nell'eseguire il trasferimento di un intero insieme di servizi e del relativo stato di esecuzione, dalla zona in cui si è verificato il fallimento, ad una differente porzione dell'infrastruttura.

Il Service Switching è un modello per la gestione di servizi di rete innovativi, che offre la possibilità ad un *Application Service Provider* (ASP) di fornire servizi ai proprio utenti sfruttando le risorse messe a disposizione dal sistema, ed un supporto alla mobilità di servizio che ne rende trasparente la posizione corrente all'interno di una architettura geograficamente distribuita.

La principale componente del modello proposto è il Service Switch, un nodo di rete che oltre ad implementare i meccanismi tradizionali di *packet* e *flow switching*, possiede caratteristiche innovative, tra le quali la capacità di ospitare servizi ed applicazioni assieme ai rispettivi dati, fornendo loro un ambiente di esecuzione sicuro e

differenziato. Ogni nodo coordina il proprio comportamento considerando uno scenario globale e le azione compiute dagli altri Service Switch, con l'obiettivo di garantire un uso ottimale delle risorse disponibili e di bilanciare il carico computazionale sull'intera architettura. Un nodo Service Switch può essere localizzato nel cuore della rete oppure sulla sua frontiera, ed implementare quindi, a seconda dei casi, un preciso insieme di funzionalità. Un ulteriore incarico assegnato al nodo, è quello di rilevare eventuali condizioni di malfunzionamento e di reagire a queste ultime attivando un meccanismo di recupero e riallocazione dei servizi.

La nascita di efficienti tecniche di virtualizzazione per moderni e potenti processori conduce ad un largo impiego di tali tecniche nel contesto del *service hosting*; le macchine virtuali offrono la possibilità di creare e sfruttare ambienti di esecuzione di tipo *self-contained*, isolati, e completamente indipendenti dall'hardware di sistema. Consapevoli delle potenzialità offerte da tali strumenti, e in accordo con simili scelte progettuali, proponiamo una particolare implementazione del modello Service Switching basata sull'uso della virtualizzazione. L'architettura proposta dispone di un meccanismo di instradamento ispirato al modello Mobile IP, finalizzato a garantire la corretta consegna di pacchetti destinati a servizi migrati, e sfrutta le potenzialità del protocollo MPLS e di alcuni meccanismi di forwarding innovativi opportunamente realizzati, per offrire canali sicuri ed affidabili alle comunicazioni tra i nodi Service Switch.

Capitolo 1
Il paradigma Service Switching

1.1 I modelli di switching tradizionali

Il più semplice modello di comunicazione è costituito da una interconnessione diretta tra gli interlocutori normalmente identificati come sorgente e destinazione, attraverso cui le informazioni possano viaggiare senza necessariamente richiedere l'uso di alcun intermediario. L'utilizzo di tale modello richiede ovviamente l'impiego di un particolare supporto che consenta di eseguire il trasferimento diretto di informazioni. In caso alternativo, modelli di comunicazione più avanzati richiedono l'introduzione di entità intermedie che siano in grado di inoltrare dati in accordo ad uno specifico meccanismo di *switching* (commutazione).

Nel conteso dei sistemi di rete, lo switching può essere definito come la proprietà degli elementi di inoltrare dati, ed esistono oggi svariate categorie di reti di comunicazione basate su differenti tecnologie e differenti modelli di switching. Il principale compito affidato alla rete è quello di implementare un meccanismo di switching coordinato, mediante il quale le informazioni possano essere inoltrate da una o più sorgenti ad una o più destinazioni. Con l'obiettivo di fornire differenti servizi a differenti tipologie di traffico, durante gli anni sono stati proposti diversi modelli di switching.

Nelle reti per la telefonia, i terminali sono i telefoni e le entità intermedie i commutatori; ciò che viene trasmesso sottoforma di segnale elettrico, continuo e modulato, è la voce. La telefonia tradizionale è basata sul modello del *circuit switching* (commutazione di circuito), tramite il quale è possibile scambiare messaggi vocali mediante l'uso temporaneo ed esclusivo di supporti di comunicazione, come ad esempio

cavi in rame o fibra ottica, connessi tra loro grazie ad una opportuna configurazione dei commutatori. Si ottiene commutazione di circuito quando una frazione fissa della capacità trasmissiva è stabilmente allocata a ciascun canale, in modo che ciascun utilizzatore abbia a disposizione un canale trasmissivo dedicato con la garanzia di poterne utilizzare l'intera capacità fino a quando la connessione non venga abbattuta. Risulta evidente che in un simile contesto i terminali sono entità molto semplici, l'unico compito ad essi assegnato è infatti quello di eseguire la trasduzione del segnale vocale, e la conversione in binario nel caso si tratti di telefoni digitali. Le entità intermedie, i commutatori, non devono invece agire sul segnale trasmesso; ad essi è affidato il semplice compito di configurare un percorso fisico tra sorgente e destinazione, in maniera trasparente sia nei riguardi degli interlocutori che delle informazioni trasportate.

Quando invece le entità coinvolte nella comunicazione sono dei calcolatori, l'interazione avviene in accordo allo schema "question/thinking/answer/thinking", implementando il quale il canale per la comunicazione non risulta costantemente occupato; le comunicazioni tra calcolatori, inoltre, avvengono di solito scambiando dati in maniera non continua ma discreta. Fatte tali ipotesi, è probabilmente più conveniente usare risorse di comunicazione in accordo ad un modello condiviso piuttosto che esclusivo; il modello di comunicazione da imitare che forse più si avvicina a tali requisiti, è quello postale, in cui le informazioni passano attraverso dei nodi di spedizione prima di raggiungere la destinazione finale. Ha effettivamente senso eseguire tale accostamento considerato che nelle comunicazioni tra calcolatori le informazioni sono atomiche e discrete, ogni pezzo di informazione è identificabile, e l'invio dei dati può essere eseguito pezzo per pezzo. Nasce quindi il modello del *packet switching* (commutazione di pacchetto), una tecnica di accesso multiplo a ripartizione nel tempo, utilizzata per condividere un canale di comunicazione tra più stazioni in modo non deterministico. Col packet switching l'informazione da trasmettere viene suddivisa in pacchetti ad ognuno dei quali è associata un'intestazione contenente informazioni necessarie ad inoltrare il pacchetto stesso verso la destinazione finale. Il flusso di dati viene quindi suddiviso in pacchetti inoltrati individualmente attraverso la rete, e

successivamente riassemblato nella sua forma originale all'arrivo sulla stazione di destinazione. Nel momento in cui un pacchetto raggiunge un nodo di commutazione all'interno della rete, quest'ultimo sceglierà il percorso migliore su cui inoltrarlo verso la destinazione, decisione che potrà fornire risultati differenti a seconda delle condizioni della rete, anche per pacchetti appartenenti al medesimo flusso di dati. Nel caso in cui ci siano più pacchetti da processare contemporaneamente, questi verranno memorizzati all'interno di una coda in attesa che arrivi il loro turno; in sostanza, un pacchetto che attraversa la rete subisce un ritardo, in parte legato alle caratteristiche del percorso sul quale viene inoltrato, ed in parte allo stato di carico della rete. In definitiva, se con la commutazione di circuito la capacità del canale trasmissivo è interamente dedicata ad una specifica comunicazione, con la commutazione di pacchetto i canali trasmissivi sono condivisi e vengono utilizzati solo per il tempo strettamente necessario. D'altro canto, è necessario che i nodi di commutazione siano in grado di implementare delle funzionalità complesse: devono poter memorizzare temporaneamente i pezzi di informazione per consentire il loro instradamento in modalità asincrona, e devono essere in grado di configurare il loro comportamento in accordo ad un approccio di tipo coordinato.

Ancora, in relazione a particolari tipologie di applicazione come ad esempio quelle multimediali, i calcolatori non scambiano tra loro dati atomici, bensì flussi continui di informazione. In tal caso è conveniente prendere in considerazione un modello di comunicazione in cui i dati non vengano scambiati sottoforma di bit, bensì sottoforma di strem; il sistema di comunicazione deve assicurare che ogni singola componente dello stream riceva il medesimo trattamento dagli elementi di rete. Supposto che i dati continuino ad essere scambiati in modo discreto, questo tipo di modello risulta essere effettivamente realizzabile purché venga introdotto un meccanismo per l'identificazione del flusso o dell'aggregato di flussi, e purché l'instradamento venga eseguito rispettando un approccio di tipo per-flow (per-flusso). Nasce quindi il *flow switching* (commutazione di flusso). I requisiti di tale modello richiedono l'aggiunta di una certa complessità sui nodi di commutazione, i quali devono essere in grado di implementare delle funzionalità complesse; dovranno innanzitutto eseguire un meccanismo di

commutazione veloce e preciso, in modo da ridurre il più possibile il tempo necessario ad instradare i flussi, ed essere in grado inoltre di memorizzare e manipolare in maniera differenziata differenti unità di dati. Ad ogni flusso o aggregato di flussi sono associate delle informazioni di stato, indispensabili per l'identificazione ed il trattamento differenziato degli stessi; i nodi di commutazione devono essere in grado di memorizzare e gestire tali informazioni. Inoltre, in riferimento ai modelli precedentemente descritti, con la commutazione di flusso è necessario incrementare la capacità dei nodi a coordinare il proprio comportamento al fine di garantire la selezione del percorso più appropriato per l'instradamento dei flussi.

1.2 Introduzione al paradigma Service Switching

Esistono tuttavia alcune categorie di applicazioni e servizi, per i quali i modelli di switching tradizionali non risultano pienamente adeguati. Servizi quali ad esempio il Transcoding o il Content Delivery, che molto spesso conviene implementare nel cuore della rete piuttosto che su end-system; oppure servizi come la Remote Autentication, da eseguire in maniera automatica e completamente trasparente. Servizi di rete innovativi come ad esempio il VoIP, oppure l'Intrusion Detection, che presentano requisiti molto stringenti in relazione ad aspetti quali la resilienza, la scalabilità e la sicurezza, e per i quali può nascere l'esigenza di implementare meccanismi di migrazione, riallocazione o replicazione, affinché tali requisiti siano verificati, e affinché sia possibile gestire condizioni critiche.

Inoltre, sono oggi disponibili diverse innovazioni tecnologiche che possono agevolare l'esecuzione e la gestione di servizi complessi come quelli sopra elencati; l'enorme capacità di trasferimento dati all'interno della rete ottenibile utilizzando ad esempio l'optical switching, abbatte i vincoli legati alla quantità di dati da trasferire ed invita ad incrementare l'utilizzo di tecniche quali ad esempio la migrazione. La grossa capacità computazionale offerta da sistemi multi computer cluster-based, assieme alla disponibilità di memorie a basso costo e di dispositivi per lo storage ad accesso rapido,

consente di costruire una intera server farm utilizzando un singolo cluster di computer. Ancora, processori veloci e riconfigurabili completamente dedicati al networking, consentono di disaccoppiare il processing legato all'instradamento di traffico dati dal carico computazionale di tipo general pourpose, ed incrementare fortemente le prestazioni di un sistema che ne faccia uso.

Da tali considerazioni nasce l'idea del Service Switching, alla base del quale compare il concetto di mobilità delle applicazioni, di spostamento di servizi piuttosto che di dati atomici o flussi di dati. Il Service Switching è un paradigma innovativo per moderni servizi di rete, un nuovo modello di comunicazione che supporta la migrazione di servizio tra nodi distribuiti geograficamente, procedura eseguita in maniera del tutto trasparente nei riguardi dell'utente e del servizio stesso.

Nel contesto in esame, un servizio può essere identificato come un insieme di applicazioni in esecuzione sulla medesima piattaforma di cui sfrutta le risorse disponibili, e la cui esecuzione risulta subordinata ad una specifica richiesta e finalizzata al raggiungimento di uno specifico risultato/obiettivo. L'entità *servizio* nel modello Service Switching è da intendere come appena descritto, quindi come l'insieme di un certo numero di applicazioni in esecuzione concorrente sul medesimo ambiente di esecuzione, che siano in grado di fornire una risposta consistente ad una richiesta specifica.

La principale componente del modello proposto è il Service Switch, un nodo di rete che oltre ad implementare i meccanismi tradizionali di packet e flow switching, possiede caratteristiche innovative; il Service Switch è in grado di ospitare servizi ed applicazioni assieme ai rispettivi dati, fornendo loro un ambiente di esecuzione sicuro, isolato, e differenziato a seconda delle particolari necessità, ed abilitando per ognuno meccanismi per l'attivazione on-demand ed il bootstrap automatico. Sfruttando un opportuno meccanismo di prenotazione implementato in accordo a politiche di scheduling ben definite, il Service Switch assegna risorse computazionali, memoria e risorse di rete ad ogni servizio ospitato. Ogni nodo coordina il proprio comportamento considerando uno scenario globale e le azione compiute dagli altri Service Switch, con l'obiettivo di

garantire un uso ottimale delle risorse disponibili e di bilanciare il carico computazionale sull'intera architettura. Interagendo tra loro attraverso canali sicuri realizzati utilizzando specifiche tecnologie, i nodi sono in grado di gestire in maniera automatica tutte le fasi del ciclo di vita del generico servizio, e di rilevare eventuali condizioni critiche legate ad esempio al verificarsi del fallimento di una specifica componente, e reagire a queste modificando lo schema di allocazione dei servizi.

Un Service Switch può, a seconda dei casi, essere localizzato sulla frontiera di un *Autonomous Sytem Domain* per gestire e monitorare i servizi, oppure nel core della rete per supportarne la migrazione e la riconfigurazione rapida, e per riallocare il carico in condizioni critiche.

La principale prerogativa del modello proposto è quella di supportare attraverso le funzionalità offerte dai Service Switch, un meccanismo di migrazione di servizio tra nodi geograficamente distribuiti, in maniera del tutto trasparente nei riguardi dell'utente e dell'applicazione. Eseguire la migrazione di servizio può avere scopi molteplici: agevola il processo di riallocazione dei servizi ed aiuta quindi ad incrementare il livello di resilienza, e consente di implementare schemi per il bilanciamento del carico computazionale e quindi per l'uso ottimo delle risorse. Inoltre, esistono in letteratura diversi lavori recentemente introdotti che propongono la migrazione come soluzione al problema del posizionamento ottimale del punto di fornitura di un servizio all'interno di una infrastruttura [19, 20, 21]. Tali modelli si basano sull'impiego di politiche per la mobilità dei servizi basate su informazioni globali; utilizzando un approccio di tipo step-by-step, i meccanismi in questione muovono i servizi verso la posizione ottimale sfruttando la migliore traiettoria di migrazione, con l'obiettivo di avvicinare il punto di fornitura del servizio all'area da cui proviene maggiore richiesta per il servizio stesso, in modo da minimizzare l'uso delle risorse di comunicazione e di migliorare la QoS (*Quality of Service*) per i servizi offerti. Tali meccanismi sono di tipo adattativo, nel senso che supportano la riconfigurazione dello schema di localizzazione dei servizi in funzione di variazioni alla topologia e all'intensità della domanda dei servizi.

1.3 Definizione di un accordo per la fornitura di servizi

Cerchiamo adesso di individuare le entità con cui i nodi Service Switch interagiscono, e le modalità con cui tali interazioni vengono eseguite; l'architettura Service Switching offre la possibilità ad un Application Service Provider (ASP) di fornire servizi ai proprio clienti sfruttando le risorse messe a disposizione dal sistema, per cui il generico ASP può essere identificato come il fruitore o cliente dell'architettura Service Switching. L'interazione tra i nodi Service Switch e l'ASP è completamente trasparente nei riguardi delle applicazioni e dei loro utenti.

Introduciamo il concetto di *Service Level Agreement* (accordo sul livello di servizio) [50]. Lo SLA è uno strumento contrattuale stabilito tra due soggetti, utilizzando il quale è possibile definire le metriche di servizio che devono essere rispettate da un Service Provider in relazione alla fornitura di un particolare servizio di cui uno o più utenti usufruisce. Lo SLA impone che il particolare servizio venga fornito a livelli prestazionali pre-negoziati, e impone il pagamento di penalità in caso di mancato raggiungimento di tali livelli. Gli SLA stipulati in riferimento all'erogazione di servizi di rete, si definiscono *SLA prestazionali*; il monitoraggio di questi ultimi consente di verificare sia l'andamento dei livelli di servizio da rispettare, sia la presenza di anomalie tecniche e applicative che potrebbero causare disservizi nei confronti dell'utente finale. I principali indicatori utilizzati come SLA prestazionali sono i seguenti: performance di rete, disponibilità delle componenti di sistema, funzionalità e tempi di risposta delle transazioni su Web.

All'aumentare del numero di richiedenti per uno specifico servizio, non c'è modo di cambiare lo SLA al fine di adattarlo alle mutate condizioni della rete, per cui gli SLA risultano essere non scalabili. Un'alternativa agli SLA è costituita dai *Provisioning Level Agreement*, contratti stipulati nel momento in cui la richiesta di servizio viene generata. I PLA possono essere modificati dinamicamente a seconda delle differenti caratteristiche e dei differenti requisiti del servizio; una volta stipulato un PLA, il servizio dev'essere fornito rispettando le *provisioning level specification* (specifiche del livello di fornitura).

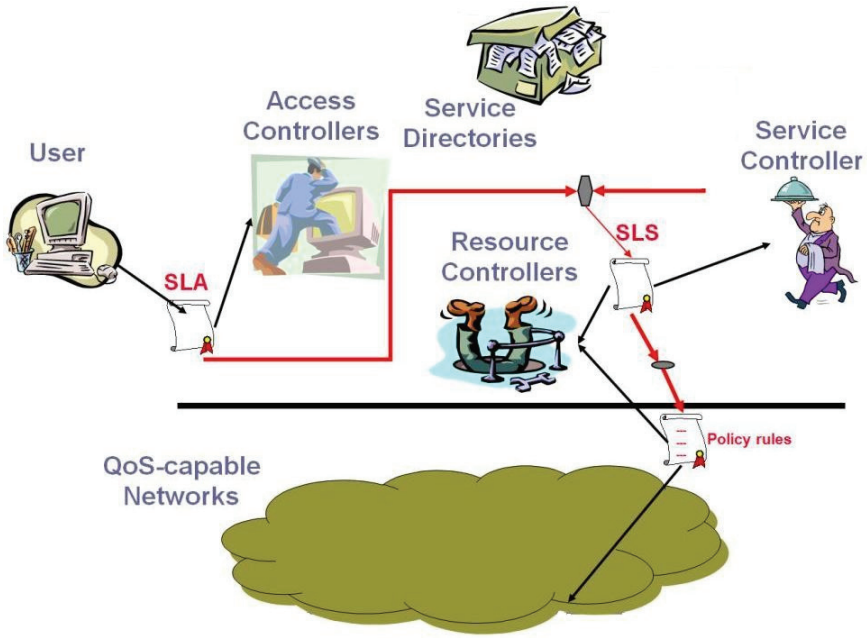

Figura 1 - SLA tra ASP e utenti

Per ogni singolo servizio fornito ai propri utenti, l'ASP stipulerà con questi ultimi un Service Level Agreement. Allo stesso modo, per ogni singolo servizio di cui l'ASP richiede l'hosting, stipulerà un Provisioning Level Agreement con la piattaforma Service Switching.

Attraverso l'accordo contrattuale, l'ASP fornisce diverse informazioni: dettagli relativi alle risorse da allocare, il modello di business da implementare, i requisiti relativi alla localizzazione iniziale e la qualità del servizio, i dettagli relativi al trattamento che la piattaforma deve garantire, le classi di SLA che possono essere sottoscritti dal generico utente. A seguito di tale trattativa, una determinata porzione delle risorse offerte dal sistema sarà a disposizione degli utenti dell'ASP.

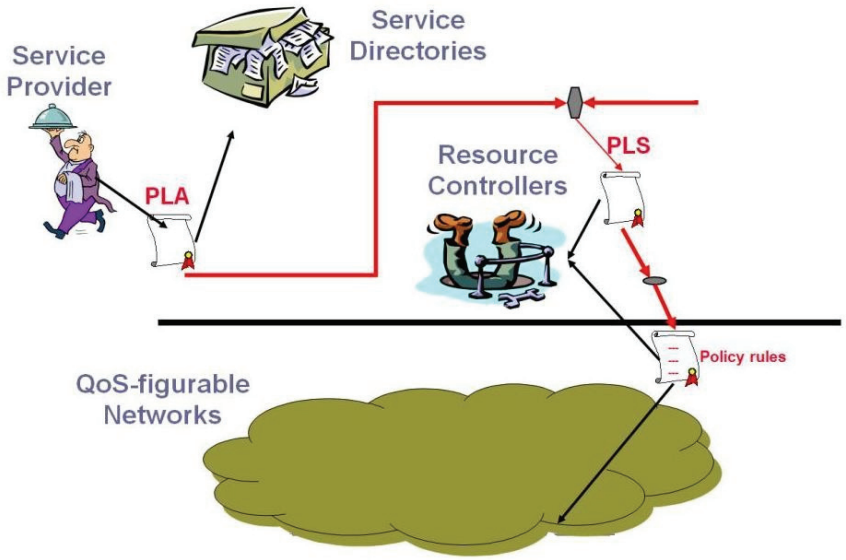

Figura 2 - PLA tra ASP e infrastruttura Service Switching

1.4 Il ciclo di vita del servizio

In un tradizionale sistema per la gestione dei servizi, è possibile individuare un certo numero di fasi che compongono il ciclo di vita del servizio; risulta evidente che le operazioni svolte durante l'esecuzione di tali fasi e la precisa sequenza con cui vengono implementate, dipende fortemente dalla natura e dalle caratteristiche del particolare sistema preso in esame. Ad ogni modo, facciamo riferimento al servizio come alla generica applicazione software, e consideriamo il generico ciclo di vita composto dalle fasi riportate di seguito:

Creazione: riassumiamo in quest'unica fase buona parte del ciclo di vita del software, quindi: analisi dei requisiti, progettazione, sviluppo, collaudo; supponiamo che la manutenzione venga invece eseguita in maniera contestuale alla fornitura del servizio.

Negoziazione: consiste nell'eseguire delle operazioni di business, quindi nello stabilire degli accordi contrattuali tra il fornitore ed il richiedente.

Fornitura: il fornitore si impegna ad erogare il servizio rispettando le regole stabilite durante la fase di Negoziazione; è durate tale fase che viene eseguita la manutenzione del servizio.

Disinstallazione: la fornitura del servizio viene sospesa; a seguito di tale fase il fornitore non avrà più la possibilità di avviare alcuna istanza del servizio in questione.

Occupiamoci adesso di formalizzare invece il ciclo di vita in riferimento al modello Service Switching, ed individuiamo quindi le fasi di cui si compone:

Attivazione: l'Application Service Provider ha stipulato un Provisioning Service Agreement fornendo tutte le informazioni necessarie all'attivazione e la distribuzione del servizio. L'attivazione consiste nel selezionare un nodo Service Switch su cui avviare l'applicazione in accordo a determinate politiche per l'allocazione delle risorse. Il servizio rimarrà attivo sul nodo selezionato fino a quando non sia necessario eseguirne la migrazione verso un Service Switch differente, oppure fino a quando l'ASP non stabilisce sia necessario eseguirne la sospensione.

Migrazione: il processo di migrazione non viene esplicitamente richiesto dall'ASP, ma eventualmente attivato in maniera autonoma ed automatica dalla rete di Service Switch in accordo a determinate politiche di gestione delle risorse. Le ragioni che possono portare alla migrazione di servizio sono differenti, e possono essere legate all'applicazione di specifici criteri per l'utilizzo ottimo delle risorse, al tentativo di incrementare la qualità del servizio percepita dall'utente, all'applicazione di procedure di *recovery* per la risoluzione di condizioni critiche.

Sospensione: tale evento può essere generato dalla richiesta esplicita dell'ASP che ne ha precedentemente richiesto l'attivazione, oppure dal verificarsi di determinate condizioni stabilite nell'accordo contrattuale. La

sospensione dell'erogazione di servizio non deve necessariamente coincidere con la disinstallazione; anche a seguito della sospensione, l'immagine del servizio può continuare a rimanere disponibile all'interno della piattaforma.

Considerando quanto riportato, è possibile verificare che le fasi del ciclo di vita del servizio in relazione al modello Service Switching, non si sovrappongono ne si sostituiscono a quelle del ciclo di vita generico. Nel caso in cui il servizio in questione debba essere ospitato dalla piattaforma per il Service Switching, possiamo immaginare che la Negoziazione e la Fornitura vengano eseguite con le regole definite dal paradigma; la fase di Fornitura, in particolare, racchiude tutte quelle che compongono il ciclo di vita in relazione al modello Service Switching. In seguito faremo riferimento solo a procedure strettamente legate al modello proposto, quindi attivazione, migrazione e sospensione di servizio, dando per assunto che tutte le operazioni preliminari oppure successive a quelle elencate, siano al di fuori del nostro interesse.

1.5 Casi d'uso

Eseguiamo una panoramica sui possibili casi d'uso in relazione al modello proposto, consideriamo quindi alcuni servizi di rete innovativi per i quali l'introduzione delle funzionalità offerte dai nodi Service Switch possa risultare efficace.

1.5.1 Il servizio di transcodifica

Per transcodifica [51, 57] s'intende la conversione diretta digitale a digitale da un segnale sorgente ad un segnale target, ed è un'operazione solitamente eseguita su dati incompatibili oppure obsoleti al fine di convertirli in un formato più comodo da utilizzare. È un processo impiegato in differenti contesti in cui sia necessario implementare un processo di adattamento dei contenuti. Viene ad esempio utilizzato per la conversione diretta di codice assembler; seppure sia solitamente preferibile utilizzare

codice sorgente e poi ricompilare l'applicazione quando necessario, ci sono casi in cui utilizzare tale approccio risulta scomodo o addirittura impossibile. Questo è vero ad esempio quando il codice sorgente non è disponibile; in tal caso può risultare opportuno eseguire la transcodifica del codice assembler per offrire differenti versioni di codice eseguibile per differenti piattaforme. La transcodifica è una tecnica molto utilizzata anche nel contesto dei Multimedia Messaging Service (servizi di messaggistica multimediale), tecnologia utilizzata per inviare o ricevere messaggi con immagini, video, audio e testo. In tale contesto la transcodifica viene impiegata per risolvere problemi legati alla incompatibilità tra dispositivi di diverso tipo; molto spesso è necessario introdurre uno stato intermedio di content adaptation per garantire che il particolare contenuto inviato dal dispositivo sorgente sia effettivamente compatibile col dispositivo target.

In relazione a tale servizio, un Service Switch può ospitare ed eseguire codice, ed essere utilizzato come transcodificatore multimediale; ancora, può essere impiegato assieme ad altri nodi Service Switch per implementare un'intera famiglia di transcodificatori ad ognuno dei quali sia assegnato uno specifico servizio per una particolare classe di traffico.

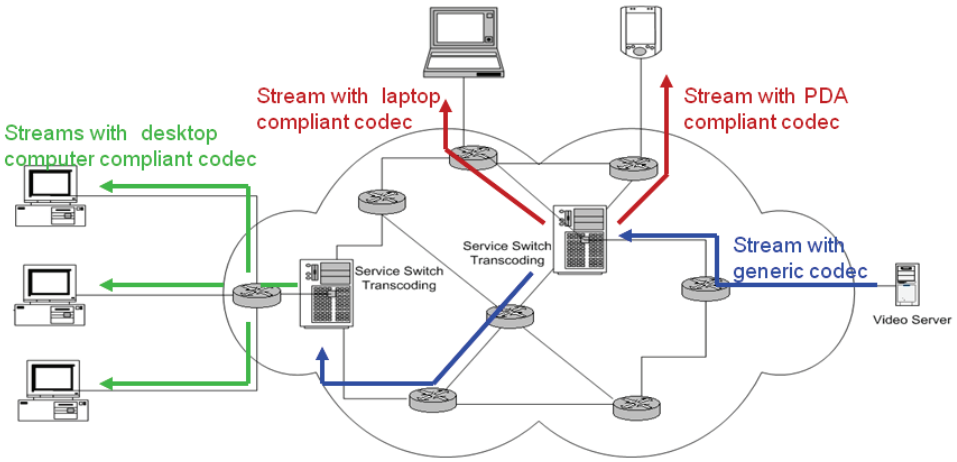

Figura 3 – Servizio di transcodifica

Prendiamo in esame lo scenario riportato in Figura 3; lo stream video inoltrato sulla rete dal Video Server, dovrà raggiungere diversi terminali sui quali sono in esecuzione client che utilizzano dei codec di tipo differente. Uno o più Service Switch sul percorso tra il server e i diversi utenti, può essere in grado di eseguire la transcodifica dello stream video in funzione della particolare tipologia di client che trova a valle, in modo che sia compatibile con quest'ultimo e quindi visualizzabile dall'utente che utilizza il terminale. In un simile scenario, potrebbe nascere l'esigenza di migrare la funzionalità di transcodifica ed avvicinarla alla rete gestita da un particolare Internet Service Provider.

1.5.2 Servizi CDN per distribuzione dei contenuti

Le Content Delivery Network [52] (reti per la distribuzione dei contenuti) sono sistemi di computer collegati in rete che collaborano in maniera trasparente per distribuire contenuti agli utenti finali. Il meccanismo consiste nell'instradare una richiesta di contenuto verso il nodo "ottimo". A seconda dei casi, il risultato che s'intende raggiungere può essere quello di ridurre i costi per l'utilizzo della banda impegnata, oppure quello di ottimizzare le prestazioni in relazione al processo di consegna dei contenuti. Se l'obiettivo da raggiungere è quest'ultimo, per nodo ottimo s'intende quello che riesce a soddisfare la richiesta nel minor tempo possibile, ad esempio il nodo geograficamente più vicino al richiedente, oppure quello con un minor carico di lavoro. Se invece l'obiettivo è quello di ridurre i costi, il nodo ottimo è quello utilizzando il quale non risulta necessario passare attraverso determinati link.

Pur essendo statica la localizzazione dei nodi che compongono una CDN, quella dei dati ospitati è invece dinamica; nasce quindi l'esigenza di controllare esplicitamente lo schema di localizzazione in accordo a determinate politiche di Content Adaptation. In un simile contesto, un Service Switch può ospitare sia dati che servizi, ed implementare quindi funzionalità necessarie alla organizzazione dei contenuti all'interno della rete; tali funzionalità possono se necessario essere migrate da un punto all'altro della rete, quindi da un Service Switch ad un altro.

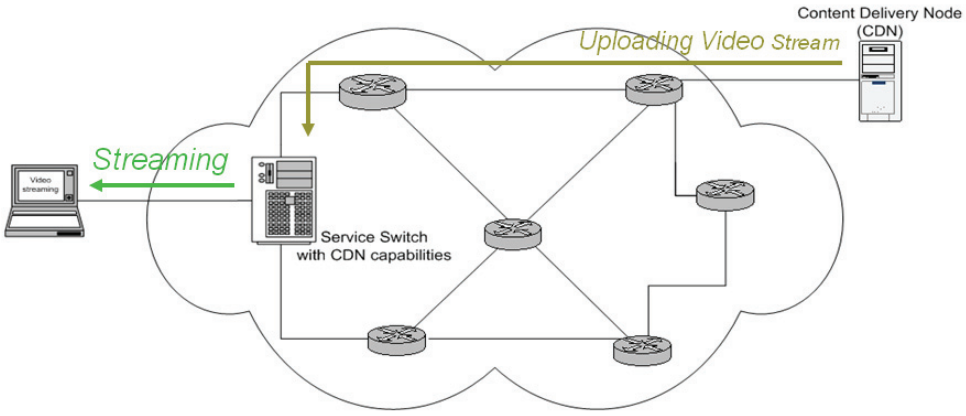

Figura 4 – Il servizio Content Delivery

Prendiamo in esame lo scenario rappresentato in Figura 4; c'è da consegnare un contenuto multimediale ad un utente, in particolare uno strem viedeo. Un Service Switch con funzionalità integrate per il content delivery, collaborando con altri Content Delivery Node, oppure interagendo direttamente col Content Provider, può agevolare la consegna del contenuto e nel caso specifico eseguire lo streaming del video verso il terminale dell'utente.

Questo particolare caso d'uso è stato utilizzato per eseguire le prove sperimentali che verranno descritte nel capitolo V.

1.5.3 Servizi per la rilevazione delle intrusioni

Esistono diverse tipologie di applicazione che trovano la giusta collocazione all'interno della rete, un esempio è quello degli IDS (Intrusion Detection System – sistemi per la rilevazione delle intrusioni) [53], sistemi composti da dispositivi software e hardware, utilizzati per identificare accessi non autorizzati a computer o intere reti locali. Le intrusioni o attacchi sono generalmente pilotate da hacker esperti, e vengono eseguite mediante l'invio di dati malformati oppure applicazioni malevoli; molto spesso il

metodo utilizzato per eseguire un attacco consiste nell'accedere agli host mediante utilizzo illecito dei privilegi di utente, oppure mediante accessi non autorizzati eseguiti tramite programmi malevoli come virus, trojan e worm. Esistono due grosse categorie di sistemi per la rilevazione delle intrusioni: sistemi basati sulle firme (signature) e sistemi basati sulle anomalie (anomaly); la prima tecnica è in qualche modo analoga a quella utilizzata per il rilevamento dei virus, mentre la seconda si basa sull'uso di un insieme di regole che consentono di distinguere ciò che è "normale" da ciò che è "anormale". Le tecniche per la rilevazione delle intrusioni vengono generalmente classificate come *misuse detection*, che usano pattern di attacchi ben noti, e *anomaly detection*, che cercano invece di determinare una possibile deviazione rispetto ai pattern di uso normale del sistema. Un IDS presenta tre componenti fondamentali: uno o più sensori utilizzati per ricevere le informazioni dalla rete o dai computer, una console utilizzata per monitorare lo stato della rete e dei computer, un motore che analizza i dati prelevati dai sensori e provvede ad individuare eventuali falle nella sicurezza informatica.

In tale contesto, Service Switch arricchiti con funzionalità per l'Intrusion Detection, possono essere integrati o sostituiti a quelli tradizionali, ad esempio in uno scenario in cui le funzionalità da implementare siano differenziate e quindi distribuite sui vari nodi.

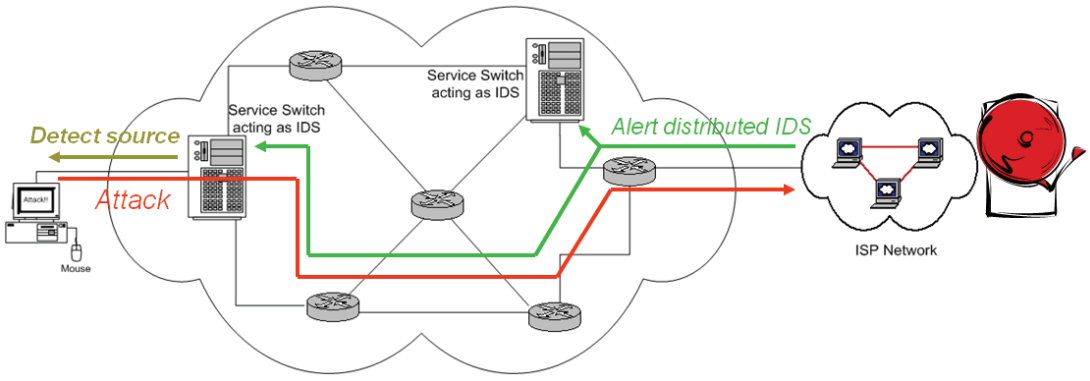

Figura 5 – Servizi per l'Intrusion Detection

Si consideri lo scenario rappresentato in Figura 5; supposto che un certo numero di nodi Service Switch con funzionalità per la rilevazione delle intrusioni siano localizzati sul percorso tra il potenziale attaccante e la rete gestita da un particolare Internet Service Provider, tali nodi saranno in grado di rilevare un eventuale attacco, segnalare l'attacco alla rete vittima, rilevare l'attaccante. In un simile scenario, potrebbe nascere l'esigenza di migrare la funzionalità di Intrusion Detection e posizionarla sul percorso tra un diverso potenziale attaccante e la rete da proteggere.

1.5.4 Servizi basati sull'uso di tecnologia Peer-to-peer

Per rete peer-to-peer [54] s'intende una rete di computer all'interno della quale non è implementata una organizzazione di tipo gerarchico, e dove non esistano client o server ma nodi tra loro paritetici che fungono a seconda dei casi sia da cliente che da servente. I nodi paritetici avranno differenti caratteristiche in termini di configurazione locale, velocità di elaborazione, capacità di memorizzazione. Il contesto in cui tale modello viene maggiormente utilizzato è quello del File Sharing (condivisione di file), ma negli ultimi anni trova impiego anche in quello del video-on-demand e delle webTV; queste ultime si basano sull'utilizzo della banda di trasmissione di cui dispongono i singoli utenti, per la trasmissione dei flussi di traffico verso gli altri fruitori all'interno della rete. È possibile sfruttare il peer-to-peer per una simile applicazione in quanto non è richiesto l'uso di server dotati di elevate prestazioni; ogni peer inoltra i flussi video solo ad un numero limitato di utenti che a loro volta li distribuiscono ad altri.

Mediante l'uso di nodi Service Switch è possibile implementare un sistema peer-to-peer sicuro e differenziato sul modello di Bit-Torrent o Gnutella. Ad ogni nodo verrebbe assegnata la funzione di caching-peer, quindi il compito di memorizzare e condividere file o porzioni di file, implementando un meccanismo di allocazione dinamico che rispetti uno schema centralizzato al fine di ridurre lo sbilanciamento nello scambio di dati.

1.5.5 Il servizio di autenticazione remota

RADIUS (Remote Authentication Dial-In User Service) [55] è un protocollo AAA (Authentication, Authorization, Accounting) utilizzato in applicazioni di accesso alle reti, ed è attualmente lo standard de-facto per l'autenticazione remota. RADIUS è un protocollo ampiamente utilizzato negli ambienti distribuiti e in sistemi integrati in cui sia necessario gestire una grossa quantità di utenti con informazioni di autenticazione distinte. Fornisce inoltre alcuni livelli di protezione contro attacchi attivi e contro lo sniffing. Il processo di autenticazione in RADIUS ha inizio con la creazione da parte del generico clinet di un pacchetto Access-Request completo di User-Name e User-Password; il primo campo viene trasmesso in chiaro mentre il secondo è protetto da un meccanismo basato sull'uso di una chiave condivisa da client e server. Una volta ricevuto il pacchetto Access-Request, il server verifica l'autenticità dei dati forniti dal client, e a seconda dei casi risponde con un pacchetto Access-Accept oppure un pacchetto Access-Reject, o addirittura ignorando la richiesta di autenticazione.

Un nodo Service Switch può essere utilizzato per implementare la funzione di server RADIUS dinamico, quindi rimpiazzarlo per un breve o lungo periodo, oppure diventare parte integrante di un sistema per l'autenticazione remota.

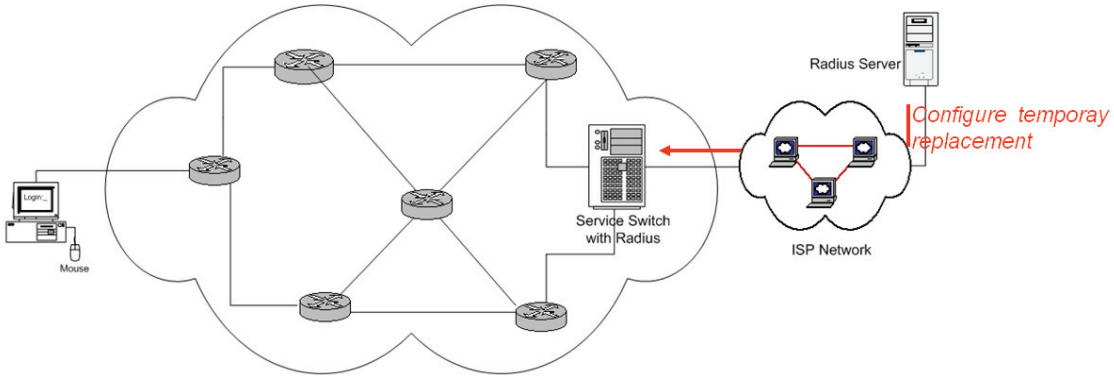

Figura 6 – Servizio di autenticazione remota

Si prenda in esame lo scenario rappresentato in Figura 6; un Service Switch sostituisce temporaneamente un server RADIUS in manutenzione, e gestisce l'autenticazione remota di una rete gestita da uno specifico Internet Service Provider.

1.5.6 Il traffico voce su IP

Consideriamo un particolare esempio di servizio per il quale bisogna garantire un elevato livello di resilienza considerata la tipologia di traffico generato e trasportato, il Voice over IP (VoIP) [56]. Il VoIP è una tecnologia che rende possibile effettuare una conversazione telefonica sfruttando una connessione Internet oppure una rete dedicata che utilizzi il protocollo IP anziché la tradizionale rete telefonica. Il segnale vocale viene digitalizzato, suddiviso in pacchetti, e instradato sulla rete; tale operazione non viene eseguita durante l'intero intervallo in cui è in corso una conversazione, ma solo quando uno degli utenti produce del suono. La tecnologia VoIP si basa sull'uso dello stack protocollare H.323, il quale specifica le componenti e le procedure da implementare per le comunicazioni multimediali, real-time e point-to-multipoint, attraverso una rete packet-based. I sistemi che prevedono l'uso di telefoni VoIP, anche definiti PBX (Private Branch eXchange), sono costituiti dalle seguenti quattro componenti: terminale utente, gateway, gatekeeper, MCU (Multipoint Control Unit). Il terminale è un end-point su cui vengono generati e terminati flussi H.323; può trattarsi quindi di un PC oppure di un PBX. Il gateway assume il ruolo di interfaccia, esegue quindi la traduzione dei messaggi che viaggiano attraverso la rete, e ne esegue la compressione e decompressione. Il gatekeeper esegue il routing e si occupa della gestione centralizzata di tutti i terminali presenti all'interno di una zona specifica. L'MCU, unità di controllo multipunto, è in grado di abilitare una conferenza tra tre o più terminali; la funzionalità svolta dall'MCU può eventualmente essere integrata in un terminale, un gateway oppure un gatekeeper.

Fatta naturalmente eccezione per il terminale VoIP, le funzionalità svolte dalle altre entità che compongono un sistema VoIP, possono essere implementate su un nodo Service Switch.

Si consideri ad esempio il caso in cui un gatekeeper sia vittima di un fallimento; le funzionalità svolte da tale componente possono migrare in maniera trasparente e quindi essere ospitate da un nodo Service Switch.

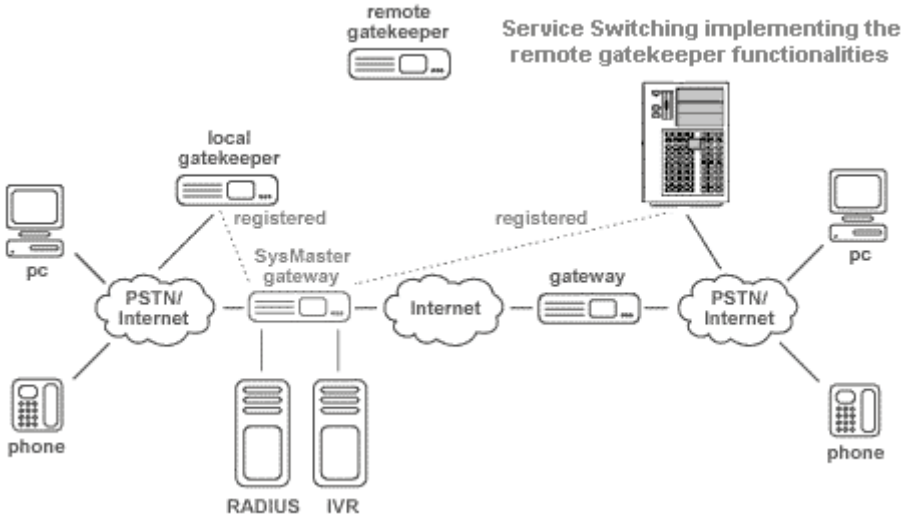

Figura 7 - VoIP

Nel Capitoli III presenteremo una particolare implementazione del modello Service Switching, basata sull'utilizzo dei meccanismi di virtualizzazione e sul modello Mobile IP. Tuttavia, la definizione del modello è stata eseguita mantenendo un livello di dettaglio tale da non limitare la realizzazione del modello stesso a quella suggerita di seguito, ma con l'obiettivo di fornire delle specifiche aperte ad eventuali soluzioni alternative.

Capitolo 2
Stato dell'arte e lavori correlati

Il concetto di mobilità di servizio copre un ruolo di rilevante importanza nelle reti di nuova generazione, e tale aspetto tenderà a rinforzansi ulteriormente nel prossimo futuro.

Nel contesto dell'ubiquitous computing, in cui utenti nomadi si spostano all'interno della rete muovendosi tra ambienti eterogenei, il supporto alla mobilità di servizio aiuta ad incrementare la qualità percepita nella fruizione del servizio stesso. Il servizio è in grado di rilevare i movimenti dell'utente, ricercare una nuova posizione ed un ambiente di esecuzione adeguati, quindi migrare ed avvicinarsi all'utente stesso per garantire una fornitura più efficiente.

In piattaforme per l'hosting di servizi è di fondamentale importanza curare aspetti legati alla gestione efficiente delle risorse disponibili, ed implementare meccanismi finalizzati al bilanciamento del carico computazionale a beneficio di risorse particolarmente sovraccariche. Poter migrare servizi su nodi differenti di una infrastruttura o in generale della rete, a prescindere dalla specifica tecnologia utilizzata per implementare tale meccanismo, consente di raggiungere tali obiettivi.

In sistemi di rete per la distribuzione di servizi è indispensabile poter fare affidamento su meccanismi che si occupino di rilevare eventuali condizioni critiche e reagire a tali condizioni in modo da garantire continuità nella fornitura dei servizi, quindi un adeguato livello di resilienza. Molto spesso, la migliore soluzione per far fronte a condizioni di questo tipo consiste nell'eseguire il trasferimento di un intero insieme di

servizi e del relativo stato di esecuzione, dalla zona in cui si è verificato il fallimento ad una differente porzione dell'infrastruttura.

Il paradigma Service Switching introdotto al capitolo precedente, si basa sul concetto di mobilità di servizio e propone un meccanismo di migrazione implementato tenendo in considerazione diversi aspetti legati al contesto della fornitura di servizi in sistemi di rete di nuova generazione. Proporremo di seguito una panoramica su modelli proposti in letteratura basati sull'utilizzo di differenti tecnologie, e in cui il concetto di mobilità di servizio assume un ruolo fondamentale; parleremo in particolare della gestione di servizi context-aware in reti Ad Hoc e in sistemi basati sull'uso di agenti mobili.

Introdurremo poi le Active Network, una particolare infrastruttura di rete costituita da nodi in grado di ospitare programmi la cui esecuzione può comportare una modifica o la sostituzione delle politiche di instradamento adottate. Le Active Network presentano diversi punti in comune col modello che noi proponiamo: similmente ad un Service Switch, un nodo attivo è in grado di processare il traffico in maniera intelligente e di eseguire funzionalità aggiuntive oltre al tradizionale meccanismo di packet switching. Descriveremo un modello per la fornitura di servizi basato sulle Active Network, e successivamente una infrastruttura basata invece sull'uso di tecnologia peer-to-peer; ciò che accomuna i due modelli è il fatto che propongono uno schema innovativo per la fornitura di servizio alternativo al tradizionale modello client-server, ed uno schema innovativo per l'interazione tra utente e fornitore del servizio.

Introdurremo infine il Grid Computing ed il modello Internet Suspend/Resume (ISR). Il Grid Computing rientra nel contesto dei sistemi di rete per la fornitura di servizi; si tratta una potente infrastruttura che offre servizi computazionali e si basa sul concetto di condivisione delle risorse. ISR è invece un sistema che si basa sul meccanismo di migrazione di macchine virtuali, e propone una potenziale alternativa all'uso dei computer portatili. In riferimento ad entrambi i modelli, presenteremo il modo in cui fanno uso dei meccanismi di virtualizzazione, strumento che assume un ruolo fondamentale nell'implementazione del paradigma Service Switching che proporremo nel Capitolo seguente.

2.1 Migrazione di servizi context-aware in reti Ad Hoc

L'impiego massiccio di tecnologia e dispositivi wireless conduce all'esigenza di ampliare sempre più il settore di ricerca e sviluppo relativo all'ubiquitous networking, quindi alla creazione di piattaforme che siano in grado di supportare la nomadicità di utenti che fanno uso di strumenti mobili quali ad esempio computer portatili, palmari, telefoni cellulari di ultima generazione.

Tali considerazioni conducono al concetto di rete Ad Hoc. Nei modelli tradizionali per la fornitura di servizi, si da per assunto che il servizio sia localizzato sempre nella medesima posizione, quindi su un nodo statico ben identificato, e che l'utente faccia sempre riferimento a tale nodo per interagire con il servizio; un esempio è quello dei web server in Internet che girano su nodi identificati da indirizzi IP statici. Lo scenario tipico di una rete Ad Hoc è costituito invece da nodi che ospitano servizi e che possono aggiungersi oppure lasciare una rete in maniera dinamica, modificando quindi di continuo l'insieme di servizi disponibili all'interno della regione di interesse. Ancora, considerata la limitata disponibilità di risorse, in una rete Ad Hoc un nodo può interrompere la fornitura di un servizio in esecuzione, oppure un nuovo servizio può essere attivato. Le reti Ad Hoc possono formarsi in maniera spontanea oppure essere create opportunamente per eseguire determinate funzioni ed essere usate per fornire un insieme di servizi. La progettazione e la fornitura di servizi per reti Ad Hoc consente da un lato agli utenti in movimento di ottenere informazioni che non potrebbero altrimenti reperire in assenza di una simile infrastruttura, e dall'altro la possibilità ai service provider di acquisire in tempo reale informazioni catturate ed inoltrate da nodi localizzati nell'immediata prossimità di una regione geografica, un oggetto, o un'attività di interesse.

In un simile scenario, considerando la dinamicità con cui i nodi che eseguono servizi offrono la propria disponibilità, un service provider può trovarsi in condizioni di dover decrementare il livello di qualità del servizio garantita, come risultato, ad esempio, di un

cambiamento al contesto del servizio oppure a quello dell'utente; per contesto s'intende un insieme di parametri quali: posizione, velocità, potenzialità degli strumenti, topologia di rete. Si consideri un servizio che esegue il monitoraggio di una regione specifica, e che il nodo su cui è in esecuzione lasci la regione in questione; in tal caso il cambiamento al contesto del servizio provoca un decremento della qualità del servizio offerto, oppure addirittura l'impossibilità di eseguire il servizio stesso. Ancora, si consideri un oggetto che insegue un servizio; la QoS garantita resta accettabile fino a quando il nodo su cui il servizio è in esecuzione resta vicino all'oggetto, ma se l'oggetto si muove, la QoS decresce. Anche i cambiamenti al contesto dell'utente possono rendere il servizio inadeguato.

In situazioni di questo tipo, la soluzione tradizionalmente adottata consiste nell'interrompere l'iterazione corrente tra utente e servizio, e consentire all'utente di trovare un nuovo servizio che possa assolvere la propria richiesta; tale approccio risulta però poco efficiente. Una soluzione innovativa è invece quella di rendere i servizi context-aware, quindi capaci di adattarsi al contesto modificato, migrando su un nodo diverso dove possano implementare le loro funzionalità in maniera più efficiente.

A differenza di un servizio statico che resta localizzato sempre sul medesimo nodo, un servizio mobile è in grado di migrare verso nodi differenti della rete al fine di eseguire la propria funzione. Tale migrazione avviene in maniera trasparente nei riguardi dell'utente, e, fatta eccezione per un piccolo ritardo dovuto alla migrazione stessa, la fornitura del servizio non subisce alcuna interruzione. Utilizzando un simile modello, l'utente non ha bisogno di ricercare nuovi servizi nel caso in cui quello attualmente in uso non sia più disponibile. Un servizio mobile appare all'utente costantemente localizzato sul medesimo end-point virtuale, mentre in realtà migra nel tempo da un nodo fisico all'altro. In simili contesti la migrazione di servizio è di tipo context-aware, nel senso che viene attivata in seguito ad un eventuale cambiamento al contesto del servizio oppure a quello dell'applicazione utente. In linea di massima, si da per assunto che l'applicazione utente resti localizzata sempre sul medesimo nodo fisico, il quale potrebbe però essere un dispositivo mobile. Un servizio context-aware è quindi

progettato in modo tale che sia capace di apprendere i cambiamenti subiti dal contesto, e quindi migrare nel caso in cui il nodo che lo ospita non fosse più in grado di verificarne i requisiti per l'esecuzione. Durante la migrazione, il servizio trasporta con se tutte le informazioni di stato necessarie alla riattivazione sul nodo destinazione.

Sono presenti in letteratura diversi modelli che supportano la gestione di servizi distribuiti e che abilitano utenti nomadi ad accedere a tali servizi in maniera efficiente. Implementando tali modelli si offre all'utente la possibilità di muoversi senza necessariamente dover rinunciare alla fruizione del o dei servizi; sistemi di questo tipo propongono in genere delle procedure per la ricerca e la selezione di servizi in ambienti mobili, e consentono l'uso di tali servizi attraverso l'impiego di meccanismi flessibili che adattano il servizio stesso alle caratteristiche del dispositivo mobile su cui dovrà essere eseguito, e alla qualità dei canali di comunicazione disponibili.

In [29] viene ad esempio proposto un modello innovativo che supporta la distribuzione di servizi su reti dinamiche Ad Hoc, incentrato quindi sul concetto di mobile service (servizio mobile). Il modello è caratterizzato da una corrispondenza uno a uno tra il servizio mobile e l'applicazione utente. Nel momento in cui viene trovato un servizio che presenta risorse sufficienti a servire le richieste dell'utente, tale servizio ne istanzia uno mobile che è in grado di migrare nella rete con l'obiettivo di assolvere in maniera efficiente le richieste dell'utente; al servizio iniziale è assegnato l'unico compito di generare quello mobile, non è in grado quindi di assolve le richieste dell'utente. Il modello si basa sull'uso di un middleware che supporta lo sviluppo e l'esecuzione di servizi mobili. Tale middleware fornisce primitive per lo scambio di messaggi di tipo content-based (basati cui contenuti), il controllo dei servizi mobili, la gestione dei contesti. Il sistema propone inoltre un meccanismo di routing content-based, e ovviamente un meccanismo di migrazione.

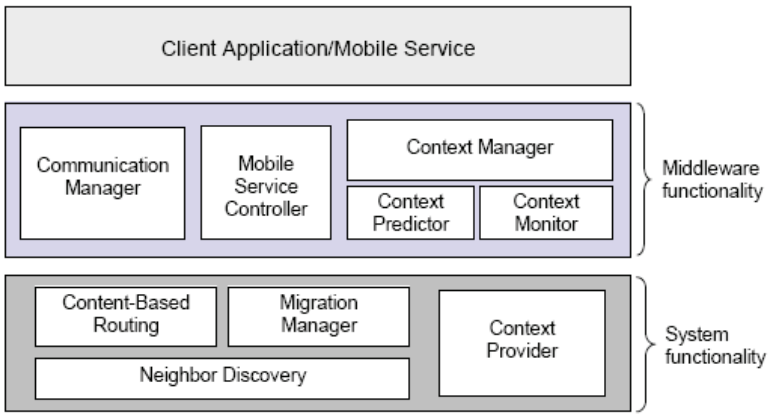

Figura 8 - Il framework Mobile Services

2.2 Gestione dei servizi in sistemi ad agenti mobili

Un software Agent [59] è una entità software autonoma che può interagire con il proprio ambiente di esecuzione e con altre entità, compresi operatori umani, macchine, e altri software agent in esecuzione su ambienti di esecuzione o piattaforme differenti. Da un punto di vista pratico, un software agent è un design pattern, un modello di progettazione per il software. L'implementazione di un design pattern agent-based può essere implementato utilizzando sia strumenti agent-based che qualsiasi altra categoria di strumenti che supporti entità software autonome, interattive e adattative; in questo secondo caso non è però possibile supportare l'autonomia, l'interattività, e la natura adattativa richiesta dagli agent.

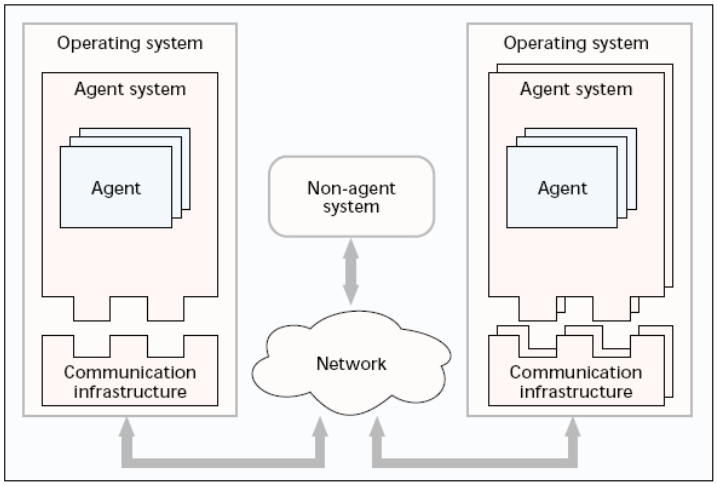

Figura 9 - Un'architettura basata su agenti mobili

Gli agenti software possono essere classificati come stazionari oppure mobili; si definiscono stazionari quando vengono eseguiti su un singolo computer, si definiscono invece mobili quando è possibile trasferire il loro codice da un host ad un altro sul quale verranno riesumati. La mobilità viene introdotta per fornire la possibilità di avvicinare l'agente ai servizi attivi sul nodo destinazione, e quindi contribuire a migliorare le prestazioni.

Supponiamo che un'agente abbia necessità di ottenere informazioni da differenti sorgenti su differenti piattaforme; sarà sicuramente in grado di inviare richieste ad ognuna di queste utilizzando l'equivalente di una chiamata a procedura remota. Se però la quantità di informazioni da trasferire è troppo elevata, bisogna tenere in conto l'impatto sulla banda. Tra l'altro, con buon probabilità, processare i dati remoti sul nodo remoto stesso, sfruttando i servizi offerti da quest'ultimo, potrebbe risultare maggiormente efficiente. Nel caso in cui simili condizioni debbano essere verificate, riallocare l'agente sulla piattaforma remota potrebbe risultare la migliore soluzione. L'inevitabile prezzo da pagare è che il sito remoto deve mettere a disposizione dei cicli di CPU al fine di supportare l'esecuzione dell'agente migrato. Questo non solo

comporta un processing addizionale sul nodo remoto, ma crea anche problemi legati alla sicurezza e alla scalabilità.

Come accennato al paragrafo precedente, negli ambienti di ubiquitous computing un importante aspetto è quello della context-awareness, quindi la conoscenza del contesto in cui il servizio viene eseguito. Un servizio context-aware deve necessariamente essere anche location-aware, deve quindi conoscere il punto in cui è localizzato. Gestire servizi di questo tipo implica l'introduzione di meccanismi di migrazione adattativi, mediante i quali sia possibile selezionare un dispositivo su cui spostare il servizio, quindi migrare il servizio stesso.

Sono presenti in letteratura diversi modelli che propongono differenti soluzioni al problema della migrazione, ma che in genere si limitano a scenari in cui i nodi coinvolti sono tra loro omogenei. In [60] è invece proposto un meccanismo basato sull'uso di agenti che offre la possibilità di migrare servizi multimediali tra dispositivi eterogenei. Tale modello propone un sistema che supporta la fornitura di servizi multimediali di tipo location-aware, adattabili al dispositivo utilizzato dall'utente e sulla base di informazioni complesse relative al contesto, senza richiedere alcun intervento dell'utente stesso.

In [61] è proposto un sistema per la fornitura di servizi ad utenti mobili basato ancora sull'uso di agenti. Il sistema è composto da un certo numero di agenti distribuiti su differenti siti, che cooperano tra loro per fornire servizi personalizzati ad utenti mobili. Lo scenario considerato è quello del generico utente che deve poter utilizzare un ambiente di lavoro uguale a quello dell'ufficio, magari a casa oppure sfruttando una qualsiasi postazione temporanea. L'agente che rappresenta l'utente ed un agente di sistema, saranno entrambi coinvolti in un processo di negoziazione che faciliti l'accesso a servizi personalizzati su vai ambienti di esecuzione. Un agente Adaptive Service Presentation (presentazione del servizio adattivo) è utilizzato per adattare il servizio da ospitare alle potenzialità della workstation utilizzata dall'utente.

2.3 Gestione dei servizi con tecnologia Active Network

Il generico end-system collegato alla rete può essere classificato come aperto, nel senso che può essere programmato utilizzando un generico linguaggio di programmazione. Al contrario, i tradizionali nodi di rete come ad esempio gli switch ATM oppure i router IP, sono sistemi integrati e chiusi, le cui funzionalità e le cui interfacce sono definiti attraverso delle procedure standardizzate. Il compito affidato a tali nodi è quello di trasportare pacchetti o celle tra sistemi. Il processing dei pacchetti all'interno della rete è limitato ad operazioni eseguite sull'header e finalizzate fondamentalmente all'instradamento; in sostanza, i nodi della rete non possono interpretare ne modificare il payload dei pacchetti.

Le Active Network (reti attive) [62, 63] rompono tale tradizione consentendo alla rete di eseguire delle computazioni personalizzate su interi pacchetti, payload incluso. Attraverso le funzionalità offerte dai nodi che le compongono, le reti attive consentono di eseguire computazioni sui dati dell'utente ed il processing intelligente dei pacchetti in accordo ai requisiti specifici del particolare servizio.

I nodi di una Active Network si definiscono Active Node (nodi attivi), mentre i pacchetti trasferiti al suo intero vengono identificati come Active Packet (pacchetti attivi), oppure Capsule (capsule). Oltre alla banda disponibile, un nodo attivo è caratterizzato da quantità di memoria, CPU e risorse per il processing. Un pacchetto attivo può contenere un programma che modifica o rimpiazza la funzionalità di processing implementata dal nodo che lo esegue. L'esecuzione di tali programmi sugli Active Node, produce come risultato una modifica allo stato del nodo e la generazione di altri pacchetti attivi da inviare sui link di uscita.

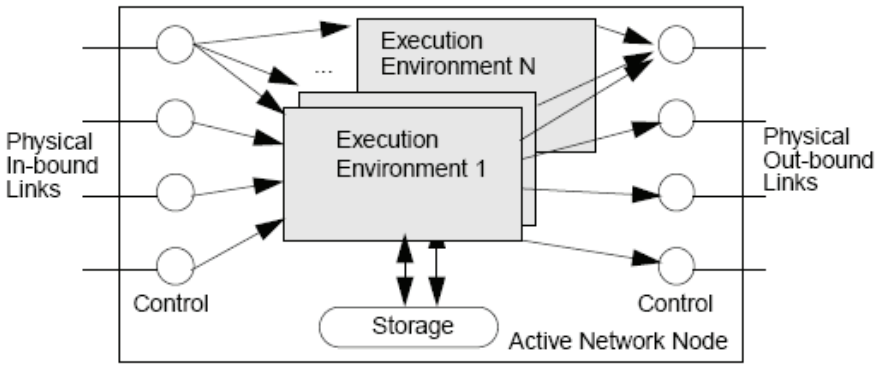

Figura 10 - L'architettura di un Active Node

La Figura 10 mostra il modello di un Active Node; le funzionalità basilari implementate da tale nodo sono le seguenti: (1) controllo sui pacchetti in ingresso; (2) controllo sui pacchetti in uscita; (3) packet processing; (4) accesso alla memoria.

Un nodo attivo può essere visto come la generalizzazione di un nodo di rete tradizionale; un router IP, ad esempio, ospita un singolo ambiente di esecuzione ed un singolo insieme di funzioni preinstallate ed abilitate ad eseguire il packet processing. Un Active Node è invece in grado di implementare diversi ambienti di esecuzione in modo parallelo; ogni pacchetto attivo in arrivo su un link di ingresso, contiene l'identificativo di uno specifico ambiente di esecuzione sul quale verrà eseguito il processing del pacchetto stesso.

Esistono due grosse aree di ricerca nel contesto delle Active Network, la prima è finalizzata alla progettazione di ambienti di esecuzione per nodi attivi, la seconda alla realizzazione di funzionalità da integrare a livello rete, ad esempio per eseguire il filtraggio e l'instradamento dei pacchetti in funzione dell'applicazione.

È possibile trovare in letteratura alcuni modelli che propongono l'uso delle reti attive nell'ambito delle telecomunicazioni; la sfida è quella di costruire nodi che siano in grado di processare pacchetti ad una velocità comparabile a quella dei router IP o degli switch ATM attualmente in commercio, e di realizzare un ambiente sicuro in cui differenti entità possano condividere risorse per lo storage, per il processing e per le

comunicazioni. Nel seguito mostreremo un framework [24, 25] per l'interazione tra utente e fornitore di servizio in un ambiente di rete attiva.

Il modello in questione si distingue da quello tradizionale per due grosse ragioni, innanzitutto perché consente di ottenere una netta separazione tra processo di fornitura e processo di gestione di servizio, secondo perché fornisce la possibilità di eseguire la gestione di servizio attraverso l'interfaccia del servizio stesso. La fornitura di servizio si basa sulla cooperazione tra utente e provider attraverso l'uso di una interfaccia di gestione, e non ne è richiesta la sospensione per implementare processi operazionali quali ad esempio la rinegoziazione della quantità di banda. In un tradizionale sistema per le telecomunicazioni, il processo di fornitura è di tipo service-specific (dipendente dal servizio), e consiste nel configurare dei link virtuali tra i vari utenti ed allocare risorse a tali link; in uno scenario di questo tipo l'utente vede un'astrazione del servizio costituita da un insieme di link ad ognuno dei quali è associato uno specifico livello di QoS e una specifica quantità di banda. Al contrario, la fornitura di servizio in ambiente Active Network, fornisce all'utente un'astrazione del servizio più complessa ma allo stesso tempo più potente, costituita da un grafo di nodi attivi virtuali, interconnessi tra loro mediante l'uso di link virtuali. Durante la fornitura del servizio l'utente ed il fornitore configurano i grafi e stabiliscono le risorse da allocare ai servizi; tali risorse includono la banda per i virtual link, le risorse necessarie per eseguire il processing, la memoria per ogni nodo virtuale attivo. L'astrazione di servizio definita in questo modo viene identificata come Virtual Active Node (VAN); il VAN è in sostanza un piattaforma per l'esecuzione di servizi a disposizione dell'utente, ed è caratterizzato da una specifica topologia e da un determinato insieme di risorse allocate; potrà essere usato dall'utente per configurare ed eseguire servizi di rete senza richiedere ulteriori interazioni con il fornitore. Non esistono vincoli sulla particolare topologia che il VAN deve implementare, ciò consente ai fornitori di configurare l'astrazione di servizio a differenti livelli di dettaglio in accordo ai requisiti richiesti dagli utenti. Il VAN è in sostanza uno strumento che consente di partizionare opportunamente le risorse disponibili sull'Active Network e di assegnarle agli utenti.

Per quanto riguarda invece la gestione, il linea generale consiste nel creare, supervisionare, aggiornare e rimuovere servizi all'interno di una piattaforma di rete. Un utente che debba usufruire del VAN fornito da un provider, deve, almeno in fase preliminare, cooperare con quest'ultimo; per quanto riguarda invece la gestione del servizio di cui un utente vuole usufruire su un particolare VAN, può essere implementata attraverso l'interfaccia del servizio stesso, non richiede quindi l'interazione con il sistema di gestione del fornitore. In un tradizionale sistema per le telecomunicazioni, l'utente è in possesso di una astrazione di servizio semplice, e vede essenzialmente il servizio come una black box con punti di accesso. La gestione di servizio viene eseguita dal fornitore attraverso quest'ultima, ed è limitata generalmente alle funzioni per il monitoraggio. In ambiente Active Network, un utente può installare funzionalità per la distribuzione e per la gestione di servizio direttamente sul VAN fornito dal provider. Il VAN fornisce un'astrazione di servizio complessa, che consente di eseguire il controllo ad un livello di dettaglio molto elevato, tanto da permettere la modifica delle politiche di scheduling per la gestione dei pacchetti nell'ambiente di esecuzione. Attraverso le funzionalità di gestione disponibili in relazione ad un VAN specifico, un utente può ad esempio installare e monitorare dei router virtuali, aggiornare servizi esistenti, eseguire operazioni di bilanciamento di carico, creare connessioni end-to-end con garanzie di qualità del servizio.

2.4 Una infrastrutture basata su tecnologia peer-to-peer per la gestione di servizi distribuiti

I fornitori di servizio nel settore dell'Information Technology, hanno la necessità di delegare in qualche modo l'hosting di servizi critici molto spesso implementati sottoforma di applicazioni Web, ad un costo ridotto e con elevato livello di qualità del servizio garantito. Un service provider che offre servizi computazionali, può ospitare tali servizi sfruttando i propri data center, ognuno dei quali è in grado di ospitare centinaia di applicazioni. Infrastrutture di questo tipo devono necessariamente

presentare requisiti essenziali quali la scalabilità, la resilienza, l'autonomia, e la loro organizzazione deve essere basata sui Service Level Agreement. D'altro canto, la dimensione e l'eterogeneità dell'hardware e del software che caratterizza ogni singolo data center, comporta evidenti problemi di gestione.

In [64] viene proposta una infrastruttura innovativa basata su tecnologia peer-to-peer, per la gestione di servizi di nuova generazione ospitati all'interno di data center. Sebbene la maggior parte dei sistemi per la fornitura di servizi siano basati sul classico modello client-server, il peer-to-peer presenta dei vantaggi competitivi se impiegato per la realizzazione di infrastrutture di grosse dimensioni.

I modelli P2P sono ampiamente utilizzati per applicazioni che richiedono la gestione di grosse quantità di utenti, quindi per il file sharing (KaZaA), oppure per la telefonia attraverso internet (Skype). In modelli che implementano il peer-to-peer, tutte le entità attive che eseguono computazione sono trattate allo stesso modo, ed ogni nodo è in grado si eseguire sia le funzionalità di client che quelle di server. I nodi sono organizzati all'interno di una rete di overlay ed eseguono il routing collettivo del traffico ed il processing di richieste. In un sistema organizzato in questo modo non esiste alcun single point of failure, ne tanto meno un sistema centralizzato che possa costituire un collo di bottiglia. Normalmente il modello P2P si basa sull'uso della potenza computazionale offerta dai computer desktop degli utenti finali dislocati ai margini della rete. Tale tecnologia può essere però impiegata in maniera efficiente per la realizzazione di data center di grosse dimensioni, contesto in cui probabilmente si riesce a sfruttarne le reali potenzialità.

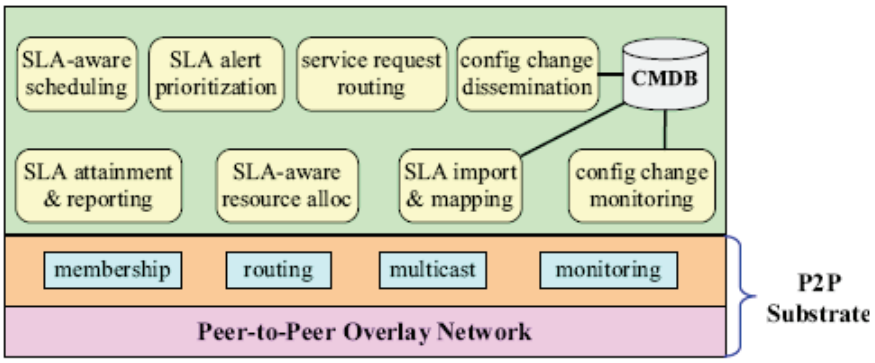

Figura 11 - L'architettura di BISE

Il sistema proposto viene identificato con l'acronimo BISE (Business-Aligned IT Service Environment), e si basa sull'uso di algoritmi peer-to-peer sostanzialmente differenti rispetto a quelli normalmente utilizzati ad esempio per applicazioni di file sharing. Tali algoritmi sono progettati e realizzati sulla base di due importanti aspetti: i nodi che compongono un data center sono caratterizzati da un'architettura e da potenzialità molto differenti rispetto ad un normale computer desktop, e la gestione di data center è certamente più complessa rispetto alla gestione di un insieme di peer che scambiano tra loro contenuti multimediali.

2.5 Il Grid Computing

Il Grid computing [34] (calcolo a griglia) è un paradigma del calcolo distribuito; basato su una infrastruttura altamente decentralizzata e di natura variegata, è in grado di consentire ad un vasto numero di utenti l'utilizzo di risorse disponibili su calcolatori connessi alla rete. La scelta del termine "griglia" discende dal fatto che gli ideatori del Grid Computing sostengono che nel prossimo futuro sarà possibile reperire risorse computazionali con la stessa semplicità con cui oggi si usufruisce dell'energia elettrica, ovvero semplicemente collegandosi ad una presa di rete. Il Grid Computing viene impiegato prevalentemente per il calcolo computazionale necessario alla risoluzione di

problemi complessi, sia in ambito scientifico che ingegneristico; uno dei primi contesti in cui è stato utilizzato, è quello della fisica delle alte energie. Una grid è in grado di fornire agli utenti la possibilità di accedere alla capacità di calcolo e di memoria di un sistema distribuito, garantendo un accesso coordinato e controllato alle risorse condivise, ed offrendo all'utente la visibilità di un unico sistema di calcolo logico cui sottomettere i propri job.

L'idea del Grid computing nasce da una semplice constatazione: l'utilizzo medio che una piccola o media impresa fa delle proprie risorse computazionali è pari a circa il 5% della reale potenza di calcolo. Da tale osservazione nasce il concetto di Virtual Organization (VO), organizzazioni virtuali dinamiche e multi-istituzionali sulle quali si basa il funzionamento del Grid Computing. Una VO può essere identificata come un insieme di istituzioni che mettono a disposizione le proprie risorse a beneficio delle altre componenti; la condivisione di tali risorse è fortemente controllata dai rispettivi fornitori, i quali definiscono le regole secondo cui la condivisione deve avvenire. Un apposito protocollo descrive il comportamento che gli elementi distribuiti all'interno del sistema devono assumere, il modo in cui possono interagire tra loro, la struttura delle informazioni da scambiare durante tale interazione.

Le procedure di creazione, manutenzione e messa in opera dei servizi Grid relativi ad una specifica VO, sono supportate dall'OGSA (Open Grid Service Architecture), un modello service-oriented in cui la generica componente viene identificata come Grid service; in OGSA un servizio Grid è definito come Web service che fornisce un insieme di interfacce ben note, e che rispetta specifiche convenzioni. Un servizio Grid può implementarne una o più interfacce, dove per interfaccia s'intende un insieme di operazioni che possono essere invocate attraverso lo scambio di una particolare sequenze di messaggi. La definizione dell'interfaccia è completamente indipendente dalla implementazione delle relative funzionalità; questa prerogativa consente ad un servizio di poter essere eseguito su differenti ambienti e non solo sulla piattaforma nativa: è sufficiente associare alla specifica interfaccia una opportuna implementazione a seconda dei casi. Ad ogni servizio è inoltre associato uno stato interno, mediante il

quale è possibile distinguere differenti istanze di servizio che utilizzano la medesima interfaccia.

La creazione di griglie computazionali può essere eseguita mediante l'uso di uno strumento open source fornito dalla Globus Alliance, il Globus Toolkit [37].

Considerato che il Grid offre accesso ad ambienti differenti, e che tipicamente una specifica applicazione richiede un ambiente di esecuzione con particolari caratteristiche, risulta evidente che la generica applicazione utente può potenzialmente sfruttare solo una piccola frazione delle risorse effettivamente disponibili. Tale considerazione, assieme all'esigenza di sfruttare meccanismi per il controllo dinamico, isolato e a grana fine, conduce al concetto di Virtual Workspace [65, 66]. Tramite l'uso dei Virtual Workspace il generico utente della Grid ha la possibilità di definire un ambiente di esecuzione personalizzato, e pretendere quindi determinati requisiti hardware e specifiche configurazioni software. L'obiettivo nell'introduzione dei Virtual Workspace è dunque quello di fornire un meccanismo che consenta di catturare le richieste relative ai requisiti per la creazione di un ambiente di esecuzione che meglio si adatti alle richieste dell'utente e alle caratteristiche dei job di cui richiederà l'esecuzione. I Virtual Workspace sono tipicamente realizzati sfruttando macchine virtuali. In aggiunta alle evidenti proprietà di isolamento che è possibile ottenere mediante l'uso di tale tecnologia, la virtualizzazione dell'hardware consente di istanziare su un host degli ambienti completamente personalizzabili. Le macchine virtuali possono essere rapidamente sospese ed il loro stato congelato, e quindi facilmente migrate verso nodi remoti in modo completamente trasparente nei riguardi dell'utente, per garantire ad esempio la disponibilità di un particolare Virtual Workspace in una posizione differente della rete.

2.6 Il modello Internet Suspend/Resume

Il termine mobile computing viene ormai utilizzato per identificare l'uso di computer portatili o palmari. D'altro canto, il precipitare dei prezzi dell'hardware conduce all'idea

che forse un giorno le infrastrutture per il pervasive computing possano magari eliminare la necessità di portarsi dietro un pc.

Il modello Internet Suspend/Resume [7, 8] (ISR) introduce un approccio innovativo per il mobile computing; si basa in sostanza sull'emulazione delle procedure di sospensione e riesumazione di un computer portatile, e sull'ipotesi che sia possibile trovare ovunque hardware disponibile, da utilizzare in sostituzione al proprio pc portatile.

Il meccanismo implementato in ISR consiste nel sospendere il Sistema Operativo e l'intero insieme di processi attivi su una macchina fisica, eseguirne la migrazione e successivamente la riesumazione su una macchina differente. Risulta evidente che un importante requisito da garantire per l'implementazione di un simile meccanismo è il seguente: non è sufficiente congelare e salvare lo stato persistente del Sistema Operativo, ma è necessario farlo anche per quello volatile, che comprende ad esempio lo stato di esecuzione delle applicazioni interattive.

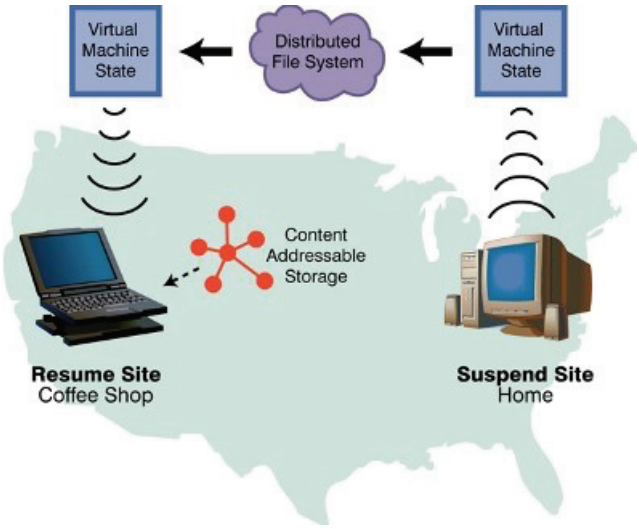

Figura 12 - Il modello ISR

Il modello ISR utilizza i meccanismi di virtualizzazione, alla base dei quali compare un particolare strato software che si definisce Virtual Machine Monitor (VMM), il quale si occupa tipicamente di mappare lo stato volatile delle macchine virtuali sul file system locale dell'host che le ospita. Nel caso in cui l'esecuzione di una macchina virtuale venga sospesa, il file associato alla macchina viene aggiornato in modo da riflettere lo stato di esecuzione volatile al momento della sospensione. Successivamente sarà possibile copiare il file sul file system di un nodo remoto, sul quale un VMM in esecuzione si occuperà di eseguire la riesumazione della macchina virtuale; tale procedura corrisponde in sostanza alla migrazione della macchina virtuale e dello stato di esecuzione volatile del sistema operativo ospite.

Eseguire la migrazione mediante l'uso delle macchine virtuali è certamente più semplice ed efficace rispetto al caso in cui vengano invece utilizzati meccanismi per la migrazione di processo. Eseguendo la migrazione a livello macchina, lo stato di esecuzione volatile viene accuratamente incapsulato, risolvendo in tal modo problemi di residual dependency legati ad esempio alla gestione di descrittori di file aperti. La migrazione di macchine virtuali non soffre poi della eventuale eterogeneità dei sistemi, mentre invece con la migrazione di processo è necessario ci sia una forte corrispondenza tra nodo sorgente e nodo destinazione.

ISR propone inoltre l'uso di file system distribuiti per facilitare il meccanismo di migrazione delle macchine virtuali. In definitiva, con ISR la migrazione sarà per il Sistema Operativo ospite come la chiusura, lo spostamento e la riapertura di un computer portatile.

Capitolo 3
Descrizione dell'architettura per il Service Switching

Descriveremo in questo capitolo ed in quello successivo, il risultato del lavoro di progettazione del modello e realizzazione dell'architettura, in relazione al paradigma Service Switching. Riporteremo dapprima l'analisi dei requisiti ed una breve introduzione agli strumenti utilizzati. Passeremo quindi alla descrizione delle funzionalità offerte dalla principale componente dell'architettura, il nodo di rete Service Switch, considerando in un primo momento solo scenari di tipo statico, o meglio, prescindendo da quali siano le entità che si occupano di eseguire il service management e da come venga implementato. Ci occuperemo successivamente di analizzare il ruolo svolto dal Service Switch durante le fasi fondamentali del ciclo di vita del servizio, descriveremo quindi le operazioni di attivazione e migrazione; analizzeremo le suddette procedure, poi in seguito parleremo dei criteri con cui viene eseguita la selezione dei nodi in esse coinvolti. Presenteremo poi una particolare implementazione del nodo Service Switch ottenuta da una caratterizzazione del modello proposto. Nel Capitolo successivo ci occuperemo invece di aspetti legati a resilienza, affidabilità e sicurezza, ed introdurremo gli strumenti utilizzati ed i meccanismi implementati per garantire tali requisiti.

3.1 Analisi dei requisiti

Obiettivo del nostro lavoro è quello di progettare un modello che supporti il paradigma del Service Switching, quindi realizzare una architettura che ne verifichi le specifiche.

Il primo passo eseguito è stato quello di classificare i requisiti che il sistema da realizzare dovrà necessariamente garantire. In relazione a quanto detto nel Capitolo I ed in riferimento al generico servizio, al nodo Service Switch è assegnato il compito di eseguire le seguenti funzionalità:

a) ospitare servizi ai quali fornire un ambiente di esecuzione sicuro e differenziato;

b) assegnare ai servizi risorse computazionali, memoria e risorse per le comunicazioni;

c) fornire supporto alla migrazione di servizio tra nodi geograficamente distribuiti.

Essendo tali funzionalità delle fondamentali prerogative che il sistema proposto deve presentare, le identifichiamo come requisiti funzionali. In relazione a questi, a seguito di una lunga e attenta fase di studio, valutazione e confronto tra diversi strumenti presi in esame, ne abbiamo selezionati alcuni che di sicuro potranno agevolarci nella realizzazione della nostra architettura:

a) meccanismi di virtualizzazione;

b) modello Mobile IP.

Grazie all'uso dei meccanismi di virtualizzazione siamo in grado di garantire i due requisiti in testa alla lista di cui sopra; ispirandoci al modello Mobile IP, invece, implementiamo un meccanismo di re-direzione di flussi di traffico grazie al quale siamo in grado di supportare la migrazione di servizio tra nodi geograficamente distribuiti.

Prendiamo poi in esame aspetti legati alla sicurezza e all'affidabilità in relazione alle comunicazioni tra nodi Service Switch. Anche in riferimento a tale contesto eseguiamo la selezione di alcuni strumenti da sfruttare nella nostra implementazione al fine di garantire quelli che invece possiamo classificare come requisiti non funzionali; utilizzeremo in particolare il protocollo MPLS (MultiProtocol Label Switching).

Introdurremo nel seguito di tale capitolo i meccanismi di virtualizzazione "system level" ed il modello Mobile IP. In quello successivo, invece, ci occuperemo di aspetti

legati alla resilienza e all'affidabilità, introdurremo quindi MPLS ed alcuni meccanismi opportunamente realizzati e sfruttati nell'architettura proposta.

3.1.1 Introduzione ai meccanismi di virtualizzazione e Xen

Le architetture dei moderni calcolatori sono sufficientemente potenti per supportare la virtualizzazione; virtualizzare un sistema consiste nel partizionare una macchina fisica e renderla in grado di supportare l'esecuzione concorrente di differenti Sistemi Operativi, fornendo in questo modo l'illusione che un certo numero di macchine virtuali (Virtual Machine - VM) siano contemporaneamente attive sul medesimo nodo fisico. Implementare questo tipo di meccanismo fornisce diversi vantaggi, in particolare un elevato livello di isolamento tra applicazioni in esecuzione su differenti macchine virtuali ospitate dal medesimo nodo fisico: le applicazioni in esecuzione su una macchina virtuale non compromettono in alcun modo quelle in esecuzione su altri domini virtuali. In secondo luogo, la virtualizzazione fornisce un supporto a differenti Sistemi Operativi, e consente quindi di ospitare applicazioni tra loro eterogenee sul medesimo calcolatore; il prezzo da pagare per ottenere tali vantaggi, è solo un piccolo overhead aggiuntivo legato all'esecuzione di software per la gestione delle macchine virtuali.

Il supporto all'esecuzione concorrente di diversi Sistemi Operativi richiede l'introduzione di uno strato software che si interpone tra l'hardware della macchina fisica e le macchine virtuali, e che viene identificato come Virtual Machine Monitor (VMM) o Hypervisor. Tale modulo si occupa di fornire alle macchine virtuali un'astrazione della macchina fisica sottostante, fornendo ad ogni Sistema Operativo ospite l'impressione di essere l'unico in esecuzione sull'intera macchina.

Nei sistemi di virtualizzazione tradizionali, anche noti come Full Virtualization system, il VMM fornisce un'astrazione del tutto identica all'hardware reale, e ciò consente di ottenere un importante vantaggio: non è necessario apportare alcuna modifica al generico Sistema Operativo ospite. Questo tipo di approccio comporta però problemi

nelle architetture x86-based, le quali non presentano alcun supporto alla virtualizzazione.

In alternativa è possibile utilizzare i sistemi di paravirtualizzation, in cui il VMM implementa un'astrazione simile ma non del tutto identica all'hardware sottostante. In tal caso è però necessario apportare alcune modifiche al kernel del Sistema Operativo ospite; vediamo il perché: determinate istruzioni di livello "supervisor" devono necessariamente essere intercettate dal VMM, ma eseguire tali istruzioni con privilegi insufficienti comporta dei fallimenti; al fine di risolvere tale problema, alcuni sistemi di paravirtualizzazione come ad esempio VMWare, riscrivono in maniera dinamica alcune porzioni del codice del kernel, con l'obiettivo di inserire delle trap ovunque sia necessario l'intervento del VMM.

Tuttavia, considerata la complessità dei sistemi x-86 based, la paravirtualizzazione offre comunque grossi vantaggi se paragonata ai tradizionali meccanismi di Full Virtualization. Pur essendo necessario apportare modifiche al Sistema Operativo ospite, con la paravirtualizzazione non infatti è necessario modificare l'Application Binary Interface (ABI), ne tanto meno la generica applicazione in esecuzione sulla VM.

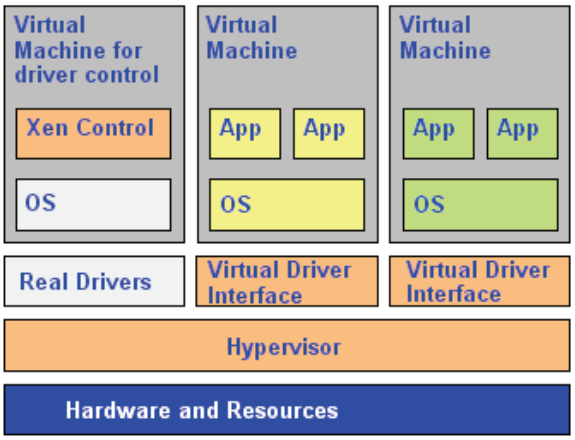

Figura 13 - Virtualizzazione a livello sistema

Per l'implementazione della nostra architettura abbiamo selezionato Xen [9], un Virtual Machine Monitor realizzato all'Università di Cambridge che consente a differenti Sistemi Operativi di condividere il medesimo hardware in modo sicuro, e senza alcun impatto sulle prestazioni e le funzionalità. In Xen, il generico Sistema Operativo ospite viene identificato come XenoLinux; ogni istanza di quest'ultimo esporta una ABI identica ad un kernel linux non virtualizzato. Xen implementa il multiplexing delle risorse fisiche a livello di Sistema Operativo, ed è in grado di fornire un elevato livello di isolamento tra differenti macchine virtuali. Rispetto alla virtualizzazione process-level c'è ovviamente un prezzo da pagare: eseguire un intero Sistema Operativo è certamente più pesante che eseguire un processo, sia in termini di inizializzazione che in termini di utilizzo delle risorse.

Nel seguito descriveremo l'astrazione di macchina virtuale esportata da Xen, e discuteremo del modo in cui il Sistema Operativo ospite deve essere modificato. Utilizzeremo il termine domain per indicare la generica macchina virtuale, ed i termini Xen o Hypervisor per indicare invece il Virtual Machine Monitor. Faremo riferimento ai seguenti principali aspetti: gestione della memoria, CPU, dispositivi di I/O.

Virtualizzare la memoria è di certo la parte più complicata nel lavoro di paravirtualizzazione di una architettura, sia in termini di meccanismi da implementare sull'Hypervisor, che in termini di modifiche da apportare al Sistema Operativo ospite. Il compito da svolgere risulta relativamente semplice nel caso in cui l'architettura presenti un TLB (Translation Lookaside Buffer) manipolabile via software, in quanto risulta semplice da virtualizzare; purtroppo le architetture x86-based non presentano tale caratteristica. Considerata tale limitazione, in Xen vengono imposte le seguenti regole: (i) i Sistemi Operativi ospite sono responsabili dell'allocazione e della gestione delle hardware page table, con un minimo contributo di Xen per garantire sicurezza e isolamento; (ii) Xen è alloggiato in una sezione di 64MB in testa ad ogni spazio di indirizzamento, evitando in tal modo la cancellazione del TLB in corrispondenza dell'entrata o l'uscita dall'Hypervisor. Ogni qual volta un Sistema Operativo ospite richiede una nuova tabella delle pagine, ad esempio nel caso in cui venga creato un

nuovo processo, deve allocare ed inizializzare una pagina della propria memoria e registrarla su Xen. Inoltre, il Sistema Operativo deve rinunciare ai privilegi per la scrittura diretta sulla tabella delle pagine della memoria: tutti gli aggiornamenti devono essere validati da Xen.

Virtualizzare la CPU comporta diverse implicazioni per il Sistema Operativo ospite: l'inserimento di un Hypervisor al di sotto di quest'ultimo, viola l'usuale assunzione secondo cui il Sistema Operativo è l'entità più privilegiata all'interno del sistema. Al fine di proteggere l'Hypervisor dal malfunzionamento dei Sistemi Operativi ospite, questi ultimi devono essere modificati ed essere eseguiti su un livello di privilegio inferiore. Nei sistemi x86-based, è possibile virtualizzare in maniera efficiente i livelli di privilegio perché ne supportano 4 in hardware, solitamente identificati come ring (anelli), e numerati da 0 (più privilegiato) a 3 (meno privilegiato). Tipicamente il codice del Sistema Operativo viene eseguito sul ring 0, in quanto nessun altro ring può eseguire istruzioni privilegiate, mentre invece il ring 3 viene utilizzato per il codice delle applicazioni; i ring 1 e 2 di norma non vengono utilizzati. Ogni Sistema Operativo che rispetta questa organizzazione, se modificato ed eseguito sul ring 1, può essere portato su Xen, in questo modo resta quindi isolato dalle applicazioni. Le istruzioni privilegiate vanno validate da Xen prima di poter essere eseguite; tale procedura viene identificata come paravirtualizzazione delle istruzioni.

Per quanto riguarda infine la virtualizzazione dei dispositivi hardware, anziché emularne di esistenti così come viene fatto con la Full Virtualization, Xen ne esporta una serie di semplici e chiare astrazioni; utilizzando memoria condivisa, l'I/O viene trasferito da e verso ogni dominio attraverso Xen.

L'architettura di Xen è illustrata in Figura 14. Uno dei domini gestiti dall'Hypervisor viene creato durante l'inizializzazione della macchina, e a tale dominio è consentito l'uso delle interfacce di controllo. Il Domain0, questo è il nome con cui viene identificato nella terminologia Xen, ospita il software di gestione di livello applicazione. Le interfacce di controllo forniscono al Domain0 l'abilità di creare e terminare altri

domini virtuali, di controllare i parametri di scheduling ad essi associati, di allocare la memoria e di fornire l'accesso ai dispositivi che la macchina fisica mette a disposizione.

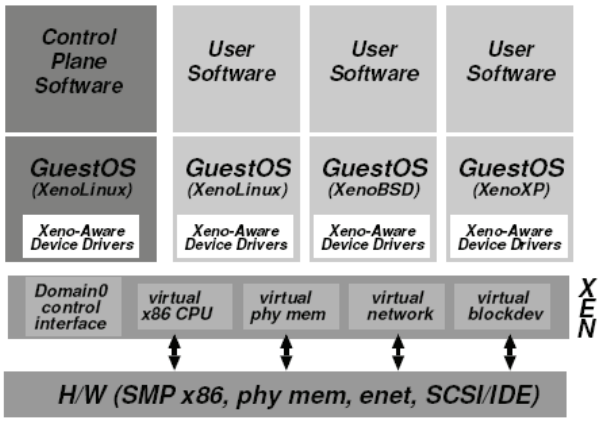

Figura 14 - L'architettura di Xen

La scelta di Xen è il risultato di una attenta fase di selezione. Facendo riferimento alla letteratura, abbiamo dapprima individuato gli strumenti per la virtualizzazione maggiormente utilizzati in virtù delle potenzialità offerte, quindi, considerato un insieme ridotto di questi ultimi, abbiamo eseguito dei test con l'obiettivo di metterne in evidenza le caratteristiche; i parametri presi in considerazione sono i seguenti: scalabilità, flessibilità, semplicità d'uso. La descrizione ed i risultati dei test che ci hanno portato alla scelta dello strumento utilizzato, sono riportati nel Capitolo relativo alla valutazione delle prestazioni.

Data la centralità dell'utilizzo delle tecniche di virtualizzazione nel paradigma Service Switching, ci siamo inoltre occupati di esplorare le potenzialità ed i limiti dell'uso di tali tecnologie in contesti analoghi, quali i sistemi per la Network Emulation ed il Grid computing.

Nel contesto dell'emulazione di rete, gli strumenti per la virtualizzazione sono largamente impiegati al fine di incrementare l'isolamento e di consentire l'esecuzione concorrente di differenti esperimenti, ed il loro uso è alla base dei meccanismi per la gestione delle risorse in sistemi cluster based come Emulab, oppure testbed distribuiti come Planetlab. Il nostro contributo in tale contesto è stato quello di realizzare NEPTUNE (Network Emulation for Protocol TUNing and Evaluation) [45], un sistema flessibile e scalabile per l'emulazione di scenari di rete.

Con i Virtual Workspace si introduce l'uso dei meccanismi di virtualizzazione anche in ambiente Grid, al fine di adeguare l'ambiente di esecuzione alle particolari necessità dell'utente. Rimanendo ne contesto della Network Emulation, ed in riferimento al concetto di Virtual Workspace, il nostro contributo [46] è stato quello di definire un modello in cui un Globus Virtual Workspace possa essere utilizzato alla base dell'implementazione di un sistema distribuito per l'emulazione di rete.

3.1.2 Introduzione al modello Mobile IP

In IP versione 4 l'indirizzo associato ad un nodo di rete identifica univocamente il punto in cui il nodo stesso è attaccato ad Internet. Al fine di ricevere datagrammi ad esso destinati, il nodo deve quindi necessariamente essere localizzato all'interno della rete indicata dal proprio indirizzo IP; nel caso in cui tale condizione non fosse rispettata, diventerebbe impossibile consegnare datagrammi al nodo in questione. Esistono due soluzioni in grado di consentire ad un nodo che abbia necessità di cambiare il punto di accesso ad Internet, di spostarsi senza necessariamente perdere la capacità di comunicare:

a) il nodo deve cambiare il proprio indirizzo IP;
b) bisogna propagare delle rotte di tipo host-specific in tutta la rete Internet, o almeno in buona parte.

Entrambe le alternative risultano comunque inaccettabili: la prima non consente al nodo di mantenere connessioni attive, mentre la seconda presenta invece degli evidenti problemi di scalabilità.

Il modello IP Mobility Support [RFC2002] nasce con l'obiettivo di fornire un supporto alla mobilità dei nodi tra due sottoreti eterogenee, senza soffrire dei problemi appena elencati.

Mobile IP introduce le seguenti entità:

Mobile Node: si tratta di un router o di un host che può cambiare il punto di accesso ad Internet e spostarsi da una sottorete ad un'altra senza modificare il proprio indirizzo IP; tale nodo è in grado di mantenere attive le comunicazioni con altri nodi sfruttando un indirizzo IP costante, assumendo che la connettività link-layer sia disponibile al nuovo punto di accesso. Ad ogni Mobile Node è associata una rete identificata come Home Network.

Home Agent: è un router sulla Home Network del generico Mobile Node in grado di implementare un meccanismo di tunneling mediante il quale inoltrare datagrammi destinati al Mobile Node che sia all'esterno della propria Home Network. L'Home Agent è inoltre in grado di mantenere informazioni relative alla posizione attuale del Mobile Node.

Foreign Agent: si tratta di un router sulla rete visitata dal generico Mobile Node, al quale fornisce servizi fin tanto che quest'ultimo risulta registrato. Il Foreign Agent esegue il detunneling e consegna datagrammi destinati al Mobile Node e ricevuti dal suo Home Agent. In relazione invece ai datagrammi inviati dal Mobile Node, il Foreign Agent implementa la funzione di default router.

Ad ogni Mobile Node è associato un indirizzo IP della propria Home Network, e tale indirizzo viene identificato come Home Address. Quando risulta essere fuori dalla Home Network, al Mobile Node viene associato un Care-of Address che ne riflette la

posizione corrente. Il Mobile Node utilizza sempre il proprio Home Address come indirizzo sorgente nei datagrammi IP inviati.

Quando il Mobile Node è fuori dalla Home Network, Mobile IP utilizza un protocollo di tunneling per nasconderne l'Home Address ai router che si trovano sul percorso tra la Home Network e la posizione corrente del Mobile Node. Il tunnel termina sul Care-of Address associato al Mobile Node, e deve necessariamente trattarsi di un indirizzo verso cui è possibile inoltrare datagrammi utilizzando il routing IP convenzionale. In corrispondenza del Care-of Address, il datagramma originale viene rimosso dal tunnel e consegnato al Mobile Node.

Esistono due vincoli da rispettare per l'uso del modello Mobile IP:

a) un Home Agent deve essere in grado di attirare e intercettare datagrammi destinati all'Home Address di ogni Mobile Node registrato;

b) Mobile Node e Foreign Agent devono poter scambiare datagrammi senza affidarsi ai meccanismi di routing IP. Tale requisito può essere soddisfatto imponendo che Foreign Agent e Mobile Node abbiano entrambi una interfaccia sul medesimo link, in modo tale che possano bypassare i meccanismi di routing IP semplicemente utilizzando quelli di livello data-link.

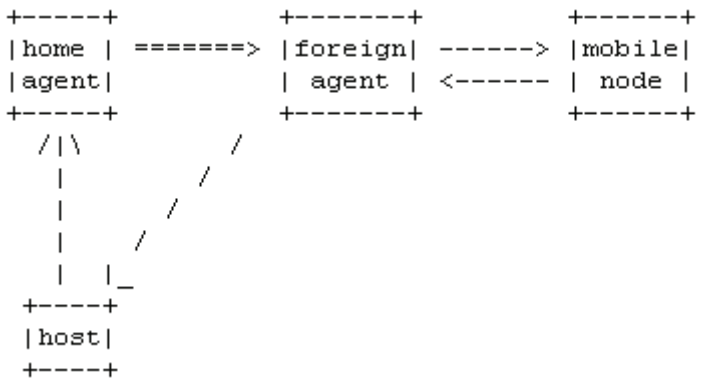

Figura 15 - Il modello IP Mobility Support

3.2 Introduzione alle funzionalità del nodo Service Switch

Ci occuperemo in questo paragrafo di descrivere il modello Service Switching, ed in particolare le funzionalità offerte dalla sua principale componente, il nodo di rete Service Switch. Faremo riferimento ai meccanismi di virtualizzazione ed al modello Mobile IP, di cui riprendiamo in parte la terminologia. Impiegheremo per la descrizione un approccio di tipo step-by-step: partendo da uno scenario in cui i nodi Service Switch sono dislocati esclusivamente ai margini della rete, descriveremo le caratteristiche di tale nodo considerando inizialmente solo un insieme ridotto delle funzionalità offerte. Tale scenario verrà progressivamente esteso introducendo dapprima l'utilizzo dei nodi Service Switch nel cuore della rete, e successivamente il supporto alla coesistenza di macchine virtuali associate ad indirizzi pubblici o privati. Ci occuperemo di volta in volta di descrivere le funzionalità aggiuntive del nodo Service Switch necessarie a verificare le specifiche di ogni scenario preso in esame.

3.2.1 Scenario semplificato

Prima ancora di partire con la descrizione del modello, forniamo innanzitutto alcuni dettagli relativi alla struttura e all'organizzazione del generico nodo fisico con supporto alla virtualizzazione, con particolare riferimento ai meccanismi per la gestione delle risorse, alle comunicazioni intranodo, e a quelle internodo tra macchine virtuali.

Per quanto riguarda il primo punto, sfruttiamo alcuni strumenti noti in letteratura [5]: XenMon, SEDF-DC, ShareGuard.

XenMon è uno strumento che esegue il monitoraggio delle performance e fornisce informazioni relative al livello di utilizzo della CPU da parte di ogni singolo dominio virtuale attivo sul nodo fisico, compreso quelli su cui girano i processi per la gestione del sistema. SEDF-DC (Simple Earliest Deadline First – Debt Collector) è uno schedulatore della CPU basato su prenotazione, mediante il quale un amministratore può specificare la quantità di CPU da allocare ad ogni macchina virtuale. Tale modulo

riceve periodicamente dei feedback da XenMon al fine di riorganizzare lo schema di allocazione della risorse e ottimizzarne quindi l'utilizzo. ShareGuard implementa un meccanismo finalizzato al controllo sull'utilizzo della CPU; interroga periodicamente XenMon, e se necessario è in grado di imporre l'interruzione del traffico da o verso lo specifico dominio virtuale che ha superato il limite prefissato.

Vediamo adesso in che modo vengono implementate le comunicazioni tra macchine virtuali attive all'interno del medesimo nodo fisico. Il Domain0 di Xen gestisce i driver dei dispositivi hardware, comprese le interfacce di rete; ad ogni interfaccia fisica è collegato un certo numero di canali di I/O. Uno dei terminali del canale è gestito dal Domain0 e viene identificato come interfaccia di back-end; l'altro terminale è gestito dal generico dominio virtuale e viene identificato come interfaccia di front-end. Xen fornisce un meccanismo che si definisce host networking via bridging [4], mediante il quale le interfacce di back-end vengono collegate alla relativa interfaccia reale mediante l'uso di un bridge virtuale. Ogni qual volta un pacchetto venga ricevuto sull'interfaccia reale, viene eseguito un meccanismo di demultiplexing in funzione dell'indirizzo di livello Data Link (MAC Address) della destinazione, che sarà ovviamente diverso per ogni singola macchina virtuale ospitata sul nodo fisico. Utilizzando tale approccio, non risulta indispensabile configurare le intefacce di back-end con un indirizzo IP, per cui lo saranno solo quelle di front-end. Anche il Domain0 riceve ed invia traffico attraverso le interfacce virtuali di front-end collegate al bridge virtuale e quindi alle interfacce reali.

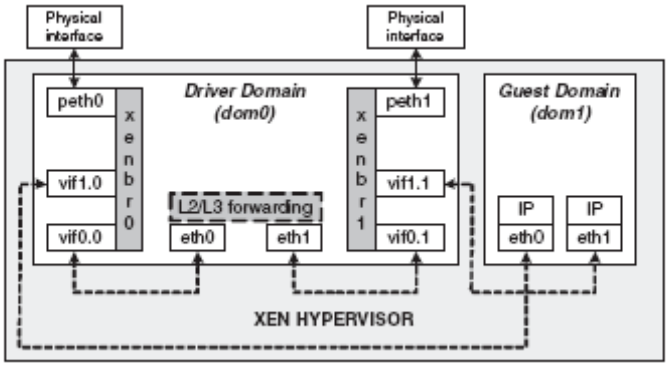

Figura 16 - Host Networking via Bridging

Le comunicazioni internodo vengono invece supportate implementando Link Multiplexing sfruttando un particolare meccanismo che definiamo IP-aliasing and destination MAC address filtering [45]. Tale meccanismo consiste nell'attivare una virtual NIC (Network Interface Card) ed associarla ad una interfaccia virtuale della Virtual Machine. Il processing del traffico viene eseguito implementando delle discipline di accodamento direttamente sulle interfacce virtuali.

Passiamo adesso alla descrizione di un primo scenario che indichiamo come semplificato; in tale scenario è previsto l'uso di un certo numero di nodi Service Switch localizzati ai margini di uno stesso Administrative Domain (ad esempio lo stesso Autonomous System). Per conferire scalabilità nella allocazione dei servizi, si assume che i nodi a cui è assegnato tale compito possano essere parte di cluster di computer. Una delle peculiarità del paradigma consiste nell'utilizzo di macchine virtuali per ospitare i servizi, pertanto si assume che ciascuno dei cluster suddetti costituisca un Virtual Execution Environment (VEE) nel quale sia possibile istanziare macchine virtuali. Il nostro modello assume la presenza di un nodo Service Switch per ogni VEE. Nel seguito chiameremo Correspondent Node i terminali della rete mediante i quali gli utenti accedono ai servizi ospitati dalla piattaforma.

Su ognuna delle VM è attivo un singolo servizio, oppure n servizi che abbiano tra loro una specifica relazione. Al servizio (o insieme di servizi) è associato univocamente un

indirizzo IP pubblico che identifichiamo come “Home Address”, e che viene utilizzato per configurare il virtual NIC della VM che lo ospita; l’indirizzo MAC associato alla VM dovrà essere unico all’interno dell’intera piattaforma. Nel modello proposto la relazione tra servizio e macchina virtuale è molto forte, si farà infatti riferimento ad una entità piuttosto che all’altra in maniera indistinta.

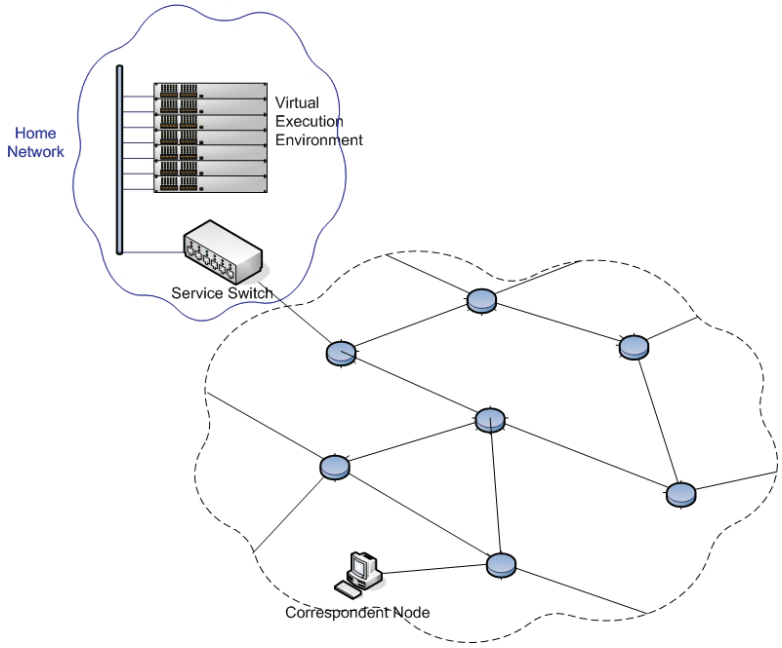

Figura 17 - Le componenti dell’architettura

Le VM attive sul medesimo server fisico sono interconnesse tra loro mediante il meccanismo dell’host networking via bridging. Le comunicazioni tra VM attive su nodi differenti del medesimo VEE vengono supportate implementando Link Multiplexing col meccanismo dell’IP-aliasing and destination MAC address filtering. Essendo configurata con un IP Pubblico, la generica VM sarà inoltre visibile e raggiungibile anche dall’esterno della rete LAN di cui fa parte il VEE che la ospita.

Posto a monte del VEE in relazione al traffico destinato alle VM, il Service Switch è un router che svolge funzioni aggiuntive rispetto al tipico forwarding IP-based. Assumiamo

che il Service Switch sia il Default Gateway della rete LAN di cui fa parte; successivamente giustificheremo tale assunzione.

Indichiamo col termine “Home Network” l’insieme composto dal VEE e dal Service Switch che ne gestisce il traffico.

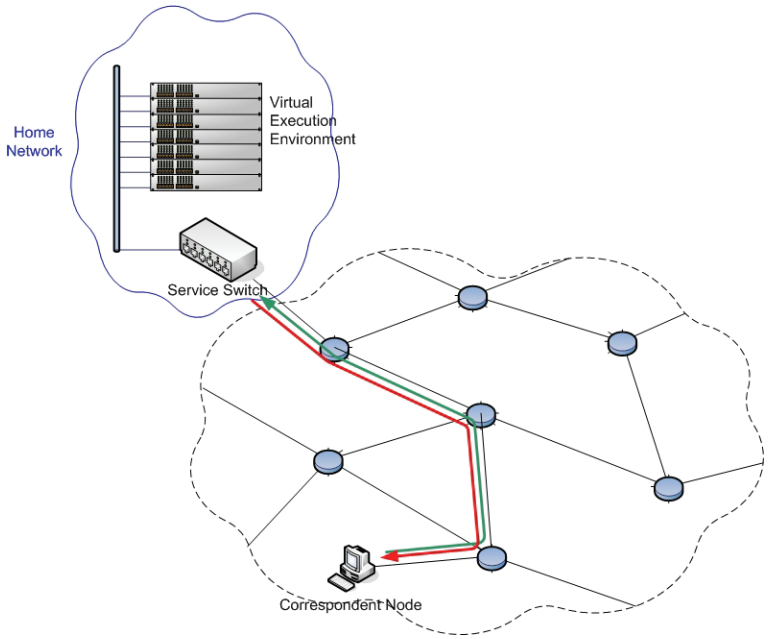

Figura 18 - Traffico da e verso le macchine virtuali

Il generico Correspondent Node interessato a fruire di uno specifico servizio, inoltra pacchetti verso la VM che lo ospita, specificando come IP destinazione l’Home Address associato alla VM stessa.

Tali pacchetti verranno processati dal Service Switch sulla Home Network, il quale si occupa innanzitutto di verificare se la VM a cui i datagrammi sono destinati, si trovi o meno all’interno della Home Network; esegue tale operazione interrogando una tabella che chiamiamo *Mobile Binding Table*, la quale possiede una entry per ogni VM sul VEE, e mantiene una corrispondenza tra l’Home Address della VM ed una

informazione relativa al punto di accesso ad Internet che la VM utilizza nel caso sia fuori dalla propria Home Network. Supponiamo per il momento che la VM si trovi all'interno di quest'ultima; in tal caso il Service Switch non farà altro che consegnare i pacchetti processati alla VM, la quale risponderà inoltrando pacchetti direttamente verso il Correspondent Node.

Home Address	Care-of Address
143.225.229.123	NULL
143.225.229.134	143.225.81.254
…	…

Tabella 1 - Mobile Binding Table

Supponiamo adesso che nasca l'esigenza di migrare uno specifico servizio, quindi la VM che lo ospita, verso un differente nodo fisico; prescindiamo per il momento da quali siano le entità che decidono di attivare tale processo e da quelle che si occupano di gestire e monitorare la procedura stessa. Possono presentarsi due casi: il physical node destinazione della migrazione fa parte del medesimo VEE; il physical node destinazione fa parte di un VEE differente. Nel primo caso la VM continuerà a far parte della medesima Home Network, per cui il Service Switch non dovrà eseguire alcuna funzione aggiuntiva rispetto alla situazione descritta in precedenza. Nel secondo caso, invece, la VM verrà ospitata da una differente rete che identifichiamo come "Foreign Network"; "Care-of Address" sarà l'indirizzo IP del Service Switch su quest'ultima.

La VM non è consapevole dello spostamento che ha subito, e continuerà ad utilizzare il proprio Home Address come indirizzo sorgente per i datagrammi in uscita; risulta evidente che sussiste un problema di incompatibilità tra l'IP della VM ospite con quelli

utilizzati nella Foreign Network. Tale problema risulta però implicitamente risolto considerando che il Service Switch ed il nodo fisico su cui è attiva la VM ospite, si trovano sulla medesima LAN, e che la rete virtuale che interconnette le varie VM sul nodo è realizzata col meccanismo del bridging; le comunicazioni verranno quindi implementate a livello Data-Link.

Come vedremo in seguito, la migrazione di servizio si completata con l'esecuzione di una procedura definita di *registrazione*, che si compone di alcune operazioni da eseguire lato Home Network, ed alcune altre lato Foreign Network. Sulla prima bisogna aggiornare la Mobile Binding Table in modo da riflettere lo spostamento della VM, quindi associando il Care-of Address della Foreign Network con l'Home Address della VM. Durante la permanenza della VM all'esterno della Home Network, il Service Switch svolgerà per quest'ultima la funzione di Proxy ARP, quindi intercetta i pacchetti inoltrati dagli host della rete locale, e li instrada verso la VM migrata sfruttando il medesimo meccanismo descritto di seguito ed utilizzato nel caso in cui la sorgente sia il generico Correspondent Node esterno alla Home Network. Come detto in precedenza, la VM migrata non è cosciente dello spostamento a cui è stata sottoposta, per cui bisogna garantire in qualche modo che i pacchetti inviati dalla VM siano correttamente consegnati, sia nel caso in cui la destinazione faccia parte della Home Network della VM, sia nel caso in cui la destinazione sia invece il generico Correspondent Node all'esterno della Home Network. In questa seconda circostanza, la VM tenta di inviare i pacchetti verso il proprio Default Gateway, quindi il Service Switch sulla propria Home Network, il quale però non risulta direttamente raggiungibile. Al fine di risolvere tali problemi, fin tanto che la VM è ospite all'interno di una specifica Foreign Network, il Service Switch su quest'ultima deve intercettare tutti i pacchetti in uscita dalla VM. La procedura di registrazione lato Foreign Network consiste quindi nell'abilitare un IP Aliasing sull'interfaccia con cui il Service Switch si collega alla rete locale, in modo che i pacchetti inoltrati dalla VM ospite passino tutti attraverso il Service Switch.

A seguito della migrazione, il Correspondent Node continuerà ad inoltrare pacchetti verso la VM utilizzando l'Home Address ad essa associato; tale caratteristica costituisce

un enorme vantaggio nell'uso del modello proposto, in quanto garantisce la trasparenza dell'operazione di migrazione nei riguardi dell'utente finale. I pacchetti inoltrati dal Correspondent Node continueranno ad essere instradati sul medesimo percorso, e continueranno quindi a passare attraverso il Service Switch della Home Network; quest'ultimo, interrogando la Mobile Binding Table, verrà a conoscenza della posizione attuale della VM migrata, creerà un tunnel il cui end-point corrisponde col Service Switch della Foreign Network che ospita la VM, ed attraverso il tunnel reinoltrerà i pacchetti verso quest'ultima. L'end point del tunnel eseguirà il detunneling del traffico ricevuto e provvederà a consegnare i pacchetti alla VM ospite.

Per quanto riguarda il traffico inverso, la VM migrata continuerà ad inoltrare pacchetti direttamente verso il Correspondent Node utilizzando come Source Address il proprio Home Address.

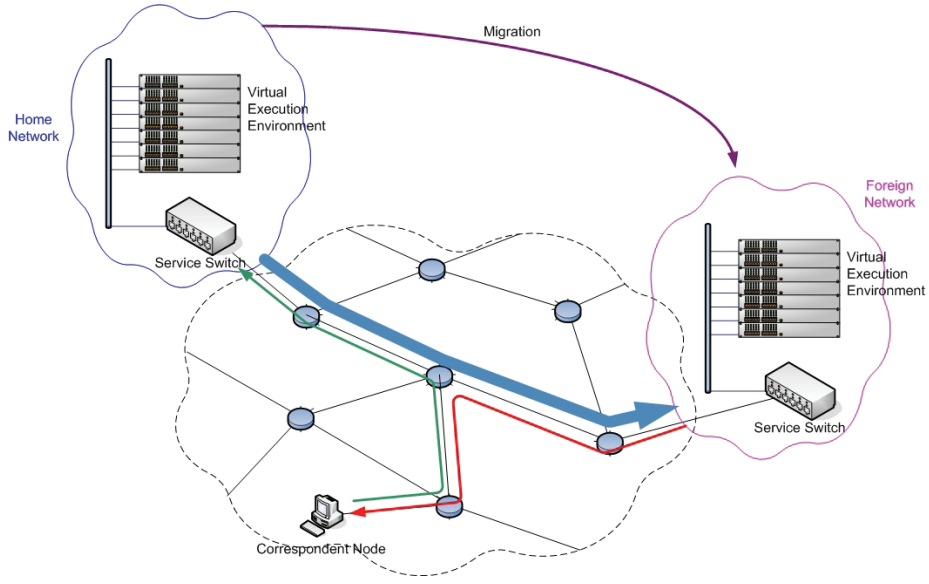

Figura 19 - Reindirizzamento del traffico destinato a macchine virtuali migrate

Occupiamoci adesso di giustificare la scelta di imporre che il Service Switch sia il Default Gateway della rete LAN di cui fa parte.

Nel paragrafo che introduce Mobile IP, abbiamo parlato di alcuni vincoli che vanno necessariamente verificati nel caso si cui si abbia necessità di utilizzare tale modello; li riportiamo di seguito:

a) L' Home Agent deve essere in grado di attirare e intercettare datagrammi destinati all'Home Address di ogni Mobile Node registrato.

b) Mobile Node e Foreign Agent devono poter scambiare datagrammi senza affidarsi ai meccanismi di routing IP. Tale requisito può essere soddisfatto imponendo che Foreign Agent e Mobile Node abbiano entrambi una interfaccia sul medesimo link, in modo tale che possano bypassare i meccanismi di routing IP semplicemente utilizzando quelli di livello data-link.

Nello scenario introdotto, il nodo Service Switch svolge, assieme ad altre funzionalità, quella di Home Agent sulla Home Network, e di Foreign Agent sulla Foreign Network. Assumere che il Service Switch sia il Default Gateway della rete LAN, ci consente di garantire che entrambi i requisiti siano rispettati. Il Service Switch è sul percorso tra Correspondent Node e VM, ed è quindi in grado di intercettare tutti i pacchetti inoltrati dall'uno all'altra; è inoltre connesso alla VM in modo che possa comunicare con quest'ultima sfruttando i meccanismi di forwarding di livello Data-Link.

3.2.2 Scenario evoluto

Descriviamo adesso uno scenario più esteso rispetto al precedente, e localizziamo i nodi Service Switch anche nel core della rete, pur considerando ancora valido il vincolo secondo cui il Default Gateway della generica LAN all'interno della quale compare un VEE, deve necessariamente essere un Service Switch. Identifichiamo i due nodi rispettivamente come Core Service Switch ed Edge Service Switch.

Se l'Edge Service Switch gestisce solo ed esclusivamente informazioni legate alle VM attive sulla propria Home Network, il Core Service Switch gestisce quelle relative a tutti i servizi ospitati dalla piattaforma; in altre parole, la Mobile Binding Table del Core

Service Switch possiede una entry per ogni singolo servizio in esecuzione su tutti i VEE.

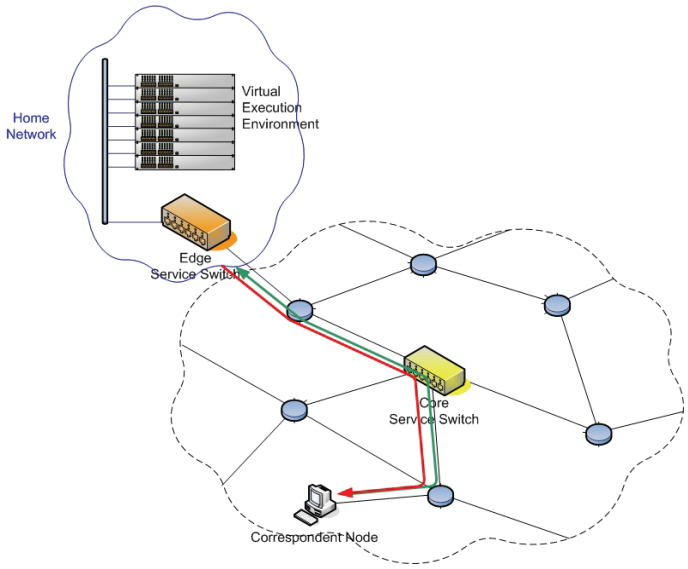

Figura 20 – Introduzione del Servce Switch Core

Considerato un simile scenario, è possibile ritrovare un numero variabile di Service Switch sul percorso che congiunge il generico Correspondent Node con la generica VM; nel caso limite ce ne sarà solo uno, l'Edge Service Switch sulla Home Network della VM in questione.

Prendiamo in considerazione uno specifico servizio e la VM che lo ospita, ed uno dei Correspondent Node che fruisce di tale servizio; supponiamo per semplicità che sul percorso tra Correspondent Node e VM ci sia un unico Core Service Switch.

Consideriamo dapprima il caso in cui la VM risieda all'interno della propria Home Network: il Core Servce Switch eseguirà il normale forwarding IP based rispettando la rotta definita dall'algoritmo di routing utilizzato.

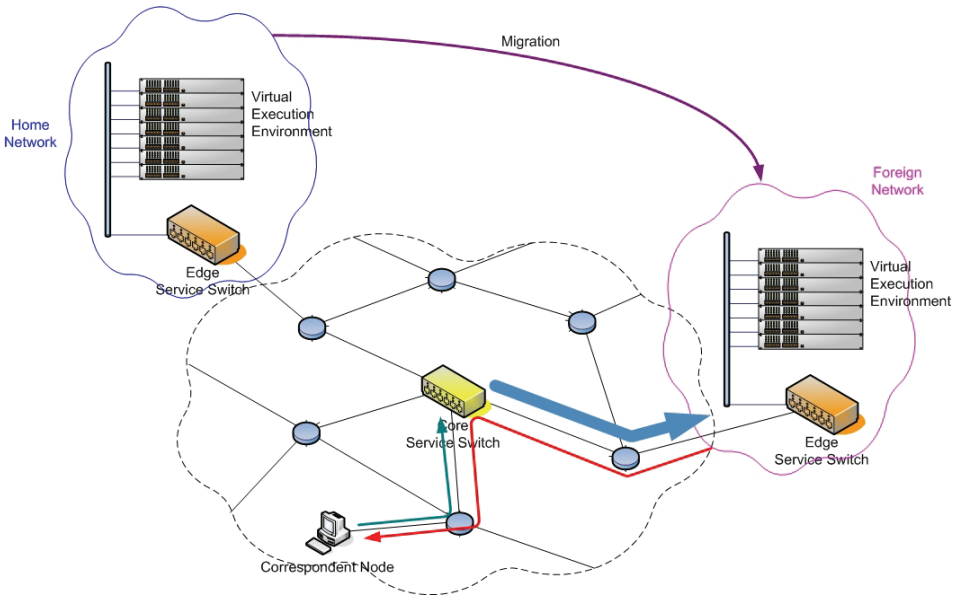

Figura 21 - Ottimizzazione del routing per il traffico destinato a macchine virtuali migrate

Supponiamo adesso che la VM venga migrata su una Foreign Network. Il Core Service Switch sul percorso tra Correspondent Node e VM, così come tutti gli altri presenti all'interno del dominio, eseguirà la procedura di registrazione ed aggiornerà la propria Mobile Binding Table associando il Care-of Address all'Home Address della VM migrata.

Da questo punto in poi, tutto il traffico processato dal Core Service Switch destinato alla VM migrata, verrà inoltrato attraverso un tunnel il cui end-point corrisponde col Service Switch della Foreign Network, il quale eseguirà il detunneling del traffico ricevuto e lo consegnerà alla VM.

L'introduzione di Core Service Switch nella rete, viene quindi eseguito allo scopo di ottimizzare l'instradamento di pacchetti destinati a VM migrate; il traffico destinato alla generica VM ospitata da una Foreign Network, non dovrà necessariamente passare attraverso l'Edge Service Switch sulla Home Network, ma potrà essere inoltrato su un percorso meno costoso.

Ci occuperemo in futuro di determinare un criterio secondo cui stabilire il posizionamento ottimale dei Core Service Switch all'interno della rete; per il momento possiamo comunque affermare che la funzionalità svolta da tali nodi e tanto più performante quanto più siano vicini ad un'area da cui vengono inoltrati datagrammi destinati alle VM ospitate dall'architettura.

3.2.3 Scenario generalizzato

Estendiamo ulteriormente lo scenario; fino ad ora abbiamo ipotizzato che tutte le VM attive sull'intera piattaforma siano associate ad indirizzi IP pubblici. Tale ipotesi limita fortemente la scalabilità del sistema, e pone vincoli sul numero massimo di servizi che la piattaforma può ospitare. Consideriamo quindi uno scenario in cui sia possibile far coesistere VM associate ad indirizzi pubblici e VM associate ad indirizzi privati, ed estendiamo le funzionalità del Service Switch affinchè sia possibile supportare tale condizione. In relazione agli aspetti affrontati in precedenza, il modello rimarrà del tutto invariato. Un importante requisito che va inevitabilmente rispettato è il seguente: l'associazione tra IP privato e VM deve essere univoco all'interno dell'intera architettura. È indispensabile imporre tale condizione al fine di evitare problemi di conflitto nell'uso di uno specifico indirizzo a seguito del processo di migrazione di una VM.

Uno dei problemi da affrontare in un simile scenario è quello relativo alla coesistenza sul medesimo server fisico, e quindi sulla medesima rete LAN, di VM associate ad indirizzi IP appartenenti a differenti classi di indirizzamento. In realtà abbiamo già affrontato il problema, in quanto una situazione del tutto analoga si verifica quando una generica VM viene ospitata da una Foreign Network. Anche in questo caso, quindi, la questione è implicitamente risolta: il Service Switch ed il nodo fisico che ospita la VM si trovano sulla medesima rete LAN, e le interconnessioni tra differenti VM sullo stesso nodo fisico sono implementate col meccanismo dell'host networking via bridging.

Nella rimanente parte di questo sottoparagrafo, faremo riferimento esclusivamente al sottoinsieme di VM associate ad indirizzi privati; per quanto riguarda il processing del traffico destinato alle VM associate ad un indirizzo pubblico, non è necessario considerare alcuna variazione rispetto a quanto descritto in precedenza.

Facciamo riferimento al caso in cui il generico Correspondent Node abbia necessità di sfruttare un servizio ospitato da una VM associata ad un indirizzo privato. Considerato che la validità di un IP privato è limitata alla rete locale di cui il physical o virtual node fa parte, risulta evidente che il Correspondent Node dovrà in qualche modo utilizzare un IP pubblico per instradare pacchetti verso la VM destinazione. Introduciamo allora la seguente condizione: l'Home Address delle macchine virtuali a cui è associato un indirizzo privato, è l'IP dell'Edge Service Switch sulla Home Network.

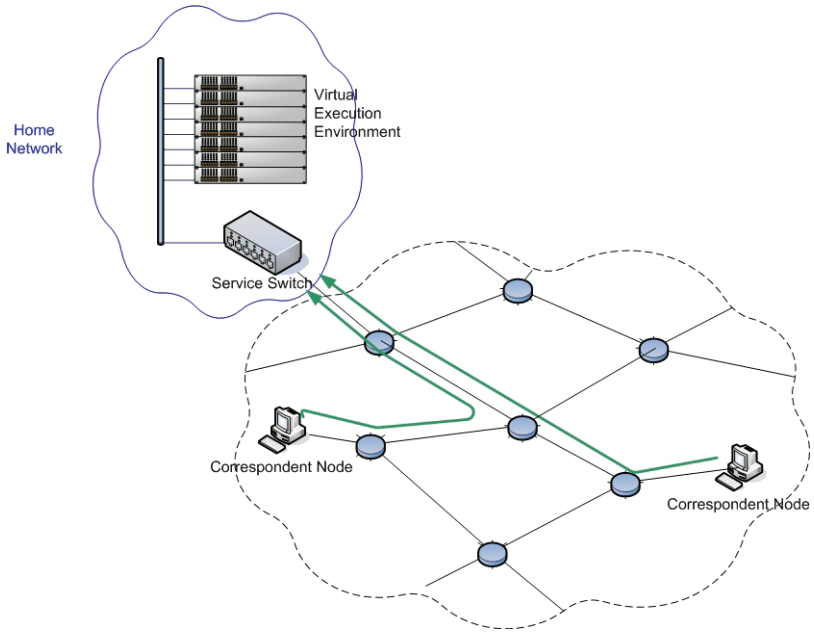

Figura 22 - Processing del traffico destinato a macchine virtuali con indirizzo privato

Fatta tale ipotesi, ne consegue che pacchetti destinati a differenti VM dello stesso VEE, presenteranno il medesimo indirizzo destinazione IP; è necessario quindi implementare un meccanismo che consenta di eseguire lo smistamento verso le differenti VM.

È possibile ottenere tale risultato implementando un meccanismo di Natting basato sul numero di porto, ed eseguendo quindi il demultiplexing del traffico in funzione della coppia di parametri <Protocol - Port Number>, campi rispettivamente riportati nell'header di livello rete ed in quello di livello trasporto. Affinché sia possibile utilizzare tale approccio, è però necessario che esista una associazione esclusiva tra la VM e la coppia <Protocol – Port Number>, requisito che in taluni casi non è possibile garantire. Prendiamo in esame un caso particolare, e cioè quello in cui vengano ospitati servizi associati ad un numero di porto well known, e che più VM all'interno della medesima Home Network offrano lo stesso servizio. Il traffico destinato alle VM in questione, non può essere smistato utilizzando il meccanismo proposto, bisogna quindi utilizzare un artificio per ovviare a tale problema; la soluzione che proponiamo consiste nel creare il mapping tra un singolo Port Number che indentificheremo come *Target*, ed un insieme di *Source* Port Number.

Consideriamo il seguente esempio: all'interno della medesima Home Network ci sono due VM con IP privato che ospitano entrambe un web server in ascolto sul porto 80; alla prima verrà associato il Source Port Number 80 e alla seconda il Source Port Number 8080. I Correspondent Node che intendono inoltrare pacchetti verso le VM dovranno utilizzare l'uno piuttosto che l'altro a seconda dei casi, di conseguenza, il Service Switch che esegue il processing dei pacchetti potrà tranquillamente distinguere quelli destinati ad una VM oppure all'altra. Prima di reinoltrare i pacchetti verso le VM, il Service Switch dovrà sostituire il Source Port Number nell'header di livello trasporto, col Target Port Number, altrimenti il server in esecuzione sulla VM li ignorerebbe.

Introduciamo una versione estesa della Mobile Binding Table:

Home Address	Protocol	Source Port Number	Private IP Address	Target Port Number	Care-of Address
143.225.229.254	TCP	80	192.168.10.10	80	NULL
143.225.229.254	TCP	8080	192.168.10.14	80	NULL
143.225.229.254	UDP	5060	192.168.10.12	5060	143.225.172.254
…	…	…	…	…	…

Tabella 2 - Mobile Binding Table estesa

Al solito, consideriamo dapprima il caso in cui la VM si trovi sella Home Network, e successivamente quello in cui venga ospitata da una Foreign Network a seguito di un processo di migrazione.

In entrambi i casi il Correspondent Node inoltrerà traffico destinato alla VM utilizzando il suo Home Address (l'IP del Service Switch sulla Home Network) come IP destinazione. In funzione poi del particolare servizio richiesto, verranno utilizzati un opportuno protocollo di livello trasporto ed un opportuno Port Number, parametri riportati nei rispettivi campi degli header di livello rete e di livello trasporto.

Se la VM risiede nella Home Network, a prescindere dalla presenza di Core Service Switch sul percorso tra Correspondent Node e VM, il traffico inoltrato dal Correspondent Node raggiungerà sempre l'Edge Service Switch sulla Home Network; quest'ultimo interrogherà la propria Mobile Binding Table in funzione della coppia di parametri <Protocol – Port Number> estrapolati dal pacchetto ricevuto. L'interrogazione è finalizzata alla ricerca di una entry associata al nodo, e quindi dell'IP

privato che il Service Switch utilizzerà per consegnare il pacchetto alla VM dopo aver eseguito la sostituzione del Port Number.

Per quanto riguarda il traffico inverso, la VM instraderà pacchetti direttamente verso il Correspondent Node, codificando l'Home Address nel campo Source Address dell'header di livello rete.

Consideriamo adesso il caso in cui la VM sia stata migrata su una Foreign Network; la entry della Mobile Binding Table ad essa associata subisce una variazione, in particolare il campo Care-of Address verrà settato con l'IP del Service Switch sulla Foreign Network. Ciò accade sia nella tabella dell'Edge Service Switch sulla Home Network, che in quelle di tutti i Core Service Switch distribuiti sulla rete. Consideriamo il caso generale, e supponiamo quindi che sul percorso tra Correspondent Node e VM migrata ci siano n Core Service Switch; raggiunto il primo di questi, i pacchetti vengono opportunamente processati e reinstradati verso la Foreign Network che ospita la VM. Analizziamo in dettaglio i vari passi: il Core Service Switch riceve in ingresso un pacchetto e ne estrapola IP Destination Address (IP dell'Edge Service Switch sulla Home Network), Protocol (livello rete), Port Number (livello trasporto); l'insieme di questi tre parametri fornisce al Service Switch la possibilità di identificare univocamente una VM all'interno dell'intera architettura. Interrogando la propria Mobile Binding Table, il Service Switch estrapola l'IP privato ed il Care-of Address associati alla VM; dopo aver eseguito la sostituzione del Port Number, procederà reinoltrando il pacchetto verso quest'ultima attraverso un tunnel il cui end-point corrisponde con l'Edge Service Switch della Foreign Network. Il Service Switch sulla Foreign Network eseguirà il detunneling del traffico ricevuto e consegnerà i pacchetti alla VM.

Per quanto riguarda il traffico inverso, così come nel caso precedente, la VM instraderà pacchetti direttamente verso il Correspondent Node, codificando il proprio Home Address nel campo IP Source Address.

3.3 Gestione dei servizi

Abbiamo fino ad ora analizzato le funzionalità offerte dal Service Switch in relazione a situazioni di tipo stazionario; passiamo adesso a considerare le seguenti operazioni: attivazione e migrazione di servizio. Introdurremo entrambe le procedure prescindendo in un primo momento da quali siano le entità che si occupano di eseguire la selezione dei nodi coinvolti nell'operazione, e quindi dai criteri con cui la selezione viene eseguita; ci occuperemo in seguito di descrivere tale procedura.

3.3.1 Attivazione di servizio

Diamo per assunto che sia stato selezionato il VEE che dovrà ospitare il servizio da attivare, che sia sufficiente attivare un'unica istanza dell'applicazione, e che il protocollo di livello trasporto ed il Port Number siano implicitamente determinati in funzione del particolare servizio.

L'Edge Service Switch del VEE gestisce un insieme di IP address sia pubblici che privati; ne assocerà uno alla VM da attivare in funzione delle indicazioni ricevute dall'ASP (Application Service Provider).

Per poter richiedere l'hosting di un servizio S, l'ASP deve preparare:

a) l'immagine del servizio, inclusi gli eseguibili ed i file dati opportunamente organizzati all'interno di un file system;

b) le risorse richieste per il servizio S.

Portiamoci adesso sul nodo fisico su cui la VM verrà attivata. La procedura di attivazione di servizio richiede una continua cooperazione tra l'Hypervisor del nodo fisico ed il Service Switch che gestisce il VEE di cui fa parte. L'Hypervisor eseguirà la prenotazione delle risorse, la creazione ed il bootstrap della Virtual Machine, quindi l'attivazione dell'applicazione. Ancora, eseguirà la configurazione del virtual NIC con l'indirizzo fornito dal Service Switch.

La procedura si completa con un aggiornamento della Mobile Binding Table dell'Edge Service Switch che gestisce il VEE, e di quelle dei Core Service Switch all'interno della rete.

3.3.2 Migrazione di servizio

Considerando quanto detto fino ad ora, risulta evidente che la migrazione di un servizio corrisponde con la migrazione della VM che lo ospita. Tale operazione comporta l'ibernazione di tutti i processi attivi di cui verrà mantenuto lo stato, e la loro successiva riesumazione.

Ci affidiamo al meccanismo di migrazione offerto in Xen [2], uno strumento che supporta il trasferimento di istanze di Sistema Operativo "live" ed "in-service".

Il processo di migrazione di una VM coinvolge due nodi fisici che vengono identificati rispettivamente come sorgente e destinazione; come per la procedura di attivazione, diamo per assunto che le entità coinvolte siano state precedentemente individuate. Migrare una VM consiste nel: trasferire l'immagine del disco, trasferire la porzione di memoria ad essa associata, connettere l'istanza di Sistema Operativo ai dispositivi locali del nodo destinazione. Tra queste, l'operazione da cui dipende l'effettivo service downtime, è la migrazione delle pagine di memoria; poniamo particolare attenzione su tale procedura, e descriviamo in dettaglio il meccanismo implementato in Xen.

Spostare il contenuto della memoria associata ad una VM da un nodo fisico ad un'altro, è una operazione potenzialmente realizzabile utilizzando differenti approcci. Tuttavia, se sulla VM è in esecuzione un live service, è importante che tale trasferimento venga eseguito in modo da minimizzare il downtime ed il total migration time. Il primo parametro corrisponde all'intervallo di tempo durante cui il servizio è non disponibile in quanto non esiste alcuna istanza della VM in esecuzione; tale intervallo è direttamente visibile dall'utente dell'applicazione come interruzione del servizio. Il secondo parametro corrisponde invece con l'intervallo tra l'istante in cui il processo di

migrazione viene inizializzato, e quello in cui la VM originale può essere disabilitata ed il nodo sorgente spento.

L'approccio implementato in Xen viene classificato come pre-copy; il meccanismo di migrazione bilancia i due intervalli di tempo combinando una fase iterativa di pre-copy ad una fase molto breve di stop-and-copy.

La principale caratteristica della migrazione pre-copy risiede nel fato che l'immagine della memoria viene trasferita mentre il Sistema Operativo sul nodo sorgente continua ad essere in esecuzione. Tuttavia, c'è un inevitabile svantaggio dovuto all'overhead causato dal trasferimento di pagine di memoria che verranno successivamente modificate e che andranno nuovamente trasferite. Esistono degli studi in letteratura eseguiti in riferimento a differenti tipi di workload, che, considerata la generica VM, confermano l'esistenza un insieme di pagine alle quali si fa accesso in scrittura con frequenza molto bassa e che quindi risultano essere degli ottimi candidati per il trasferimento in pre-copy, ed una rimanente porzione di pagine aggiornate invece con elevata frequenza che vanno quindi necessariamente trasferite durante la fase di stop-and-copy.

Il processo di migrazione implementato in Xen, che riprendiamo nel nostro modello, è quindi composto dalle seguenti fasi:

pre-migration: durante tale fase viene eseguito il trasferimento dell'immagine del disco associato alla VM da migrare;

iterative pre-copy: durante la prima iterazione tutte le pagine di memoria vengono trasferite da sorgente a destinazione; durante le successive, invece, verranno trasferite solo quelle modificate nella iterazione precedente;

stop-and-copy: l'esecuzione dell'istanza di Sistema Operativo sul nodo sorgente viene sospesa, lo stato della CPU e le rimanenti pagine di memoria vengono quindi trasferite. Alla fine di tale fase c'è una copia

consistente della VM sospesa su entrambi i nodi, e la copia sul nodo sorgente continua ad essere considerata come primaria. Nel nostro modello, in realtà, questa condizione è vera solo nel caso in cui il nodo sorgente sia parte del VEE della Home Network della VM migrata; in questo modo siamo certi del fatto che sulla Home Network è sempre disponibile una copia dell'immagine della VM.

Activation: la VM migrata sul nodo destinazione viene quindi attivata. Viene eseguito del codice di post-migrazione per aggiornare l'immagine del disco e per connettere i driver dei dispositivi all'istanza del Sistema Operativo sulla VM riesumata.

Al fine di ottimizzare l'uso delle risorse di rete, durante la migrazione della memoria viene implementato il meccanismo del Dynamic Rate-Limiting, che consiste in effetti nell'utilizzare la banda in maniera graduale al fine di evitare la saturazione della rete che circonda i nodo coinvolti.

L'obiettivo è quello di adattare in maniera dinamica il limite di banda durante ogni iterazione di pre-copy. Fassati un minimo ed un massimo limite di banda, durante la prima iterazione viene utilizzato il limite inferiore; ad ogni iterazione successiva viene calcolato il dirtying rate dividendo il numero di pagine "sporcate" durante l'iterazione precedente per la durata di tale iterazione. Il limite di banda per ogni iterazione verrà quindi calcolato sommando un valore costante al dirtying rate di quella precedente. La fase di pre-copy termina quando il valore calcolato supera il limite massimo, oppure quando la quantità di memoria che resta da trasferire è minore o uguale a 256KB. Durante la fase di stop-and-copy, invece, il service downtime viene minimizzato sfruttando la banda fino al limite massimo.

Il processo di migrazione viene implementato sfruttando un approccio di tipo managed: sui nodi coinvolti è attiva una macchina virtuale su cui è in esecuzione un software, il Migration Daemon, al quale è affidata la gestione della procedura. Il Migration Daemon

è responsabile della creazione delle VM sul nodo destinazione e del trasferimento del live system state attraverso la rete.

Sfruttare il meccanismo di migrazione disponibile in Xen comporta un requisito fondamentale: i due nodi fisici coinvolti nel processo devono poter comunicare tra loro. Fin tanto che si considera la migrazione di una VM tra nodi appartenenti al medesimo VEE, il requisito risulta certamente verificato; se consideriamo invece la migrazione tra nodi geografici, è indispensabile imporre che i nodi coinvolti siano associati ad indirizzi pubblici. Affinché il modello possa effettivamente supportare la migrazione di servizio, è quindi necessario imporre che i nodi fisici che compongono i vari VEE siano tutti associati ad indirizzi pubblici.

La procedura di migrazione di servizio viene completata con l'operazione di registrazione precedentemente introdotta, e con un aggiornamento delle Mobile Binding Table dei Core Service Switch della rete.

Cerchiamo di giustificare il motivo per cui scegliamo di sfruttare meccanismi per la migrazione di macchine virtuali per implementare la mobilità dei servizi. Esistono delle tecnologie alternative con le quali è possibile eseguire la migrazione di applicazioni, come ad esempio quelli per la migrazione di processo; tale argomento è stato largamente affrontato negli anni 80, ma nonostante ciò l'utilizzo di questo strumento per applicazioni di tipo "real-world" è rimasto molto scarso. Diversi studi sono stati condotti con l'obiettivo di trovare le ragioni alla base dello scarso successo di tali tecnologie; probabilmente il principale difetto è legato alla residual dependency di cui soffre un processo migrato. Per residual dependency si intende in generale la dipendenza da risorse locali sul nodo sorgente, come ad esempio open file descriptor, oppure segmenti di memoria condivisi. La residual dependency comporta diversi problemi, innanzitutto perché impone che la macchina sorgente debba necessariamente rimanere attiva, e poi perché causa un impatto negativo sulle performance del processo migrato.

L'esecuzione di un processo su un sistema Sprite [47], ad esempio, può richiedere l'invocazione di alcune system call che vanno necessariamente inoltrate ed eseguite sul nodo sorgente, riducendo pesantemente le prestazioni, o addirittura comportando un fallimento nel caso in cui il nodo sorgente non sia più disponibile. Un simile problema si presenta nei sistemi MOSIX [48]: sul nodo sorgente deve rimanere attivo un processo che supporta l'esecuzione del processo remoto migrato.

Ancora, eseguire la migrazione di un'applicazione utilizzando invece le tecnologie ad Agenti Mobili, comporta molto spesso il compito di riscrivere buona parte del codice, ed in taluni casi è necessario che la macchina su cui verrà attivata l'applicazione a seguito della migrazione, sia completamente compatibile con quella su cui era in esecuzione precedentemente. Entrambe le macchine devono poi necessariamente fornire caratteristiche quali: esecuzione concorrente, riutilizzo di codice, protezione e fault isolation tra applicazioni.

La migrazione di interi sistemi operativi comporta invece dipendenze pressoché inesistenti; la ragione è che la migrazione viene eseguita a livello macchina e lo stato di esecuzione volatile viene incapsulato all'interno dell'immagine della macchina stessa. Tale caratteristica conferisce ai meccanismi di migrazione system-level elevata resilienza e robustezza.

3.3.3 Selezione dei nodi coinvolti

Occupiamoci adesso della questione legata alla selezione dei nodi coinvolti nelle procedure di attivazione e migrazione di servizio.

Esistono in letteratura diversi lavori introdotti con l'obiettivo di proporre soluzioni al problema del posizionamento ottimale del punto di fornitura di un servizio. Nell'ambito di tale contesto, abbiamo approfondito alcuni modelli per la localizzazione ottimale di servizi in reti di larga scala [19, 20, 21]. Tali modelli propongono l'impiego di politiche per la mobilità dei servizi basate su informazioni globali anzi che locali. Utilizzando un

approccio di tipo step-by-step, tali meccanismi muovono i servizi verso la posizione ottimale sfruttando la migliore traiettoria di migrazione. Considerando il profilo della domanda, l'obiettivo è quello di avvicinare il punto di fornitura del servizio all'area da cui proviene maggiore richiesta per il servizio stesso, in modo da minimizzare l'uso delle risorse di comunicazione e di migliorare la QoS (*Quality of Service*) per i servizi offerti. I meccanismi descritti sono di tipo adattativo, nel senso che supportano la riconfigurazione dello schema di localizzazione dei servizi in funzione di variazioni alla topologia e all'intensità della domanda dei servizi.

Il meccanismo per la localizzazione dei nodi coinvolti nelle procedure di attivazione e migrazione di servizio attualmente implementato nel nostro modello, non è ispirato a politiche di questo tipo, è invece finalizzato al bilanciamento del carico computazionale all'interno dell'architettura. La nostra intenzione è quella di lavorare nel prossimo futuro alla fusione del criterio da noi utilizzato con quello del posizionamento ottimale descritto in precedenza, e realizzare quindi un meccanismo di localizzazione che venga fuori dalla convergenza di entrambi gli approcci.

Il nostro modello non presenta alcuna entità centralizzata, per cui l'intelligenza necessaria ad eseguire la selezioni di nodi da coinvolgere in processi di attivazione o migrazione di servizio, è distribuita sugli Edge Service Switch della rete. Vediamo in che modo si estendono le funzionalità dell'Edge Service Switch rispetto a quanto detto in precedenza; assumiamo innanzitutto che sia in grado di monitorare le risorse fisiche disponibili sul VEE della Home Network di cui fa parte, interagendo con gli Hypervisor dei nodi fisici che lo compongono.

Le richieste per attivazione e migrazione di servizio vengono entrambe accompagnate dai requisiti per l'attivazione, e vengono processate dagli Edge Service Switch alla stessa maniera. Ricevuta una richiesta, il nodo calcola un indice di suitability in funzione delle risorse disponibili sul VEE; fatto ciò, inoltrerà tale informazione a tutti gli altri Edge Service Switch della rete. Ognuno di questi eseguirà un confronto tra l'indice calcolato e quello ricevuto dagli altri. Supponendo che i valori siano tutti differenti tra loro, risulterà implicitamente selezionato l'Edge Service Switch che avrà

fornito l'indice più elevato; esso stesso attiverà la procedura relativa alla richiesta processata. Identifichiamo tale procedura come "selection".

Considerato che la sorgente per le richieste di attivazione di servizio è sempre un ASP, vediamo adesso chi e per quale motivo inoltra invece una richiesta di migrazione.

Individuiamo due differenti tipologie di migrazione ed eseguiamo la seguente classificazione: migrazione locale tra nodi appartenenti al medesimo VEE, migrazione geografica. La prima è finalizzata semplicemente ad un bilanciamento del carico computazionale tra i nodi che compongono il VEE, ed è l'Edge Service Switch appartenente alla Home Network ad occuparsi della selezione dei nodi coinvolti e ad attivare il processo. Nel secondo caso, invece, sono svariate le condizioni che potrebbero motivare una richiesta di migrazione di servizio; noi consideriamo le seguenti: le risorse disponibili sul VEE scendono al di sotto di un valore di soglia prestabilito; i nodi del VEE (solo alcuni oppure tutti) andranno in manutenzione.

La seconda condizione può comportare in generale la migrazione di un intero insieme di VM, che risulteranno però banalmente individuate. Per risolvere la prima, invece, bisogna porsi i seguenti interrogativi: quante VM vanno migrate? Come scegliere le VM da migrare? Il criterio che adottiamo nel nostro modello è il seguente: verrà migrata un'unica VM, in particolare quella che comporta un utilizzo maggiore delle risorse disponibili sul VEE. Individuarla è semplice, è sufficiente che l'Edge Service Switch sulla rete in questione raccolga informazioni dagli Hypervisor dei nodi fisici che compongono il VEE, i quali a loro volta eseguono il monitoraggio delle risorse fisiche locali e gestiscono lo schema di allocazione di tali risorse alle VM attive.

3.4 Classificazione delle funzionalità dei nodi Service Switch

Completata la descrizione delle principali funzioni svolte dal Service Switch, eseguiamo adesso una classificazione di tali funzioni ed individuiamo le entità che compongono il nodo alle quali verranno assegnate.

È necessario considerare Edge Service Switch e Core Service Switch in maniera separata. Partiamo dal primo e facciamo una ulteriore considerazione: l'Edge Service Switch mette a disposizione differenti funzionalità a seconda di come viene identificata la rete di cui fa parte; la situazione cambia infatti in maniera significativa se tale rete viene identificata come Home Network oppure come Foreign Network.

Nei riguardi della generica Home Network, l'Edge Service Switch che ne fa parte svolge la funzione di Home Agent, quindi, processa il traffico destinato alle VM che compongono il VEE sotto la sua responsabilità, e lo reinoltra opportunamente nel caso in cui siano state migrate. Inoltre, implementa un meccanismo di Natting in funzione del numero di porto, in modo che l'architettura possa supportare la coesistenza di VM a cui sono assegnati indirizzi pubblici ed indirizzi privati. Le due macrofunzioni appena elencate vengono implementate da un modulo software che identifichiamo come "Owner Agent".

Consideriamo adesso il ruolo svolto dall'Edge Service Switch quando, in relazione ad una generica VM ospite, la rete di cui fa parte viene identificata come Foreign Network: il nodo svolge la funzione di Foreign Agent ed implementa un meccanismo di IP Aliasing sulle interfacce con le quali si collega agli host della rete locale, per fare in modo che le VM ospitate possano ignorare di essere all'esterno della propria Home Network. Le due macrofunzioni elencate vengono implementate da un modulo software che identifichiamo come "Hosting Agent".

Ancora, all'Edge Service Switch sono affidati alcuni compiti legati alla gestione ed al monitoraggio delle risorse locali di ogni rete, del ciclo di vita dei servizi, di particolari condizioni di funzionamento. Queste funzionalità vengono implementate da un modulo software che identifichiamo come "Environment Manager".

Passiamo adesso al Core Service Switch. Le funzionalità svolte da tale nodo si sovrappongono in parte a quelle dell'Edge Service Switch, abbiamo detto infatti che, come quest'ultimo, è in grado di aprire un tunnel e reindirizzare pacchetti destinati a VM migrate al fine di ottimizzarne l'instradamento. Anche il Core Service Switch

svolge quindi la funzione di Home Agent, ma non in maniera limitata ad una specifica rete, bensì in relazione a tutte le VM attive all'interno dell'architettura. Ancora, come l'Edge Service Switch, implementa il Natting per consentire la coesistenza di IP pubblici e privati. In definitiva, anche sul Core Service Switch è attivo un Owner Agent.

Al Core Service Switch è assegnata una ulteriore fondamentale funzione: utilizzando un particolare meccanismo che verrà descritto in dettaglio nel Capitolo successivo, il nodo è in grado di rilevare l'eventuale crash di un Edge Service Switch all'interno della rete, e di avviare la procedura di riattivazione per ogni servizio in esecuzione sul VEE gestito da tale nodo. Questa funzionalità è implementata da un modulo software che identifichiamo come "Detection Agent".

3.5 Una particolare implementazione: il Service Switch BOX

Abbiamo fino ad ora descritto il modello facendo riferimento all'implementazione basata sull'utilizzo dei meccanismi di virtualizzazione, e considerando uno scenario del tutto generale sia in relazione alla tipologia di servizio ospitata dalla piattaforma, sia in relazione alla tipologia di cliente che possa avere necessità di sfruttarne le potenzialità.

Consideriamo adesso una caratterizzazione del modello precedentemente descritto, e adottiamo alcune particolari scelte implementative.

In riferimento alla generica Home Network, supponiamo che il VEE sia composto da un unico nodo fisico anziché un intero cluster, e che le funzionalità assegnate all'Edge Service Switch responsabile del VEE, siano implementate all'interno del medesimo nodo fisico. Fatte tali ipotesi, risulta in definitiva che l'intera Home Network è in realtà localizzata all'interno di un unico nodo, o meglio, all'interno di un'unica macchina Xen che identifichiamo come "Service Switch Box".

Nel paragrafo precedente abbiamo classificato e suddiviso le funzionalità offerte dai Service Switch, ed abbiamo identificato le entità che le implementano. Consideriamo in particolare quelle dell'Edge Service Switch: Owner Agent, Hosting Agent ed

Environment Manager. Assumiamo che tali moduli siano in esecuzione su un'unica VM dedicata; tale VM può fare accesso diretto ad una interfaccia di rete fisica per uso esclusivo che verrà configurata con il public IP Address associato al Service Switch. La VM in questione avrà inoltre l'incarico di implementare il meccanismo dell'host forwarding via bridging, e di gestire e monitorare quindi gli incoming e gli outcoming packet in transito attraverso l'interfaccia. Facendo tali ipotesi, risultano anche in questo caso verificati i due vincoli imposti da Mobile IP: il Service Switch (che implementa la funzione di Home Agent) è in grado di intercettare tutto il traffico destinato alle VM, ed inoltre può comunicare con le VM bypassando il routing IP.

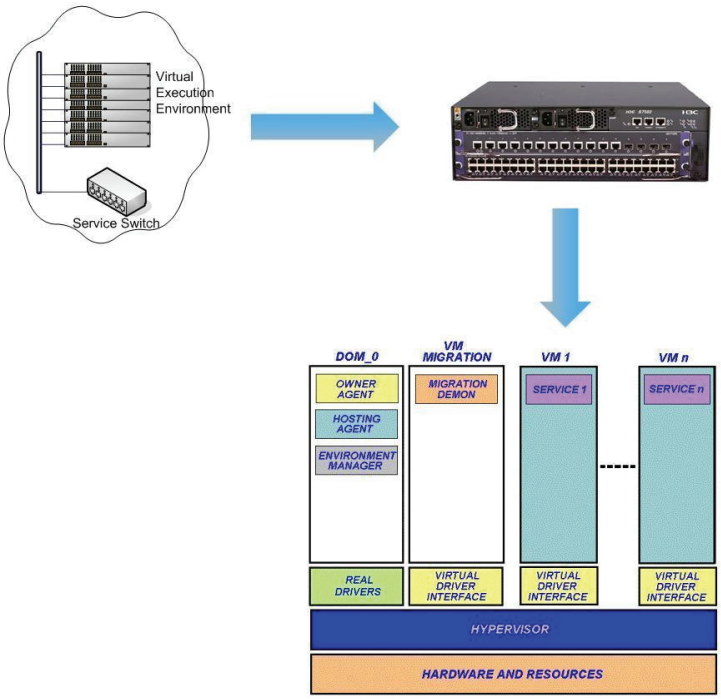

Figura 23 - Il Service Switch Box

Non è necessario imporre che il Service Switch Box sia il Default Gateway della rete di cui fa parte. Considerato il modo in cui è strutturato il nodo, tra l'altro, tale imposizione non avrebbe alcun significato: il Service Switch non è più un router, bensì un end-

system. Risulta evidente, infine, che in un simile scenario non ha alcun senso considerare la migrazione di una VM all'interno del medesimo VEE.

Utilizzeremo questa particolare implementazione del Service Switch per eseguire le prove sperimentali descritte al Capitolo V.

Capitolo 4
Resilienza, scalabilità e sicurezza

4.1 Canali sicuri e affidabili per le comunicazioni tra Service Switch

Ci occuperemo in questo paragrafo di aspetti legati alla sicurezza in relazione alle comunicazioni tra nodi Service Switch, facendo riferimento a strumenti utilizzati e meccanismi opportunamente implementati per generare canali di comunicazione affidabili. Riporteremo quindi una breve panoramica su MPLS (MultiProtocol Label Switching) e su una particolare implementazione Open Source di tale protocollo; introdurremo poi delle politiche di forwarding innovative: lo Splitting ed il Multicast.

4.1.1 Introduzione a MPLS

Il MultiProtocol Label Switching (MPLS) [RFC3031] nasce con l'obiettivo di integrare il mondo a commutazione di circuito di ATM e quello a commutazione di pacchetto di IP, e rappresenta in qualche modo la convergenza di due modelli fondamentali delle reti di calcolatori, *datagram* e *virtual circuit*. MPLS è una tecnica di instradamento che introduce i circuiti virtuali nell'ambiente connection-less di IP, e fornisce la possibilità di utilizzare dei percorsi espliciti precalcolati. Consente inoltre di superare le barriere tra diverse tecnologie, e di ridurre la complessità ed i costi delle comunicazioni.

Analizziamo brevemente i vantaggi offerti da tale strumento, con particolare riferimento al meccanismo di instradamento implementato, il *label swapping*.

Introduciamo innanzitutto un po' di terminologia: in MPLS un circuito virtuale viene identificato come LSP (Label Switched Path), i router del core come LSR (Label Switching Router), e quelli della frontiera come LER (Label Edge Router). Il meccanismo di forwarding implementato in MPLS si basa sull'uso di una etichetta, la quale non è altro che un identificatore locale corto di lunghezza fissa e non strutturato; l'ambito di validità di una etichetta può essere un link, un Autonomous System, o un intero dominio MPLS. A seconda della particolare tecnologia utilizzata, l'etichetta MPLS può essere o meno inserita in campi dell'header già esistenti; in ATM, ad esempio, viene codificata nel campo VCI/VPI.

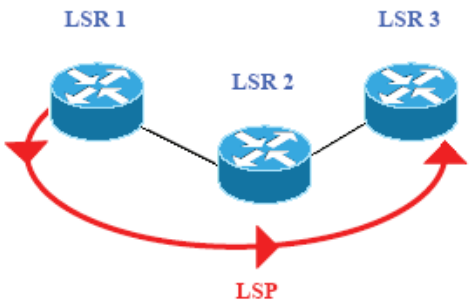

Figura 24 – MPLS: LSR ed LSP

In alternativa, è possibile sfruttare un meccanismo di incapsulamento opportunamente definito e standardizzato, il quale prevede l'uso di un header aggiuntivo definito shim label header, all'interno del quale viene trasportata l'etichetta MPLS assieme ad altre informazioni necessarie ad eseguire il label swapping; lo shim label header si colloca tra l'header di livello data-link e quello di livello rete.

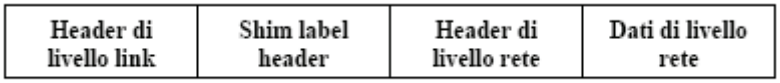

Figura 25 – MPLS: lo Shim Label Header

Analizziamo in dettaglio i vari campi dell'header MPLS:

Label : è l'etichetta vera e propria; ha dimensione pari a 20 bit ed è utilizzata dall'LSR che riceve il pacchetto come indice per la ricerca nelle tabelle.

Exp : è costituito da 3 bit ed è un campo riservato per usi sperimentali.

Bottom (S) : è un campo di un bit; se posto a zero indica che l'etichetta processata è l'ultima della pila.

Time To Live (TTL) : è un campo di 8 bit contenente il TTL.

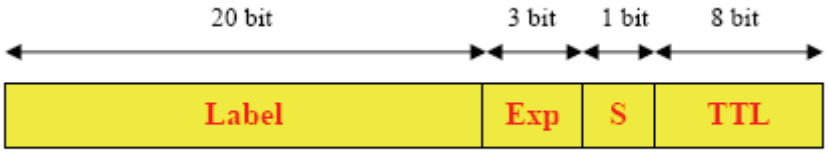

Figura 26 - Formato di una etichetta MPLS

Ogni etichetta è associata univocamente ad una FEC (Forwarding Equivalence Class), la quale identifica un gruppo di datagrammi da inoltrare lungo il medesimo percorso e che riceveranno il medesimo trattamento. Una FEC può essere definita a differenti livelli di granularità, ed è in sostanza un insieme di regole di classificazione necessarie a determinare l'appartenenza del pacchetto. L'associazione tra etichetta e FEC è biunivoca.

Il tradizionale meccanismo di routing IP–based, impone che tutti i pacchetti aventi uguale destinazione vengano manipolati allo stesso modo. In alcuni casi potrebbe invece risultare comodo poter garantire un trattamento differenziato a diversi flussi di traffico seppur diretti alla medesima destinazione. Il label swapping offre questa opportunità: è sufficiente assegnare a diversi flussi di traffico delle etichette differenti, per ottenere come risultato l'instradamento su differenti circuiti virtuali. MPLS consente in sostanza di implementare il routing esplicito.

LER ed LSR eseguono il forwarding dei pacchetti in funzione del valore dell'etichetta riportata nel pacchetto processato ed in funzione delle informazioni contenute nella Forwarding Information Base (FIB), costituita dalle seguenti tre tabelle: ILM (Incoming Label Map) o anche tabella di ingresso, NHLFE (Next-Hop Label Forwarding Entry) o anche tabella di uscita, FTN (FEC To NHLFE).

La ILM contiene l'associazione tra etichetta d'ingresso, corrispondente etichetta di uscita, e interfaccia di uscita; ogni LSR ne gestisce una. La NHLFE contiene le operazioni da effettuare sui pacchetti (pop, push, dlv, set) identificati da una specifica etichetta, oltre che il next-hop e l'etichetta di uscita. La FTN contiene una associazione tra l'identificativo della FEC e la chiave di accesso ad una particolare entry della NHLFE.

Analizziamo le varie operazioni da eseguire in corrispondenza di ogni nodo della nuvola MPLS. Il Label Edge Router di ingresso interroga la FTN in funzione delle informazioni contenute nell'header IP del pacchetto processato, con l'obiettivo di determinare la classe di equivalenza (FEC) a cui appartiene ed il corrispondente percorso (LSP) su cui inoltrarlo; incapsula il pacchetto IP all'interno di un apposito header MPLS e lo invia verso la destinazione. Utilizzando l'etichetta MPLS come indice della ILM, il Label Switching Router determina link ed etichetta di uscita, effettua lo swapping delle etichette, instrada il pacchetto. Il Label Edge Router di uscita interroga la tabella di ingresso in funzione del valore dell'etichetta MPLS; nel caso in cui il next-hop verso cui inoltrare il pacchetto sia al di fuori della nuvola MPLS,

provvederà ad eliminare lo shim label header e a ripristinare il tradizionale forwarding di livello 3.

In MPLS il compito di creare LSP è affidato ai meccanismi di segnalazione; l'operazione consiste nel creare l'associazione tra una classe di equivalenza ed un insieme di etichette, e nel distribuire le etichette sui nodi della rete MPLS utilizzando un apposito protocollo. Il compito assegnato ad un protocollo per la distribuzione delle etichette è quello di determinare la sequenza di LSR sul percorso tra sorgente e destinazione. I due protocolli maggiormente utilizzati sono i seguenti: LDP (Label Distribution Protocol) e RSVP (Resource reSerVation Protocol). Per quanto riguarda in particolare il primo, offre due diverse modalità per la creazione di percorsi virtuali, il downstream on demand ed il downstream unsolicited.

Introduciamo brevemente il progetto MPLS-Linux (http://mpls-linux.sourceforge.net/), una particolare implementazione Open Source del protocollo MPLS.

MPLS-Linux fornisce due componenti: una patch per il kernel di Linux che introduce uno stack MPLS completo e funzionale, ed un programma lato utente che consente di configurare manualmente dei percorsi virtuali.

MPLS-Linux introduce le seguenti istruzioni per il processing dei pacchetti:

- push: aggiunge un header MPLS ad un pacchetto con una specifica etichetta;
- set: inoltra un pacchetto verso il livello data-link causandone in questo modo la trasmissione verso il next-hop;
- pop: rimuove l'header MPLS dal pacchetto;
- fwd: sottomette il pacchetto alle istruzioni contenute in una specifica entry della NHLFE;
- dlv: invia il pacchetto a livello IP.

Rispettando le specifiche del protocollo MPLS, la patch per il kernel introduce le seguenti tabelle:

- FTN: è costituita da un certo numero di entry, ognuna delle quali fa riferimento ad una specifica FEC e contiene la chiave per l'accesso ad una entry della NHLFE.

- ILM: la tabella viene interrogata ogni qual volta si esegue il processing di un pacchetto in ingresso che contenga un header MPLS. Le entry della tabella sono identificate da una chiave e puntano ad una entry della NHLFE.

- NHLFE: ogni entry della tabella descrive il modo con cui processare il pacchetto; può essere puntata dalla FTN, dalla ILM, oppure da una ulteriore entry della NHLFE stessa.

Illustriamo il modo in cui tali tabelle vengono utilizzate per processare i pacchetti facendo riferimento ai differenti tipi di nodo che costituiscono la generica nuvola MPLS:

- LER di ingresso: riceve pacchetti senza etichetta MPLS, aggiunge quindi un header ed inoltra il pacchetto sull'LSP. Il primo passo è dunque quello di classificare il pacchetto come appartenente ad una specifica classe di equivalenza, e di ricercare una entry nella FTN associata a tale FEC. Si ottiene come risultato una chiave di accesso alla tabella di uscita, la NHLFE, alla quale si fa accesso, e, sfruttando le informazioni in essa contenute, si inoltra il pacchetto verso il next-hop.

- LSR: riceve pacchetti con etichetta, esegue uno scambio di etichette ed inoltra il pacchetto verso un altro LSR. La commutazione di etichetta richiede un accesso alla ILM ed un successivo accesso alla NHLFE.

- LER di uscita: anche quest'ultimo riceve pacchetti con etichetta, quindi rimuove l'header MPLS e li passa al livello IP. La sequenza di istruzioni (pop - dlv) necessaria ad eseguire tale operazione, è contenuta nella tabella di ingresso.

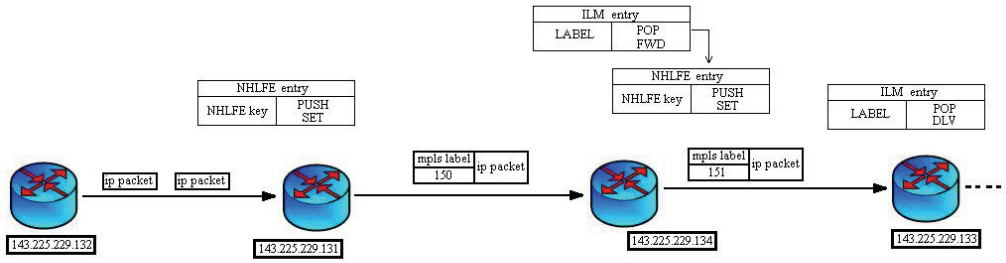

Figura 27 – MPLS: esempio di configurazione di in LSP

4.1.2 Politiche di instradamento innovative

Il nostro contributo originale nell'ambito delle reti MPLS, è stato quello di proporre delle politiche innovative di forwarding basate sull'utilizzo di percorsi di tipo point-to-multipoint, che verranno poi sfruttate nel modello Service Switching per incrementare sicurezza e affidabilità nelle comunicazioni. Ci occuperemo in questo paragrafo di introdurre i meccanismi di forwarding realizzati; parleremo dei principali requisiti necessari per eseguire il set-up di percorsi virtuali di tipo point-to-multipoint, e proporremo un nuovo set di istruzioni per router MPLS necessarie alla gestione di percorsi di questo tipo. Presenteremo successivamente il meccanismo da implementare sul generico LSR per la gestione di informazioni extra necessarie alla selezione di un LSP in un insieme di n LSP. Introdurremo quindi due esempi di applicazioni che possono richiedere l'utilizzo di percorsi point-to-multipoint: lo Splitting ed il Multicast. Nel Capitolo relativo alle prove sperimentali dimostreremo infine l'efficacia delle soluzioni proposte ed implementate sullo stack MPLS-Linux.

Lo Splitting

Il meccanismo di Splitting consente di inoltrare differenti flussi di traffico su percorsi multipli di egual costo, al fine di ottimizzare l'uso delle risorse. Il meccanismo viene implementato sfruttando la tecnica round robin per la selezione del percorso di uscita in funzione della classe di equivalenza a cui il pacchetto appartiene, e consiste nell'inoltrare su percorsi differenti pacchetti processati in sequenza in corrispondenza di un LER d'ingresso alla nuvola MPLS.

In MPLS, flussi appartenenti alla medesima classe di equivalenza vengono considerati come una entità singola e manipolati allo stesso modo; associare una FEC ad una singola entry NHLFE equivale ad imporre che pacchetti appartenenti alla medesima classe di equivalenza debbano essere inoltrati lungo il medesimo LSP. Risulta quindi evidente che per poter implementare lo Splitting è necessario che l'ingress LER possa associare una singola FEC ad entry multiple della NHLFE. Lo standard IETF prevede che una FEC possa essere associata ad etichette di uscita multiple, ma non specifica come questa associazione possa essere realizzata. Il nostro principale contributo è stato quello di progettare e realizzare un meccanismo che consenta di eseguire un'associazione multipla di questo tipo.

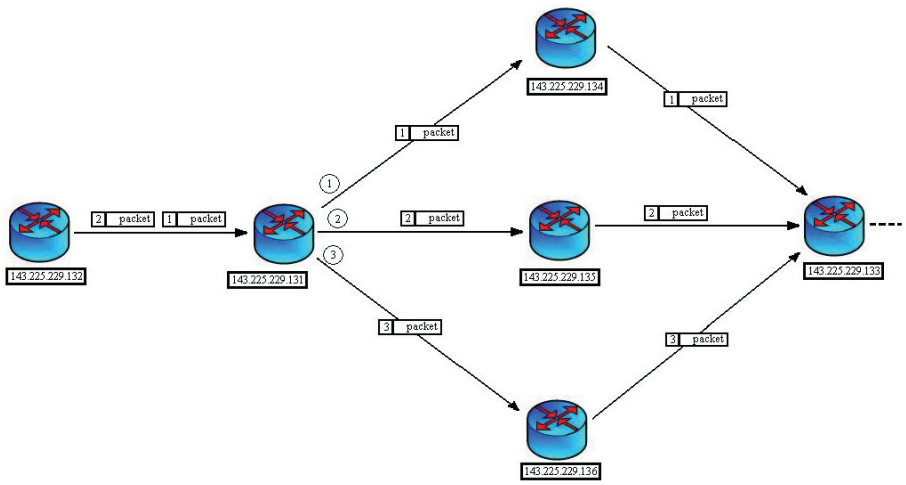

Figura 28 - MPLS Splitting

In una situazione ideale, tutti i percorsi su cui eseguire lo Splitting di un flusso specifico, presentano le stesse proprietà in termini di banda residua, packet loss e lunghezza del percorso. L'utilizzo di percorsi che presentino invece caratteristiche significativamente differenti tra loro, può comportare il fatto che i pacchetti arrivino fuori sequenza, condizione che ovviamente non può verificarsi nel caso in cui vengano invece inoltrati tutti sul medesimo percorso. È necessario quindi introdurre un meccanismo finalizzato ad eseguire la ricostruzione del flusso di traffico originale in corrispondenza del LER di uscita della nuvola MPLS, al fine di emulare il comportamento di una tradizionale rete MPLS dove non sia implementato lo Splitting.

Per consentire ad un egress LER di riordinare i pacchetti in maniera corretta, il nodo d'ingresso che esegue lo Splitting ha il compito di aggiungere informazioni extra all'interno del pacchetto, possibilità offerta dal meccanismo di stacking delle etichette MPLS; l'ingress LER aggiunge quindi una etichetta addizionale nell'header MPLS contenente tali informazioni. Considerato che all'interno del core, gli LSR non fanno altro che estrarre e processare solo la prima etichetta dello stack MPLS, si può essere certi del fatto che non andranno mai a fare riferimento ad informazioni necessarie ad eseguire la ricostruzione dei flussi, che arriveranno quindi inalterate al LER di uscita.

Eseguire lo splitting di traffico secondo l'approccio che proponiamo, comporta il vantaggio legato al fatto che tutte le informazioni necessarie ad implementare la procedura risultano distribuite esclusivamente sulla frontiera della nuvola MPLS, e che l'intero processo sia completamente trasparente al core, dove va semplicemente eseguito il normale label swapping.

Il Multicast

Il progetto MPLS, nella sua versione nativa, supporta il multicasting, tant'è vero che nell'header di livello Data-link sono stati assegnati due differenti codici a pacchetti

MPLS unicast e pacchetti MPLS multicast; tuttavia, il multicast MPLS non è mai stato implementato in sistemi Linux.

È possibile introdurre il multicast in MPLS utilizzando due differenti approcci: considerato che i router MPLS non sono altro che dei router IP con funzionalità aggiuntive, si potrebbe implementare il multicast MPLS basandosi su quello di IP; in alternativa, è possibile implementare un meccanismo di multicast via unicast. Il modello che proponiamo si basa su questo secondo approccio. In corrispondenza dei nodi dell'albero multicast, il nostro meccanismo duplica i pacchetti ed inoltra copie del medesimo pacchetto su differenti link di uscita dopo aver associato ad ognuna una differente etichetta. In accordo col meccanismo di routing utilizzato alla base, sia i pacchetti in ingresso che quelli in uscita sono unicast, così come le etichette ad essi associati. Non è quindi prevista la possibilità di associare una singola etichetta ad un intero gruppo multicast.

Gli LSP utilizzati vengono costruiti in maniera statica, non è infatti disponibile in MPLS alcun protocollo di segnalazione per la costruzione di un albero multicast.

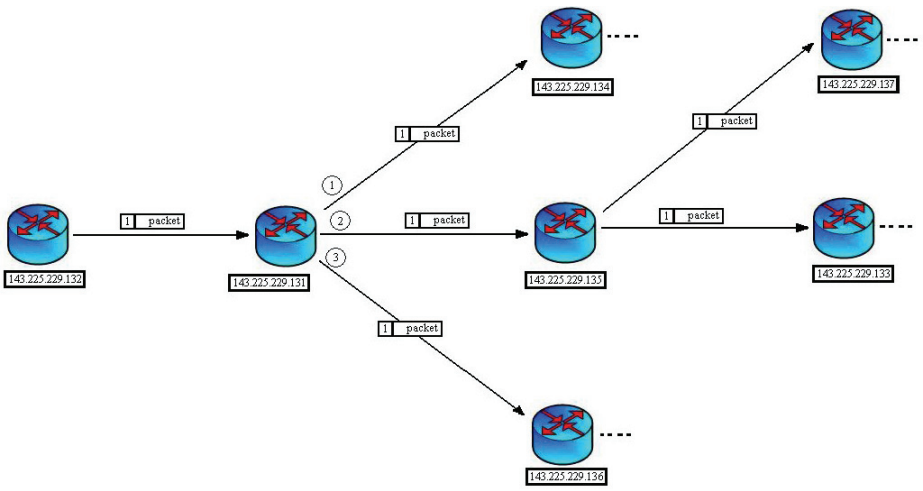

Figura 29 - MPLS Multicast

Dettagli relativi all'implementazione

Entrambe le politiche di forwarding introdotte si basano sulla possibilità di associare ad una singola classe di equivalenza dei percorsi multipli, quindi differenti LSP alla medesima etichetta. Una volta individuato l'intero insieme di percorsi associato alla FEC, bisogna in qualche modo selezionare l'LSP su cui inoltrare il pacchetto processato. Bisogna quindi definire un algoritmo di selezione che a seconda del criterio che si intende utilizzare, restituisca un unico LSP, oppure l'intero insieme di LSP. In modalità "Splitting", l'algoritmo restituisce un LSP alla volta, selezionato utilizzando il criterio round robin. In modalità "Multicast" restituisce invece tutti gli LSP.

In definitiva, il nostro compito è quello di implementare meccanismi mediante i quali sia possibile:

1. inoltrare pacchetti appartenenti alla medesima FEC su differenti percorsi;
2. scegliere a run-time su quanti e quali LSP inoltrare il pacchetto.

Per raggiungere il primo obiettivo abbiamo introdotto uno step aggiuntivo nel processing di un pacchetto in corrispondenza di un nodo di ingresso alla nuvola MPLS; la normale sequenza prevede un accesso alla FTN ed un successivo accesso alla NHLFE per il processing vero e proprio, quindi l'invio del pacchetto attraverso l'interfaccia di rete. Introduciamo un ulteriore accesso alla tabella NHLFE: nella entry alla quale si esegue il primo accesso, ci sarà una nuova istruzione, la "mforward", la quale implementare una associazione statica 1 a n con ulteriori entry NHLFE; il secondo accesso verrà fatto su una di queste, la quale conterrà le istruzioni con cui il pacchetto verrà processato.

Per raggiungere il secondo obiettivo, abbiamo aggiunto una nuova struttura che mantiene tutti i dati necessari all'esecuzione dell'algoritmo di selezione, ed un riferimento all'algoritmo stesso. L'algoritmo esegue il processing della lista di entry NHLFE, e a seguito di determinate operazioni restituisce la chiave d'accesso ad una

singola entry oppure ad un array di entry, a seconda della politica di forwarding implementata.

In riferimento allo Splitting, per consentire il corretto riordinamento dei pacchetti in corrispondenza del LER di uscita, quindi la ricostruzione del traffico originale, bisogna poter identificare univocamente il traffico appartenente ad una specifica classe di equivalenza, e l'ordine con cui i pacchetti sono stati inoltrati dall'ingress LER. Utilizziamo rispettivamente i seguenti due parametri: splitting identifier, sequence number.

Vediamo di seguito la struttura di un'etichetta per lo splitting:

Label: sequence number;

Exp: contiene una stringa di bit (111) necessaria a specificare che l'etichetta contiene informazioni relative al meccanismi di Splitting;

Bottom (S): indica che l'etichetta non costituisce la testa dello stack;

Time To Live (TTL): splitting identifier.

Il meccanismo per la ricostruzione dei flussi non è obligatorio, può essere selezionato come funzionalità aggiuntiva. Mettiamo a disposizione tale alternativa perché il reordering dei pacchetti comporta una penalità sia in termini di ritardo nel processing che in termini di buffer utilizzato; d'altro canto, non rilascia a TCP la responsabilità della ricostruzione del flusso di dati originale al livello TCP. Le politiche di forwarding che implementano lo Splitting con e senza ricostruzione, sono identificate rispettivamente come LRR (Label Round Robin) e RR (Round Robin).

Vediamo adesso invece alcuni dettagli relativi all'implementazione del Multicast. In MPLS-Linux esiste un apposito header per il multicast, ed esiste inoltre l'interfaccia della funzione che ne implementa le funzionalità (mpls_skb_recv_mc), ma tale funzione non è mai stata realizzata ed i pacchetti che la invocano vengono di conseguenza scartati.

Come accennato in precedenza, il nostro è un meccanismo di multicast via unicast, per cui i router MPLS con la funzionalità multicast abilitata, ricevono pacchetti unicast, li duplicano, ed inoltrano copie dei pacchetti su differenti LSP. Il codice settato nell'header Ethernet non è quello relativo alle trasmissioni multicast, bensì quello relativo alle trasmissioni unicast, e l'etichetta inserita nello stack MPLS è di tipo unicast. Tale approccio non impone limiti di alcun tipo: tutti i router MPLS possono essere integrati con la funzionalità per il multicast, in quanto non dovranno far altro che implementare il meccanismo di forwarding MPLS unicast.

4.1.3 Il Multicast e lo Splitting per il Service Switching

Ci occuperemo in questo paragrafo di analizzare il modo in cui MPLS ed i meccanismi di forwarding proposti, vengono utilizzati nel modello Service Switching, quindi le motivazioni che ci hanno suggerito di impiegare tali strumenti in un simile contesto.

Scegliamo di utilizzare MPLS in virtù delle potenzialità offerte da tale tecnologia, con particolare riferimento alla possibilità di eseguire routing esplicito per particolari classi di traffico grazie a meccanismi di Traffic Engineering, e a quella di gestire flussi di traffico in maniera differenziata e garantire quindi qualità del servizio. L'introduzione delle nostre politiche di forwarding ci consente inoltre di ampliare tali potenzialità, quindi di: incrementare affidabilità e sicurezza in relazione all'uso dei canali di comunicazione, implementare load balancing per ottimizzare l'uso delle risorse di rete, configurare percorsi di back-up.

In riferimento al Service Switching, utilizzeremo il Multicast MPLS per supportare le comunicazioni punto-multipunto tra i nodi Service Switch che compongono la nostra architettura; riportiamo di seguito una lista di circostanze in relazione alle quali risulta indispensabile poter fare affidamento su tale strumento:

- Richiesta di attivazione di servizio: la richiesta viene inoltrata dal generico Application Service Provider e dev'essere consegnata all'intera classe di nodi Edge Service Switch;
- Richiesta di migrazione: la richiesta viene inoltrata da un singolo Edge Servcie Switch e dev'essere consegnata a tutti gli atri;
- Diffusione dell'indice di suitability: come al punto precedente;
- Aggiornamento delle Mobile Binding Table: a seguito delle procedure di attivazione e migrazione di servizio, l'Edge Service Switch che gestisce il VEE su cui è stata eseguita l'attivazione o riesumazione di una VM, invierà la nuova associazione Home Address – Care-of Address verso l'intera classe di Core Service Switch della rete.

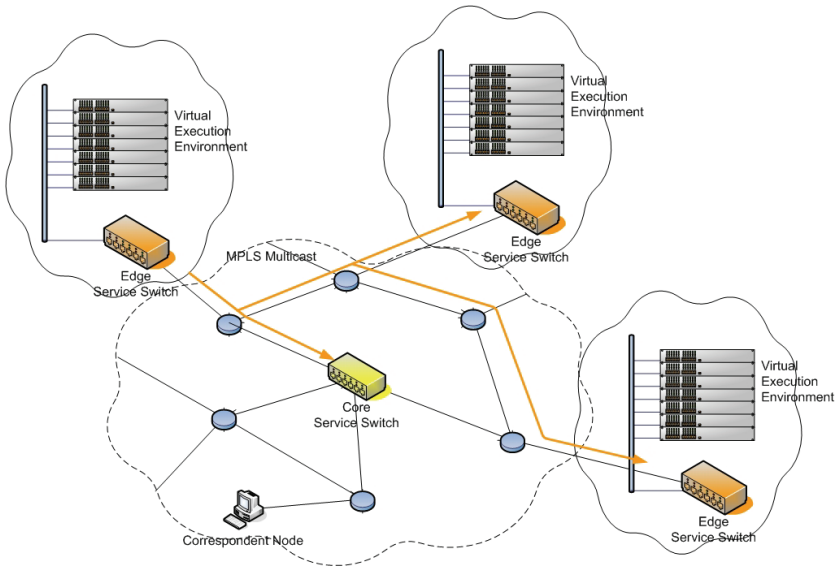

Figura 30 - Il Multicast MPLS per le comunicazioni punto-multipunto tra nodi Service Switch

Utilizzeremo invece lo Splitting per implementare il meccanismo di tunneling necessario ad eseguire il reinstradamento del traffico destinato a VM migrate;

riportiamo in Figura 31 il tipico scenario in relazione al quale il meccanismo di tunneling viene impiegato; l'esigenza è quella di reinstradare in maniera trasparente nei riguardi dell'utente, i pacchetti destinati ad una VM migrata ed ospitata da una Foreign Network. L'Edge Service Switch sulla Home Network inoltra tali pacchetti attraverso un tunnel il cui end point corrisponde con l'Edge Service Switch sulla Foreign Network, il quale esegue il detunneling dei pacchetti e li consegna alla VM.

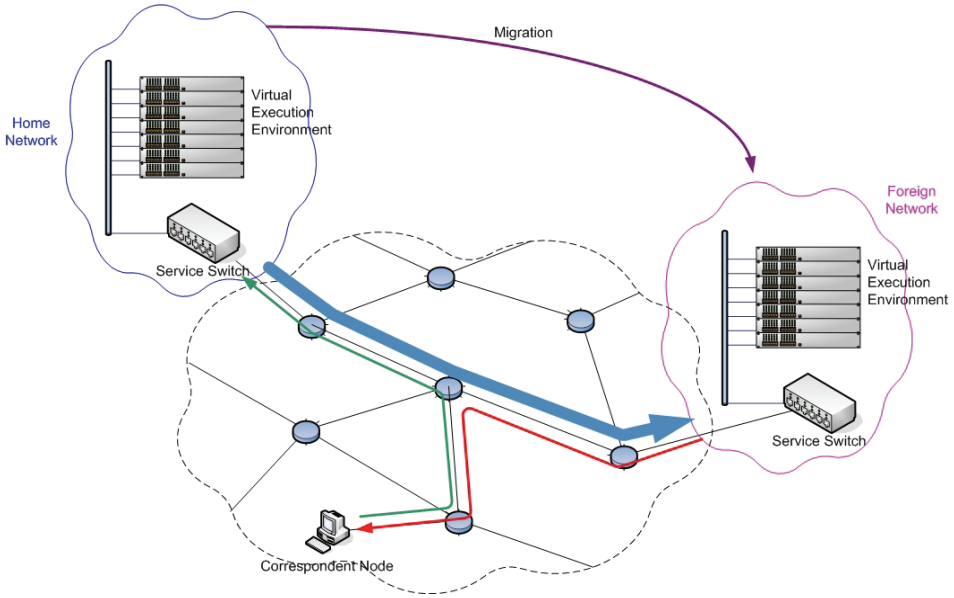

Figura 31 – Meccanismo di tunneling tradizionale

Implementiamo il tunneling utilizzando MPLS ed il meccanismo di Splitting; il tunnel non verrà costruito sfruttando un percorso singolo tra sorgente e destinazione, bensì due o più percorsi. In corrispondenza dell'Edge Service Switch sulla Home Network, il traffico verrà suddiviso su tali percorsi, e poi ricostruito in corrispondenza dell'Edge Service Switch sulla Foreign Network.

Tale meccanismo ci consente di suddividere il traffico sfruttando un approccio di tipo per-packet, e di incrementare quindi notevolmente il livello di sicurezza sui dati trasferiti; considerato ad esempio il caso in cui l'eventuale utente malevolo riuscisse a

sniffare il traffico su uno dei due percorsi, non sarebbe comunque in grado risalire al flusso originale. Lo Splitting consente inoltre di sfruttare percorsi multipli su cui inoltrare solo una porzione dei flussi di traffico e ottimizzare quindi l'uso delle risorse di rete, e di risolvere problemi legati all'eventuale malfunzionamento di un link dirottando il traffico sui percorsi di Splitting, ed eliminando quindi la necessità di configurarne degli altri.

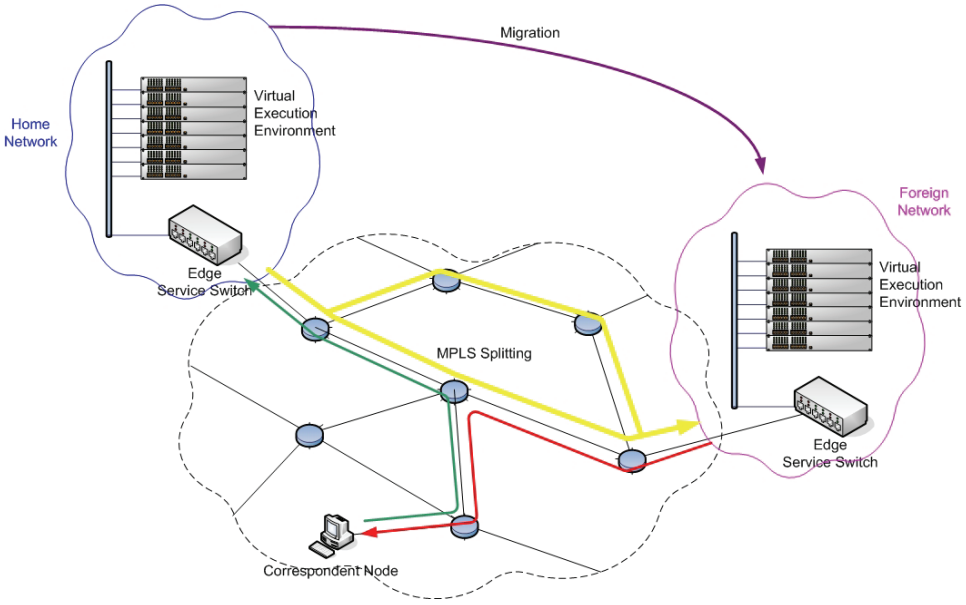

Figura 32 - Tunneling con MPLS Splitting

4.2 Gestione delle condizioni critiche

Abbiamo fino ad ora considerato le funzionalità svolte dai Servce Switch in normali condizioni di funzionamento; prendiamo adesso in esame il caso in cui un Service Switch sia vittima di un fallimento e diventi quindi irraggiungibile o comunque inutilizzabile.

Le conseguenze derivanti da un tale evento sono ovviamente legate alla tipologia di nodo soggetto al fallimento. Nel caso in cui si tratti di un Core Service Switch, a farne

le spese è l'ottimizzazione dell'instradamento di traffico destinato a macchine virtuali migrate. Eseguire una operazione di recupero non è in realtà indispensabile, in quanto, il reindirizzamento verso la Foreign Network che ospita la VM migrata, verrà comunque eseguito da altri Core Service Switch sul percorso tra Correspondent Node e Home Network della VM migrata, oppure, nel peggiore dei casi, dall'Edge Service Switch sulla Home Network.

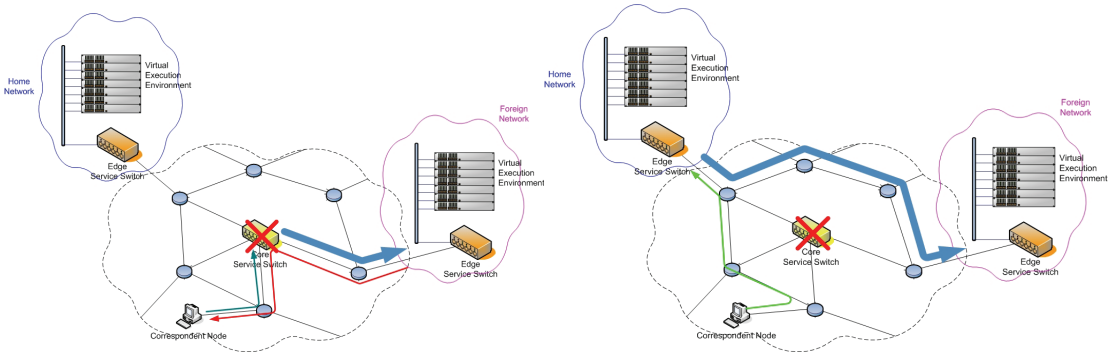

Figura 33 - Crash di un Core Service Switch

Ben più delicato è invece il caso in cui ad essere vittima di un fallimento sia un Edge Service Switch; nel caso in cui dovesse verificarsi una simile condizione, tutti i servizi ospitati dal VEE da esso gestito diventerebbero inutilizzabili.

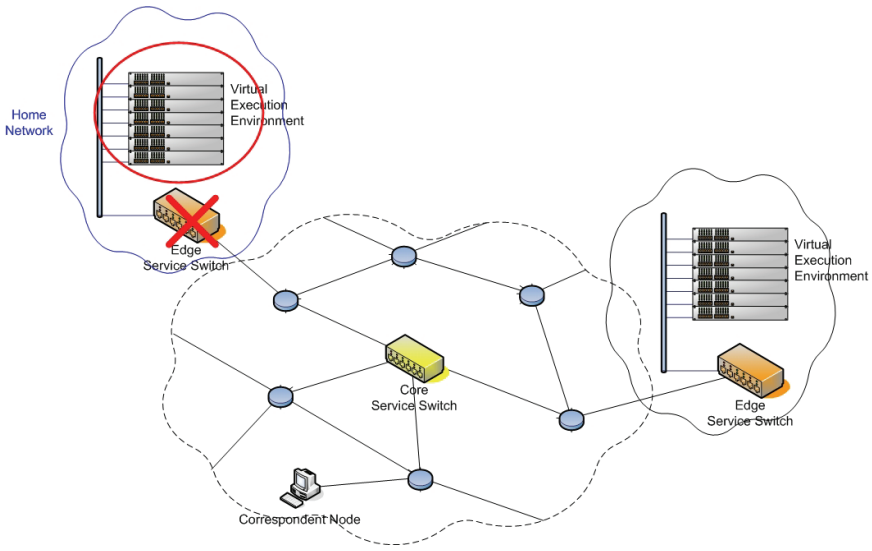

Figura 34 - Crash di un Edge Service Switch

Vediamo in che modo bisogna estendere le funzionalità dei Service Switch per individuare un simile evento e risolverne le relative conseguenze.

Assegniamo ai Core Service Switch il compito di inviare periodicamente un messaggio di tipo “are you up?” verso l’intera rete di Edge Service Switch, i quali reagiranno inviando un segnale che indichi il loro stato di funzionamento; nel caso in cui il generico Core Service Switch non dovesse ricevere alcuna risposta entro lo scadere di un time out prestabilito, il procedimento verrà reiterato per un numero di volte tale da fornire garanzia sul fatto che lo specifico Edge Service Switch sia effettivamente down. Il primo Core Service Switch che rileva tale condizione, si preoccuperà di avviare la procedura di recovery.

La soluzione che proponiamo al problema dell’isolamento di un intero insieme servizi, consiste nel riattivare tali servizi sui rimanenti VEE della piattaforma. Ogni Core Service Switch gestisce all’interno della propria Mobile Binding Table informazioni relative a tutti i sevizi attivi, ed è in grado quindi di localizzare ogni singolo servizio sullo specifico VEE. Una volta individuato il crash di un Edge Service Switch, il Core

Service Switch sarà in grado di identificare tutti i servizi ospitati dal VEE rimasto isolato, e quindi di avviare per ognuno di questi la procedura di riattivazione.

Consideriamo il caso generico, e supponiamo che solo parte dei servizi attivi sul VEE ormai irraggiungibile, si trovi effettivamente all'interno della propria Home Network, e che la restante parte sia invece ospite. Abbiamo detto in precedenza che, anche a seguito di migrazione, sul VEE della Home Network è sempre disponibile una copia dell'immagine della VM; i servizi che all'istante del crash dell'Edge Service Switch erano ospiti all'interno del VEE da esso gestito, verranno quindi riattivati sulla rispettiva Home Network.

Per gli altri servizi la procedura da implementare è differente. Per poter riattivare un servizio, o meglio, la VM che lo ospita, è indispensabile poter disporre dell'immagine della VM stessa, ma per quanto detto finora, tale oggetto risulta essere disponibile solo sul cluster a valle dell'Edge Service Switch fuori uso. Per rendere la procedura di riattivazione effettivamente implementabile, è invece necessario disporre di una copia di back-up; utilizziamo a tale scopo dei dispositivi di storage condiviso, sfruttiamo in particolare la tecnologia NAS (Network Attached Storage) [49]. L'immagine di ogni singola VM attiva sull'intera architettura sarà quindi disponibile sul NAS.

Analizziamo in dettaglio il processo di riattivazione. La procedura si compone di una sequenza di passi, il primo dei quali consiste nell'inoltrare la relativa richiesta verso l'intera classe di Edge Service Switch, i quali si occuperanno di processarla allo stesso modo di una richiesta di attivazione. Identificato il VEE su cui il servizio verrà riattivato, l'Edge Service Switch che lo gestisce procederà recuperando l'immagine della VM dal NAS, prenotando le risorse necessarie, e infine riesumando la VM. La rete che la ospita sarà per la VM a tutti gli effetti una Foreign Network, quindi, la procedura di riattivazione si chiuderà con la riconfigurazione delle Mobile Binding Table in modo da riflettere la nuova posizione del servizio.

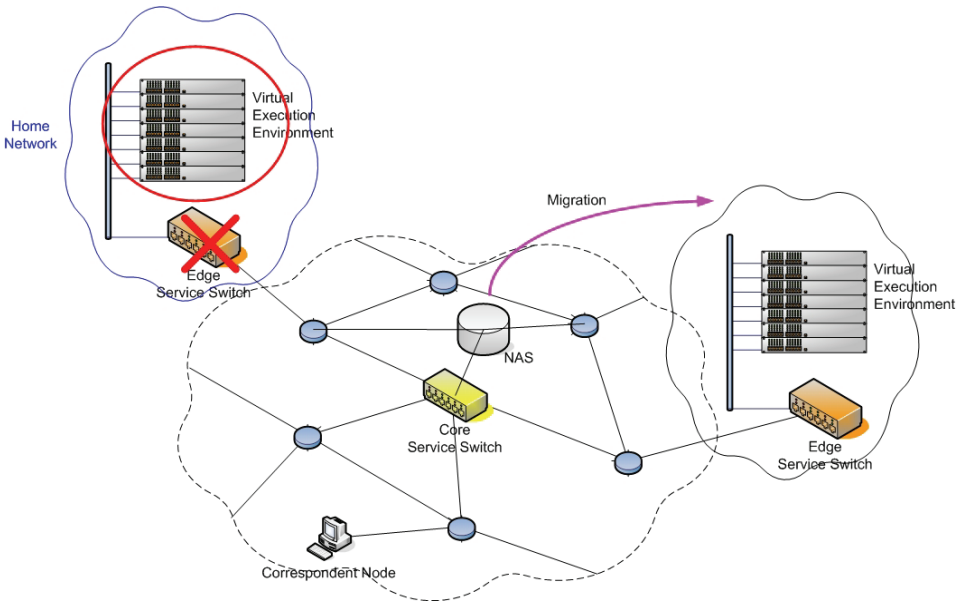

Figura 35 - Introduzione dei NAS

La nostra intenzione è quella di garantire che la procedura di riattivazione dei servizi venga implementata in maniera trasparente rispetto al generico utente, il quale resterà di conseguenza allo scuro del guasto che ha causato l'isolamento del VEE. Raggiungiamo tale obiettivo affidandoci nuovamente al meccanismo di reindirizzamento utilizzato per supportare la migrazione di servizio. Imponiamo quindi il vincolo che sul percorso tra il generico Correspondent Node e la Home Network della VM che ospita il servizio a cui è interessato, sia sempre localizzato almeno un Core Service Switch che sia in grado di implementare il meccanismo di reindirizzamento del traffico destinato a VM migrate.

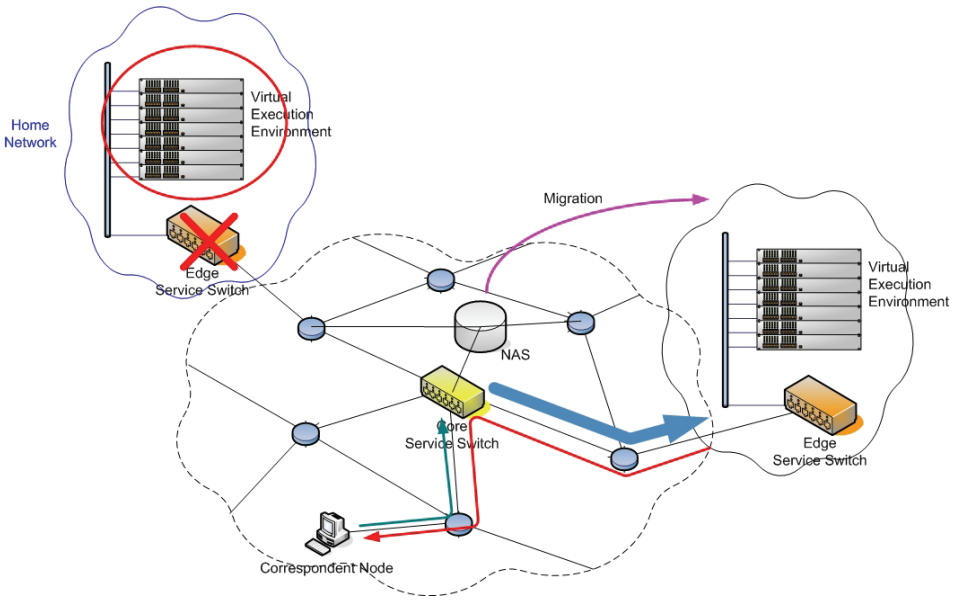

Figura 36 - Reindirizzamento di traffico destinato a VM riattivate

Risulta evidente che lo stato in cui il servizio verrà riattivato sarà differente rispetto a quello in cui si trovava nell'stante del crash dell'Edge Service Switch. È possibile in ogni caso fare in modo che i servizi vengano riabilitati con uno stato relativamente recente facendo in modo che le copie di back-up delle immagini delle VM sul NAS, vengano periodicamente rinfrescate. L'operazione di refresh ricade sotto la responsabilità dell'Edge Servce Switch che gestisce il VEE su cui la generica VM è attiva.

Sono diversi gli approcci che avremmo potuto implementare per la risoluzione del problema trattato in questo paragrafo, e per arrivare alla selezione e quindi alla realizzazione del meccanismo proposto, abbiamo preso in esame differenti alternative. Una di queste è la riconfigurazione del client dell'utente in sostituzione al reindirizzamento di traffico destinato a VM riesumate. Con tale soluzione avremmo probabilmente guadagnato in robustezza, considerato che col nostro approccio esistono condizioni estreme in cui non c'è possibilità di risolvere il problema dell'isolamento di servizi; dall'altro canto, a differenza dell'alternativa, il meccanismo da noi implementato è del tutto trasparente rispetto al generico utente.

Ancora, abbiamo preso in esame il concetto di replicazione di servizio in alternativa a quello di riattivazione. In futuro ci occuperemo di mettere a confronto i due diversi approcci per valutare quale dei due fornisca migliori risultati in funzione di determinate classi di servizi.

Capitolo 5
Valutazione delle prestazioni

Ci occuperemo in questo capitolo di descrivere alcune sessioni sperimentali eseguite per valutare il modello Service Switching, con particolare riferimento all'implementazione basata sull'uso dei meccanismi di virtualizzazione presentata ai Capitoli III e IV.

Partiremo col presentare i risultati sperimentali del confronto eseguito tra due differenti tecniche di virtualizzazione, OpenVZ e Xen, grazie ai quali abbiamo selezionato lo strumento utilizzato per l'implementazione della nostra architettura. Ci occuperemo successivamente di descrivere un primo esperimento basato sull'uso di Xen, finalizzato a verificare l'efficacia di tale strumento in contesti legati al Networking; L'esperimento consiste nella realizzazione di un router modulare che esegue il filtraggio di frame video. Passeremo quindi alla sessione sperimentale sul Service Switching, eseguita in riferimento ad un particolare caso d'uso, la distribuzione di contenuti web. Utilizzando una architettura per il caching e la distribuzione di contenuti web a comunità di utenti, preventivamente testata sui nodi di una slice Planetlab, abbiamo sperimentato la piattaforma Service Switching implementando le funzionalità assegnate alle componenti della CDN come servizi ospitati all'interno della nostra piattaforma. Riportiamo infine una descrizione delle prove sperimentali eseguite sui meccanismi di forwarding utilizzati nell'architettura Service Switching: MPLS Splitting ed MPLS Multicast; ci occuperemo innanzitutto di misurare l'overhead introdotto sui nodi che implementano tali meccanismi, e successivamente di eseguire una analisi delle prestazioni.

5.1 Confronto tra differenti tecniche di virtualizzazione

L'implementazione dell'architettura Serivice Switching proposta al Capitolo III, è basata sull'uso dei meccanismi di virtualizzazione, in particolare sull'uso di Xen. La selezione di Xen è stata eseguita a valle di una fase di valutazione e di confronto tra differenti strumenti open source messi a disposizione della comunità scientifica. Considerati quelli maggiormente utilizzati in virtù delle potenzialità offerte, abbiamo eseguito dei test con l'obiettivo di metterne in evidenza le caratteristiche; i parametri presi in considerazione sono i seguenti: scalabilità, flessibilità, semplicità d'uso. Riportiamo di seguito i dettagli relativi ad alcuni degli strumenti di virtualizzazione maggiormente utilizzati, e successivamente la descrizione del confronto eseguito tra due di questi, OpenVZ e ovviamente Xen.

FreeBSD Jail è lo strumento di virtualizzazione alla base dell'implementazione di Emulab [67], un sistema che supporta simulazioni di rete multiple in esecuzione concorrente sul medesimo nodo fisico. FreeBSD Jail fornisce la possibilità di partizionare l'ambiente di esecuzione, in particolare, l'amministratore del sistema può creare diversi minisistemi tra loro indipendenti, denominati jail; l'account super-user ha diritto di accesso su ognuno di questi. Il generico jail è un ambiente virtuale in esecuzione sulla macchina host, caratterizzato dai propri file, i propri processi attivi, uno o più account utente ed un account superutente. Dal punto di vista del generico processo in esecuzione, il jail non è differente da un sistema reale. All'atto della creazione, il jail viene associato ad un root file system; i processi in esecuzione sul jail non possono manipolare file che non sono in grado di indirizzare, quindi, i file che si trovano al di fuori del root file system del jail, risultano protetti da quest'ultimo. La sicurezza è garantita dal fatto che l'ambiente jail è separato dal resto del sistema: nel caso in cui un jail dovesse essere vittima di fallimenti o cattive configurazioni che generano errori, tale evento non intacca l'integrità del resto del sistema.

Linux-VServer implementa un partizionamento "leggero" basato sul concetto di Security Context, mediante il quale è possibile creare dei Virtual Private Server (VPS) indipendenti, in esecuzione concorrente sul medesimo server fisico. Un VPS fornisce un

ambiente operativo identico a quello di un server Linux convenzionale. Tutti i servizi possono essere attivati su tale VPN, senza richiedere alcuna modifica preventiva, mentre invece l'implementazione del Security Context richiede alcune modifiche al kernel di Linux. Il Context nasconde tutti i processi in esecuzione che si trovano al di fuori della sua copertura, e proibisce ogni interazione indesiderata con processi appartenenti ad un Context differente. Il problema legato all'uso di VServer è che il Networking è basato sull'isolamento e non sulla virtualizzazione; ciò impedisce ad ogni VPS di creare il proprio routing interno e di assegnare differenti indirizzi MAC.

OpenVZ [68] è uno strumento per la virtualizzazione a livello sistema operativo che fornisce la possibilità di eseguire istanze di sistema multiple, anche in questo caso denominate Virtual Private Server (VPS), oppure Virtual Environment (VE). Se paragonato ad altri meccanismi di virtualizzazione come VMWare o Xen, OpenVZ non offre la medesima flessibilità in termini di tipologie di Sistema Operativo ospitabile, ma può comunque rivelarsi una buona alternativa in relazione a differenti contesti. Il networking in OpenVZ è implementato attraverso un virtual device identificato come veth, un dispositivo Ethernet-like che può essere istanziato all'interno di un VE. Al veth è possibile assegnare un MAC address, ed è possibile utilizzarlo per implementare una configurazione bridg, che consiste nell'emulare una sorta di vitual switch all'interno di un host, e al quale tutte le interfacce create all'interno del VE sono connesse. Il veth può essere configurato attraverso un DHCP server durante il boot del VE.

Xen è un sistema di paravirtualizzazione realizzato all'Università di Cambridge; fornisce un Virtual Machine Monitor (VMM) per processori della famiglia x86, e attraverso questi l'esecuzione multipla di diversi sistemi operativi. Ulteriori dettagli relativi a Xen sono stati precedentemente introdotti al Capitolo III.

Riportiamo di seguito il confronto tra due dei sistemi di virtualizzazione appena elencati, OpenVZ e Xen, eseguito nel contesto della emulazione di rete. Entrambi gli strumenti consentono di configurare una interfaccia di rete virtuale all'interno di una macchina virtuale. OpenVZ appartiene alla categoria delle soluzioni per la

virtualizzazione di server a livello di sistema operativo, mentre Xen a quello degli hypervisor basati sul concetto di paravirtualizzazione.

L'esperimento eseguito consiste nella realizzazione di una rete composta da due end-sysyem interconnessi attraverso l'uso di una coppia di router IP intermedi (Figura 37). Tale scenario è stato realizzato sfruttando differenti configurazioni: in una di queste i router sono implementati su macchine virtuali Xen, in un'altra invece, su macchine virtuali OpenVZ; in entrambi i casi le macchine virtuali sono state allocate all'interno di due diverse box Linux. In un ulteriore scenario che verrà utilizzato come riferimento, i router intermedi sono implementati senza l'uso della virtualizzazione, quindi direttamente su box Linux.

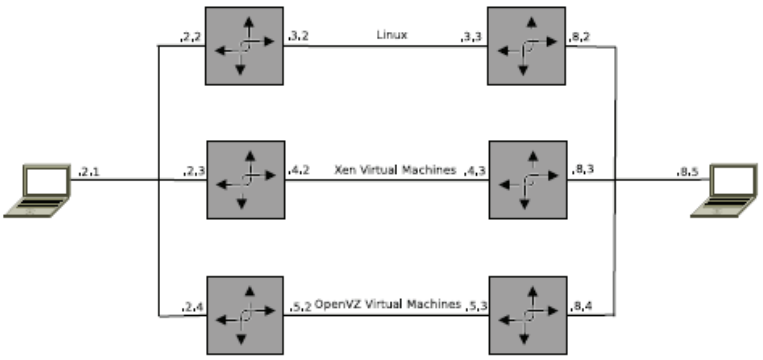

Figura 37 – Confronto OpenVZ – Xen: testbed sperimentale

Le macchine fisiche utilizzate per l'esperimento hanno tra loro una configurazione hardware identica: ogni macchina è equipaggiata con due Intel Xeon da 2.8Ghz, 5GB di RAM, ed una interfaccia Tigon 3 gigabit ethernet. La virtualizzazione della rete è ottenuta attraverso il meccanismo dell'IP-aliasing.

Per quanto riguarda la configurazione di Xen, abbiamo creato per ogni router virtuale due virtual NIC (Network Inteface Card), ed il networking è stato configurato con la modalità host-networking via bridging. Discorso del tutto analogo per le veth nella configurazione OpenVZ.

Negli esperimenti eseguiti abbiamo utilizzato D-ITG, un generatore di traffico sintetico realizzato all'Università di Napoli [69]. D-ITG offre diversi gradi di libertà nella configurazione del traffico da inoltrare da sorgente a destinazione, consente infatti di impostare parametri quali: dimensione del pacchetto, quantità complessiva di dati trasmessi, velocità di trasmissione dei pacchetti appartenenti ai flussi di traffico generati, eccetera. Il software è costituito da una componente in esecuzione sul sender e che si occupa della generazione del traffico, ed una in esecuzione sul receiver alla quale è affidato il compito di calcolare delle statistiche relative alle caratteristiche del traffico ricevuto. D-ITG supporta entrambi i protocolli di livello trasporto.

I parametri selezionati per la configurazione dei pacchetti in relazione ai nostri esperimenti, sono: packet size (dimensione del pacchetto) e inter-departure time (IDT – velocità di trasmissione dei pacchetti). Gli esperimenti sono stati eseguiti generando stream CBR costituiti da pacchetti di dimensione costante pari a 1 KB in maniera continua per 30 secondi, facendo variare l'IDT.

In Figura 38 è riportato un diagramma che indica la quantità di pacchetti persi al variare della velocità di trasmissione. I risultati mostrano che per rate di trasmissione che non superano i 30000 pacchetti al secondo, non c'è alcuna perdita, la quale diventa invece significativa non appena tale soglia viene superata. Risulta inoltre evidente che la percentuale di pacchetti persi utilizzando Xen, è più elevata rispetto a quella che si ottiene utilizzando invece OpenVZ.

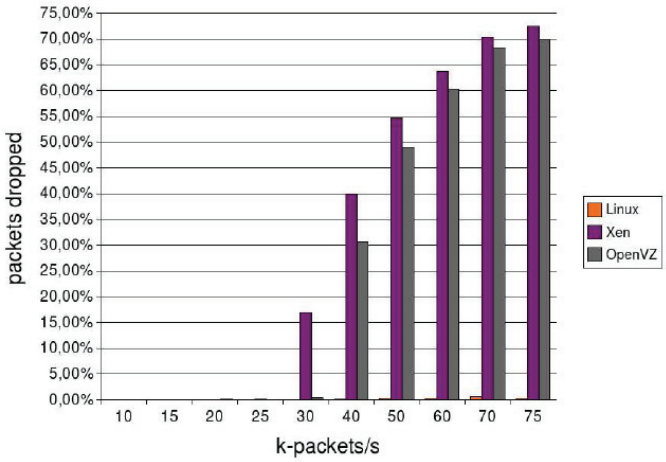

Figura 38 – Confronto OpenVZ – Xen: pacchetti persi in funzione della velocità di trasmissione

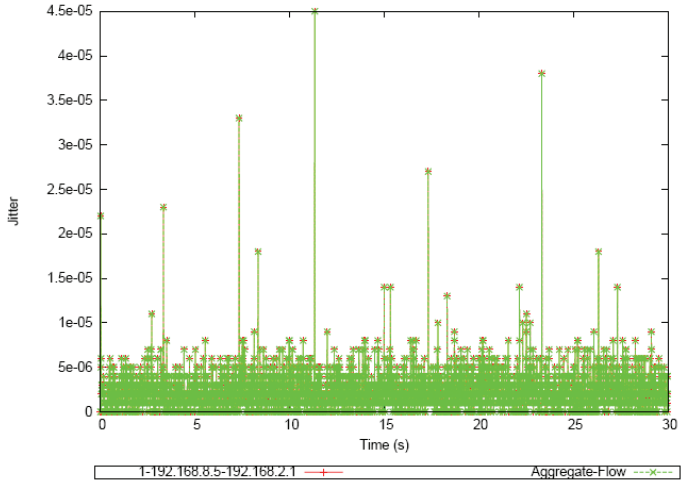

Figura 39 - Confronto OpenVZ – Xen: jitter introdotto dal router Linux

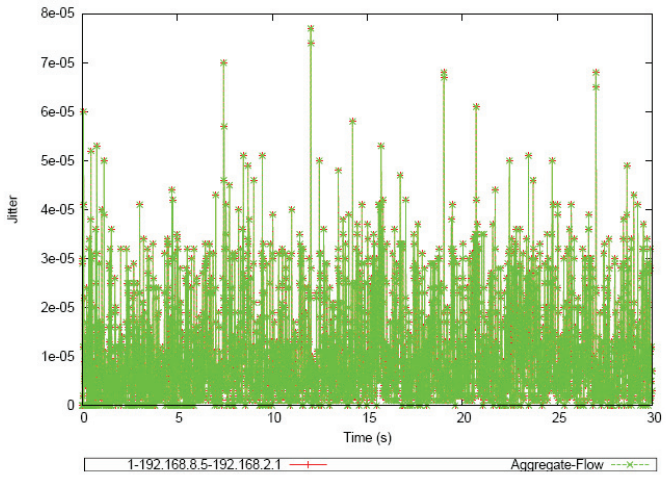

Figura 40 - Confronto OpenVZ – Xen: jitter introdotto dal router Xen

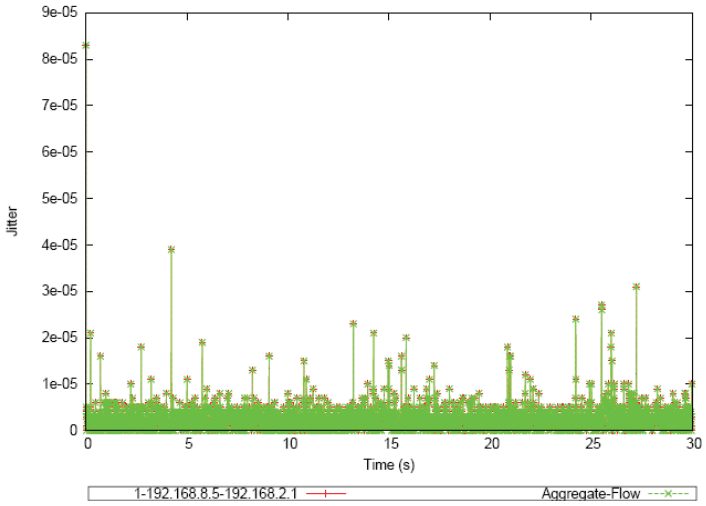

Figura 41 - Confronto OpenVZ – Xen: jitter introdotto dal router OpenVz

Entrambi gli strumenti comportano una riduzione del throughput effettivamente sostenibile considerate le configurazioni implementate. Riportiamo nelle figure 39, 40 e 41, il jitter misurato sul nodo destinazione ad intervalli regolari di 10 millisecondi, in riferimento rispettivamente agli scenari: router Linux, Xen, OpenVZ. Tenuto in conto

che gli esperimenti sono stati eseguiti in completa assenza di interferenze, il jitter è interamente dovuto al fatto che il processing dei pacchetti sui router intermedi richiede un tempo variabile. I grafici mostrano che i router OpenVZ introducono mediamente un jitter inferiore del 50% rispetto a quello generato invece dai router Xen.

I risultati dei test ci consentono di affermare che entrambe le tecniche di virtualizzazione considerate, presentano alcuni problemi nel caso in cui siano utilizzate per la gestione di una grossa quantità di traffico di rete. Seppure i risultati sperimentali ottenuti mostrano che OpenVZ offre prestazioni migliori rispetto a Xen, quest'ultimo risulta essere molto più flessibile e semplice da utilizzare, aspetti che costituiscono un grosso punto a favore in special modo nel caso in cui si debbano gestire sistemi cluster-based composti da un elevato numero di nodi, come nel caso dei Virtual Execution Environment presenti nell'architettura Service Switching. Inoltre, la fault-isolation che i sistemi di paravirtualizzazione sono in grado di garantire, spingono la scelta ancora in favore di Xen.

5.2 Una prima sperimentazione dell'uso di Xen

Descriviamo in questo paragrafo alcune prove sperimentali eseguite per verificare ancora una volta l'efficacia dell'utilizzo dei meccanismi di virtualizzazione, e in particolare di Xen. L'obiettivo è quello di valutare la possibilità di utilizzare tecniche di virtualizzazione nell'ambito di uno scenario di reti attive.

L'applicazione scelta come target per la nostra sperimentazione, consiste nel trasmettere un flusso video tra uno o più sender ed uno o più receiver, utilizzando un codec a "livelli". Un video codificato a livelli produce uno stream formato da più flussi, cui corrispondono diversi gradi di qualità; ogni livello trasporta un'informazione di dettaglio sul flusso che lo precede. Ciò significa che, a parte il livello base che trasporta una versione del video autosufficiente ma codificata in maniera grossolana, tutti gli altri livelli (detti di enhancement) trasportano pacchetti accessori; nel caso in cui tali

pacchetti venissero ricevuti, contribuiscono a migliorare la qualità del video, in caso contrario, la decodifica non risulta pregiudicata.

Il nostro lavoro è stato quello di dotare un nodo di rete attivo di una capacità di filtraggio intelligente. Nel caso in cui ci sia congestione della rete, oppure in quello in cui il destinatario dichiari esplicitamente di non essere in grado di supportare un rate specifico, il router deve poter selezionare in maniera razionale quali pacchetti eliminare per ridurre il rate dello streaming. La scelta cadrà ovviamente su quelli dei flussi di maggior dettaglio.

Il tool utilizzato per la generazione del video è vic, un tool di videoconferenza progettato da un gruppo di ricerca dell'Unversità di Berkeley in California; vic implementa una codifica a livelli identificata come PVH (Progressive video with Hybrid Transform). Il meccanismo di distribuzione implementato da vic, utilizza il multicast per offrire un elevato livello di scalabilità: ogni ricevitore interessato ai contenuti, in base alle risorse elaborative di cui dispone, può effettuare il join ad un numero arbitrario di gruppi, determinando in questo modo il livello di qualità desiderato per la fruizione del video. Il sender invierà solo una copia di ogni flusso, saranno poi i router multicast sul percorso ad inoltrare opportunamente i singoli flussi in maniera indipendente e in base ai join "selettivi" effettuati dagli utenti. L'utilizzo di banda risulta quindi ottimizzato in quanto non c'è replicazione di contenuti sui link.

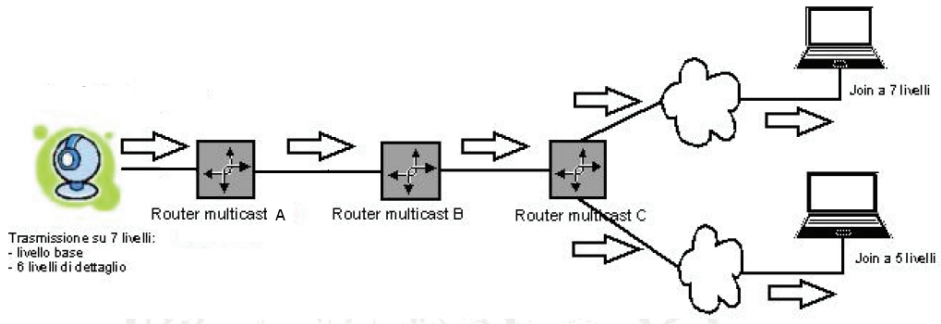

Figura 42 – Realizzazione di un nodo attivo per il filtering di frame video: scenario realizzato

La programmazione del router attivo è stata effettuata utilizzando Click [70], una architettura software che permette di realizzare router modulari. Un router modulare è costituito da più componenti elementari, ognuna delle quali svolge solo una semplice e precisa operazione di processing sui pacchetti. Click fornisce delle componenti basilari (elements) necessarie per classificare, accodare, schedulare pacchetti; editando un file di configurazione, è possibile selezionare gli elements da utilizzare, e definire i potenziali percorsi che i pacchetti entranti nelle interfacce del router dovranno seguire; modificare i file di configurazione consente quindi di costruire router estremamente flessibili ed adattabili al particolare contesto applicativo. Utilizziamo per la nostra sperimentazione una macchina Xen (versione 3.1.0) su cui è attiva un'unica macchina virtuale che ospita il nostro router modulare implementato con Click.

L'obiettivo delle sperimentazioni è quello di valutare sia quantitativamente che qualitativamente l'impatto del filtraggio sui rate dei flussi dello streaming. Abbiamo quindi eseguito il logging dei pacchetti in transito sulla dom0 di Xen a seguito dell'elaborazione implementata sulla domU, quindi immediatamente prima che lascino il nodo attivo. Misuriamo dunque l'incidenza dei flussi di streaming sul traffico a valle del router attivo, in modo da valutare il rate di trasmissione (packet rate e bit rate) con o senza operazione di filtraggio. Il nodo attivo è collegato al receiver mediante un cavo cross, il canale è quindi altamente affidabile.

Il sender impiegato trasmette sempre con il massimo numero di livelli, quindi 6 di dettaglio su 7 complessivi.

Il logging è stato effettuato per tre casistiche:

- assenza di filtraggio (ritroviamo il pieno rate prodotto dal sender);
- filtraggio che taglia due livelli di dettaglio;
- filtraggio che taglia tre livelli di dettaglio.

Tale scelta è dettata dalla necessità di valutare empiricamente l'incidenza del taglio di un livello sul rate, e determinare di conseguenza un valore di compromesso fra qualità accettabile del video e risparmio di banda. Nelle immagini 7 e 8 riportiamo i risultati delle misurazioni effettuate in un intervallo di tempo campione:

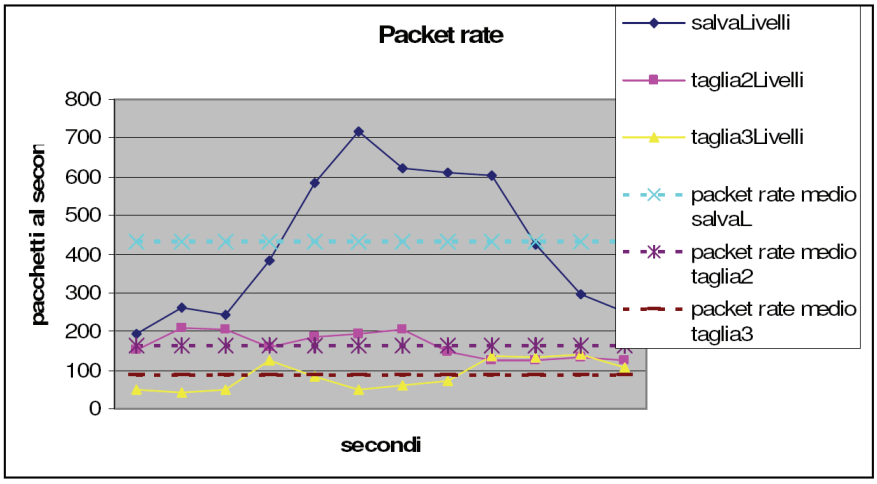

Figura 43 - Realizzazione di un nodo attivo per il filtering di frame video: packet rate sul nodo destinazione

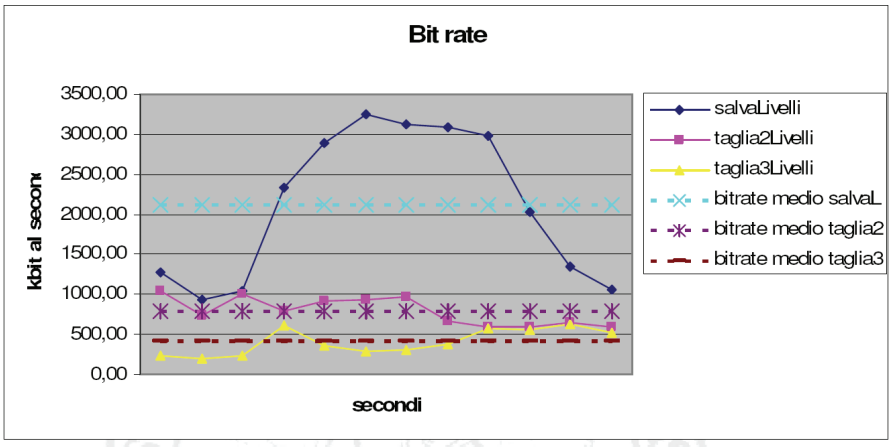

Figura 44 - Realizzazione di un nodo attivo per il filtering di frame video: bit rate sul nodo destinazione

Dai grafici è possibile verificare che la curva associata alla modalità salva livelli, presenta un andamento differente rispetto alle altre, in particolare, in un determinato intervallo assume valori più elevati. Ciò è dovuto al fatto che in tale intervallo, l'occorrenza di pacchetti dei livelli di maggior dettaglio è superiore, e dunque tali pacchetti vengono bloccati nelle altre due modalità di funzionamento. Tagliare solo i due livelli di maggior dettaglio comporta una riduzione del numero medio di pacchetti al secondo inoltrati, quindi un risparmio in termini di bitrate medio del 62% (da 2112 a 790 kbit/sec). Bloccando invece i 3 livelli di maggior dettaglio, la riduzione del packet rate medio è ancora maggiore, e comporta un risparmio in termini di bit rate medio dell'80% (da 2112 a 406 kbit/sec).

5.3 Prove sperimentali sull'architettura Service Switching

Le prove sperimentali per il testing dell'architettura Service Switching sono state eseguite facendo riferimento ad un particolare caso d'uso, le reti per la distribuzione di contenuti web. Nel seguito verrà riportata una breve descrizione del modello di riferimento, delle entità che ne fanno parte, e del modo in cui la sessione sperimentale è stata condotta.

5.3.1 Caso d'uso: il caching e la distribuzione di contenuti web

Le aziende che offrono servizi legati alla distribuzione di contenuti web sono alla continua ricerca di soluzioni sempre più performanti che garantiscano prestazioni elevate, e che implementino tecniche di Web Caching per applicazioni dinamiche complesse. In relazione a tale contesto, e nell'ambito di una collaborazione col Computing Department dell'Università di Lancaster (UK), abbiamo lavorato alla progettazione e realizzazione di un'architettura per la distribuzione di contenuti Web a comunità di utenti. Il modello proposto offre un meccanismo per il caching on-demand di contenuti Web, implementato rispettando delle politiche di content adaptation

opportunamente definite, ed un meccanismo per la distribuzione di tali contenuti a comunità di utenti attraverso l'uso di entità opportunamente progettate e realizzate.

Gli utenti a cui sono indirizzati i benefici introdotti dall'architettura proposta, non costituiscono un'aggregazione disomogenea, ma sono accumunati dai medesimi interessi; indicheremo con community (comunità) tale insieme di utenti. Indicheremo invece col termine cluster un sottoinsieme della community composto da utenti dislocati nella medesima zona e che condividono l'infrastruttura di rete per l'accesso ad Internet. Gli utenti sono accomunati dalla necessità di usufruire dei medesimi servizi Internet, in particolare dei medesimi contenuti messi a disposizione di uno o più Content Provider.

L'obiettivo nell'utilizzo di tale architettura è quello di migliorare la qualità dell'esperienza percepita da tali utenti; l'implementazione del meccanismo proposto comporta una riduzione della latenza percepita durante l'accesso ad un contenuto specifico. L'architettura fa uso di un insieme di cache messe a disposizione dal generico Service Provider, il quale si impegna ad utilizzare la propria infrastruttura di rete per distribuire i contenuti richiesti agli utenti della comunità. Ogni cluster di utenti si affida alle funzionalità offerte da un'altra componente, il nodo Proxy; quest'ultimo, nota l'allocazione dei contenuti presso la rete di cache, è in grado di indirizzare ogni singola richiesta verso la cache specifica, in modo da ottimizzare l'accesso al contenuto richiesto. I client utilizzati dagli utenti della community devono essere configurati in modo tale da inoltrare automaticamente le proprie richieste verso il Proxy associato al cluster di cui fanno parte; sarà compito di tale componente quello di indirizzare ogni richiesta verso l'opportuna cache. Il nodo Proxy implementa un meccanismo di content-aware request routing (instradamento delle richieste in funzione del contenuto), le cui regole vengono stabilite in maniera dinamica in base all'effettiva allocazione delle risorse all'interno della rete di cache.

Il Proxy interpreta le richieste ricevute e stabilisce in che modo instradarle: utilizza a tale scopo una tabella di routing che identifichiamo come Content Routing Table, alla quale è affidato il compito di mantenere la corrispondenza tra lo specifico contenuto e la cache su cui è allocato. Essendo inoltre destinazione di tutte le richieste eseguite dagli

utenti, il Proxy deve essere in grado di instradare anche quelle relative a contenuti non disponibili all'interno del sistema di cache, e quindi di reinoltare verso il server d'origine del contenuto.

La Content Routing Table è costituita da coppie del tipo <chiave, valore>; la chiave di accesso corrisponde con la URL della risorsa, il valore alla cache verso cui deve essere rediretta la richiesta per il contenuto. L'URL è un'informazione strutturata complessa, ed è composta dai seguenti campi elementari: Protocol, Host, Port, Path, Query string. Le informazioni fornite per l'individuazione della cache da interrogare sono invece le seguenti: hostname (o l'indirizzo IP), port number.

Protocol	Host	Port	Path	Query string	Host	Port
http	www.someho st.com	80	*	*	Cache1	8080
http	www.someho st.com	80	/path/video.avi	*	Cache2	3128
http	*	80	/path	*	Cache3	8080

Tabella 3 – Rete CDN: Content Routing Table

Sebbene la tabella sia utilizzata dal Proxy, è il CDN-Coordinator (che presenteremo di seguito) ad occuparsi della gestione da remoto di quest'ultima, quindi dell'inserimento, della modifica o della cancellazione di entry.

Introduciamo una ulteriore componente, il CDN-Coordinator; il compito affidato a tale entità è quello di: gestire l'intera architettura, stabilire lo schema di allocazione delle risorse presso il sistema di cache, definire le politiche di request routing, istruire i Proxy di ogni cluster. Il CDN-Coordinator definisce lo schema di localizzazione ottimale dei contenuti in accordo alle politiche di content adaptation adottate. Fatto ciò, coordina la rete affinché vengano rispettate le politiche di request routing stabilite. Il problema della

localizzazione richiede diverse informazioni per poter essere risolto, riguardanti: lo stato della rete, la banda a disposizione, il traffico generato dagli utenti di ogni cluster, il numero di accessi ad ogni risorsa. Il CDN-Coordinator si occupa della raccolta di tali informazioni, lavoro che esegue attraverso l'uso di determinate componenti software alle quali è assegnato il compito di eseguire determinate misurazioni.

Calcolare lo schema di localizzazione ottimo consiste nel risolvere un problema classico di localizzazione su rete, quindi nel definire un opportuno modello di programmazione lineare binaria. La funzione obiettivo minimizza il tempo necessario per accedere ai contenuti. L'interazione tra Proxy e CDN-Coordinator è supportata da un protocollo di comunicazione opportunamente realizzato.

Le componenti che costituiscono l'architettura appena descritta, sono state distribuite, eseguite e testate sui nodi appartenenti ad una slice PlanetLab [27, 33]. L'obiettivo della campagna sperimentale eseguita, è semplicemente quello di verificare l'efficacia del modello proposto. Consideriamo a tale scopo un caso particolare, consideriamo un unico cluster di utenti costituito da un unico utente; all'interno della medesima LAN ci sarà anche un nodo Proxy a cui l'unico utente del cluster indirizza le proprie richieste. Utilizziamo due cache ed un web-server, il quale mette a disposizione dell'utente un unico oggetto di dimensioni pari a 4.162.048 byte.

Lo scenario implementato è costituito quindi da: 1 nodo Proxy, 2 cache, 1 web server Apache.

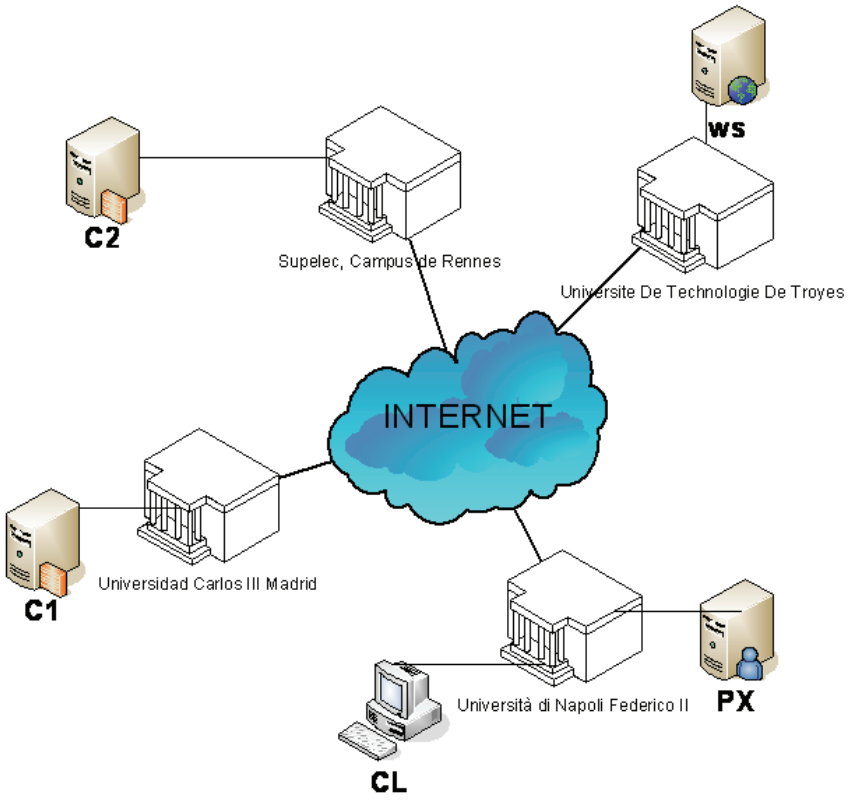

Figura 45 – Test della CDN su Planetlab: topologia di riferimento

Il WebServer è in esecuzione su un nodo Planetlab al quale è associata una quantità di banda limitata, al fine si valutare la convenienza nell'uso delle cache. La quantità di banda associata alla cache C2 è confrontabile con quella del Web Server, quella associata alla C1 è più elevata.

Ricordiamo che nel caso in cui il contenuto richiesto dall'utente non sia disponibile sulla cache, il Proxy reindirizzerà la richiesta direttamente sul Web Server di origine, il quale provvede ad assolverla; in tal caso la cache approfitterà per salvare una copia del contenuto.

Eseguiamo una valutazione dei tempi impiegati per scaricare un oggetto considerando i seguenti casi:

- richiesta inoltrata direttamente al Web Server di origine (CL →WS);

- richiesta inoltrata ad una cache su cui il contenuto non è disponibile (CL→C1→WS e CL→C2→WS);
- richiesta inoltrata ad una cache su cui il contenuto è disponibile (CL→C1 e C1→C2);
- richiesta inoltrata al Proxy che scarica il contenuto direttamente dal Web Server di origine in quanto non dispone nella propria tabella di alcuna entry associata al contenuto richiesto (Cl→PX→WS)
- richiesta inoltrata al Proxy che scarica il contenuto da una delle due cache, la quale però non ha a disposizione il contenuto richiesto (CL→PX→C1→WS e Cl→PX→C1→WS)
- richiesta inoltrata al Proxy che scarica il contenuto da una delle due cache sulla quale il contenuto richiesto è disponibile (Cl→PX→C1 e CL→PX→C1).

I valori ottenuti dalle misurazioni effettuate, sono riportati nelle tabelle e nei grafici che seguono.

esperimento	μ	σ
CL->WS	47,9	9,6
CL>PX->WS	73,2	7,7
CL->C1->WS	69,9	10,6
CL->C1	3,5	0,6
CL->PX->C1->WS	58,7	7,5
CL->PX->C1	3,4	0,2
CL->C2->WS	128,6	14,1
CL->C2	78,7	13,5
CL->PX->C2->WS	125,5	25,0
CL->PX->C2	72,8	7,9

Tabella 4 - Test della CDN su Planetlab: tempi di trasferimento

esperimento	μ	σ
CL->WS	88,0	16,3
CL>PX->WS	56,3	5,8
CL->C1->WS	59,5	7,9
CL->C1	1159,6	90,5
CL->PX->C1->WS	70,8	9,5
CL->PX->C1	1212,5	46,2
CL->C2->WS	32,1	3,6
CL->C2	53,1	8,5
CL->PX->C2->WS	33,7	5,6
CL->PX->C2	56,7	5,8

Tabella 5 - Test della CDN su Planetlab: velocità di trasferimento

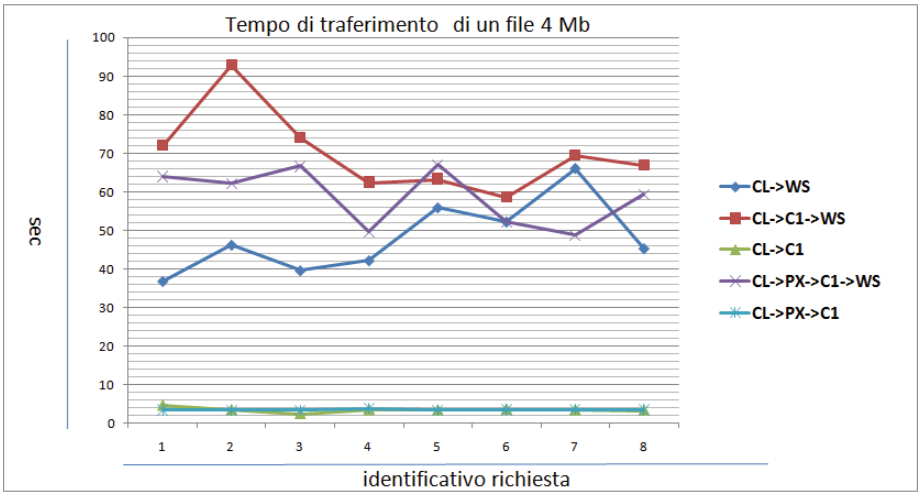

Figura 46 - Test della CDN su Planetlab: tempo di trasferimento di un file tramite la cache C1

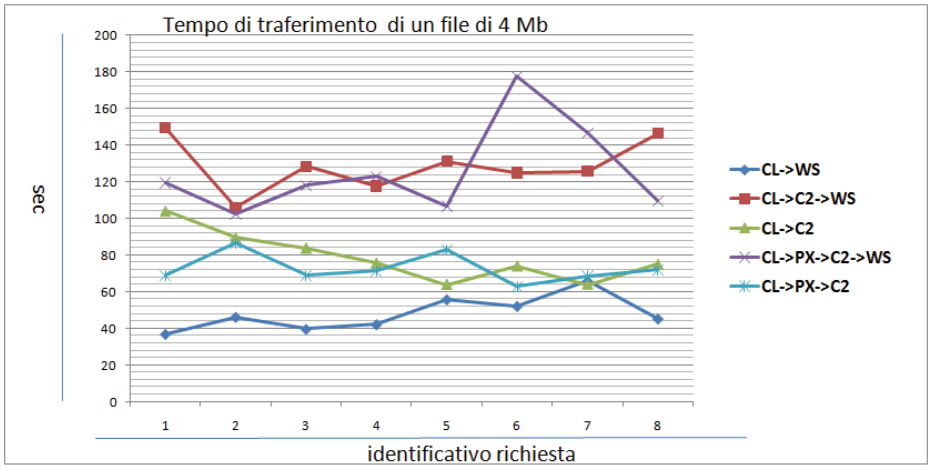

Figura 47 - Test della CDN su Planetlab: tempo di trasferimento di un file tramite la cache C2

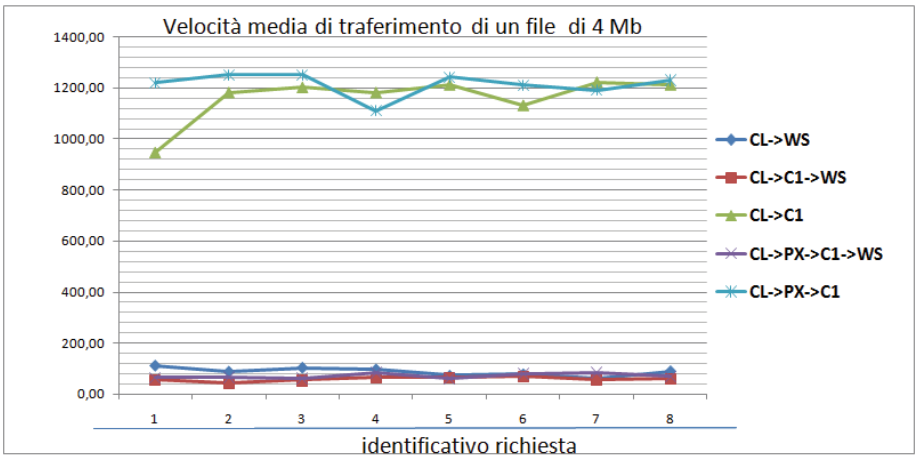

Figura 48 - Test della CDN su Planetlab: velocità di trasferimento di un file tramite la cache C1

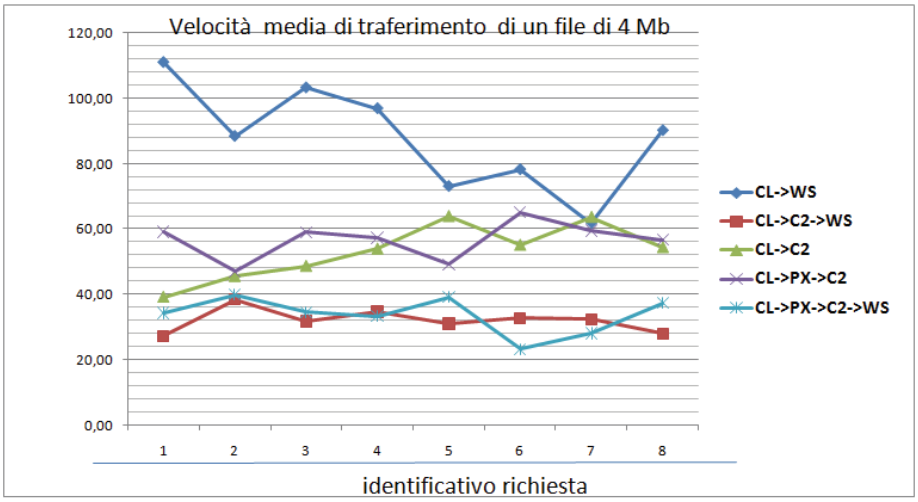

Figura 49 - Test della CDN su Planetlab: velocità di trasferimento di un file tramite la cache2

I grafici evidenziano che l'uso della cache, in taluni casi, riduce in maniera significativa i tempi di accesso ad un contenuto. È possibile verificare inoltre che la prossimità geografica non costituisce necessariamente un vantaggio: nonostante la cache C1 rispetto alla C2 sia più distante dal Web Server, riesce a reperire il contenuto richiesto in meno tempo; questo è certamente legato alla banda disponibile. In linea di massima, risulta che l'utilizzo della cache C1 comporta effettivamente dei vantaggi, al contrario di quanto accade invece con la cache C2; utilizzare quest'ultima, infatti, comporta addirittura un decremento delle prestazioni rispetto al caso in cui il contenuto richiesto venga direttamente scaricato dal Web Server di origine, oppure indirettamente attraverso il Proxy.

5.3.2 Descrizione della sessione sperimentale sul Service Switching

Come anticipato in precedenza, l'obiettivo della sessione sperimentale è quello di verificare l'efficacia dell'implementazione del modello Service Switching basata sull'uso dei meccanismi di virtualizzazione e presentata ai Capitoli III e IV, utilizzando come caso d'uso i servizi offerti dall'architettura CDN appena introdotta.

Cominciamo quindi col considerare le principali componenti di tale architettura, e verifichiamo per ognuna di esse se ha senso implementarne le funzionalità come servizio sulla piattaforma Service Switching. Successivamente descriveremo le prove sperimentali eseguite.

Nodo Proxy

Nell'architettura CDN è previsto un nodo Proxy per ogni cluster di utenti; il Proxy fa parte della medesima rete LAN ed utilizza la stessa rete d'accesso ad Internet degli utenti. Ha certamente senso ospitare le funzionalità del Proxy su un VEE dell'architettura Service Switching, supposto che la Home Network del Proxy sia la rete LAN in cui sono localizzati gli utenti di cui processa le richieste. Per quanto riguarda la migrazione del servizio, e quindi della macchina virtuale su cui è in esecuzione, è probabilmente necessario limitarsi a considerare la migrazione locale tra due nodi del medesimo VEE; in tal caso il Proxy continuerebbe ad essere localizzato all'interno della stessa rete locale in cui si trovano gli utenti che compongono il cluster. Supponiamo venga invece migrato su un nodo remoto e localizzato in una LAN differente: il meccanismo di readdressing implementato dal Service Switch della Home Network del Proxy, garantisce che tutto il traffico destinato a quest'ultimo venga correttamente consegnato; ciò che si verifica, però, è un inevitabile calo delle prestazioni nell'implementazione del caching dei contenuti Web. Il vantaggio ottenibile da tale meccanismo risiede nel fatto che la richiesta dell'utente per l'accesso ad uno specifico contenuto, viene rediretta dal Proxy verso una cache, piuttosto che essere inoltrata direttamente al server di origine; supposto che la posizione della cache e la quantità di banda disponibile siano favorevoli rispetto alla situazione in cui si trova invece il Server Web, ottenere il contenuto dalla cache comporta di certo una latenza inferiore. Se però il Proxy non si trova, come suggerito dal modello, nella medesima rete locale degli utenti che compongono il cluster, sebbene il traffico destinato al Proxy venga correttamente consegnato a quest'ultimo, c'è da implementare due ulteriori passaggi. La sequenza diventa quindi: Client → Service Switch Home Network → Service Switch Foreign

Network → Proxy → Cache; risulta evidente che il vantaggio nell'uso del meccanismo di caching risulterebbe fortemente ridotto.

CDN-Coordinator

Il modello non impone alcun vincolo sulla localizzazione del CDNCoordinator, per cui le funzionalità implementate da tale nodo possono essere ospitate dalla piattaforma Service Switching. La VM che ospita il servizio può essere sottoposta sia a migrazione locale che a migrazione geografica.

Cache

Anche in questo caso il modello non impone vincoli sulla localizzazione di tale componente, per cui le funzionalità implementate dalla cache possono essere ospitate dalla piattaforma Service Switching, e la VM che ospita il servizio può essere sottoposta sia a migrazione locale che geografica. C'è però da fare una considerazione: tenuto in conto la natura del servizio offerto dalla cache, è possibile supporre che la quantità di dati contenuta nel disco sia molto elevata; tale condizione influisce ovviamente sui tempi necessari ad eseguire la migrazione, in particolar modo nel caso in cui sorgente e destinazione siano distribuiti geograficamente.

Le prove sperimentali sono state eseguite sfruttando tre macchine Xen nella configurazione Service Switch BOX introdotta al Capitolo III. In particolare, abbiamo utilizzato un nodo Acer con architettura Dual Core con processori Penium 4 Xeon da 3.20GHz, 2GB di RAM e due interfacce Fast Ethernet, e due nodi HP con due Intel Xeon da 2.8Ghz, 5GB di RAM, ed una interfaccia Tigon 3 Gigabit Ethernet; su ognuna delle macchine è installata una distribuzione CentOS con kernel 2.6.18-92.1.18.el5xen.

I due nodi HP, che indicheremo rispettivamente come nodo A e nodo B, sono localizzati all'interno della medesima rete LAN, mentre il nodo Acer, che indicheremo invece

come nodo C, è localizzato in una rete locale differente; utilizzeremo il nodo A come sorgente, e i nodi B e C come destinazione, al fine di sperimentare quindi sia la migrazione locale che quella geografica. Gli hop intermedi tra il nodo A ed il nodo C sono complessivamente 5; nella seguente Tabella riportiamo l'output del comando traceroute lanciato dal nodo A:

1	192.168.74.198
2	192.168.0.253
3	143.225.81.254
4	viaclaudio.r190.unina.it
5	143.225.229.135

Tabella 6 - Test Service Switching: output Traceroute

Nelle Tabelle 7 e 8 riportiamo invece le caratteristiche dei canali rispettivamente dal nodo A al nodo B, e dal nodo A al nodo C, calcolate in tre fasi differenti inoltrando 1000 pacchetti da sorgente a destinazione.

% pacchetti persi	Dutara complessiva (ms)	RTT min (ms)	RTT medio (ms)	RTT max (ms)	Deviazione standard (ms)
0	999072	0,223611	0,2625	5.191	0,111806
0	999063	0,234028	0,260417	0,451389	0.024
0	999061	0,23125	0,250694	0,393056	0.027

Tabella 7 - Test Service Switching: caratteristiche canale nodo A – nodo B

% pacchetti persi	Dutara complessiva (ms)	RTT min (ms)	RTT medio (ms)	RTT max (ms)	Deviazione standard (ms)
0	999060	0,536111	0,604167	6.741	0,150694
0	999062	0,538194	0,59375	1.273	0.058
0	999073	0,5375	0,582639	2.340	0.066

Tabella 8 - Test Service Switching: caratteristiche canale nodo A – nodo C

Le macchine virtuali utilizzate per eseguire le prove sono tutte configurate con 512MB di memoria ed un disco da 1 GB; la distribuzione istallata sulle VM è una Debian lenny.

L'obiettivo degli esperimenti eseguiti è quello di misurare il tempo necessario ad eseguire la migrazione delle macchine virtuali che ospitano i servizi implementati dalle componenti dell'architettura CDN, e valutare quindi il downtime imposto ai servizi. Abbiamo quindi utilizzato due macchine virtuali: sulla prima, che identifichiamo come VM1, è allocato il componente Cache, e sull'altra, che identifichiamo invece come VM2, i componenti CDN-Coordinator e Proxy.

Allo stato iniziale le macchine virtuali sono entrambe ospitate dal nodo A; eseguiamo un primo test facendo migrare la VM1 verso il nodo B (migrazione locale), e la VM2 verso il nodo C (migrazione geografica). Successivamente, eseguiamo la migrazione della VM1 verso il nodo C (migrazione geografica) , e della VM2 nuovamente verso il nodo A (migrazione geografica).

In riferimento agli scenari appena descritti, abbiamo tralasciato la migrazione del disco ed abbiamo invece calcolato i tempi necessari ad eseguire la migrazione della memoria, quindi l'intervallo di tempo durante il quale le applicazioni in esecuzione sulla macchina virtuale risultano disabilitati.

I risultati ottenuti sono riportati nelle Tabelle 9, 10, 11 e 12; sulle colonne sono riportati i seguenti valori:

- real: tempo necessario all'esecuzione dell'istruzione "xm migrate";
- user: tempo necessario all'esecuzione di processi user-level;
- sys: tempo necessario all'esecuzione di processi system-level;
- delta: tempo necessario ad eseguire la migrazione della memoria;
- sent: rate di trasmissione impiegato dal nodo sorgente.

real (ms)	user (ms)	sys (ms)	delta (ms)	sent (Mb/s)
50.083	176	28	49250	87
49002	160	32	48048	89
49614	140	60	48739	88
49080	164	28	48226	89

Tabella 9 - Test Service Switching: VM1 da nodo A a nodo B

real (ms)	User (ms)	sys (ms)	Delta (ms)	sent (Mb/s)
50885	164	44	50098	85
49254	144	56	48374	88
49050	148	32	48206	89
48884	160	52	48034	89

Tabella 10 - Test Service Switching: VM2 da nodo A a nodo C

real (ms)	User (ms)	sys (ms)	Delta (ms)	sent (Mb/s)
48705	152	52	47849	89
49126	160	40	48214	89
48628	192	20	47713	89
48848	176	32	47883	89

Tabella 11 - Test Service Switching: VM1 da nodo B a nodo C

real (ms)	user (ms)	sys (ms)	Delta (ms)	sent (Mb/s)
46596	100	36	46189	91
46833	100	20	46405	92
46556	80	40	46101	93
46621	116	28	46168	92

Tabella 12 - Test Service Switching: VM2 da nodo C a nodo A

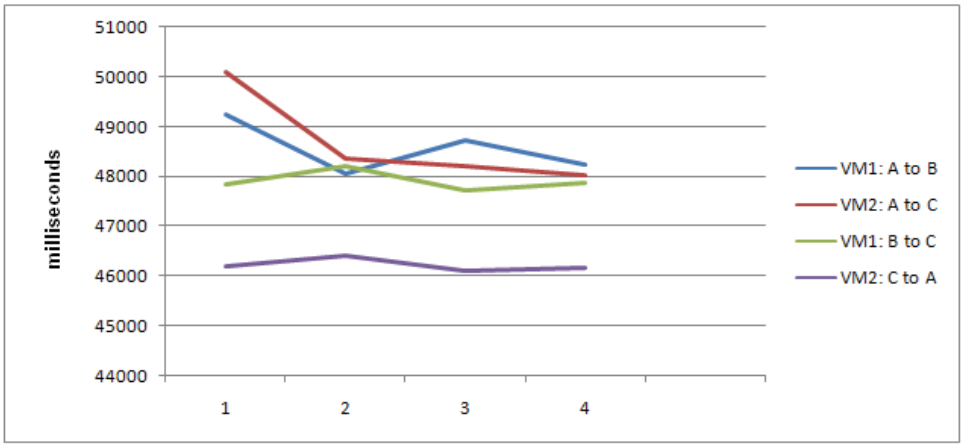

Figura 50 - Test Service Switching: tempi di migrazione a confronto

Il grafico in Figura 50 mette in risalto il confronto tra i tempi di migrazione relativi ai vari scenari. Consultando tale grafico, risulta evidente che il tempo necessario per eseguire la migrazione delle macchine virtuali non dipende dalla particolare applicazione in esecuzione; inoltre, almeno in riferimento a condizioni simili a quelle in cui ci siamo posti per eseguire i nostri esperimenti, il tempo di migrazione non dipende nemmeno dalla distanza geografica tra nodo sorgente e nodo destinazione. Ciò che invece influisce fortemente, è la dimensione della memoria della macchina virtuale da migrare, e ovviamente le condizioni della rete, parametri in relazione ai quali il nodo sorgente determina il rate di trasmissione.

Utilizzando ancora le funzionalità implementate dalla rete CDN come caso d'uso, ci occuperemo in futuro di eseguire delle ulteriori sperimentazioni finalizzate a valutare le prestazioni in termini di latenza relativa alla consegna di contenuti percepita dall'utente, considerando sia il caso in cui le macchine virtuali si troviano all'interno della Home Network, sia quello in cui siano invece ospitate da un Foreign Network.

5.4 Prove sperimentali su MPLS Splitting ed MPLS Multicast

Descriveremo in questo paragrafo le prove sperimentali eseguite sui meccanismi di forwarding utilizzati nell'architettura Service Switching: MPLS Splitting ed MPLS Multicast. Ci occuperemo innanzitutto di misurare l'overhead introdotto da tali meccanismi considerando l'intero insieme di politiche di forwarding implementate, ed eseguiremo successivamente una analisi delle prestazioni con particolare riferimento al meccanismo di Splitting.

5.4.1 Misura dell'overhead introdotto dai meccanismi proposti

Ci occuperemo in questo paragrafo di eseguire una valutazione dell'efficienza dei meccanismi di Splitting e Multicast, misurando l' effettivo overhead computazionale introdotto.

Per eseguire le sperimentazioni abbiamo realizzato un testbed composto da 8 macchine con configurazione hardware identica. Ogni nodo è equipaggiato con un processore Pentium 4 Xeon su cui è in esecuzione il Sistema Operativo Linux ed il pacchetto MPLS-Linux versione 1.950 con l'estensione per lo Splitting ed il Multicast. Ogni macchina possiede inoltre due interfacce di rete Fast Ethernet (100BaseTX) connesse ad uno switch.

Abbiamo utilizzato tre differenti tool: il generatore di traffico D-ITG, gli analizzatori di pacchetto Ethereal e tcpdump, il misuratore delle performance del kernel OProfile.

Dettagli relativi a D-ITG sono già stati forniti nella sezione relativa alla descrizione delle prove sperimentali eseguite per la selezione di Xen; forniamo di seguito alcuni dettagli relativi invece agli atri strumenti utilizzati.

Ethereal e tcpdump costituiscono lo standard de-facto nel contesto degli analizzatori di pacchetto; entrambi gli strumenti consentono di catturare pacchetti dalla rete e visualizzarli in maniera semplice e dettagliata. Utilizzeremo tcpdump per la cattura ed Ethereal per la visualizzazione, in quanto quest'ultimo mette a disposizione una interfaccia molto comoda, che consente tra l'altro di visualizzare i campi che compongono l'header di livello MPLS.

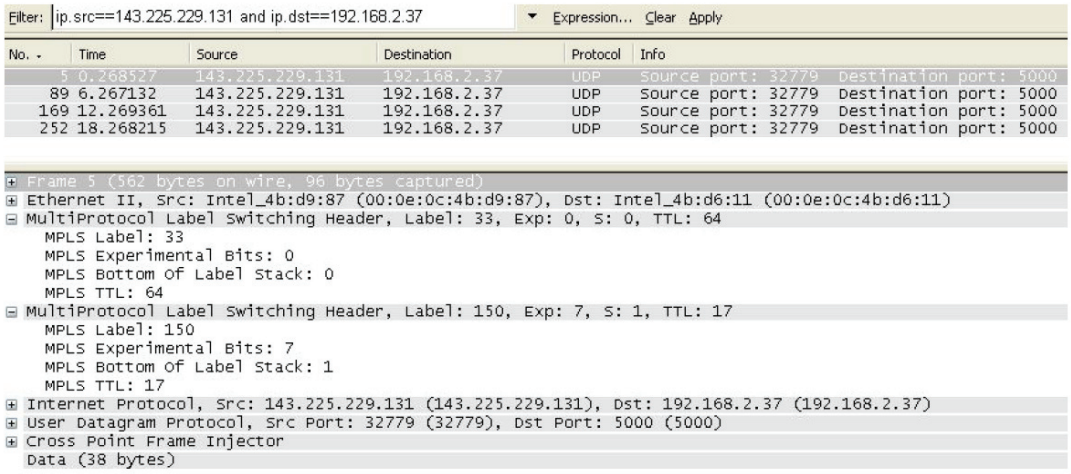

Figura 51 - L'interfaccia grafica dell'analizzatore di traffic Ethereal

OProfile è uno strumento disponibile in Linux che consente di eseguire alcune misure per la valutazione delle prestazioni del sistema. OProfile utilizza l'hardware del processore per monitorare tutte le parti in esecuzione sul sistema, a partire dal kernel fino ad arrivare alle applicazioni utente. Possiede la capacità di eseguire delle analisi in relazione a differenti tipologie di eventi di sistema. Per le nostre misure abbiamo utilizzato in particolare la componente CPU performance counter; il Sistema Operativo invia degli interrupt alla CPU ad intervalli regolari (time slice), ed il demone OProfile memorizza l'istruzione in esecuzione in quell'istante, operazione che comporta ovviamente un piccolo overhead. Utilizzando le informazioni di debug, OProfile associa gli execution point memorizzati con la routine o il codice sorgente del quale fa parte.

L'obiettivo degli esperimenti eseguiti è quello di valutare l'overhead computazionale introdotto dal meccanismo che proponiamo, e confrontarlo con quello introdotto dalla versione originale di MPLS-Linux nel caso in cui il traffico venga inoltrato su un LSP singolo.

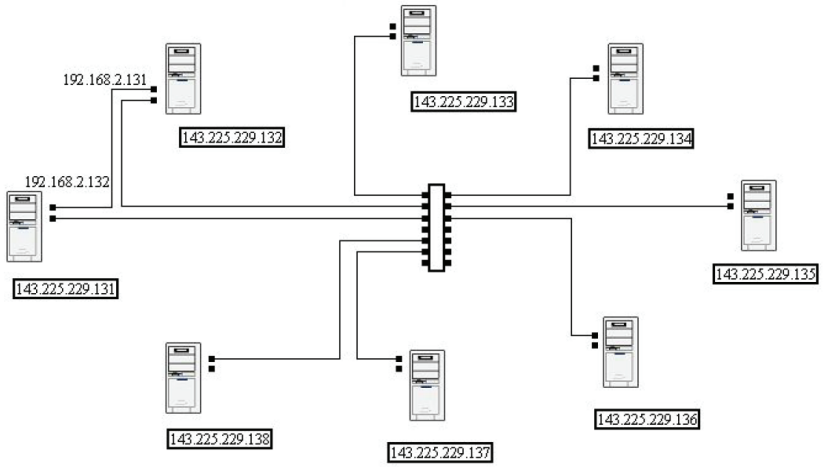

Figura 52 – Test su MPLS Splitting e MPLS Multicast: la rete utilizzata per gli esperimenti

Tutti gli esperimenti sono stati eseguiti utilizzando la medesima configurazione di rete, costituita da:

- Sender: utilizzando D-ITG genera il traffico di rete indirizzato a differenti destinazioni;
- Ingress LER: esegue lo splitting del traffico generato dal Sender in accordo a politiche prestabilite;
- Router MPLS (LSR): simulano il backbone di una rete;
- Egress LER: raccoglie il traffico e lo reinoltra verso un host specifico;
- Host destinazione (non rappresentato nell'immagine).

I diversi scenari implementativi realizzati, differiscono tra loro in funzione della policy applicata sull'ingress LER.

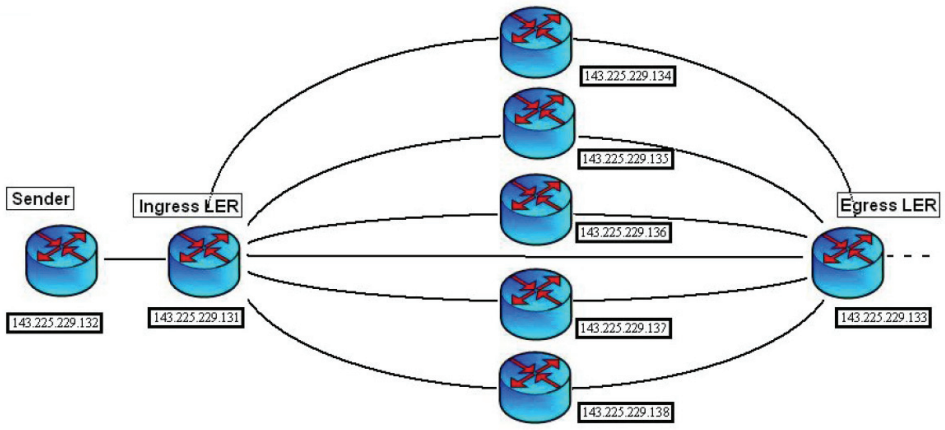

Figura 53 - Test su MPLS Splitting e MPLS Multicast: la rete utilizzata per eseguire gli esperimenti

Le policy associano percorsi differenti a flussi differenti; identifichiamo il flusso in funzione della particolare coppia sorgente/destinazione. Indichiamo di seguito le varie configurazioni implementate sull'Ingress LER:

- LS (Label Switch): tutto il traffico che appartiene al flusso viene inoltrato sul medesimo percorso virtuale (LSP). Si tratta del meccanismo di forwarding tradizionale offerto dall'implementazioen MPLS-Linux, per cui verrà utilizzato come termine di paragone per misurare l'overhead addizionale introdotto dal nostro codice.

- RR (Round Robin): il traffico appartenente ad un flusso viene inoltrato su tutti i link di uscita che connettono l'Ingress LER all'Egress LER, sfruttando una politica di tipo Round Robin. L'Egress LER colleziona i pacchetti, ma non esegue il reordering, quindi non ricostruisce il flusso originale, rilasciando tale compito ai protocolli di livello trasporto (TCP).

- LRR (label round robin): anche in questo caso il traffico che appartiene ad un flusso viene inoltrato su tutti i link che collegano l'Ingress LER all'Egress LER, con la significativa differenza che stavolta viene introdotta una etichetta MPLS all'interno di ogni pacchetto, contenente informazioni necessarie ad eseguire il reordering in corrispondenza dell'Egress LER.

- MCAST (Multicast): il generico pacchetto viene inoltrato su tutti i link di uscita; l'Egress LER non esegue alcun meccanismo di riggetto di pacchetti duplicati.

I risultati sperimentali sono raggruppati sulla base del numero di flussi generati dal Sender.

8 flussi

In Tabella 13 sono riportati i risultati ottenuti in riferimento ad un intervallo di 20 minuti.

Le funzioni MPLS sono riportate nella colonna "functions", e nelle altre colonne sono riportate invece informazioni relative al tempo di esecuzione di ognuna delle funzioni in riferimento all'intervallo di tempo considerato. Com'era prevedibile, i grafici mostrano

che il Label Switching è il meccanismo meno costoso in termini di risorse CPU impegnate. La policy RR è meno costosa rispetto alla policy LRR, in quanto quest'ultima richiede il processing di una etichetta aggiuntiva. La policy MCAST è più pesante rispetto alle altre, in quanto implementa la duplicazione dei pacchetti ed il forwarding su percorsi multipli.

12 e 16 flussi

Riportiamo nelle Tabelle 14 e 15 i risultati ottenuti.

Non ci sono differenze significative rispetto al caso precedente.

8 FLOWS				
label-switching	policy RR	policy LRR	policy MCAST	functions
0.0169	0.0167	0.0162	0.0277	mpls_output2
0.0047	0.0054	0.0047	0.0072	mpls_send
0.0113	0.0075	0.0073	0.0112	mpls_push
0.0044	0.0038	0.0050	0.0036	mpls_output
0.0043	0.0057	0.0056	0.0085	mpls_out_op_set
0.0047	0.0049	0.0048	0.0028	mpls_output_shim
0.0033	0.0024	0.0037	0.0016	mpls_proto_find_by_ethertype
9.5e-06	9.6e-4	7.9e-4	0.0016	mpls_op_push
8.3e-06	3.0e-4	7.2e-4	9.5e-5	mpls4_get_ttl
0	0.0024	0.0024	0.0011	mpls_op_mfwd
0	0	0.0057	0	mpls_ipush
0	0	0	0.0020	mpls_multicast
0	0	0	0.0290	pskb_copy
0	0.0095	0.0099	0.0064	mpls_policy_get_nhlfe_list
0	0	0.0039	0	split_label_round_robin
0	0.0038	0	0	split_round_robin
0.05322	0.064334	0.07071	0.102842	TOT

Tabella 13 - Test su MPLS Splitting e MPLS Multicast: 8 flussi di traffico

12 FLOWS				
label-switching	policy RR	policy LRR	policy MCAST	functions
0.0122	0.0169	0.0164	0.0293	mpls_output2
0.0056	0.0063	0.0055	0.0027	mpls_send
0.0073	0.0079	0.0071	0.0116	mpls_push
0.0040	0.0044	0.0045	0.0046	mpls_output
0.0049	0.0049	0.0053	0.0078	mpls_out_op_set
0.0040	0.0063	0.0068	0.0058	mpls_output_shim
0.0022	0.0030	0.0028	0.0018	mpls_proto_find_by_ethertype
9.0e-04	9.7e-4	6.2e-4	0.0016	mpls_op_push
1.2e-04	3.2e-4	4.0e-4	4.1e-5	mpls4_get_ttl
0	0.0024	0.0023	0.0021	mpls_op_mfwd
0	0	0.0051	0	mpls_ipush
0	0	0	0.0020	mpls_multicast
0	0	0	0.0315	pskb_copy
0	0.0104	0.0093	0.0064	mpls_policy_get_nhlfe_list
0	0	0.0063	0	split_label_round_robin
0	0.0057	0	0	split_round_robin
0.05426	0.06829	0.07147	0.10722	TOT

Tabella 14 - Test su MPLS Splitting e MPLS Multicast: 12 flussi di traffico

16 FLOWS				
label-switching	policy RR	policy LRR	policy MCAST	functions
0.0193	0.0252	0.0248	0.0429	mpls_output2
0.0083	0.0044	0.0079	0.0102	mpls_send
0.0135	0.0108	0.0124	0.0146	mpls_push
0.0077	0.0062	0.0079	0.0048	mpls_output
0.0090	0.0073	0.0080	0.0137	mpls_out_op_set
0.0065	0.0075	0.0081	0.0049	mpls_output_shim
0.0040	0.0038	0.0047	0.0034	mpls_proto_find_by_ethertype
0.0025	0.0012	0.0025	0.0002	mpls_op_push
2.8e-04	1.7e-4	6.2e-4	3.4e-4	mpls4_get_ttl
0	0.0028	0.0034	0.0021	mpls_op_mfwd
0	0	0.0102	0	mpls_ipush
0	0	0	0.0032	mpls_multicast
0	0	0	0.0432	pskb_copy
0	0.0119	0.0108	0.0075	mpls_policy_get_nhlfe_list
0	0	0.0063	0	split_label_round_robin
0	0.0053	0	0	split_round_robin
0.07108	0.08647	0.10702	0.15274	TOT

Tabella 15 - Test su MPLS Splitting e MPLS Multicast: 16 flussi di traffico

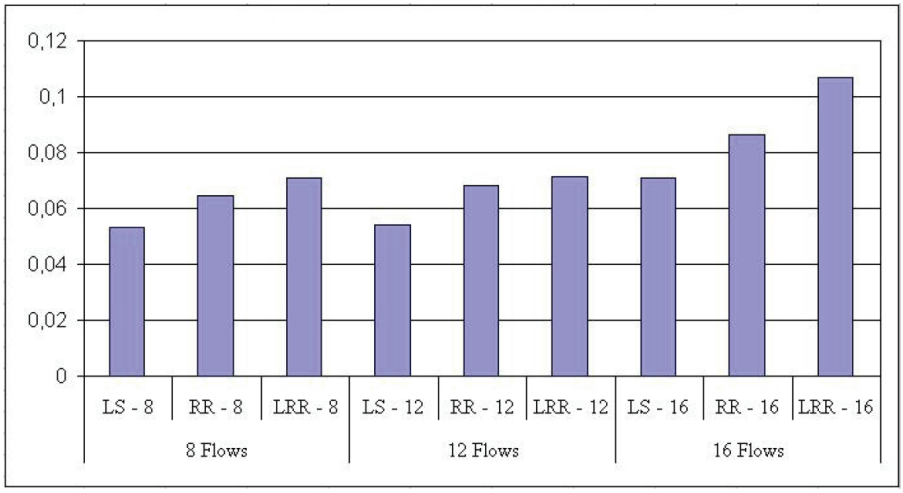

Figura 54 - Test su MPLS Splitting e MPLS Multicast: grafico Splitting

Dal diagrama riportato in Figura 54, è possibile notare che la policy LRR comporta un overhead pari al 100%-120% in più rispetto alla policy LS; il prezzo da pagare è comunque giustificato dai benefici legati all'utilizzo di tale policy.

I risultati delle misure eseguite in riferimento alla policy MCAST, sono state riportate in un diagramma a parte, in quanto l'overhead relativo a tale policy è molto più elevato rispetto a quello prodotto dalle altre.

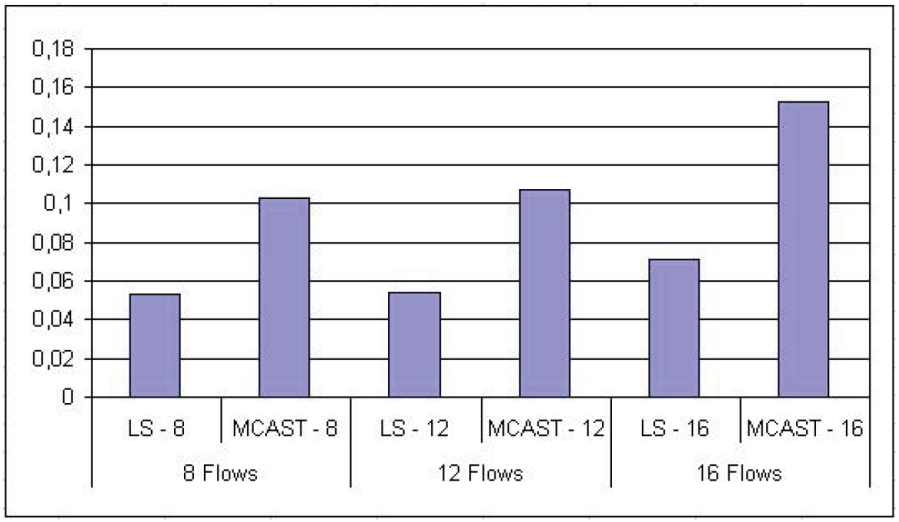

Figura 55 - Test su MPLS Splitting e MPLS Multicast: grafico Multicast

5.4.2 Valutazione delle prestazioni offerte dal meccanismo di Splitting

Occupiamoci adesso di valutare le prestazioni offerte dai meccanismi proposti; ci limiteremo in particolare a considerare la policy Round Robin, quindi lo Splitting senza la ricostruzione del flusso originale.

Le prove sperimentali sono state eseguite facendo riferimento ai seguenti parametri:

- Percentuale di pacchetti persi durante la trasmissione (% packet loss);
- Valore medio del ritardo dei pacchetti in ricezione sull'Egress LER (average delay);
- Valore medio del jitter misurato sull'Egress LER (average jitter).

Le prove sono state eseguite utilizzando dei router Linux-MPLS, il generatore di traffico D-ITG, e l'applicazione Traffic Control (TC) del pacchetto iproute2. Quest'ultimo strumento consente di configurare la quantità di banda residua su un link, con l'obiettivo di simulare una congestione del canale. Per i nostri esperimenti abbiamo sfruttato in particolare la modalità qdisc, mediante la quale è possibile associare ad ogni interfaccia di rete una disciplina di accodamento che implementa uno scheduling dei

pacchetti in uscita. TC consente di impostare la dimensione massima della coda, il rate di uscita e la latenza dei pacchetti accodati.

Il testbed utilizzato è costituito da un Sender, un Ingress LER che implementa le policy Round Robin e Label Switching, due router MPLS-Linux intermedi, ed un Egress LER. La topologia implementata è rappresentata in Figura 56.

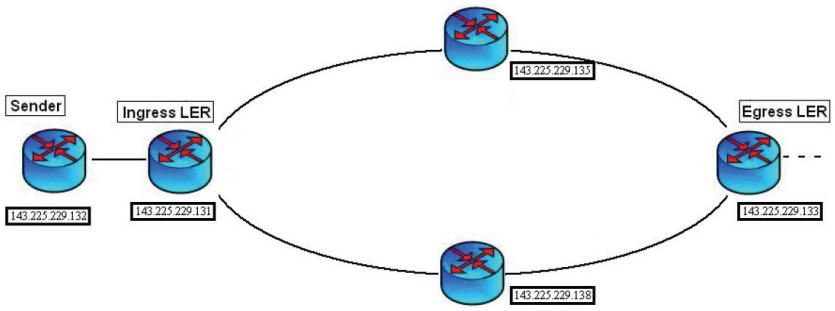

Figura 56 – Valutazione prestazioni Splitting: topologia implementata

Abbiamo suddiviso la sessione sperimentale in 4 prove successive, che presentano come obiettivo la valutazione delle prestazioni offerte dalla policy Round Robin nel confronto con la policy Label Switching, facendo riferimento ogni volta ad una particolare tipologia di traffico e a determinate le caratteristiche dei link che costituiscono la topologia.

Riportiamo di seguito un riepilogo dei risultati ottenuti dalle prove sperimentali.

Test 1

Obiettivo: Valutazione delle prestazioni in termini di percentuale di pacchetti persi, ritardo medio e jitter medio.

Parametri relativi alla configurazione dei link di uscita dell'ingress LER:

- rate: 2Mbit/sec
- latency: 100ms
- burst: 400Kbyte

Il traffico generato è composto da 8 flussi UDP con le seguenti caratteristiche:

- rate: 100pkt/sec
- size pkt: 512byte/pkt

Riportiamo i valori ottenuti dalle misurazioni nelle Tabelle 10 e 11, e in Figura 57 i confronti in funzione dei parametri sopra elencati.

Durata	% packet loss	Avg Delay	Avg Jitter
10sec	24,56%	976ms	6,3ms
1min	40,27%	1530ms	3,8ms
5min	42,76%	1648ms	4,0ms
10min	43,07%	1664ms	3,4ms
20min	43,24%	1673ms	5,7ms

Tabella 16 - Valutazione prestazioni Splitting, Test 1: policy Label Switching

Durata	% packet loss	Avg Delay	Avg Jitter
10sec	0%	2,5ms	1,2ms
1min	0%	2,3ms	1,0ms
5min	0%	2,4ms	1,1ms
10min	0%	2,6ms	1,2ms
20min	0,01%	11,6ms	3,0ms

Tabella 17 - Valutazione prestazioni Splitting, Test 1: policy Round Robin

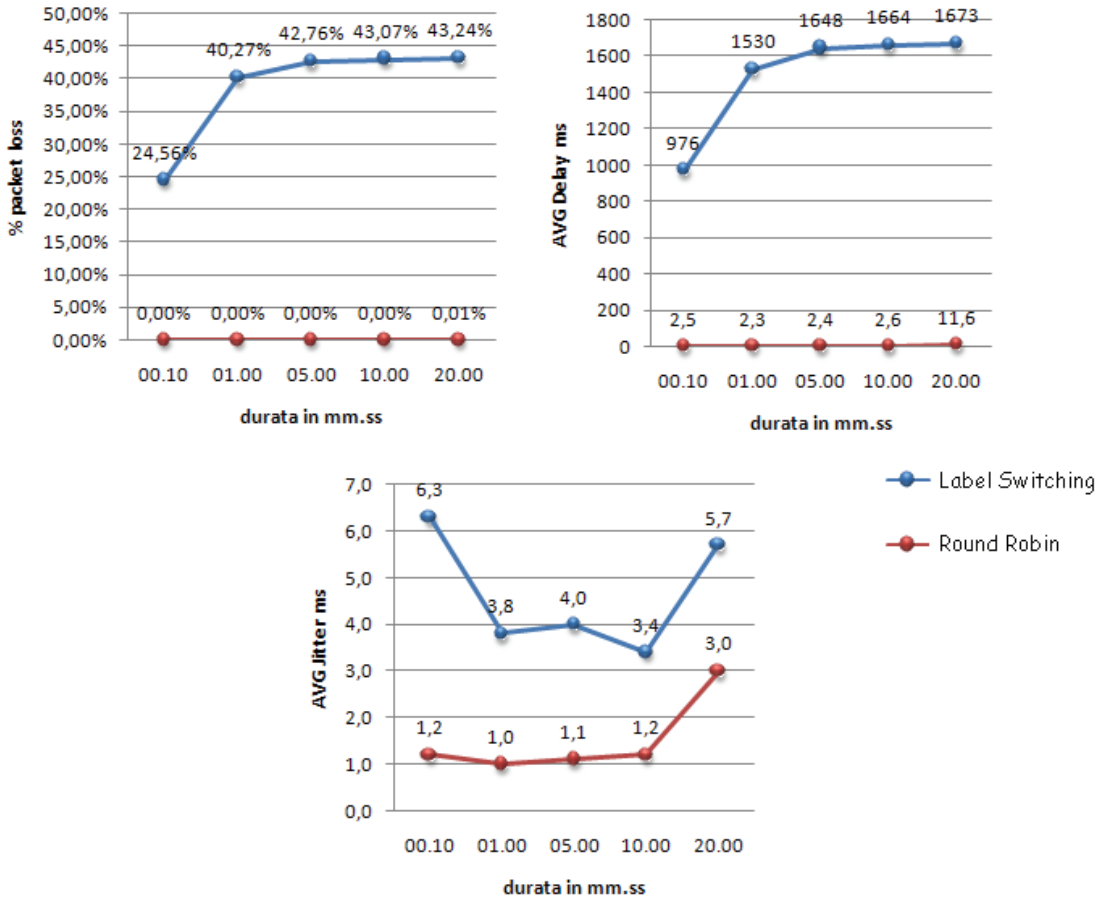

Figura 57 - Valutazione prestazioni Splitting, Test 1: % pacchetti persi, ritardo medio, valore medio del jitter

Con lo Splitting si ottiene l'abbattimento della percentuale di pacchetti persi e del ritardo medio, ed una forte riduzione del valore medio del jitter.

Test 2

Obiettivo: valutazione del guadagno in termini di numero di flussi supportati.

Le condizioni sono le stesse rispetto a quelle del test precedente, sia in termini di configurazione dei link, che in termini di caratteristiche dei flussi di traffico generati.

Le misure sono state eseguite al variare del numero di flussi inoltrati da sorgente a destinazione.

Numero flussi	% packet loss	Avg Delay	Avg Jitter
5	0%	965ms	20ms
6	0,01%	1181ms	25ms
7	1,92%	1392ms	29ms
8	43,07%	1664ms	3,4ms
9	49,40%	1665ms	3,3ms

Tabella 18 - Valutazione prestazioni Splitting, Test 2: policy Label Switching

Numero flussi	% packet loss	Avg Delay	Avg Jitter
8	0%	2,6ms	1,2ms
9	0,02%	9,3ms	3,0ms
10	8,93%	1610ms	5,4ms
11	17,21%	1645ms	5,7ms
12	24,11%	1657ms	6,8ms

Tabella 19 - Valutazione prestazioni Splitting, Test 2: policy Round Robin

È facile verificare che utilizzando un singolo LSP, si ottiene a partire da 5 flussi di traffico una percentuale di pacchetti persi non nulla. La soglia sale invece a 8 nel caso invece in cui venga utilizzato lo Splitting.

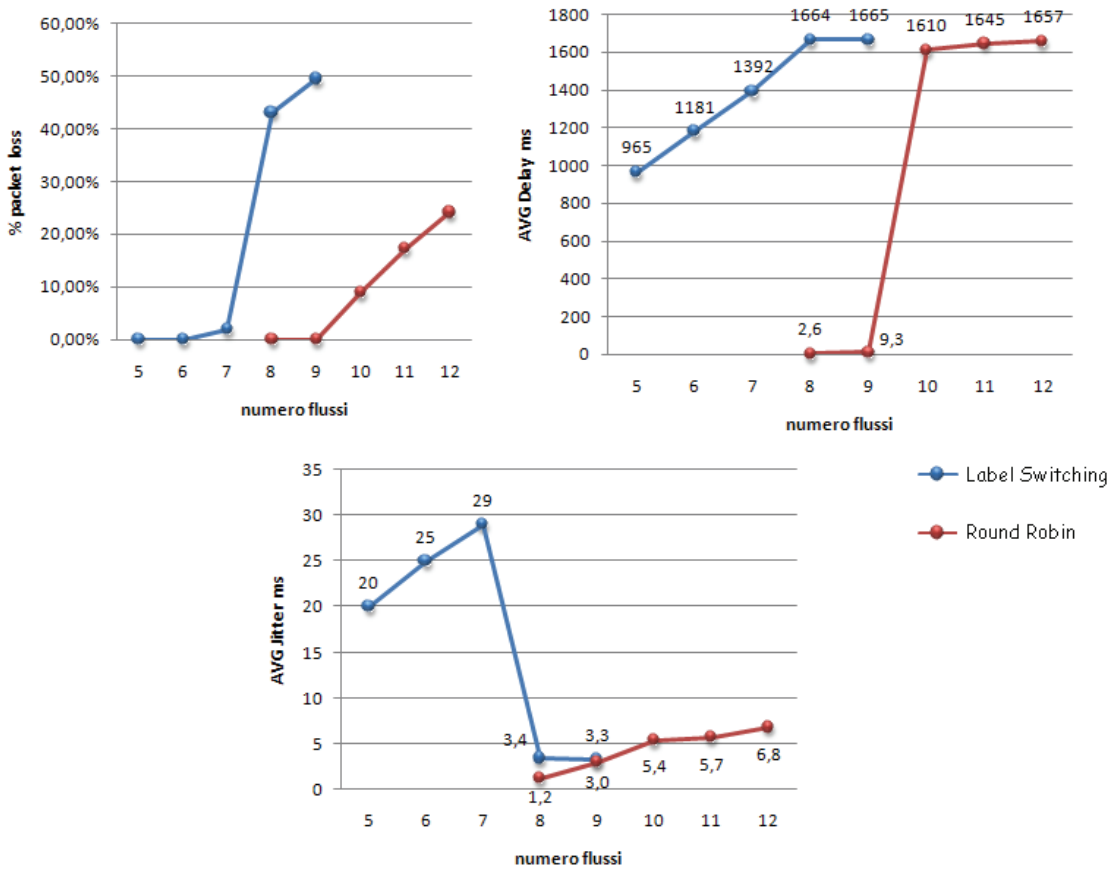

Figura 58 - Valutazione prestazioni Splitting, Test 2: % pacchetti persi, ritardo medio, valore medio del jitter

Nel caso in cui venga utilizzato un singolo LSP, il valore medio del jitter diminuisce in maniera brusca nel passaggio da 7 a 8 flussi di traffico perché in corrispondenza di tale transizione si verifica un forte incremento della percentuale di pacchetti persi, e quindi un forte decremento dei pacchetti in relazione ai quali viene misurato il jitter

Test 3

Obiettivo: come al test 1.

Verrà generato un singolo flusso di traffico VoIP con codec G.7.11.2 e due campioni per pacchetto; la congestione dei link è simulata configurando l'interfaccia di uscita dell'Ingress LER con i seguenti parametri:

- rate: 50Kbit/sec;
- latency: 200ms
- burst: 5600byte

La scelta di tali valori è stata effettuata in base alle caratteristiche ed ai requisiti relativi al traffico VoIP; in particolare, il rate è stato impostato a 50Kbit/sec per fare in modo che il flusso di traffico in ingresso trovasse una rete già congestionata, e la latenza a 200ms in modo che coincidesse col ritardo massimo consentito per traffico di questo tipo.

Per facilitare la lettura dei risultati ottenuti, riassumiamo brevemente i requisiti minimi da garantire per traffico VoIP:

- % packet loss: < 1%;
- avg delay: < 200ms;
- avg jitter: < 20ms;

Durata	% packet loss	Avg Delay	Avg Jitter
10sec	2,80%	395ms	3,1ms
1min	15,33%	914ms	6,1ms
5min	17,35%	1012ms	6,65ms
10min	17,60%	1024ms	6,72ms
20min	17,71%	1031ms	6,76ms
30min	17,76%	1033ms	6,78ms

Tabella 20 - Valutazione prestazioni Splitting, Test 3: policy Label Switching

Durata	% packet loss	Avg Delay	Avg Jitter
10sec	0%	0,89ms	0,29ms
1min	0%	0,90ms	0,26ms
5min	0%	0,82ms	0,21ms
10min	0%	0,84ms	0,22ms
20min	0%	0,93ms	0,26ms
30min	0%	0,96ms	0,37ms

Tabella 21 - Valutazione prestazioni Splitting, Test 3: policy Round Robin

I grafici riportati in Figura 59, mettono in evidenza il fatto che in ciascuno dei casi considerati, a differenza di quelli in cui venga usato un singolo LSP, l'utilizzo dello Splitting consente di verificare i requisiti richiesti dal VoIP.

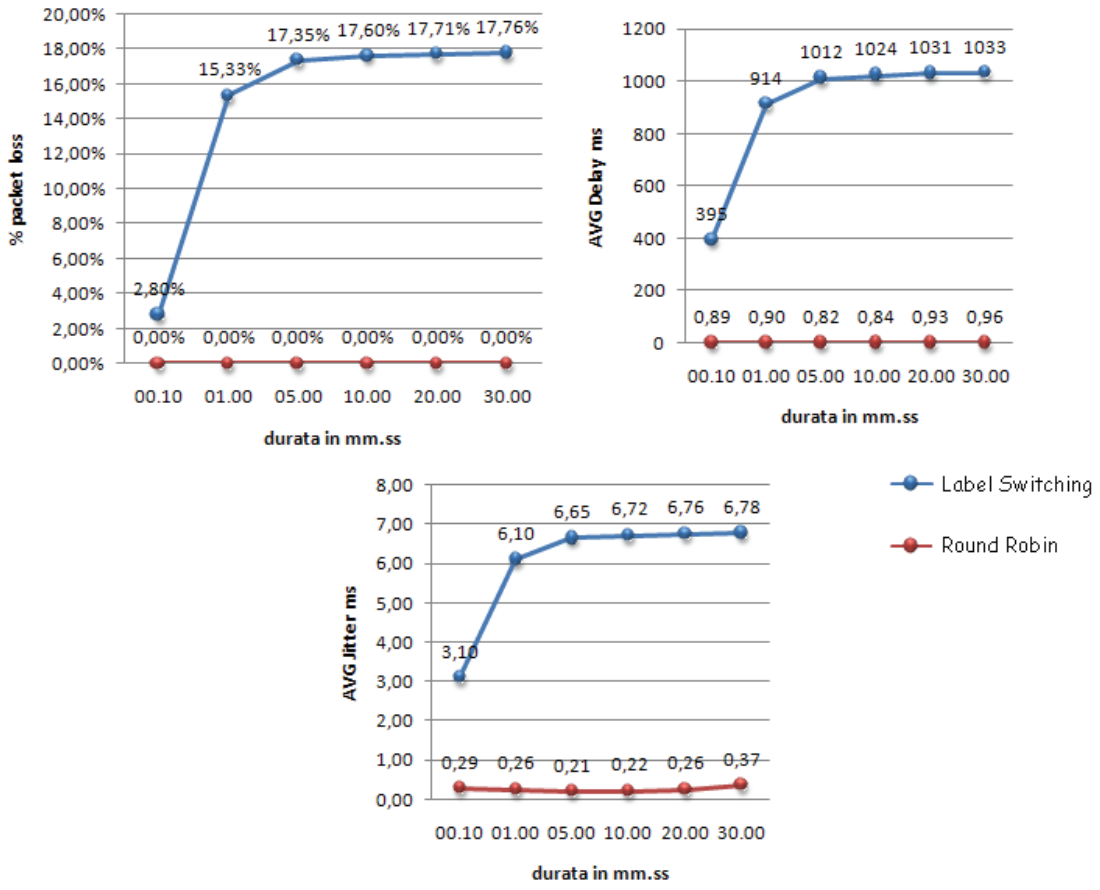

Figura 59 - Valutazione prestazioni Splitting, Test 3: % pacchetti persi, ritardo medio, valore medio del jitter

Test 4

Obiettivo: come al test 1.

Utilizziamo stavolta traffico misto, quindi flussi con le seguenti caratteristiche:

- 1 flusso UDP: 1000pkt/sec, 512byte/pkt;
- 1 flusso UDP: 500pkt/sec, dimensione dei pacchetti modellabile come variabile aleatoria uniforme appartenente all'intervallo [500, 1000];
- 2 flussi VoIP: il primo con codec G.711.1 (un campione per pacchetto) ed il secondo con codec G.711.2 (due campioni per pacchetto);

- 1 flusso TCP: 1000pky/sec, di 512byte/pkt;
- 1 flusso TCP: 500pkt/sec, dimensione dei pacchetti modellabile come variabile aleatoria esponenziale con valore medio pari a 200 byte.

Durata	% packet loss	Avg Delay	Avg Jitter
10sec	9,25%	177ms	1,9ms
1min	17,60%	206ms	2,2ms
5min	13,50%	190ms	1,8ms
10min	15,79%	212ms	2,2ms
20min	14,05%	191ms	1,9ms
40min	13,51%	121ms	2,1ms
1ora	14,99%	173ms	2,5ms

Tabella 22 - Valutazione prestazioni Splitting, Test 4: policy Label Switching

Durata	% packet loss	Avg Delay	Avg Jitter
10sec	0%	3,5ms	2ms
1min	0%	3,1ms	1,8ms
5min	0%	3,2ms	1,9ms
10min	0%	3,2ms	1,8ms
20min	0%	4,3ms	2,3ms
40min	0%	4,0ms	2,3ms
1ora	0%	3,6ms	2,2ms

Tabella 23 - Valutazione prestazioni Splitting, Test 4: policy Round Robin

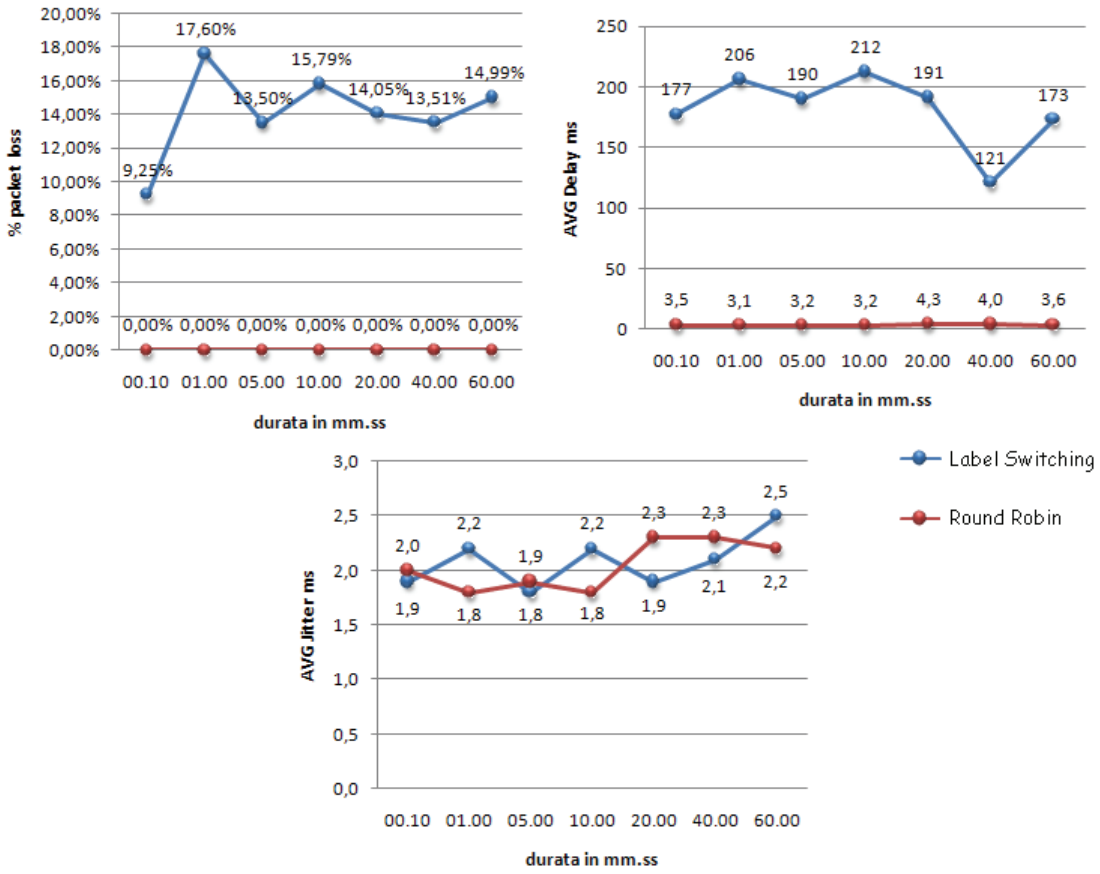

Figura 60 - Valutazione prestazioni Splitting, Test 4: % pacchetti persi, ritardo medio, valore medio del jitter

Capitolo 6
Conclusioni e sviluppi futuri

Nel lavoro di tesi presentato, ci siamo occupati della definizione di un paradigma innovativo per moderni servizi di rete, il Service Switching, un modello di comunicazione alla cui base ritroviamo il concetto di mobilità dei servizi, il quale ricopre un ruolo di rilevante importanza nelle reti di nuova generazione.

Nel contesto dell'ubiquitous computing, ad esempio, in cui utenti nomadi si spostano all'interno della rete muovendosi tra ambienti eterogenei, il supporto alla mobilità di servizio aiuta ad incrementare la qualità percepita nella fruizione del servizio stesso. In piattaforme per l'hosting di servizi, invece, è di fondamentale importanza curare aspetti legati alla gestione efficiente delle risorse disponibili, ed implementare meccanismi finalizzati al bilanciamento del carico computazionale a beneficio di risorse particolarmente sovraccariche; poter migrare servizi su nodi differenti di una infrastruttura o in generale della rete, consente di raggiungere tali obiettivi. In sistemi di rete per la distribuzione di servizi, è inoltre indispensabile poter fare affidamento su meccanismi che si occupano di rilevare eventuali condizioni critiche e reagire a tali condizioni in modo da garantire continuità nella fornitura dei servizi, quindi un adeguato livello di resilienza. Molto spesso, la migliore soluzione per far fronte a condizioni di questo tipo consiste proprio nell'eseguire un trasferimento dalla zona in cui si è verificato il fallimento ad una zona differente dell'infrastruttura, di un intero insieme di servizi e del relativo stato di esecuzione.

Il Service Switching è un modello per la gestione di servizi di rete innovativi, che offre la possibilità ad un Application Service Provider (ASP) di fornire servizi ai proprio

utenti sfruttando le risorse messe a disposizione dal sistema, ed un supporto alla mobilità di servizio che ne rende trasparente la posizione corrente all'interno di una architettura geograficamente distribuita.

La principale componente del modello proposto è il Service Switch, un nodo di rete che oltre ad implementare i meccanismi tradizionali di packet e flow switching, possiede caratteristiche innovative, tra le quali la possibilità ospitare servizi ed applicazioni assieme ai rispettivi dati, fornendo loro un ambiente di esecuzione sicuro, isolato, e differenziato. Il Service Switch supporta la migrazione di servizio tra nodi geograficamente distribuiti, ed implementa tale operazione in maniera completamente trasparente nei riguardi dell'utente e dell'applicazione stessa. Ogni nodo coordina il proprio comportamento considerando uno scenario globale e le azione compiute dagli altri Service Switch, con l'obiettivo di garantire un uso ottimale delle risorse disponibili e di bilanciare il carico computazionale sull'intera architettura. È possibile localizzare un nodo Service Switch nel cuore della rete oppure sulla sua frontiera, e a seconda dei casi implementerà un preciso insieme di funzionalità. L'intera rete di Service Switch è in grado di rilevare eventuali condizioni di malfunzionamento, e di reagire a tali condizioni avviando un meccanismo di riattivazione dei servizi.

Abbiamo proposto una particolare implementazione per il modello Service Switching che fa uso delle tecniche di virtualizzazione, e che offre un meccanismo di instradamento ispirato al modello Mobile IP per garantire la corretta consegna di pacchetti destinati a servizi migrati. Tale implementazione sfrutta inoltre le potenzialità del protocollo MPLS e di alcuni meccanismi di forwarding innovativi opportunamente realizzati, ed è quindi in grado di garantire che le comunicazioni tra i nodi che compongono la rete di Service Switch, avvengano attraverso l'uso di canali sicuri ed affidabili.

Nella parte conclusiva della tesi abbiamo poi riportato e documentato i risultati ottenuti da alcune sessioni sperimentali eseguite con l'obiettivo di valutare il modello Service Switching e l'architettura che lo implementa, in riferimento ad un particolare caso d'uso, il caching e la distribuzione di contenuti web a comunità di utenti.

È nostra intenzione per il prossimo futuro continuare a lavorare su diversi aspetti legati al lavoro svolto e al modello che proponiamo, con l'intenzione di estendere ulteriormente le potenzialità dell'architettura Service Switching; uno dei temi sul quale intendiamo continuare a lavorare è quello della migrazione di servizio. In letteratura sono stati di recente introdotti alcuni modelli per la localizzazione ottimale di servizi in reti di larga scala. Tali modelli propongono l'impiego di politiche per la mobilità dei servizi basate su informazioni globali anziché locali. Utilizzando un approccio di tipo step-by-step, tali meccanismi muovono i servizi verso la posizione ottimale sfruttando la migliore traiettoria di migrazione, con l'obiettivo di avvicinare il punto di fornitura del servizio alla domanda, in modo da minimizzare l'uso delle risorse di comunicazione e di migliorare la qualità del servizio per i servizi offerti. I meccanismi in questione sono di tipo adattativo, nel senso che supportano la riconfigurazione dello schema di localizzazione dei servizi in funzione di variazioni alla topologia, oppure in funzione dell'intensità della domanda. Il meccanismo di migrazione proposto col nostro modello, si basa invece su informazioni relative alle risorse disponibili all'interno dell'infrastruttura, ed è finalizzato al bilanciamento del carico computazionale su quest'ultima. È nostra intenzione estendere il meccanismo di migrazione supportato dal Service Switching, con l'obiettivo di integrare le politiche appena introdotte con quelle adottate nel nostro modello, ed implementare quindi la mobilità dei servizi sia per migliorare le prestazioni percepite dall'utente, che per ottimizzare l'uso delle risorse che il sistema mette a disposizione.

Un altro importante aspetto in relazione al quale abbiamo intenzione di estendere la nostra architettura, riguarda la gestione di condizioni critiche quali ad esempio il crash di un nodo Edge Service Switch. Il meccanismo attualmente implementato a seguito del rilevamento di una condizione di questo tipo, consiste nell'avviare una procedura di riattivazione dei servizi ospitati dal Virtual Execution Environment gestito dal nodo vittima del fallimento, meccanismo basato sulla disponibilità di copie dell'immagine delle macchie virtuali da riattivare. Sebbene tali immagini vengano rinfrescate periodicamente, implementare questo tipo di procedura non consente di salvare lo stato dei servizi prima del fallimento. Una alternativa da considerare in tali circostanze,

potrebbe essere quella di replicare i servizi e quindi redirigere, se necessario, le richieste verso dei server replica evitando di perderne lo stato; in tal caso ci sarebbe però un prezzo da pagare, quello di dover impiegare una quantità di risorse maggiore rispetto a quella strettamente necessaria. La nostra intenzione è quella di eseguire delle simulazioni che ci consentano di valutare i risultati ottenibili da tale alternativa, nel confronto con la soluzione adottata dal nostro modello; potrebbe infatti risultare opportuno supportare entrambi i meccanismi, ed attivarne uno più tosto che l'altro in funzione della particolare tipologia di servizio che l'architettura deve ospitare.

In riferimento alla rete CDN introdotta al Capitolo V, e relativamente alla quale sono state eseguite le prove sperimentali sull'architettura Service Switching, stiamo attualmente lavorando alla realizzazione di una nuova infrastruttura per il caching e la distribuzione di contenuti web, basata però su tecnologie peer-to-peer. Tale infrastruttura introduce rispetto alla precedente un ulteriore livello di caching al fine di ottimizzare ulteriormente le prestazioni del sistema, ed implementa le funzionalità dei componenti Proxy e CDN-Coordinator in maniera distribuita sui vari nodi che la compongono. A seguito della realizzazione di tale infrastruttura, la nostra intenzione è quella di eseguire delle sperimentazioni ospitando le funzionalità implementate dalle componenti di quest'ultima, sui Service Switch della nostra piattaforma, i quali saranno quindi organizzati secondo le regole dei protocolli peer-to-peer. Lo scopo di tale sperimentazione è quello di confrontare il supporto alla mobilità offerto dalla nostra architettura, con le prestazioni ottenibili invece dai sistemi peer-to-peer self-organized [22, 23].

Ancora, ci occuperemo di determinare un criterio secondo cui stabilire il posizionamento ottimale dei Core Service Switch all'interno della rete; per il momento possiamo in ogni caso affermare che la funzionalità svolta da tali nodi e tanto più performante quanto più siano vicini ad un'area da cui vengono inoltrati datagrammi destinati alle VM ospitate dall'architettura.

Infine, abbiamo intenzione di estendere l'uso dei dispositivi di storage condiviso per ottimizzare il trasferimento dell'immagine della macchina virtuale durante il processo di

migrazione; l'idea è quella di sfruttare la copia disponibile sul NAS e trasferirla sul nodo destinazione a seguito di un opportuno aggiornamento. Anche in relazione a tale aspetto, la nostra intenzione è quella di eseguire delle simulazioni che ci aiutino a determinare la migliore soluzione da implementare.

Bibliografia

[1] M. Nelson, B. Lim, G. Hutchins, “Fast Transparent Migration for Virtual Machines”, in Proceedings of USENIX 2005.

[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, “Live Migration of Virtual Machines”, in Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, May 2005.

[3] J. Hansen, E. Jul, ”Self-migration of operating systems”, in Proceedings of the 11th ACM SIGOPS European Workshop (EW 2004), pages 126–130, 2004.

[4] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, T. Schooley, “Evaluating Xen for Router Virtualization”, in proceedings of 16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007.

[5] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, “Enforcing performance isolation across virtual machines in Xen”, in Proceedings of the ACM/IFIP/USENIX 7th International Middleware Conference, Melbourne, Australia, volume 4290 of Lecture Notes in Computer Science, pages 342– 362. Springer, November 2006.

[6] P.M. Chen, B.D. Noble, “When Virtual is Better Than Real”, in Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems, Schloss Elmau, Germany, May 2001.

[7] M. Kozuch, M. Satyanarayanan, “Internet Suspend/Resume”, in Proceeding of 4th IEEE Workshop on Mobile Computing Systems and Applications, Callicoon, NY, June 2002.

[8] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, A. Surie, D. R. O'Hallaron, A. Wolbach, J. Harkes, A. Perrig, D. J. Farber, M. A. Kozuch, C. J. Helfrich, P. Nath, H. A. Lagar-Cavilla, "Pervasive Personal Computing in an Internet Suspend/Resume System", IEEE Internet Computing, 11(2), 2007.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization", in Proceedings of 19th ACM Symposium on Operating Systems Principles, SOSP 2003.

[10] P. Willmann, J. Shafer, D. Carr, A. Menon, S. Rixner, A. L. Cox, W. Zwaenepoel, "Concurrent Direct Network Access for Virtual Machine Monitors", in Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'07), Feb. 2007.

[11] L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization", in Proceedings of ACM Workshop on Hot Topics in Networks (HotNets), 2004.

[12] X. Jiang, D. Xu, "SODA: a Service-On-Demand Architecture for Application Service Hosting Utility Platforms", IEEE International Symposium on High Performance Distributed Computing (HPDC-12), June 2003.

[13] A. Snoeren, H. Balakrishnan, "An end-to-end approach to host mobility", in Proceedings of the 6th annual international conference on Mobile computing and networking, pages 155.166. ACM Press, 2000.

[14] X. Jiang, D. Xu, "vBET: a VM-Based Emulation Testbed", in Proceedings of the ACM SIGCOMM 2003 Workshops, (Karlsruhe, Germany, August), 95–104.

[15] P. Ruth, J. Rhee, D. Xu, R. Kennell, S. Goasguen, "Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure", IEEE Int'l Conf. on Autonomic Computing (ICAC'06), June 2006.

[16] M. McNett, D. Gupta, A. Vahdat, G. M. Voelker, "Usher: An Extensible Framework for Managing Clusters of Virtual Machines", in Proceedings of the USENIX Large Installation System Administration Conference (LISA), 2007.

[17] J. Chase, L. Grit, D. Irwin, J. Moore, S. Sprenkle, “Dynamic Virtual Clusters in a Grid Site Manager”, Accepted to the 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.

[18] IBM, RFC 2002: “IP Mobility Support”, Internet Engineering Task Force, Network Working Group, category Standard Track, editor C. Perkins, October 1996.

[19] K.Oikonomou, I.Stavrakakis, ”Scalable Service Migration: The Tree Topology Case”, The Fifth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2006), Lipari, Italy, June 14-17, 2006.

[20] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, “Distributed Placement of Service Facilities in Large-Scale Networks”, in 26th Annual IEEE Conf. on Computer Communications, Anchorage, USA, May 2007.

[21] K. Oikonomou, I. Stavrakakis, A. Xydias, “Scalable service migration in general topologies”, in World of Wireless, Mobile and Multimedia Networks, 2008, WoWMoM 2008, International Symposium on a, ISBN 978-1-4244-2099-5, pages 1-6, August 2008.

[22] L. Cheng, A. Galis, B. Mathieu, K. Jean, R. Ocampo, L. Mamatas, J. Rubio-Loyola, J. Serrat, A. Berl, H. de Meer, S. Davy, Z. Movahedi, and L. Lefevre, “Self-organising Management Overlays for Future Internet Services”, LNCS: Modelling Autonomic Communications Environment, Proceedings of 3rd IEEE International Workshop on Modelling Autonomic Communications Environments (MACE2008), Samos Island, Greece, September 22-26, 2008. Springer, Berlin (Germany), September 2008.

[23] Amine M. Houyou, Hermann de Meer, “Self-Organizing Location-Aware Overlay Networks”, in Proc. Self-Organizing Systems: Demonstrations and Posters Session, Second Internation Workshop, IWSOS 2007 (David Hutchison, Randy H. Katz (Eds.)) The Lake District, UK, September 11-13, 2007.

[24] M. Brunner, B. Plattner, and Rolf Stadler: “ Service Creation and Management in Active Telecom Networks” . Communications of the, ACM – April 2001.

[25] M. Brunner, R. Stadler, “The Impact of Active Networking Technology on Service Management in a Telecom Environment,” IFIP/IEEE International Symposium on Integrated, Network Management (IM’99), Boston, USA, 1999.

[26] M. Sifalakis, S. Schmid, T. Chart, A.C. Scott, “A Framework for Developing Mobile Network Services”, in Active Networks, book series Lecture Notes in Computer Science, Volume 3912/2007, ISBN 978-3-540-71499-6, pages 126-137, SpringerLink, June 2007.

[27] Turner, J. S., Crowley, P., DeHart, J., Freestone, A., Heller, B., Kuhns, F., Kumar, S., Lockwood, J., Lu, J., Wilson, M., Wiseman, C., and Zar, D., “Supercharging PlanetLab: a high performance, multi-application, overlay network platform”, in SIGCOMM ’07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (Kyoto, Japan, Aug. 2007), pp. 85–96.

[28] Y. Wang, J. van der Merwe, and J. Rexford, “VROOM: Virtual, ROuters On the Move,” in Proc. HotNets, November 2007.

[29] O. Riva, T. Nadeem, C. Borcea and L. Iftode, “Mobile Services: Context-Aware Service Migration in Ad-Hoc Networks”, Rutgers University Technical Report DCS-TR-564.

[30] H. Bharadvaj A. Joshi, and S . Auephanwiriyakul, "An Active Transcoding Proxy to Support Mobile Web Access", Proc. 17th Symp. Reliable Dist. Sys., 1998.

[31] N. Bhatia and J. S. Vetter, “Virtual Cluster Management with Xen”, Euro-Par 2007 Workshops: Parallel Processing, book series Lecture Notes in Computer Science, Volume 4854/2008, ISBN 978-3-540-78472-2, pages 185-194, SpringerLink, March 2008.

[32] J. Xu, M. Zhao, J. Fortes, R. Carpenter, M. Yousif, “Autonomic resources management in virtualised data centers using fuzzy logic-based approaches”, journal Cluster Computing, Volume 11, Number 3, pages 213-227, SpringerLink, September 2008.

[33] M.E. Fiuczynski, “PlanetLab: Overview, history, and future directions”, ACM SIGOPS Operating Systems Review, 40(1):6–10, January 2006.

[34] Foster, C. Kesselman and S. Tuecke, “The anatomy of the Grid: Enabling scalable virtual organizations”, Int. J. High Performance Computing Applications 15(3) (2001) 200–222, http://www.globus.org/ research/papers/anatomy.pdf.

[35] D. Kotz and R. S. Gray, “Mobile agents and the future of the internet”, IEEE Trans. Automat. Contr., vol. AC-28, pp. 1081–1090, December 1983.

[36] A. Bieszczad, B. Pagurek, and T. White, ‘‘Mobile agents for network Management,’’ IEEE Commun. Surv. vol. 1, no. 1, pp. 2 – 9.

[37] Foster et al, “The Physiology of the Grid”, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[38] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, “An integrated experimental environment for distributed systems and networks”, in Proc. of the Fifth Symposium on Operating Systems Design and Implementation, pages 255–270, Boston, MA, December 2002, USENIX Association.

[39] D. Herrscher and K. Rothermel, “A Dynamic Network Scenario Emulation Tool”, in Procs. of the 11th International Conference on Computer Communications and Networks (ICCCN 2002), pages 262–267, Miami, October 2002.

[40] S. Maier, D. Herrscher, and K. Rothermel, “On node virtualization for scalable network emulation”, in Procs. of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 05), Philadelphia, PA, July 24-28, 2005, pages 917–928, July 2005.

[41] V. Darlagiannis, A. Mauthe, and R. Steinmetz, “Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems”, Journal of Networks and System Management, 12(3):371–395, 1 2004.

[42] V. Darlagiannis, A. Mauthe, R. Steinmetz, “Sampling Cluster Endurance for Peer-to-Peer based Content Distribution Networks”, Proceedings of Multimedia

Communication and Networking (MMCN 2006, part of IS&T/SPIE Electronic Imaging 2006 Symposium).

[43] S. Saroiu, K.P. Gummadi, R.J. Dunn, S.D. Gribble, H.M. Levy, “An Analysis of Internet Content Delivery Systems”, in Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002) (Boston, MA, Dec. 2002).

[44] X. Xiao, A. Hannan, B. Bailey, and L. Ni, “Traffic engineering with MPLS in the Internet,” IEEE Network Mag., pp. 28–33, Mar. 2000.

[45] R. Canonico, P. Di Gennaro, V. Manetti, G. Ventre, “Virtualization Techniques in Network Emulation Systems”, in Euro-Par 2007 Workshop: Parallel Processing, book series Lecture Notes in Computer Science, ISBN 978-3-540-78472-2, volume 4854/2008, pages 144-153, Publisher Springer Berlin / Heidelberg, March 08.

[46] R. Canonico, P. Di Gennaro, V. Manetti, G. Ventre, “Network Emulation on Globus-based Grids: mechanisms and challenges”, in Grid Enabled Remote Instrumentation, book series Signals and Communication Technology, ISBN 978-0-387-09662-9, pages 455-468, Publisher Springer, October 08.

[47] J. K. Ousterhout, A. R. Cherenson, F. Douglis, M. N. Nelson, B. B. Welch, “The Sprite network operating system”, Computer Magazine of the Computer Group News of the IEEE Computer Group Society, ACM CR 8905-0314, 21(2), 1988.

[48] A. Barak, O. La'adan, “The MOSIX multicomputer operating system for high performance cluster computing”, Journal of Future Generation Computer Systems, 13(4-5):361.372, March 1998.

[49] J. Tate, F. Lucchese, R.Moore, “Introduction to Storage Area Networks”, IBM RedBooks.

[50] ASP Industry Consortium, White Paper on Service Level Agreements, 2000.

[51] J. R. Smith, R. Mohan, C. S. Li, “Transcoding Internet content for heterogenous client devices”, in Proc. IEEE Inter. Symp. on Circuits and Syst. (ISCAS), June 1998, special session on Next Generation Internet.

[52] A. Vakali, G. Pallis, “Content delivery networks: Status and trends”, IEEE Internet Computing 7, 6 (Nov./Dec. 2003), 68–74.

[53] S. Axelsson, “Intrusion detection systems: A survey and taxonomy”, Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.

[54] V. Darlagiannis, A. Mauthe, R. Steinmetz, “Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems”, Journal of Networks and System Management, 12(3):371–395, 2004.

[55] J. Hill, “An Analysis of the RADIUS Authentication Protocol”, Retrieved from http://www.untruth.org/~josh/security/radius/radius-auth.html, 2001.

[56] B. Goode, “Voice Over Internet Protocol (VoIP)”, Proceedings of the IEEE, 90:1495–1517, September 2002.

[57] H. Bharadvaj, A. Joshi, S. Auephanwiriyakul, “An active transcoding proxy to support mobiIe Web access”, Proc. 17th lEEE Symp. Reliable Distributed Syst., October 1998.

[58] C: Tang, R. N. Chang, E. So, “A distributed Service Management Infrastructure for Enterprise Data Centers Based on Peer-to-Peer Technology”.

[59] Object Management Group, Agent Technology, Green Paper, version 0.92, OMG Agent Working Group, 25. April 2000.

[60] Won-Joo Hwang, “Design and Implementation of Multimedia Service Management Agent on Home Networks Environment”, IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.7B, July 2006.

[61] H. Harroud, “Policy-driven Personalized Multimedia Services for Mobile Users”, IEEE Transactions on Mobile computing, Vol.2, N° 1, October-March 2003.

[62] K.L. Calvert, S. Bhattacharjee, E. Zegura and G. Tech, “Directions in Active Networks”, IEEE Communications Magazine, 1998.

[63] K.L. Calvert, “Architectural Framework for Active Networks Version 1.0”, Active Network Working Group Draft, 1999.

[64] C. Tang, R. N. Chang, and E. So, “A Distributed Service Management Infrastructure for Enterprise Data Centers Based on Peer-to-Peer Technology”, IEEE International Conference on Services Computing (SCC'06).

[65] K. Keahey, I. Foster, T. Freeman, X. Zhang and D. Galron, “Virtual Workspaces in the Grid”, Lecture Notes in Computer Science, Springer, Vol. 3648, Pag. 421-431, August 2005.

[66] K. Keahey, I. Foster, T. Freeman and X. Zhang, “Virtual workspaces: Achieving quality of service and quality of life in the Grid”, Scientific Programming, IOS Press, Vol. 13, Pag. 265-275, 2005.

[67] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, A. Joglekar, “An integrated experimental environment for distributed systems and networks”, in Proc. of the Fifth Symposium on Operating Systems Design and Implementation, Boston, MA, USENIX Association (December 2002) 255–270.

[68] http://openvz.org: OpenVZ server virtualization open-source project.

[69] http://www.grid.unina.it/software/ITG/index.php.

[70] Kohler, Morris, Chen, Jannotti, “The Click modular router”, in ACM SIGOPS Operating Systems Review , Volume 33 Issue 5, pp. 217-231**,** 1999.

[71] A. Acharya, and A. Shaikh, “Using Mobility Support for Request-Routing in IPv6 CDNs”, in 7th Web Caching Workshop (Aug. 2002).