Internet and Data Centers

Appunti delle Lezioni di Internet and Data Centers

Anno Accademico: 2025/26

Giacomo Sturm

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche
Università degli Studi "Roma Tre"

Sorgente del file LaTeX ed ultima versione del testo disponibile al link:
https://github.com/00Darxk/Internet-and-Data-Centers/

Indice

1 Introduzione: ISPs ed IXPs

Gli *Internet Service Provider*, ISP, o *Autonomous System*, AS, sono gestori autonomi della rete che hanno una responsabilità su un certo territorio. Su queste reti indipendenti viaggia il traffico, gli ISP sono infatti responsabili di inoltrare i pacchetti che passano al loro interno per raggiungere la destinazione. In totale sono presenti circa 120000 ISP al mondo, alcuni di questi sono pubblicizzati e pubblici per offrire servizi ad utenti comuni, altri offrono servizi a grandi compagnie private o altri ISP. Questi ISP formano una gerarchia, non dichiarata, che si può inferire, non essendo governati da una struttura o organizzazione sovrastante. Alcuni di questi ISP sono privati e non si mostrano, per cui si può solo supporre la loro posizione in questa gerarchia. Ogni ISP si cura solamente dei suoi interessi specifici, attraverso i loro clienti, permettendo loro di connettersi ad un prezzo economico con altri servizi o utenti nel loro territorio. Questi ISP pagano altri ISP più grandi per raggiungere destinazioni sempre più remote, in questo modo si può determinare una gerarchia che parte dagli ISP più piccoli, non fornitori di altri ISP, fino ai più grandi, che offrono solo servizi ad altri ISP, stando alla radice della gerarchia.

Queste regioni geografiche che identificano i territori di un certo ISP non sono separate, ma sono fortemente sovrapposte, è solamente una divisione logica e parzialmente può essere interpretata come una suddivisione geografica.

Questi ISP comunicano tra di loro attraverso una connessione privata di rete *Private Network Interconnects*, PNI, queste possono essere dei cavi fisici di proprietà degli ISP oppure possono essere affittati, che collegano due router tra questi due ISP. Il costo per questa PNI viene pagato in base ad accordi privati tra i vari ISP ed in base al traffico. Questo è un rapporto commerciale a vari livelli, ci può essere un cliente che paga per utilizzare i servizi di un altro ISP, oppure bilaterale o peer-to-peer, dove i due ISP si scambiano pacchetti destinati all'altro ISP, ed entrambi pagano insieme il costo della connessione. In genere si paga in base al picco di traffico giornaliero.

Questo processo tuttavia non è più efficiente, con il concetto di economia di scala, invece di stendere fili singoli tra tutti gli ISP, si creano di punti di scambio comuni dove tutti i provider portano una loro macchina ed un filo che la collega alla loro rete, a questo punto per comunicare con altri ISP si crea una connessione logica con uno degli ISP presenti nel punto di scambio. Questi punti di scambio si chiamano *Internet eXchange Points* IXP. Questi sono governati da associazioni che mettono a servizio dei propri consorziati punti di scambio in modo sicuro. Quando si offre un servizio, eventualmente, si cominciano ad offrirne di ulteriori, quindi oltre alla comunicazione offrono servizi di housing, di computing, etc. formando i tradizionali *Data Centers*. In questi punti di scambio ci sono interconnessioni di diverso tipo, commerciali, peer-to-peer, gratuite, ma sempre offerti nello stesso punto di scambio. Al mondo sono presenti circa un migliaio di questi IXP, in Europa il più grande è ad Amsterdam, AMS-IX.

La rete di reti non è più una rete di macchine connesse tra di loro, ma coagula considerando questi punti di scambio comuni dove sono presenti delle macchine. Un *Data Center* è uno spazio dedicato contenente computer, risorse di massa e sistemi di telecomunicazione. Questi si trovano generalmente in scantinati, ed i più grandi in zone a basso costo energetico o dov'è facilmente procurable, oppure in zone dove la temperatura è pressoché bassa, per diminuire il costo di raffreddamento. In questi data center si utilizza una quantità molto elevata di energia. Per la loro importanza, sono costantemente sorvegliati e protetti. Ogni macchina ha una procedura di recupero

pubblica, si assume che chi è in grado di avvicinarsi ad un macchina può effettuare queste procedure di recovery ed entrarne in possesso. Quindi la sicurezza deve essere anche fisica per impedire alle persone di avvicinarsi a questi macchinari.

C'è una differenza tra l'*hosting* e l'*housing*, nel primo l'hardware è posseduto dal data center, mentre nel secondo l'hardware non è fornito dal data center me viene procurato dai clienti. Altri data center invece hanno un solo cliente che ne è il proprietario, questi generalmente vengono realizzati solo dai più grandi come Google, Microsoft, etc.

Questa tendenza è nata negli ultimi decenni, quando oltre ad offrire connettività gli ISP cominciarono ad offrire hosting ed housing, creando dei propri data center. La connettività è ormai diventata un bene di consumo a basso costo, quasi dato per scontato. Questi data center possono essere interni ad una singola rete o condivisi tra varie reti.

Dentro un IXP si può facilmente inserire un data center, avendo già la struttura e l'organizzazione per ospitarli, unendo i router e macchine dei vari ISP a risorse di calcolo offerte dal data center.

1.1 Il Cloud

La definizione di *cloud* fornita dal NIST, *National Institute of Standard and Technology*, è caratterizzata dalla possibilità di scalare su richiesta, un accesso a banda capiente, la possibilità di attivare risorse su richiesta, avere una risposta rapida e misurazione accurata.

I servizi offerti da piattaforme cloud possono essere IaaS, *Infrastructure as a Service*, macchine virtuali, risorse di massa, firewall, tutti realizzati virtualmente; PaaS *Platform as a Service*, la gestione di macchine virtuali database, risorse; Saas *Software as a Service*, dato che il software che viene eseguito su queste macchine virtuali può essere comprato o affittato direttamente da queste piattaforme cloud. Altri servizi specializzati possono offrire potenza di calcolo massiva, soprattutto nell'ambito dell'intelligenza artificiale e nel loro addestramento. Alcuni servizi di cloud sono reti pubbliche, altri possono essere privati o ibridi. I leader del settore sono AWS, *Amazon Web Services*, Microsoft Azure e Google Cloud. Questi cloud possono popolare diversi data center, AWS si trova su 25 macro-regioni geografiche. Microsoft si trova su più di 60 regioni mentre Google principalmente in America del Nord.

Quando si compra un servizio su una piattaforma cloud, questo è spesso non necessariamente su uno stesso data center, poiché trasferire una macchina virtuale è estremamente facile. Le risorse acquistate su piattaforme cloud sono distribuite ed in base alla necessità del gestore possono essere trasferite. Questo vale sia per macchine virtuali che per la memoria di massa.

In Europa è stato creato un sistema federato di ISP europei, GAIA-X, per realizzare hub nazionali indipendenti dall'hardware garantendo servizi, specificando un modello con garanzia di controllo sulla locazione delle macchine e la strada che percorrono i dati. Offrendo la possibilità ai loro clienti di mantenere i dati all'interno della propria nazione, per aderire ad eventuali normative, o esigenze specifiche.

In Italia l'Agenzia per la Cybersicurezza Nazionale ha fondato un cloud marketplace, che offre alla pubblica amministrazione la possibilità di acquistare risorse cloud da parte di provider qualificati. La CONSIP ha un servizio di supporto per i servizi cloud dedicati alla pubblica amministrazione. Inoltre c'è un programma per portare ed abilitare la pubblica amministrazione all'uso dei servizi

cloud. Questo è previsto dal PNRR, nel primo obiettivo, attraverso una migrazione ad un cloud nazionale, il PSN, oppure ad un provider certificato. Anche la stessa migrazione ad un altro cloud viene trattato come un servizio senza dover necessariamente conoscere il suo funzionamento tecnico.

1.2 Polo Strategico Nazionale: PSN

Il Polo Strategico Nazionale o PSN, è un progetto con l'obiettivo di creare un grande data center nazionale che fornisca vari servizi cloud.

I principali servizi offerti dal PSN sono housing ed hosting; cloud IaaS Private e Shared; può offrire cloud pubblica gestita dalla PSN garantendone la sicurezza, utilizzando sistemi offerti da produttori certificati, si può realizzare un sistema ibrido con elementi interni alla rete.

Questi servizi sono in parte realizzati da infrastrutture proprie del PSN ed in parte da cloud pubblici esistenti, Azure, Google Cloud ed Oracle.

Sulla rete privata del PSN, sul suo hardware, vengono offerti servizi IaaS dedicati e condivisi, CaaS e *Disaster Recovery* (DR) e PaaS, fornendo elementi applicativi e middleware come servizio.

Servizi di cloud ibridi vengono gestiti nel territorio nazionale dal PSN, utilizzando un'infrastruttura ibrida, cloud pubblici di partner commerciali e privati sulla rete del PSN, installati sull'infrastruttura locale. Questo come il *Public Cloud PSN Managed* ed il *Secure Public Cloud* garantiscono la presenza dei dati sul territorio nazionale. Per il resto dei servizi cloud, i dati sono localizzati presso il *Cloud Service Provider* (CSP) e non garantisce la *data sovereignty*.

L'infrastruttura del PSN è un data center a doppia regione, interconnessa via *Virtual Data Center Network* (VDCN), simulando una stessa LAN, duplicando i dati tra le due regioni.

Servizi PaaS sono messi a disposizione da parte del PSN su una piattaforma per erogare elementi applicativi e middleware, astraendo l'infrastruttura sottostante. Questi servizi sono *Database as a Service* (DaaS), verifica dell'identità e gestione degli accessi, big data e servizi AI. Analogamente fornisce applicazioni basate su container con servizi CaaS.

L'accesso a questi servizi avviene solamente tramite la rete TIM, uno dei consorzianti che ha vinto il contratto, questa implementa dei sistemi di sicurezza per garantire che eventuali attacchi non riescano a penetrare la rete interna del PSN, utilizzando tunnel per il traffico verso l'infrastruttura del PSN.

I quattro data centers sono collegati su due regioni, tra queste regioni il traffico viene gestito con tunnel end-to-end come il protocollo *Multi Protocol Label Switching* MPLS (5.5), sul backbone IP di TIM.

Tra data center della stessa regione le VLAN sono trasportate con VXLAN, *Virtual eXtensive LAN*, queste utilizzano espedienti tecnologici per duplicare infrastrutture fisiche utilizzando dei tag per dividere i pacchetti destinati alle varie copie. Il protocollo VXLAN permette di connettere reti diverse condividendo pacchetti di livello due tra le reti, trattandola come una singola LAN, trasparente dal punto di vista delle macchine interne.

1.3 Sistema Pubblico di Connettività: SPC

Questo è un bando multi-fornitore, il vincitore del bando ha preso la parte più grande del mercato della pubblica amministrazione, mentre i restanti fornitori hanno ottenuto il rimanente.

Il Sistema Pubblico di Connettività rappresenta una rete di grandi dimensioni dedicata a connettere le sedi di ogni singola Pubblica Amministrazione tra di loro ed all'Internet. Offre diversi servizi di trasporto wired e wireless, su rete elettrica, ottica o da parte di dati satellitari.

Per questi servizi viene definito un *Service Level Agreement* (SLA), ad ogni servizio il fornitore assegna una misura di qualità. Può essere espressa in termini di metriche come il jitter, la distanza tra i vari pacchetti, se è pari a zero questi arrivano tutti allineati. Se la misura della qualità dovesse scendere al di sotto di terminate soglie, sono previsti rimborsi, chiamati "penali" all'amministrazione. Gli SLA contemplano anche guasti o anomalie. Vari ISP realizzano questo SPC, a ciascuno è assegnato un gruppo di PA, Pubbliche Amministrazioni, l'accesso ad Internet è gestito dai singoli fornitori, in base all'esito dell'asta multi-fornitore.

Gli ISP realizzano le reti delle PA usando MPLS, per comunicare tra di loro su una rete QXN, *Qualified eXchange Network*, attualmente collocato presso gli IXP di Roma (Namex) e Milano (MIX).

2 Algoritmi di Instradamento

Si considerano algoritmi di instradamento solo per infrastrutture fisiche, connesse da reti fisse. Per queste esistono due tipi principali di algoritmi di instradamento, *distance vector* e *state packet*. In TCP si utilizza una variante del distance vector. Corrispondono a due filosofie opposte.

Si vogliono delle certe qualità da questi algoritmi:

- Si vuole avere algoritmi efficienti, per evitare che il calcolo dei cammini abbia un peso eccessivo rispetto all'instradamento dei pacchetti. La tabella di instradamento dentro ai router non viene realizzata con un tabella, ma con strutture ad albero specializzate per avere un accesso il più veloce possibile. Inoltre, le risorse computazionali dei router non sono necessariamente sufficienti per poter gestire la complessità dell'intera rete. Si vuole mantenere la maggior parte della computazione sull'hardware del router, e questa deve essere veloce data la dimensione dei singoli pacchetti. Sono di piccola dimensione, poiché durante la creazione dei vari protocolli di rete in competizione, venivano favorite schede di rete più economiche. Utilizzando meno memoria per un pacchetto si hanno schede di rete più economiche, e questo rappresenta un errore da parte del consorzio DIX, che ha creato ethernet, anche se si è affermato come lo standard per le comunicazioni via filo, è necessario avere operazioni di inoltro estremamente veloci poiché i singoli pacchetti sono estremamente piccoli. Un router è diviso nel *data plane* e nel *control plane*, gli algoritmi di routing sono presenti nel control plane, e la tabella di instradamento composta da questo control plane viene inoltrata al data plane. Parlando con le altre macchine i protocolli di routing devono determinare la tabella di instradamento. Si vuole che questi protocolli siano efficienti, poiché su un router sono presenti molti altri servizi aggiuntivi, quindi il tempo del processore è limitato.

- Inoltre, si vuole individuare un cammino ottimo, un cammino che impieghi il minor tempo possibile per raggiungere la sua destinazione. Per determinare l'ottimalità del cammino si utilizzano criteri come il numero di hop o il costo delle linee, talvolta assunto inversamente proporzionale alla velocità. Questo cammino se minimizza il numero di hop, minimizza il numero di immissioni da parte dei router che attraversa.

- Un altro criterio utile è l'utilizzo delle linee, avendo banda limitata, un router non può mandare tutti i pacchetti su una stessa linea, avendo anche un buffer limitato per le singole linee. Il carico corrente della rete è tuttavia difficile da calcolare. Se si considerasse solamente il carico della rete, si creerebbe un fenomeno di retroazione causando un oscillazione della rete incontrollabile. Altrimenti si potrebbero scegliere linee con un packet loss minimo. In genere si scelgono algoritmi che si basano sul numero di hop. Questi algoritmi devono essere robusti e dinamici, poiché alcune linee possono riscontrare malfunzionamenti, errori di configurazione da parte di amministrazioni di rete, etc.

- Allo stesso tempo, si vuole essere stabile, non si vuole cambiare il routing durante la trasmissione. Tutti i pacchetti devono essere inviati e ricevuti nello stesso ordine, anche se esiste il livello TPC per riordinarli, nessuna macchina deliberatamente invia pacchetti fuori sequenza. Per questo un'oscillazione del routing non è accettabile, poiché grava sul livello TPC, aumentando il tempo necessario per sistemare questi pacchetti. Per questo si sono realizzati

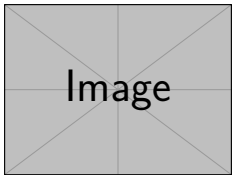
protocolli inter-dominio che possono oscillano in continuazione, dato che il loro effetto non era completamente compreso. Per cui è possibile realizzare esperimenti dove la rete diventa instabile.

- Possono effettuare scelte diverse in base al tipo di traffico, questo può essere classificato quando entra in una rete di un ISP, per determinare se si tratta di traffico utente generico o mission critical o privilegiato o VoIP.
- Inoltre, questi algoritmi devono essere equi, nessun nodo deve essere privilegiato o danneggiato rispetto a tutti gli altri.
- L'ultimo criterio è l'economicità, si vuole ridurre i costi di configurazione e manutenzione dei protocolli di routing.

Questi criteri sono talvolta contrastanti, e bisogna scegliere l'algoritmo migliore per un dato caso d'uso.

Si possono classificare questi algoritmi in statici e dinamici. Gli algoritmi dinamici applicano un instradamento in funzione della topologia e del carico della rete, mentre algoritmi statici hanno applicazione ristretta poiché prendono decisioni indipendenti dallo stato della rete. Questi algoritmi statici vengono utilizzati in casi semplici. La configurazione manuale delle macchine è una configurazione statica, questa è sempre presente anche in piccole parti in ogni rete. Se in una rete sono presenti topologie ad albero, queste non hanno bisogno di una configurazione dinamica, poiché tra due nodi in un albero è presente un solo cammino che li collega.

Invece per topologie magliate è opportuno utilizzare un algoritmo dinamico, poiché al taglio di un link o allo spegnimento di un nodo, bisogna ridistribuire il carico e riorganizzare la rete in modo veloce, senza compromettere l'operabilità delle altre macchine.



Uno di questi algoritmi statici è il *flooding* che consiste nell'inoltro di ogni pacchetto a tutte le interfacce connesse al router.

Gli algoritmi dinamici possono essere successivamente divisi in:

1. Routing isolato, dove ogni router decide senza comunicare con altri router. Algoritmi di routing isolato sono *hot potato* e *backward learning*, l'algoritmo utilizzato dai switch o bridge, che determinano in base alla provenienza dei pacchetti le destinazioni possibili rispetto alle varie interfacce.
2. Routing centralizzato, dove un router centrale determina la scelta migliore, anche questa strategia è rimasta per molto tempo sottovalutata

3. Routing distribuito, questa è la strategia corrente della rete, dove ogni router informa i propri vicini con informazioni note. I principali algoritmi di routing distribuito sono distance vector e link state packet, nel distance vector i router informano i propri vicini rispetto alla condizione globale, mentre nel link state packet i router inviano alcuni pacchetti verso tutta la rete che raccontano della topologia locale.

Per comunicare sulla rete è richiesto:

- Un indirizzo di rete ed una sua netmask, associato ad una scheda di rete. Una netmask indica da un indirizzo qual è la parte di prefisso, relativo all'intera rete, e quale di host, relativa alla singola macchina. Per mandare il pacchetto apparentemente non serve, ma è necessaria per determinare la rete dove si trova la macchina a cui può inviare pacchetti direttamente. Si controlla prima se la macchina destinazione è direttamente raggiungibile, inviando un pacchetto di livello due (MAC), come un ARP request.
- Ogni macchina almeno in questo senso ha meccanismi di routing. Inoltre per tutti i destinatari che non sono locali è presente un default gateway, il primo indirizzo disponibile nella rete.
- Inoltre, è necessario conoscere l'IP del DNS per risolvere nomi di dominio in indirizzi.

Una macchina avendo queste quattro informazioni può navigare in rete. Anche un router ha interfacce e netmask differenti per ogni rete su cui è connesso. Queste vengono inserite nella tabella di instradamento e rappresentano le reti direttamente connesse e non possono essere rimosse dalla tabella di instradamento, per configurazione.

In ogni router è presente una parte di routing statico, che rappresenta questa configurazione di interfacce.

Se il routing è completamente statico questa tabella contiene per ogni nodo da raggiungere le linee da usare, compilata dall'amministratore di rete, chiamato ad intervenire in presenza di guasti. Una variante quasi-statica permette all'amministratore di fornire più alternative in ordine di priorità.

Se il destinatario è direttamente connesso, bisogna determinare l'indirizzo MAC della scheda di destinazione con una richiesta ARP, altrimenti si ottiene il next hop per raggiungerlo. Questi pacchetti di livello due vengono creati localmente nelle varie reti locali che attraversa, mentre rimane invariato il pacchetto di livello tre, eccetto per il campo ttl ed il checksum che viene ricalcolato.

Una versione del flooding chiamato selective flooding viene utilizzato come sotto-protocollo di altri protocolli per inoltrare pacchetti solo su un insieme di linee selezionato, scartando pacchetti troppo vecchi, per esempio al suo secondo passaggio per uno stesso nodo evitando cicli.

Nel routing isolato ogni *Intermediate System* (IS) calcola in modo indipendente le proprie tabelle. L'hot potato invia il pacchetto alla linea con coda più breve, di interesse solo teorico.

Nel *backward learning*, definito in IEEE 802.1D solo su livello due, dai pacchetti in ingresso vengono determinate le macchine raggiungibili su quella linea, questi protocolli non aprono pacchetti di livello superiore. Per effettuare il backward learning è importante che non siano presenti maglie, altrimenti un pacchetto potrebbe entrare da più interfacce, spesso si accoppia ad un algoritmo per il calcolo dello spanning tree o albero ricoprente della rete. Può essere raffinato aggiungendo un campo che specifica il costo di un cammino, incrementandolo ad ogni hop, per mantenere alternative

ordinate. Quando la destinazione è ignota si effettua flooding. Per la sua necessità di avere una struttura ad albero non è utilizzato in Internet, dato che non è possibile disattivare arbitrariamente connessioni.

Il routing centralizzato è un meccanismo di routing gestito da un'entità centrale, supponendo l'esistenza di un *Routing Control Center* (RCC) che conosce la topologia della rete, spesso non è realistico. È rimasta per molto tempo teorica, ma recentemente sono apparse esigenze di trasferire un grade volume di traffico tra poche macchine. Quindi un'autorità centrale tramite dei protocolli, *Software Defined Networking* (SDN 3.4), determina la topologia della rete in quel momento e stabilisce la strada migliore per poter trasferire quella grande quantità di dati. Si può scegliere di effettuare routing guidati da protocolli noti o imporre flussi di traffico.

Nel routing distribuito non esiste un RCC, ma tutte le funzionalità sono implementate da tutti gli IS, seguendo lo stesso paradigma, comunicando con i propri vicini. Il primo distance vector invia ai propri vicini una parte della tabella di routing, una proiezione rimuovendo colonne relative a macchine locali. Mentre il link state packet invia informazioni sui suoi vicini verso il resto della rete.

2.1 Algoritmi Distance Vector

Ogni IS invia una tabella detta distance vector ad ogni IS adiacente. Questa tabella contiene la parte essenziale della sua tabella di instradamento, da cui vengono omessi dettagli locali. Ogni IS ricevendo questi distance vector dai propri vicina ricalcola la tabella di instradamento integrando queste informazioni.

Le destinazioni vengono apprese con un meccanismo di passaparola. Inizialmente la tabella di instradamento non può essere vuota, altrimenti rimarrebbe vuota ad ogni iterazione dell'algoritmo. La prima informazione da inserire sono gli indirizzi direttamente connessi, a distanza di un singolo hop, nel data plane. Queste informazioni vengono prese dal control plane ed inviate ai propri vicini. Questi effettuano lo stesso inviando i propri indirizzi direttamente connessi, inizializzando l'algoritmo.

Ogni destinazione nella tabella è corredata dal costo del cammino dalla destinazione all'IS stesso. Questo viene calcolato dai distance vector ottenuti dai vicini, sommando i costi della linea di ingresso da cui è stato ricevuto.

Molte aziende e produttori cercando di superare la competizione, aggiungono caratteristiche e modificano questi protocolli. L'algoritmo di distance vector originario appartiene a Cisco.

Per passare le informazioni bisogna inviare i vari distance vector ai propri vicini, questi possono essere inviati secondo vari criteri. Possono essere inviati periodicamente oppure ad ogni modifica della tabella di instradamento.

I nodi adiacenti da aggiornare vengono appresi in base a protocolli di servizio appositi, oppure sono ignoti, inviando i pacchetti di distance vector in multicast.

Il tempo necessario affinché tutte le macchine conoscano tutte le altre è abbastanza lungo, questo tempo si chiama tempo di convergenza, fino a decine di secondi, un tempo estremamente lungo per l'ingegneria delle reti. Non si può intasare la rete inviando distance vector molto grandi, per cui l'utilizzo di questo algoritmo è limitato a reti con non più di 1000 nodi, si preferiscono protocolli come SPF per reti più dinamiche.

Per ogni linea il distance vector è composto da due colonne, una prima che contiene la rete e la seconda il costo per raggiungerla. Questa colonna dei costi viene sommata al costo della linea ricevente. Tra questi distance vector risultanti, si scelgono le rotte migliori per raggiungere una rete in base al costo minore. Queste vengono inserite nella tabella di instradamento dell'IS.

Questi IS non devono essere sincronizzati altrimenti periodicamente tutte le macchine smetterebbero di funzionare per mandare e ricevere i vari distance vector. Per questo il tempo da attendere per inviare i distance vector viene sfasato di una certa periodo. Sono presenti ulteriori meccanismi per garantire che questo processo non sia sincrono.

Il problema cruciale che si crea con i distance vector è il *count to infinity*, riescono a reagire rapidamente a buone notizie, e lentamente alle cattive notizie.

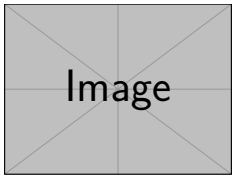
Una buona notizia può essere l'aggiunta di una nuova rete, il router invia il suo distance vector aggiornato e l'informazione viene propagata sull'intera rete in un tempo relativamente breve.

Se una connessione viene tagliata, il vicino non può aggiornare la propria tabella di instradamento, non sapendo che il vicino non sia più raggiungibile sull'intera rete. Queste macchine avrebbero raggiunto la rete ora disconnessa passando per altre macchine adiacenti. Poiché non possono eliminare questa entry nella tabella di instradamento, ad ogni nuovo distance vector la distanza alla rete disconnessa incrementa in continuazione, tendendo all'infinito. Si crea una specie di eco, ad ogni passaggio nei distance vector che riceve una macchina, il costo del cammino verso la rete disconnessa è maggiore rispetto all'iterazione precedente, e quindi deve aumentare il costo memorizzato nella sua tabella di instradamento.

Una buona notizia arriva a distanza k in k passi. Le cattive notizie invece si propagano in un tempo che è funzione del valore convenzionalmente attribuito ad infinito. Questo valore si attribuisce alla lunghezza del cammino più lungo nella rete sommato ad uno.

Se un apparato ha inviato la rotta al vicino, e questo la rimanda indietro, non dovrebbe considerarla, questo fenomeno si chiama *split horizon*, ma non sempre è efficace per impedire la propagazione dell'eco. In reti circolari l'eco si propaga in un'unica direzione perpetuamente provocando il fenomeno del *count to infinity*. Questo algoritmo non conosce la topologia della rete, quindi non potendo sapere lo stato dei link, l'unico modo per conoscere le rotte raggiungibili è attraverso i distance vector.

Il valore ottimale di infinito per ogni rete dipende dalla sua topologia, e dal modo in cui l'eco viene propagato. Bisogna determinare il costo del cammino più lungo possibile nella rete, al massimo comprende tutti i router presenti. Non tutte le reti ammettono un cammino che passa per tutti i router.



Quando il costo di un cammino supera il valore di infinito nei distance vector, viene rimossa la rotta dalla tabella di instradamento e non più condivisa. Restituisce pacchetti ICMP *destination unreachable* se si prova ad accedere alla rotta.

A rete è connessa è possibile dimostrare con il modello di Bellman e Ford che l'algoritmo converge sempre ad una soluzione ottima, anche se all'inizio le tabelle di instradamento provengono da una situazione diversa. Il numero di passi è lineare se la metrica è il numero di hop.

Ci sono vari modi per valutare l'efficienza di un algoritmo di instradamento. Si può utilizzare il numero di passi come metrica per determinare l'efficienza dell'algoritmo distance vector. In una rete dove ci sono n nodi ed m link, la metrica è relativa solo al numero di hop, una buona notizia si propaga tra i nodi più lontani, al massimo $n-1$ link, in $n-1$ passi, analogamente per una cattiva notizia, attribuita al valore n.

Un altra metrica è il lavoro svolto dai router, questi inviano ad ogni passo m distance vector, dove m è il numero massimo di linee attaccate al router. Ogni tabella ha al più $O(n)$ righe, per ciascuno dei passi il router spende tempo $O(mn)$, inviando su ciascuna delle m linee un distance vector con n righe. Dato che la convergenza richiede tempo per propagarsi, nel caso peggiore di $O(n)$ passi, ogni router calcola per un tempo $O(n^2m)$.

2.2 Algoritmi Link State Packet

Questi algoritmi utilizzano un pacchetto che viene chiamato *link state packet* (lsp) che descrive lo stato dei link di ogni IS. Un router su questa rete si calcola l'instradamento ottimale per ogni rotta costruendo la topologia completa della rete, dai vari lsp. Usano un algoritmo di calcolo di costo minimo, in questo caso usa Dijkstra, compone una tabella di instradamento per il valore del next hop per ogni rotta. La mappa della rete viene costituita usando questi lsp, qui sorge un problema, poiché i link non sono componenti attivi, al massimo un router può descrivere le macchine presenti intorno all'IS, ma deve includere anche informazioni riguardo i link connessi. Questi pacchetti vengono mandati ai vicini e si suppone che i vicini li propaghino in selective flooding ad ogni altro IS della rete.

Se viene modificata la topologia, il router che rileva questa modifica aggiorna la sua mappa ed invia lsp aggiornati sulla rete. Ogni IS conserva in un database gli lsp più recenti, per aggiornare la topologia della rete. I link hanno dei costi e per questo si utilizza un algoritmo come Dijkstra invece di un BFS, che potrebbe essere utilizzato in caso la metrica sia il numero di hop. Il database deve essere sincronizzato tra tutti gli IS, le informazioni devono essere propagate e uguali tra tutti, considerando solo la versione più recente dei vari lsp. Se la topologia non cambia non è necessario inviare periodicamente l'intero database. Si tratta di un protocollo molto più silente, è sufficiente un unico pacchetto lsp per aggiornare il database correttamente tra tutti gli IS della rete allo stesso stato.

Algoritmi distance vector inviano pacchetti molto grandi per calcolare direttamente la tabella di instradamento, mentre algoritmi link state packet inviano pacchetti piccoli con l'obiettivo di calcolare la mappa della rete, e da questa calcolare la tabella di instradamento. A differenza dei distance vector, questo algoritmo può gestire reti con migliaia e migliaia di nodi, convergendo rapidamente.

2.2.1 Algoritmo di Dijkstra

Si considera un insieme $V = \{1, \cdots, n\}$ di vertici numerati, con archi orientati. Si indica con $A(i)$ l'insieme dei vertici j per cui è presente un arco orientato (i, j). Per ogni arco è presente una metrica

$a_{ij} \geq 0$, questa è infinito se l'arco è assente. La somma di un cammino è la somma delle metriche degli archi attraversati.

La distanza corrente dal vertice 1, iniziale, al vertice i è $d[i]$, la distanza minima tra questi è $m(i)$. Si definisce S l'insieme dei vertici in cui il cammino corrente ha distanza minima $d[i] = m(i)$.

L'algoritmo di Dijkstra calcola il costo minimo tra il primo nodo ed ogni altro nodo della rete, questi vengono inseriti in S per valori crescenti della distanza da 1. All'inizio contiene solo il primo elemento. Nel secondo passo, per ogni nodo i dal successivo all'ultimo n si pone $d[i] = a_{1i}$. In seguito si ripete il seguente passaggio finché $V \setminus S$ non è vuoto, si sceglie un nodo i in questo insieme tale che $d[i]$ sia minimo, aggiungendolo ad S, e per ogni nodo j adiacente ad i, si pone $d[j] = \min(d[j], d[i] + a_{ij})$.

$$S = \{1\}$$
$$\forall i \in \{2, \cdots, n\} \rightarrow d[i] = a_{1i}$$
$$\text{se } V \setminus S \neq \emptyset,\ i \in V \setminus S \text{ t.c. } d[i] = m(i) \implies S = S \cup \{i\},\ \forall j \in A(i) :\ d[j] = \min(d[j], d[i] + a_{ij})$$

Se $d[v] = m(v)$ per ogni nodo $v \in S$, allora quando i viene aggiunto ad S, anche i ha per $d[i] = m(i)$. Supponendo per assurdo che $d[i] > m(i)$. Sia il cammino minimo p tra 1 a i necessariamente $|p| = m(i) < d[i]$. Sia j l'ultimo nodo di p, e k il nodo seguente. Allora deve essere $k \neq i$ altrimenti l'algoritmo avrebbe computato $d[i] = |p| = m(i)$, poiché il sottoinsieme di un cammino minimo è anch'esso un cammino minimo.

Ipotizzando che $d[k] = m(k)$, ovvero il cammino è ottimo, poiché è calcolato da $d[j]$, per ipotesi $d[j] = m(j)$ e l'arco (j, k) è utilizzato da p. Poiché k è un nodo intermedio di p, si ha che $m(k) \leq m(i)$. Ma si ha che $d[k] = m(k) \leq m(i) = d[i]$, e questo rappresenta un assurdo, poiché $d[i]$ non è un ottimo e non sarebbe stato scelto dall'algoritmo.

Supponendo che in una rete ci siano n nodi e m link. Un'implementazione semplice richiede tempo $O(n^2 + m) = O(n^2)$, in questa si scorrono tutti i nodi ad ogni passo, mentre un'implementazione più sofisticata scorre tutti i nodi ordinati per costo minore, senza doverli scorrere tutti ha una complessità $O(n \log n + m)$.

Calcolati i cammini minimi, questi formano un albero ricoprente dei cammini minimi, ogni IS crea la sua tabella di instradamento in base a questo albero, scartando tutte le informazioni oltre il next hop per una data rotta, poiché suppone che tutti gli altri router abbiano calcolato lo stesso albero ricoprente. Può essere che i cammini minimi calcolati non coincidano perfettamente, poiché possono esistere più cammini ottimali tra due nodi.

Alcuni algoritmi gestiti da Cisco utilizzano cammini tra due nodi dello stesso costo, *Equal Cost Multi-path*, in questo modo un router può distribuire il traffico su più rotte. Questi router hanno due entry per ogni rotta, dividono il traffico in queste linee, se non viene gestito bene i pacchetti vengono disordinati ed il livello TCP non è in grado di riordinarli. Per gestirlo correttamente prendono gli indirizzi IP del sorgente e destinazione e ne calcolano l'hash ad un bit, se è uno lo mandano su una rotta, altrimenti lo mandano all'altra. In questo modo instradano interi flussi su rotte diverse.

2.2.2 Neighbor Greetings

Poiché i link sono inerti, non è possibile sapere quando un link è attivo o disattivo. Si utilizza quindi un protocollo di *neighbor greetings*. Questo si utilizza per mantenere aggiornate la adiacenze, inviando periodicamente su tutte le interfacce dei pacchetti. Quando un IS rileva una modifica della topologia, rilevando un nuovo pacchetto di greeting oppure se una delle linee ha superato un tempo di timeout, invia un lsp per propagare la modifica.

Ogni pacchetto lsp ha un numero di versione relativo al router, il pacchetto rimane invariato durante la sua propagazione con un processo di selective flooding. Si utilizza questo numero di versione per verificare se è già stato ricevuto un pacchetto con quel numero di versione, in caso il router ricevente non compie alcuna azione, altrimenti se il pacchetto ricevuto ha una versione più recente aggiorna il suo database e lo propaga, oppure se ha una versione precedente, trasmette quello posseduto, di versione maggiore, al mittente, per allinearlo con il proprio database.

Una LAN su cui si affacciano diversi router si presta male ad essere modellata come un grafo, uno dei router si prende l'onere di rappresentare la LAN come uno pseudo nodo, e gli altri vedranno la LAN come una stella, avente al centro questo router.

Per questo alcuni interlocutori sono dei link, sono pseudo nodi che impersonano la connessione. Ogni connessione ha un *designated router* che rappresenta la LAN al resto del mondo, ogni altro router vede la LAN come se fosse attiva.

2.3 Protocolli di Routing

Il termina gateway è un termine antico che indica il router utilizzato per accedere all'Internet da una LAN. Alcuni protocolli sono detti *Interior Gateway Protocol* (IGP) all'interno di una stessa LAN, questi sono RIP, OSPF, IS-IS. Altri protocolli si chiamano *Exterior Gateway Protocol* (EGP) che sono utilizzati all'esterno delle LAN, questi sono EGP e BGP. Gli IGP vengono utilizzati per aggiornare le tabelle di instradamento all'interno del sistema, sulla stessa LAN. I protocolli usati per trasferire informazioni di routing tra diversi domini amministrativi passano per BGP, il protocollo standard esterno, ed in generale si usano protocolli EGP.

2.3.1 RIP

Il *Routing Information Protocol* (RIP) è un IGP introdotto nel TCP/IP nel 1982, implementa l'algoritmo distance vector con invio del distance vector ogni 30 secondi. Usa una sola metrica, basata sugli hop, con un numero massimo di hop permessi pari a 15, quindi può essere utilizzato solo su reti piccole. Non vengono considerate altre metriche, tuttavia è compatible con ogni macchina, e converge sempre.

Nella prima versione venivano inviati in broadcast, da RIPv2 i pacchetti vengono inviati alla lista di subnet, ottenute dal proprio indirizzo IP e la netmask, in multicast.

I pacchetti di richiesta chiedono ai vicini i rispettivi distance vector, i pacchetti di risposta trasferiscono i distance vector richiesti.

Vengono inviati ogni 30 secondi, aggiungendo un piccolo offset random, per evitare che la rete si sincronizzi globalmente, possono anche essere inviati senza una richiesta esplicita. Dopo un certo tempo di timeout un prefisso viene considerato non più raggiungibile.

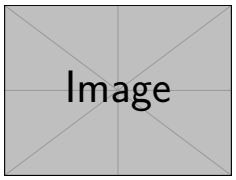
2.3.2 OPSF

L'*Open Shortest Path First* (OSPF) è un protocollo non proprietario, standardizzato dall'IETF, probabilmente il più diffuso IGP per TCP/IP. Abbastanza stabile come protocollo, con le ultime variazioni per IPv6 avvenute nel 2008. Sfrutta l'algoritmo link state packet, pensato per scalare in modo efficiente.

È organizzato gerarchicamente, dividendo la rete in aree connesse, dove il routing che avviene all'interno di una stessa area non è necessariamente uguale a quello interno ad un'altra area. Per comunicare tra queste aree si usano router confinanti su più di un'area, per agire da ponte tra queste. I router che si trovano sulla backbone, l'area principale, hanno una visione di tutto il resto della rete, mentre quelli nelle parti marginali hanno una visione molto ristretta.

Ci potrebbero essere configurazioni di nicchia, per tecnologie speciali che non corrispondono esattamente a questa suddivisione. Generalmente ogni area ha una topologia invisibile alle altre aree, i router di frontiera sono gli unici a cui si può accedere per comunicare con l'interno dell'area.

Questo protocollo assegna ogni interfacce ad un'unica area. Un router può essere interno se ha interfacce su una stessa area, backbone se ha almeno un'interfaccia backbone, area border se ha almeno due interfacce diverse su due aree diverse, e AS boundary o frontiera se ha almeno un'interfaccia verso l'esterno, altri domini amministrativi.



Ai router di frontiera vengono iniettate rotte, come se fossero locali, dentro al protocollo OSPF, che a loro volta iniettano sulla sua rete locale prendendosi la responsabilità di inoltrare eventuali messaggi alle altre aree.

Il protocollo IS-IS, *Intermediate System to Intermediate System* è simile ad OSPF, dove il confine tra due aree non sono i router di frontiera, ma i link tra due router nelle due aree.

2.3.3 Coesistenza di Protocolli Diversi

I protocolli di rete permettono la consegna di un pacchetto da una qualsiasi macchina che si trova in Internet ad un'altra qualsiasi macchina che si trova in Internet. Per effettuarlo è necessario uno schema di indirizzamento, questo viene gestito dal protocollo di livello tre. Molti protocolli hanno deciso di essere più capillari, individuando solamente un'interfaccia in una porzione specifica della rete. Si determina il formato del pacchetto di rete, questo contiene il pacchetto di trasporto di livello quattro, che non può essere letto e gestito da questi router. Offrono un servizio di consegna ai protocolli del livello di trasporto. I servizi di consegna per livello tre sono a commutazione di pacchetto.

Un protocollo di routing si riferisce ad uno specifico protocollo di rete, ed utilizza un algoritmo di instradamento per aggiornare le tabelle di instradamento a cui si riferiscono i protocolli di rete.

In generale sono presenti più protocolli di rete e quindi più protocolli di routing diversi, contemporaneamente, su una stessa macchina, anche relativi allo stesso protocollo di rete.

Queste macchine devono permettere a più protocolli di convivere. Se sono presenti due protocolli di rete le macchine vengono dette *dual stack*. Gli ES hanno due pile protocollari, in corrispondenza di due librerie API e le applicazioni devono scegliere quale pila da usare. Gli IS dual stack sono in grado di instradare pacchetti relativi ad entrambi i protocolli di rete.

La filosofia è di "navi nella nebbia", i diversi protocolli convivono in modo totalmente isolato. Per questo non è possibile effettuare una transizione completa da IPv4 a IPv6, la rete ha un'inerzia elevata, dato che le macchine devono poter operare su tutti questi protocolli.

Tipicamente le tabelle di routing sono separate ed i database dei vari protocolli sono invisibili l'uno dall'altro. Tutte queste tabelle di instradamento nel control plane determinano secondo certi criteri la tabella di instradamento nel data plane. Spesso questi protocolli utilizzano metriche tra di loro non confrontabili, tipicamente è l'amministratore di rete a decidere in che modo gestire questi vari protocolli, indicando una priorità tra le soluzioni offerte dai vari protocolli.

L'amministratore può governare queste rotte per determinare in che modo i pacchetti tra i vari protocolli comunicano l'un l'altro. Le rotte trasferite vengono viste dal protocollo ricevente come rotte statiche. Iniettare dentro un protocollo un altro protocollo è un'operazione delicata, se si usano protocolli che possono comunicare con tutto l'Internet, pubblicizzano di possedere tutte queste rotte, e se venissero inoltrate all'esterno, si creerebbe un effetto "buco nero".

Due ISP che comunicano usando due protocolli di routing differenti, possono utilizzare dei router di frontiera, appartenenti ad entrambi i loro domini. Questi possono comunicare su entrambi i protocolli per iniettare e pubblicizzare rotte tra i due domini. La mancanza di una chiara divisione delle competenze su questo router è un problema di natura gestionale.

Tuttavia protocolli diversi utilizzano metriche diverse, e se vengono condivise le rotte in questo modo, le metriche, con i loro criteri di ottimalità, vengono ignorate. Se la rotta attraversa diversi domini, questo processo di iniettare rotte ignorando le metriche, la perdita di ottimalità può essere molto rilevante.

2.3.4 EGP e Routing Inter-Dominio

Per evitare di avere questi problemi di carattere amministrativo, invece di cogestire una macchina e separare le competenze, ci si inventa una struttura dove tra le due reti si condivide un filo di cogestione più semplice, e sulle macchine affioranti si usa uno stesso protocollo EGP.

I protocolli EGP sono adottati tra domini amministrativi diversi, scegliendo un solo protocollo in comune. Può prediligere le rotte che attraversano più domini amministrativi, mantenendo i criteri di ottimalità persi usando le tecniche precedenti. Consente l'implementazione di politiche commerciali, non tutte le rotte possono essere attraversate da un dominio amministrativo, le rotte che attraversano uno specifico dominio possono essere preferite. Alcune di queste scelte sono dovute ad accordi tra i vari ISP. Inoltre, persegue la trasparenza rispetto agli IGP.

Le informazioni apprese tramite EGP sono ridistribuite in un IGP, quando sono raggiungibili. Una rotta che passa per EGP deve avere metriche di distanza, e non può essere calcolata dal protocollo IGP interno. Questa distanza deve essere misurata rispetto ad altri router di frontiera.

Queste informazioni sono condivise a tutti gli router di frontiera, che parlano EGP, dello stesso dominio amministrativo tramite connessioni TCP, per poter incrementare le metriche di distanza.

Il primo protocollo EGP ha cominciato ad essere usato nel 1984. Utilizza un protocollo simile al distance vector, ma solo con indicazione di raggiungibilità. Molto elementare, funzionava esclusivamente con reti ad albero.

2.3.5 BGP

Il protocollo BGP (5.1) è stato pensato per sostituire EGP, ridefinito nel 2006. Usa lo strato di trasporto connesso, TCP, per scambiarsi informazioni tra router BGP, sulla porta 179. Tipicamente prova a connettersi in maniera incrociata, e prova a realizzare connessione sia come client che come server, la prima che instaura una connessione la crea. In connessioni TCP non c'è differenza tra client e server, le connessioni sono essenzialmente peer-to-peer. Una connessione può essere attiva in un senso e non in un altro.

Questo protocollo BGP è uno dei protocolli più intriganti e complessi, consente di effettuare tante operazioni ed è difficile da configurare. Richiede una conoscenza molto profonda della rete, le configurazioni su questo protocollo sono molto costose.

3 Algoritmi e Protocolli di Livello Due

3.1 Spanning Tree Algorithm: STP

Questo protocollo di instradamento è pensato per essere utilizzato in una rete locale, dove sono presenti switch, quindi opera sul livello due. Gli switch compilano automaticamente le proprie tabelle di instradamento in modo backward learning, su questo fanno filtering. Questo funziona fino a quando non è presente un ciclo nella rete. Gli hub invece operano su pacchetti di livello uno, si sincronizzano sul preambolo e lo accorciano e con un tempo di ritardo molto basso lo inoltrano completamente. Se è presente un ciclo in hub, è impossibile individuare un ciclo. Gli switch invece sono macchine di tipo store and forward, è conveniente avere cicli nella rete per evitare fallimenti dovuti a guasti, ma non devono essere attivi altrimenti non funzionerebbe il meccanismo di apprendimento degli switch. La LAN si deve organizzare per chiudere questi cicli.

Con dei cicli uno stesso pacchetto può percorrere il ciclo e quindi uno switch non può sapere su quale interfaccia effettivamente si trovi l'host mittente.

Molti switch venivano venduti con lo spanning tree disabilitato, per cui era molto semplice creare cicli accidentalmente ed inviare perpetuamente pacchetti. Per evitare questi cicli bisogna bloccare temporaneamente delle porte, queste continuano ad ascoltare, ma non inoltrano i pacchetti. L'algoritmo di spanning tree è tra i più famosi poiché reti con switch sono molto comuni. Questo algoritmo rende la rete un albero e mantiene ridondanze ed è in grado di rispondere ai cambiamenti della rete. Questo albero ha una radice che viene chiamato *root bridge*, alla fine della configurazione delle porte vengono messe in *blocking*. L'amministratore della rete può configurare minimamente secondo alcune necessità. Questo algoritmo è robusto, efficace, efficiente ed è realizzato per garantire un basso costo di banda. Viene descritto nello standard IEEE 802.1D.

Si usano dei pacchetti chiamati *Bridge PDU* o BPDU, sono tra i rarissimi pacchetti che non usano lo standard ethernet 2.0, ma IEEE 802.3, dentro questi pacchetti non è neanche presente il codice per indirizzare il protocollo IP successivo. Sono pacchetti devo sono state rimosse tutti i dati superflui.

Questo protocollo è veloce, flessibile ai cambiamenti della topologia e di facile uso.

Lo standard indica con LAN il dominio dove vengono collegati gli switch, un insieme di domini di collisione. Si assume che ogni LAN possa contenere un numero arbitrario di macchine.

L'algoritmo parte da uno scenario con queste LAN e quando ci sono dei cicli li aprono disattivando alcune porte, senza chiudere nessuna LAN. Ogni bridge ha un identificatore, gli spareggi si calcolano sull'id più basso. Anche le porte hanno un identificatore, analogamente più basso meglio è. Si associa un valore di costo ad ogni LAN, definito dalla velocità della LAN. Lo standard viene aggiornato per aggiornare questa tabella dei costi quando vengono rilasciate nuove tecnologie a prestazioni più elevate. Se questi costi vengono messi a zero, l'algoritmo non necessariamente converge.

Le macchine devono essere raggiungibili da qualsiasi altra macchina senza che nessuna LAN venga isolata. Si vuole creare un albero di cammino a costo minimo, ma questo non è vero per ogni cammino individuato. Possono esserci cammini a costo minore per due specifiche macchine, ma che sono stati bloccati.

L'id del bridge, *bridge-id*, è rappresentato da due byte che indicano la priorità, di default si trova a metà dell'intervallo possibile `80:00`, i restanti sono corrispondenti all'indirizzo MAC della prima porta del bridge. Il valore di priorità si può modificare per rendere alcuni bridge più probabili ad essere scelti come radice.

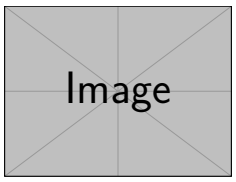
L'id della porta, *port-id*, è costituito dalla concatenazione di varie parti, un byte di priorità, spesso il valore di default è `80`, ed un byte corrispondente al numero della porta sul bridge, dato un contatore delle porte sul bridge.

Il costo di una LAN è gestito da una specie di distance vector. Non si contano gli hop, ma la somma dei costi, ogni tecnologia ha un costo inversamente proporzionale alla sua velocità. Su ogni porta viene specificato in ingresso questo costo.

Questo algoritmo è diviso in fasi, che possono essere eseguite in un singolo pacchetto. Nella prima fase viene eletto un bridge radice, si identifica per ogni bridge una porta verso il bridge radice, la più conveniente da lasciare accesa. Si determinano le porte designate per ogni LAN, una porta di un bridge viene scelta come quella che connetterà la LAN all'albero. Vengono poi bloccate tutte le porte ridondanti nell'ultima fase, tutte quelle inutilizzate dall'albero, non designate e non radici.

Sono previsti due tipi di BPDU, di configurazione, che contiene le informazioni necessarie per l'algoritmo dell'albero ricoprente e pacchetti di notifica del cambiamento della topologia, che non contengono nessun dato. I pacchetti LLC hanno un indirizzo di sorgente e destinazione pari a `0x42`, questi sono contenuti in pacchetti MAC con sorgente l'indirizzo della porta mittente, e con destinazione l'indirizzo multicast `01:80:c2:00:00:00`.

I campi importanti sono l'identificatore della radice dell'albero ricoprente, tutti riportano qual è la radice. La distanza dalla radice, l'identificatore del bridge mittente e l'identificatore della porta mittente.



1. Nella prima fase si deve capire qual è il bridge da radice, all'inizio ogni bridge si crede la radice ed invia come primo parametro il suo stesso id, ed invia questi pacchetti. Nel tempo quando un bridge legge un pacchetto con un campo radice con un id minore, si accorge di non essere la radice e propaga questa informazione nei suoi seguenti pacchetti. In questo modo la rete converge ad uno stesso bridge radice.

   Inoltre per configurare il costo dalla radice, all'inizio viene posta a zero, poiché ogni bridge si crede esso stesso la radice, se arriva un pacchetto con campo radice minore, utilizza il costo di questo pacchetto incrementato con il costo attribuito alla porta ricevente.

   Questi pacchetti vengono inoltrati modificando i campi di uno stesso pacchetto, modificando se necessario il campo radice, aggiornando il costo, e sostituendo l'id dello bridge e della porta mittente.

Questo algoritmo è perfettamente deterministico, sono presenti criteri di spareggio affinché sia convergente ad uno bridge radice specifico. Quando arrivano due pacchetti da due bridge diversi a parità di radice, si sceglie quello a costo minore, altrimenti si sceglie il pacchetto inviato dallo bridge con id più basso. In caso questi vengano mandati dallo stesso pacchetto, inoltra il pacchetto con id di porta più basso.

Alla fine di questi spareggi c'è sicuramente un bridge radice, e le porte dei bridge che ricevono questo pacchetto configurazione diventano le root port dello bridge.

2. Nella seconda fase bisogna scegliere le porte designate per ogni LAN connessa ad un bridge. Per ogni LAN connessa si scansionano le porte connesse e si elegge la porta con id più basso a porta designata.

3. Nella terza fase bisogna identificare le LAN, uno dei bridge su ogni LAN deve mantenere una porta aperta che viene chiamata designated port. Questa è quella che invia i pacchetti di configurazione con costo più basso, a parità di questo, il bridge di id più basso, altrimenti la porta ad id più basso.

4. Nell'ultima fase tutte le porte root e designated vengono messe in stato di forwarding, mentre tutte le altre vengono messe in blocco. Tuttavia si continuano a mandare BPDU anche dalle porte bloccate. Per questo la suddivisione in fasi non è propriamente corretta.

Nel momento in cui un bridge rileva un cambio di topologia genera dei pacchetti di topology change che raggiungono la radice e vengono propagati a tutti gli altri bridge nella rete.

In questi pacchetti sono contenuti a flag del cambiamento di topologia e l'acknowledgement di questo cambiamento. L'identificatore del root bridge, il costo del cammino fino alla radice, dinamico ad ogni hop, l'id del mittente e della rotta mittente. Inoltre, sono presenti metriche temporali, il tempo stimato da quando è stato emesso questo BPDU, il tempo in cui bisogna scartare la BPDU, il tempo tra due BPDU successive, il tempo da attendere per effettuare le transizioni delle porte *listening-learning-forwarding*.

Questi pacchetti vengono inviati fino a quando la radice non invia un pacchetto che conferma di aver ottenuto questa informazione. A questo punto tutti i bridge abbassano i valori di timer del protocollo in risposta temporanea all'instabilità della rete.

3.2 Virtual LAN: VLAN

Spesso nelle LAN di organizzazioni si pone il problema di separare area di dominio diverso tra di loro, per non interferire a vicenda e diminuire il rumore sulla rete. Questo poiché molti protocolli inviano pacchetti di broadcast, che possono sprecare molta banda inviando pacchetti a tutte le altre macchine, come ARP e DHCP request. Inoltre, ogni pacchetto broadcast ricevuto deve essere analizzato dalla scheda di rete della macchina, provocando un interrupt, questo può essere un problema se si intasa la rete di pacchetti broadcast, soprattutto in applicazioni time-sensitive come nei data center. Quando su una rete con switch il traffico raggiunge il 20% di utilizzo, non si può più utilizzare correttamente, mentre il traffico broadcast rimane invariato.

Normalmente per permettere l'accesso alla rete ad un dipendente, questo viene collegato da un punto di accesso ad uno switch tramite un permutatore. Ma in questo modo si uniscono dipartimenti, o società diverse tra di loro, con possibili criteri di sicurezza diversi. Infatti, tutte le macchine su una stessa LAN sono contrattabili da tutte le altre, e quando un router si trova in stato transitorio, ovvero invia il traffico in broadcast, si può osservare tutto il traffico della rete.

Utilizzando la virtualizzazione all'interno di una LAN fisica, si può partizionare la rete a piacimento, senza modificare la topologia sottostante, in modo flessibile e semplice, invece di acquistare nuovi switch per separare LAN fisiche. Si può utilizzare un solo switch per creare diverse VLAN, *Virtual LAN*, definite dallo standard IEEE 802.1Q. Queste VLAN definiscono una topologia logica indipendente, che vive sulla topologia fisica delle rete, realizzate sullo stesso switch. L'idea principale consiste nel tenere completamente separato il traffico di VLAN diverse.

All'interno di uno switch l'amministratore di rete definisce queste entità software chiamate VLAN, possono essere impostate semplicemente in base alla porta usata, ma non è un approccio flessibile. Una situazione più complessa permette di definire la VLAN a cui appartiene un pacchetto dalla porta di appartenenza, ed il suo contenuto, come l'indirizzo MAC, indirizzo o header IP, ecc. Queste diverse regole vengono fornite dal costruttore per lo specifico switch. I pacchetti indirizzati ad una VLAN vengono inoltrati solamente a quella VLAN, quindi queste entità si comportano come un filtro all'interno dello switch.

Una VLAN denota un insieme di pacchetti in transito per uno switch. Un singolo pacchetto può appartenere ad una singola VLAN, in base a regole specificate dallo switch. I pacchetti in entrata vengono partizionati in VLAN, ed in caso non siano definite VLAN, tutti questi pacchetti vengono attribuiti ad una VLAN di default.

Per assegnare un pacchetto ad una VLAN, questo deve arrivare su un certo insieme di porte chiamato *ingress port*, e rispettare certe caratteristiche. I pacchetti che sono stati assegnati ad una VLAN, invece, possono uscire solamente attraverso un altro insieme di porte chiamate *egress port*. Questi due insiemi di porte possono sovrapporsi parzialmente o totalmente.

Ad ogni VLAN è associato un identificatore, VLAN ID, nell'intervallo $[1, 4094]$, che la identifica. La default ha un ID pari ad uno. Inoltre, si può spesso attribuire un nome alfanumerico, più facile da ricordare, il nome non viene condiviso da una macchina all'altra, mentre in VLAN distribuite su più switch, il loro ID potrebbe essere distribuito. L'idea di collegare VLAN diverse, su uno stesso switch, consiste nell'usare un router che comunica a queste su porte diverse dello stesso switch. Il router deve assegnare i prefissi corretti a queste VLAN, quindi potrebbe essere consapevoli della presenza delle VLAN.

Ci potrebbe essere una singola interfaccia, ed un router VLAN aware, grazie a delle tecniche o informazioni nel pacchetto, è in grado di determinare a quale VLAN appartiene per instradare correttamente il traffico.

Nel momento che uno switch, in presenza di VLAN, riceve un pacchetto, ne determina la VLAN di appartenenza, altrimenti lo assegna alla VLAN di default. In seguito individua le porte da utilizzare per trasmettere il pacchetto, dal filtering database, ed eventualmente lo ritrasmette.

Gli switch determinano a quale VLAN assegnare un pacchetto, in base al loro filtering database. Questa tabella assomiglia ad una tabella di instradamento, e può essere configurata secondo due alternative, in base allo switch. Uno switch può operare in modalità IVL, *Independent VLAN Learning*, dove ogni VLAN ha un suo filtering database dedicato, e la modalità SVL, *Shared VLAN*

Learning, dove è presente un singolo database comune, condiviso tra tutte le VLAN. Alcuni switch possono operare solamente in modalità SVL.

Quando un pacchetto viene cercato nel filtering database, se non si trova il suo destinatario, allora viene inviato in broadcast a tutte le egress port della VLAN di appartenenza. Se invece si trova la porta da cui ritrasmettere prima controlla che la porta individuata sia configurata come egress port per la specifica VLAN, altrimenti scarta il pacchetto.

Quando una porta è sia ingress che egress per una VLAN, si parla di VLAN simmetriche, in generale l'insieme delle egress port potrebbe essere diverso da quelle delle ingress port. In generale si parla di VLAN asimmetriche, dove l'insieme delle egress port può essere diverso da quelle di ingress.

Le connessioni virtuali possono essere asimmetriche, ovvero una porta può essere una porta di ingress per una VLAN o di egress per un'altra VLAN. In questo modo si possono configurare le connessioni tra varie VLAN, impostando correttamente le porte di ingress ed egress. In presenza di VLAN asimmetriche, è conveniente usare il router in modalità SVL, altrimenti se non venisse trovata la porta corrispondente di egress per la VLAN destinazione, il pacchetto verrebbe mandato in broadcast, degradando le prestazioni. Usando un singolo filtering database per le varie VLAN si può identificare la porta di egress corretta appartenente alla VLAN di destinazione.

Avendo rete con più switch, bisogna provare a supportare le VLAN anche in questa configurazione. Si vorrebbe che ad ogni router si possa attaccare una VLAN, e che queste comunichino solamente tra di loro, o secondo certe policy. Non è possibile connettere due porte dedicate ad una stessa VLAN tra due router, poiché se si attaccasse un cavo per ogni VLAN, il bridge disattiva autonomamente tutti i link tranne uno. Il protocollo 802.1D parte prima della comunicazione, quindi non è possibile impedire disattivare tutti gli altri link superflui.

Questo si risolve nel protocollo IEEE 802.1Q, che prevede che tra due switch possa essere dichiarato un *trunk 1Q*, una somma di regole o policy, che possono essere definite a mano, per descrivere in che modo possano transitare i pacchetti tra varie VLAN, definendo ingress ed egress port corrispondenti.

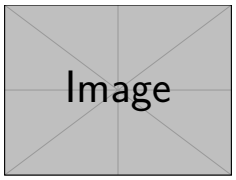
Per riconoscere se un certo pacchetto appartiene ad una certa VLAN, viene etichettato, inserendo un nuovo campo nell'header di livello due, chiamato tag, o etichetta. In base al tag il pacchetto viene estratto e consegnato alla VLAN corretta. Se il tag è vuoto è possibile configurare la porta per scartare il pacchetto, analogamente si può configurare di scartare se è presente un tag.

Per ogni porta di uno switch, si può definire a quale VLAN corrisponde, e configurare se deve etichettare i pacchetti in uscita come appartenenti a quella VLAN. In ricezione quando riceve un pacchetto, in base al tag può essere attribuito a quella VLAN oppure utilizzando le regole definite per quella porta.

Una porta può essere *access*, ricevendo ed inviando solo pacchetti non etichettati, *trunk* ricevendo ed inviando solo pacchetti etichettati, oppure ibrida.

La specifica viene definita nello standard IEEE 802.3ac, inserendo un campo di due byte, length e type, per ethernet 802.3 assume il valore `81-00`, è seguito da due byte di tag. Usando quattro byte, la lunghezza massima del pacchetto passa da 1522 a 1518. Per distinguere questi pacchetti in ethernet 2.0 e IEEE 802.3, si guarda ai byte corrispondenti al campo type o size. Questo campo in ethernet 2.0 specifica il tipo del pacchetto, e viene inserito qua il valore `81-00`, per indicare che è

stato aggiunto un tag di due byte di seguito. Altrimenti lo switch lo legge e determina se si tratta di un campo type o size, per distinguerlo da IEEE 802.3.



Se uno switch non è VLAN aware, smista il traffico guardando la destinazione e la sorgente, quindi neanche lo legge il tag, basando l'istradamento solo sull'indirizzo destinazione e sorgente, rendendolo retro-compatibile. Le macchine ci mettono molto poco ad essere VLAN aware.

Lo standard ha introdotto l'idea di *double tagging* usando due etichette consecutive, molto usato dagli ISP. Questi sovrappongono due tag, uno per identificare il pacchetto in base al cliente, C-TAG, e la seconda etichetta per associare il pacchetto ad un servizio, S-TAG.

3.3 Evoluzione dello Spanning Tree Protocol

Avendo delle VLAN, si è notato come parte prima il protocollo di spanning tree, quindi non è possibile collegare le diverse VLAN con link fisici tra due stessi switch. Questo problema non è risolubile.

Lo spanning tree ha avuto molte evoluzioni, alcune variazioni sono configurare per la velocità, in caso di esigenze particolari, come in ambito militare. Nel 2002 si è introdotto l'ultimo spanning tree protocol, chiamato *Multiple Spanning Tree Protocol*. Inoltre, Cisco introduce degli algoritmi di spanning tree proprietari per le VLAN, il problema consiste nel realizzare uno spanning tree diverso per le varie VLAN, prima che queste entrino in configurazione.

Gli switch dovrebbero conoscere prima le VLAN. Vengono usate in reti di qualsiasi dimensioni, e sono supportate da quasi tutti gli switch anche di fascia bassa. Nelle reti con VLAN potrebbe comunque essere necessario individuare uno spanning tree per evitare cicli. Se ogni trunk 1Q trasporta tutte le VLAN, uno spanning tree tradizionale non rischia di disconnettere qualche VLAN, altrimenti potrebbe disconnetterle dalla rete.

Anche con trunk 1Q completi, non rappresenta una scelta ottimale, potrebbe bloccare collegamenti veloci o solo per certe VLAN.

Si risolve calcolando in un primo periodo più spanning tree diversi, che poi verranno associati successivamente a VLAN diverse. Lo standard IEEE 802.1s prevede la presenza contemporanea di più spanning tree sulla LAN. In questo modo si può associare un'istanza dello spanning tree specifica ad ogni VLAN. Lo spanning tree di un'istanza è generalmente diverso da altre istanze, e permette di associare ogni VLAN all'istanza desiderata, ed ogni VLAN può essere associata ad un'unica istanza, mentre uno stesso spanning tree può avere più VLAN associate.

L'amministratore di rete deve configurare correttamente queste varie istanze alle VLAN, conoscendo i meccanismi dello spanning tre.

In realtà nel comportamento normale dello switch, non c'è una modifica sostanziale dello spanning tree. Modificando i parametri di priorità tra i vari switch per le varie porte configurate con le VLAN si possono calcolare istanze diverse dello spanning tree.

Il traffico etichettato da una certa VLAN esce da una porta se questa è aperta, rispetto allo spanning tree associato a quella VLAN ed è una egress port.

Utilizzando VLAN e spanning tree multiplo, si può lasciare attivo un numero considerevole di link, avendo un sistema di distribuzione del traffico migliore, sostenendo comunque la tolleranza ai guasti, ricalcolando dinamicamente solamente gli alberi la cui topologia è modificata.

Si può inserire un router, VLAN aware, che può inoltrare il traffico in maniera omogenea o eterogenea in base alla VLAN a cui è destinato.

Lo spanning tree viene associato a valle ad una singola VLAN, mentre durante il calcolo non dipende dalla topologia della stessa VLAN.

Lo standard *Multi-Spanning Tree Protocol* permette di usare questo concetto anche in LAN molto grandi, suddividendo la LAN in regioni, avendo per ogni regione un'istanza di spanning tree indipendente rispetto alle regioni vicine. Un ulteriore spanning tree a livello più alto le collega tra di loro.

3.4 Software Defined Network: SDN

Questo approccio centralizzato è ritornato di moda recentemente. Questa tipologia di algoritmi dinamici di instradamento prevede un routing controllato da un centro che elabora le informazioni disponibili, e decide l'instradamento nella rete. Questi algoritmi sono utili in reti piccole, e non è affidabile in reti di grande dimensione, con traffico intenso intorno al nodo di controllo.

A causa di questi problemi è stato sempre scarsamente adottato. Si è creata la fondazione ONF nel 2011 dedicata alla promozione e adozione di SDN.

La definizione di SDN è riportata nella RFC 7426, è un approccio programmabile di reti che supporta la separazione dei piani di controllo e di inoltro attraverso interfacce standardizzate. Questa computazione viene realizzata su di una macchina centralizzata.

Le prime idee risalgono al 2004.

Divide il routing ed il processamento del traffico, quindi il control ed il data plane.

Le funzioni del control plane vengono realizzate in un server che è in grado di determinare per ogni flusso di traffico specifico un cammino nella rete, questo percorso può essere configurabile dall'amministratore della rete. Le funzioni del data plane rimangono distribuite.

Il controllo centralizzato vuol dire che una macchina deve poter comunicare con tutti. Si suppone sia presente una linea di conversazione diretta, estranea alla rete. Questo protocollo viola la separazione tra il livello due ed il livello tre.

Si assume che sia presente questo canale di comunicazione. La parte di control plane, distribuita tra le macchine ora viene spostata su questo controllore. L'instradamento viene definito da questo. Si applica a delle reti che sono locali, di una singola amministrazione. l'operatore di rete non interviene sulle macchine modificando la configurazione, questo operazioni sono troppo costose su reti complesse.

Il numero di switch, link e macchine virtuali in un datacenter è considerevole.

L'ONF è un'organizzazione che promuove standard aperti, chiamati *OpenFlow*. L'ultima versione dello standard è la 1.5.1 uscita nel 2015, la versione 1.6 è disponibile solamente ai membri di ONF. Molti produttori, ISP e OTT, grosse reti di telefonia che offrono anche servizi di altro tipo, fanno parte di ONF. Cisco ha la sua implementazione di OpenFlow, chiamata Cisco ONE, *Open Network Environment*.

Altre attività di standardizzazione sono IETF, ITU-T, IEEE.

Le applicazioni potenziali sono molteplici, si possono realizzare applicazioni di virtualizzazione, di middleware. Oppure routing ottimizzato per certe tipologie di traffico. Si possono realizzare servizi specifici per certe applicazioni.

Il controller parla fisicamente con un API chiamato *southbound* diretto verso gli elementi della rete, e *northbound* orientato verso applicazioni di rete per determinare i requisiti dei vari servizi proposti.

Questo protocollo si basa sul concetto di flusso, i pacchetti vengono classificati ed attribuiti ad un flusso, definito sulla base di diversi criteri, la porta, l'indirizzo MAC o IP o altre informazioni contenute all'interno del pacchetto.

Lo switch inoltra il traffico secondo regole in una tabella di flussi. SDN separa il control plane dal data plane. Il controller esegue il software, anche algoritmicamente complesso su un hardware general purpose.

Le macchine per l'instradamento si comportano sia come router che come switch, in base al loro comportamento, quando utilizzano l'indirizzo MAC si comportano come switch, quando inoltrano pacchetti si comportano come router, vengono chiamati *datapath* per identificarli.

La comunicazione tra il controller ed i datapath avviene tramite lo standard OpenFlow. La tabella dei flussi aiuta a smistare il traffico di questo switch. Nel momento in cui uno switch riceve il primo pacchetto di un flusso, può effettuare una richiesta ad il suo controller per determinare il flusso di questo pacchetto. Le entry vengono quindi inserite dinamicamente nella tabella.

Lo switch si occupa solo dell'inoltro dei pacchetti secondo queste istruzioni, si occupa del data plane, mentre il controller si occupa del control plane.

Ogni switch ha uno o più tabella di flussi usate per inoltrare i pacchetti, una tabella è composta da una flow entry, una tripla di `match`, `action` e `stats`. Il primo controlla il pacchetto e determina se esiste un entry corrispondente in base a vari campi o dati contenuti nel pacchetto. L'azione da svolgere, `action` per quel pacchetto può essere di inoltrare ad una data porta, oppure inviarlo al controller, oppure operare come uno switch o un router. Queste azioni sono molte e rendono questo protocollo molto versatile.

Queste entry possono essere distribute con due tempi di timeout, *idle* e *hard*. Se c'è un flusso che non si presenta, viene cancellato dopo che scade il timer idle. Mentre l'hard timeout è non dipende dal traffico corrente, impone di scartare il flusso dopo che è passato l'intervallo di tempo indicato.

Il controller è un'entità software che deve conoscere l'intera rete in modo dinamico, producendo flow entries ed inviandole alle macchine.

3.4.1 Messaggi OpenFlow

I tipi di messaggi principali sono *PacketIn*, *PacketOut* e *FlowMod*.

Il pacchetto PacketIn viene inviato al controller quando uno switch non riesce ad effettuare match sulla sua tabella di flussi. Questo può inoltrato per intero al controller il pacchetto in questione, che può guardarlo completamente, oppure più semplicemente prende l'header del pacchetto, di default i primi 128 byte. In questo caso il controller riceve anche l'id del buffer nel quale il pacchetto viene memorizzato in attesa della risposta.

Il messaggio PacketOut può contenere un pacchetto modificato, se è in risposta ad un PacketIn viene usato dal controller per istruire allo switch l'invio di un singolo pacchetto. Quando sono in risposta indicano ad uno switch il flusso da usare, contiene oltre all'azione, l'intero pacchetto oppure l'id del buffer nel quale lo switch ha dichiarato di conservare il pacchetto. Se non è in risposta ad un PacketIn, allora contiene un pacchetto ad-hoc creato dal controller, può essere usato per ricostruire la topologia della rete.

Il messaggio FlowMod, *Flow Modification* invia una flow entry, può essere in risposta ad un datapath, per installare, modificare o rimuovere una regola in uno switch. Può avere una priorità, quindi le tabelle di flussi potrebbero essere ordinate per priorità creando una gerarchia, scegliendo la regola a priorità più alta. In caso sia uguale potrebbe essere sovrascritta, come per Open vSwitch.

Con questo pacchetto si possono modificare, rimuovere flow entries, oppure notificare al controller l'istante dell'eventuale rimozione, o può richiederne una verifica logica.

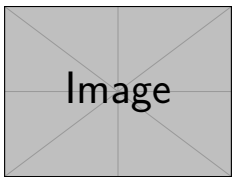
3.4.2 Topologia

Il controller deve avere una mappa della rete, per farlo utilizza dei pacchetti appartenenti al *Link Layer Discovery Protocol* (LLDP), questi appartengono al livello 2 link state, non hanno bisogno di pacchetti IP. II pacchetti vengono mantenuti all'interno della rete, sono spediti periodicamente per esplorarla. Il controller effettua un handshake con tutti gli switch, e questi dichiarano al controller l'elenco delle loro interfacce.

Usando un FlowMod manda a tutti gli switch una regola che li istruisce su come trattare i pacchetti LLDP seguenti, può indicare che tutti i pacchetti LLDP ricevuti devono essere rimandati dentro ad un messaggio PacketIn.

Dopo aver ricevuto questi pacchetti, il controller invia a ciascun switch, e per ciascun sua porta, un pacchetto LLDP, in un messaggio di tipo PacketOut, specificando nell'action che il pacchetto LLDP deve essere inviato sulla porta specificata. Lo switch connesso alla porta su cui è stato inviato questo LLDP, lo spedisce al controller in un pacchetto PacketIn, seguendo la regola inserita inizialmente, specificando la porta da cui è stato ricevuto. Questi pacchetti non possono passare attraverso ulteriori switch o link, poiché essendo al livello 2 rimangono nel link.

In questo modo il controller determina che una certa porta di uno switch è connessa ad una certa porta di un altro switch. Ottiene per ogni link una risposta simmetrica per verificare la topologia.



In base a quanto detto finora ogni rete ha un solo controller, questo ne limita la scalabilità e e l'affidabilità, rappresenta un *single point of failure*. Esistono soluzioni distribuite con controller organizzati gerarchicamente, per gestire il traffico in maniera distribuita, dove ogni controller gestisce un numero limitato di switch, oppure è collegato ad un insieme di switch a cui delega di distribuire regole ad altri switch.

Problemi di scalabilità possono derivare dalla dimensione delle tabelle dei flussi negli switch. Per un certo periodo c'è stata una grande ricerca per determinare come inserire le flow entry in modo efficiente e creare una tabella il più piccola possibile, basandosi su criteri come la priorità delle regole, per cercare di risolvere questo problema.

Aggiungere altri controller aumenta l'affidabilità del sistema, poiché solo i controller conoscono la topologia della rete, tuttavia per aggiungere controller ridondanti bisogna garantire che questi siano sincronizzati tra di loro.

In caso di un guasto un controller deve poter accorgersi del guasto su un link o apparato, determinare la nuova topologia ed installare nuove versioni regole sugli switch affetti. Questo potrebbe non competere con tecnologie consolidate come OSPF.

Anche negli switch ci sono limitazioni, non hanno un buffer illimitato, né tabelle abbastanza grandi. Su una rete SDN è comunque presente una rete sottostante con i loro protocolli di instradamento, su cui viene costruita la SDN, questo garantisce grande libertà per la gestione di una rete.

4 Esaurimento Indirizzi IPv4 ed IPv6

Lo spazio di indirizzamento IPv4 è limitato, con 32 bit di indirizzi si hanno 2^{32} indirizzi diversi, circa quattro miliardi, non è sufficiente per la crescita esponenziale degli apparati di rete, e la loro necessità di connettersi. Questo spazio non è allocato nel modo più efficiente, poiché gli indirizzi sono raggruppati in LAN, i cui indirizzi possono essere solamente potenze di due. Inoltre per molto tempo si usavano esclusivamente con classi, A B e C, con 8, 16 e 24 bit di prefisso.

Per risolvere questo esaurimento si sono sviluppate varie soluzioni.

- La prima è l'indirizzamento per barre e non per classi, in modo da frazionare delle reti in subnet, chiamato CIDR, *Classless Inter-Domain Routing*.
- Un'altra soluzione semplice è la diminuzione dell'assegnazione di indirizzi.
- Ci sono state delle campagne di renumbering su grosse organizzazioni che avevano grandi prefissi `/8`, infatti, alcuni organizzazioni con prefissi anche abbastanza grandi non si sono mai esposte verso la rete.
- Si può utilizzare diffusamente l'indirizzamento privato, con blocchi di indirizzi privati non validi in Internet e non instradati, ma validi per la LAN.
- Un'altra soluzione è il DHCP, *Dynamic Host Configuration Protocol*, che assegna dinamicamente indirizzi privati o pubblici per uso temporaneo.
- Il NAT, *Network Address Translation* è un meccanismo per la condivisione di indirizzi pubblici da parte di diverse macchine che hanno indirizzi privati diversi. Non si possono instradare direttamente i pacchetti per una destinazione remota, devono essere tradotti gli indirizzi mittenti per poter ritornare al router di inoltro, che sostituisce nuovamente l'indirizzo privato.
- Si può moltiplicare la potenza del NAT, considerando anche le porte dello specifico router con il PAT, *Port Address Translation*, che opera a livello di porte.
- La soluzione definitiva è il passaggio a IPv6, con uno spazio di indirizzamento di 64 bit.

Un altro problema che si è dovuto affrontare è l'esaurimento delle memorie dei router, queste sono molto più costose di memorie dedicate ad applicazioni general purpose. Soprattutto grazie a CIDR si sono moltiplicate le entry abbordo delle macchine, e questo problema è peggiorato ulteriormente.

Questo problema venne descritto per la prima vota nell'RFC 1338, nel 1992, anche se avrebbe avuto impatto solo decenni dopo, si cominciarono a cercare alternative, il CIDR venne proposto come soluzione già dall'anno successivo, negli anni seguenti venne affiancato dal NAT e dall'introduzione di IPv6. All'inizio del 2011 lo IANA ha assegnato l'ultimo blocco di prefissi `/8`, da allora lo spazio di indirizzamento disponibili sta diminuendo velocemente.

4.1 Classless Inter-Domain Routing: CIDR

Il CIDR consiste nell'adozione della notazione barra per la suddivisione di reti in sottoreti. Rende possibili il subnetting, ed ulteriori frazionamenti ricorsivi su una stessa LAN. Inoltre, è stato utilizzato per delle campagne di supernetting per riunire insieme diverse LAN IPv4 che condividono lo stesso instradamento, riducendo la dimensione delle tabelle dei router. Le operazioni di subnetting hanno effetti opposti sulle dimensioni delle tabelle di instradamento sulla rete.

4.2 Private Address Allocation

Definiti nell'RFC 1918, alcuni indirizzi sono dedicati ad utilizzo privato, non indirizzabili nell'Internet. Nessun pacchetto può avere questi pacchetti come indirizzo di destinazione, ma può essere presente come indirizzo sorgente. Perché se dentro una LAN si ha uno schema di indirizzamento privato, se avviene un qualche errore su una macchina intermedia, deve poter mandare un pacchetto ICMP a quell'interfaccia avendo un indirizzo privato come sorgente. Se questo intermediario è nella LAN, avrà un indirizzo privato come campo destinazione.

Questi sono sottratti agli indirizzi globalmente unici di Internet. Possono essere utilizzati anche senza essere connessi alla rete, riutilizzando la pila protocollare TCP/IP in un ambiente chiuso, sfruttando le sue capacità. Gli indirizzi IP configurati in queste reti sono indirizzi privati, univoci solo all'interno della rete.

Gli indirizzi privati sono:

- 10.0.0.0/8, da 10.0.0.0 a 10.255.255.255, equivalente ad una classe A. RFC 1918 consiglia di utilizzare due numeri casuali, x e y, per questo tipo di indirizzo: $10.x.y.0/8$, al posto dei due zeri centrali per evitare conflitti in caso due reti vengano unite in futuro.
- 172.16.0.0/12, da 172.12.0.0 a 172.31.255.255, equivalente a 16 reti di classe B.
- Infine 192.168.0.0/16, da 192.168.0.0 a 192.168.255.255, equivalenti a 256 reti di classe C.

4.3 Network Address Translation: NAT

Non è possibile comunicare sulla rete usando indirizzi privati, è necessario convertirli in indirizzi globalmente univoci, questo viene realizzato dal NAT.

Il NAT ha uno scopo più ampio, non solo di convertire indirizzi privati in indirizzi pubblici, ma di collegare due interfacce su due *realm* diversi. Un realm è una rete dove gli indirizzi IP hanno un significato univoco. In queste reti non ci sono ambiguità sui numeri IP. Poiché in Internet gli indirizzi IPv4 sono univoci, rappresenta un realm.

Diversi realm possono avere indirizzi sovrapposti, ogni rete che utilizza indirizzi privati è un realm a sé stante. Il pacchetto in transito deve essere convertito per essere compatibile tra i due realm. Questa sostituzione è completamente trasparente dal punto di vista degli ES. I realm attraversati potrebbero essere molteplici, potenzialmente applicando NAT varie volte. La macchina che applica NAT possiede un blocco di indirizzi IP univoci sui realm dove comunica, possibilmente annunciati in Internet se parla BGP. I pacchetti diretti ad un indirizzo pubblico in questo blocco vengono mandato a questo router.

Si considera un ES nella LAN gestite da un router NAT, con un indirizzo privato x in un realm, che vuole comunicare con un apparato su un realm differente. In condizioni normali il pacchetto attraverserebbe il router senza modifiche, ma per poter accedere all'altro realm bisogna sostituire l'indirizzo privato dal pacchetto. Allora il NAT sceglie tra uno dei suoi indirizzi pubblici y e lo sostituisce a x, salvando l'associazione su una tabella di traduzione, *NAT table*, $x : y$. Dopo aver modificato il pacchetto lo inoltra sulla rete di destinazione. Al ritorno del pacchetto questo viene tradotto nuovamente, sostituendo nel campo del destinatario y con x, l'indirizzo privato.

Dopo aver terminato la conversazione l'associazione può essere gestita secondo varie discipline. In alcuni NAT si usano gli stessi indirizzi pubblici per una stessa macchina privata, oppure per ogni connessione di macchine private, oppure li assegnano per un certo periodo di tempo ad una data macchina. Ci sono molte politiche di gestione, ma rimane comunque alla base l'associazione tra un indirizzo univoci su due realm.

Questo può consentire di migrare da un provider ad un altro senza effettuare renumbering, inserendo un NAT.

Il router gestisce la corrispondenza o *binding* tra gli indirizzi dei due realm tramite una tabella di traduzione. Questo binding può essere statico, dove la tabella viene configurata manualmente, oppure dinamicamente, in cui la tabella cambia nel tempo a seconda del traffico, e gli indirizzi pubblici vengono assegnati alle macchine che ne hanno bisogno.

In una rete locale gli apparecchi sono classificati in base a diverse esigenze di connettività, in base a questi criteri si individuano macchine che non richiederanno mai NAT, alcune macchine che lo richiederanno solamente come client, ed altre macchine che lo richiedono completamente, queste possono offrire servizi raggiungibili dall'esterno.

Per rendere quest'operazione invisibile agli AS bisogna modificare sia il pacchetto IP sia, in caso, il pacchetto TCP. Bisogna modificare l'indirizzo destinazione e mittente, ed anche l'header checksum. Anche se la maggior parte dei router non controlla questo campo, poiché è tempo "sprecato", molti router, infatti, sospendono il controllo del checksum. Anche se non è controllato, deve essere ricalcolato. Mentre nel pacchetto TCP bisogna ricalcolare il checksum, poiché viene calcolato anche sui campi del pacchetto IP.

4.4 Porta Address Translation: PAT

Poiché bisogna avere un numero elevato di indirizzi pubblici per effettuare il NAT, spesso si preferisce il PAT, chiamato anche *IP Masquerading* o *Network Address and Port Translation* (NAPT). Questo consente a molte macchine con indirizzamento privato di utilizzare lo stesso indirizzo pubblico, si può diminuire il numero di indirizzi pubblici assegnati ad un unico router, fino ad un singolo indirizzo pubblico. È il sistema operativo ad assegnare una porta, quando realizza una socket, per una determinata connessione. Si può influire su questa scelta, ma non è di interesse dal lato client usare una specifica porta, per cui è possibile modificare questa porta in modo arbitrario senza intralciare le connessioni.

Effettuando la traduzione l'indirizzo pubblico è necessariamente l'unico indirizzo pubblico disponibile, nel pacchetto IP si inserisce l'indirizzo IP del router. Nel pacchetto TCP bisogna modificare le porte di ingresso ed uscita e ricalcolare la checksum. Usando il PAT si cancella la porta effettiva utilizzata del client, ma la connessione ne è indipendente.

Quando si usa più di un indirizzo IP pubblico si parla anche di multi-PAT. Sulla tabella PAT quindi vengono specificate solo le porte. A questo punto non si hanno più problemi di scalabilità.

Queste due tecniche NAT e PAT sono ampiamente utilizzate, ma pongono altri tipi di problemi.

- Violano il principio end-to-end, definito nell'RFC 1958, che afferma che lo stato di una connessione deve essere gestito solamente dagli ES, e non può essere condiviso con le macchine intermedie. Il livello di trasporto si assume faccia comunicare due macchine, in modo trasparente rispetto ai livelli sottostanti. Questo ha effetti sulla connettività end-to-end, poiché le informazioni sulle porte e indirizzi IP sono memorizzati in tabelle degli apparati intermedi che effettuano NAT o PAT.
- I protocolli che menzionano gli indirizzi IP non funzionano correttamente, come FTP, per ricostruire la connessione bisognerebbe ricalcolare tutti i campi ed i sequence number che dipendono dal cambio delle porte ed indirizzi IP.
- In caso di frammentazione la sostituzione degli indirizzi devono essere coerenti in ogni frammento.
- I DNS devono rispondere in due modi diversi per le macchine interne ed esterne, per le prime i nomi privati devono corrispondere agli indirizzi privati mentre quelli pubblici ad indirizzi pubblici.
- La necessità di ricalcolare vari campi riduce notevolmente le prestazioni di router con NAT e PAT abilitato.

4.5 DHCP

Il protocollo DHCP assegna dinamicamente ed autonomamente configurazioni di rete agli ES di una rete. È un'alternativa alla configurazione manuale, distribuendo indirizzi pubblici o privati di un insieme di macchine. Può essere utilizzato per configurare utenti temporaneamente attivi sulla rete. Le configurazioni distribuite da DHCP contengono almeno l'indirizzo IP e la netmask, e gli indirizzi del default gateway e del server DNS. Queste sono le informazioni che consentono ad un host di accedere ad Internet.

Un server DHCP ha una serie di indirizzi pubblici o privati, molto spesso statici, il server risponde solo ad alcune macchine in base al loro indirizzo MAC, configurato dall'amministratore. L'assegnazione di una configurazione è temporanea, per un prefissato intervallo di tempo breve.

Questo protocollo è diviso in quattro fasi.

1. Il protocollo prevede un DHCP DISCOVER, un client manda un broadcast in cui annuncia che non ha una configurazione. Viene effettuata una discovery all'avvio di una macchina, quando scade un'assegnazione o quando richiesto dall'amministratore. In mancanza di una risposta viene iterata ogni cinque minuti. Questo pacchetto DHCP viaggia su UDP.
2. Il server, o i server, DHCP sulla LAN offrono una possibile configurazione con un DHCP OFFER. Più server possono rispondere contemporaneamente allo stesso client, che sceglie una delle possibili configurazioni. Le richieste essendo broadcast non attraversano i router,

ma un router potrebbe essere configurato come proxy DHCP che inoltra la richiesta ad uno specifico DHCP remoto.

3. Il client in base a queste configurazioni accetta esplicitamente una di queste configurazioni con un pacchetto DHCP REQUEST.

4. Il server conferma l'assegnazione temporanea con un DHCP ACK.

4.6 IPv6

Il protocollo di rete IPv6 mira a sostituire IPv4, con una versione migliore e più efficiente del protocollo di rete livello tre, per aumentarne le prestazioni. Implementa strategie per sostituire il protocollo ARP. Un indirizzo IPv6 è composto da 128 bit, ed ha meccanismi per accorciare la scrittura, utilizzando indirizzi molto corti principalmente per macchine importanti.

L'header IPv4 contiene campi rimossi da IPv6, dato che alcuni non sono mai stati utilizzati, come i campi relativi alla frammentazione. Alcune tecnologie possono richiedere una MTU più piccola, e questo richiede che il router frammenti i pacchetti di livello tre, usando questi campi IPv4, rimossi da IPv6, per tenere traccia dei frammenti. Questi frammenti vengono poi ricomposti a destinazione, aumentando il carico per la macchina destinazione. Se viene specificato che un pacchetto non può essere frammentato ed un router sul tragitto individua una strettoia, questo viene scartato e viene inviato un pacchetto ICMP. La scheda di rete di una macchina condivide le MTU dei singoli livello protocollari a quelli superiori.

È stato rimosso il campo checksum, che andava ricalcolato ed in teoria controllato ad ogni hop. Questo nella patrica non viene quasi mai controllato dai router per aumentare l'efficienza, che dovevano comunque ricalcolarlo perdendo di efficienza.

In IPv6 mancano le opzioni, campi di lunghezza dinamica, per rendere la struttura del pacchetto stabile. Se il pacchetto è privo di *extension header* è sempre di 40 byte, i primi 8 sono l'header, ed i restanti 32 per gli indirizzi sorgente e destinazione. Un pacchetto IPv6 ha i seguenti campi:

- Il campo *version* di 4 bit, di valore `0110` (6) per indicare IPv6

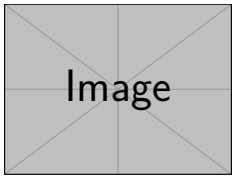
- Il campo *traffic class* di 8 bit, simile al TOS in IPv4, può rappresentare la priorità del pacchetto nel traffico Internet. Può essere usato da ISP per rispettare accordi commerciali tra di loro, assegnando ai pacchetti diverse classi di servizio

- Il campo *flow label*, di 20 bit, di utilizzo ancora poco chiaro. Quest'etichetta ha lo scopo di individuare il flusso a cui il pacchetto appartiene in modo da effettuare un instradamento veloce. Due flussi in IPv6 sono uguali solo se condividono indirizzo e porta destinazione ed anche il valore di questo campo.

- Il campo *payload length* di 16 bit, specifica la lunghezza dei dati nel pacchetto, in byte, fino ad un massimo di 64 KB. Per pacchetti di dimensioni maggiori si usa l'opzione *jumbo payload*

- Il campo *next header* di 8 bit specifica l'header successivo, se presente. Permette di specificare degli *extension header* che contengono anch'essi il campo next header, quindi è possibile realizzare una catena di header, simile alle opzioni in IPv4.

- Il campo *hop limit* di 8 bit, è essenzialmente il TTL in IPv4
- I campi *source e destination address* di 16 byte ciascuno, corrispondono agli indirizzi sorgente e destinazione



L'extension header rappresenta un nuovo metodo per implementare le opzioni, questi extension header possono essere di routing, o di codifica, o header di livelli superiori.

Ci sono alcuni header che se sono presenti devono essere letti necessariamente. Un IS guarda gli header fino a quando lo riguardano, sono ordinati in modo da avere opzioni diverse, dedicate.

1. Header IPv6
2. L'header *Hop by Hop* (00) contiene informazioni rilevanti per ogni nodo attraversato dal pacchetto, in questo header è contenuta l'opzione jumbo payload
3. L'header *Destination Options* (60) contiene informazioni opzionali, in questa posizione implica che tutti i nodi intermedi specificati nel *Routing* devono leggere queste opzioni
4. L'header *Routing* (43) indica una lista di router da attraversare, aumentando le prestazioni rispetto ad IPv4. Ogni router in questa lista aggiorna l'indirizzo destinazione del pacchetto con l'indirizzo IPv6 del prossimo router nella lista
5. L'header *Fragment* (44) viene usato dall'host mittente solo per l'host destinatario, per indicare la frammentazione del pacchetto. Implica che i router intermedi non possono frammentare ulteriormente il pacchetto. IPv6 prevede una MTU minima di 1280 byte, altrimenti bisogna frammentare al livello due. Ogni nodo deve implementare una sua strategia di *MTU Path Discovery*
6. L'header *Authentication* (51) offre il supporto nativo di IPv6 per la sicurezza, IPsec, ma non tutte le implementazioni di IPv6 seguono questa convenzione. Quest'header offre un modo per verificare l'autenticità dell'indirizzo mittente e che il pacchetto non sia stato manomesso in transito
7. L'header *Encapsulating Security Payload* (50) garantisce che solo il destinatario autorizzato possa leggere il contenuto del pacchetto, cifrando il contenuto di ciò che segue.
8. L'header *Destination Options* (60) in questa posizione, alla fine della catena, contiene opzioni esaminate solo dal nodo di destinazione

Gli indirizzi IPv6 vengono rappresentati in notazione esadecimale, in un URL gli indirizzi IPv6 sono inclusi tra parentesi quadre. Scompare l'indirizzo di broadcast, che causava per IPv4 un interrupt a tutte le macchine che lo ricevono.

In una configurazione DHCP, la macchina registra sulla sua scheda di rete gli indirizzi MAC del server DHCP in un insieme per realizzare un insieme di indirizzi multicast. Via software si può dire ad un scheda di rete di raccogliere altri tipi di pacchetti. Utilizzando uno sniffer, questo raccoglie tutti i pacchetti come se fossero direttamente inviati alla macchina. Dall'indirizzo IPv6 si può estrarre l'indirizzo MAC, per aggiungerlo alla lista di indirizzi anycast.

C'è una forte presenza di macchine con uno stesso numero, queste macchine anycast sono macchine che offrono servizi, e la richiesta viene soddisfatta dalla macchina più vicina.

Gli indirizzi IPv6 possono essere privati per vari livelli di privatezza. È molto più frazionato rispetto ad IPv4, questi si chiamano *Reserved*. Altri tipi di indirizzi sono *Global Unicast*, *Unique Local Unicast*, *Link Local Unicast* e *Multicast*. L'indirizzo `0:0:0:0:0:0:0:0`, o `::` indica l'assenza dell'indirizzo, e può essere usato nella richiesta iniziale DHCP, equivalente a `0.0.0.0` in IPv4.

Ci sono indirizzi che contengono dentro indirizzi IPv4 nascosti, usati in alcuni tipi di transizioni IPv4-Ipv6, ora deprecati. Mentre nelle reti IPv4 c'è un'alta frammentazione e subnetting applicando il CIDR, in IPv6 la lunghezza del prefisso è fissa e diversa dal'identificativo dell'host. C'è comunque la notazione barra per aggregarle a criteri di routing, ma negli indirizzi è sempre presente un prefisso di 64 bit, e quindi hanno una netmask stabile.

La parte di prefisso è utile ad identificare la subnet, mentre l'identificativo dell'host può essere specificato manualmente oppure automaticamente. Aggiungendo a questo indirizzo prefissi diversi si ottengono indirizzi differenti per la stessa macchina. Quando la macchina trasmette sceglie uno di questi indirizzi da utilizzare per la trasmissione.

L'identificativo della macchina si può ottenere automaticamente dall'*Interface ID* in formato EUI-64. Questo sfrutta la caratteristica degli indirizzi MAC di essere univoci a livello globale per realizzare identificativi per l'interfaccia altrettanto univoci. Esiste una procedura per passare da EUI-48, l'indirizzo MAC ad EUI-64, l'identificativo di un'interfaccia IPv6, aggiungendo tra i primi 24 bit la stringa `ff:fe` prima dei seguenti 24, complementando il penultimo bit del primo byte. Tuttavia essendo l'indirizzo MAC statico, anche l'EUI-64 è statico, e potrebbe potenzialmente permettere di tracciare un utente. Quindi per la privacy sono stati introdotti meccanismi per generare alternativamente l'EUI-64, con una stringa casuale di 64 bit.

Per *link* si intende una rete fisica come esempio una LAN, un collegamento punto-unto o anche una rete geografica comune. Nodi sullo stesso link sono detti neighbor, o vicini. Per *site* invece si intende un insieme di link gestiti da un unica organizzazione. Questi hanno un loro prefisso *link local unicast* ed analogamente *site local unicast*, utilizzabili dentro la LAN. Il prefisso link local è `fe80::` e site local `fec0::`. Gli indirizzi IPv6 possono avere un ambito, quindi possono essere validi solo su un link, un site, o globalmente. Non possono essere usati per comunicare con altri nodi al di fuori di questo ambito.

Mentre i prefissi *global unicast* vengono assegnati dalla IANA alle organizzazioni continentali e regionali, ed in seguito vengono assegnati facilmente ai singoli IPS, e successivamente utenti.

La distribuzione degli indirizzi verso il basso è automatica, e potrebbe essere possibile sostituire gli indirizzi in modo trasparente. Si potrebbero assegnare tutti gli indirizzi che iniziano per una certa sequenza in una zona geografica limitata. Tutti questi pacchetti vengono instradati osservando

una barra molto corta. Quindi questi prefissi vengono suddivisi in questo modo, i primi 32 bit di prefisso sono assegnati per un ISP, mentre blocchi più grandi a RIR. Successive suddivisioni in barra 48 vengono affidate alle singole organizzazioni, da 48 a 64 è possibile fare subnetting sul site dell'organizzazione per realizzare link.

Invece di avere indirizzi broadcast si utilizzano indirizzi multicast, cominciano sempre con `ff`, questo è seguito da una flag di 4 bit, per indicare se si tratta di un indirizzo permanente o temporaneo. I seguenti quattro bit individuano uno scope specifico che può andare da nodo, a link, site, fino a globale. I seguenti 112 bit identificano il group ID, un gruppo multicast valido su un certo ambito a cui un nodo si può iscrivere.

Indirizzi multicast importanti sono dati da `ff02::1:ffxx:xxxx`, questi indirizzi sono chiamati *solicited-node*. Si crea un gruppo multicast per ogni indirizzo che non è multicast.

Questo indirizzo solicited node viene usato per sostituire il protocollo ARP e ICMPv4.

Gli indirizzi anycast non sono distinguibili da indirizzi unicast, sono indirizzi unicast assegnati ad un gruppo di interfacce, normalmente di nodi diversi. Bisogna specificare ai nodi che gli è stato assegnato un indirizzo anycast.

4.6.1 ICMPv6

Pacchetti ICMPv6 riprendono le funzionalità di ICMPv4, ed assumono nuove funzionalità e responsabilità gestite da protocolli ARP e IGMP su IPv4. Il pacchetto ICMPv6 presenta una specie di header dove viene specifico il tipo ed i sottotipo, degli header precedenti, seguito dai dati di ICMP.

Contenuti dentro un pacchetto IPv6, l'header ICMPv6 è indicato dal campo next header di valore 58. L'header ICMP ha tre campi, il tipo, il codice di errore, essenzialmente un sottotipo, e la checksum. Il primo bit del campo type distingue tra due classi di messaggio, di segnalazione di errore o di messaggi informativi.

I pacchetti IPv6 vengono frammentati end-to-end, quindi bisogna garantire che non vengano scartati durante il percorso, non potendo frammentare il pacchetto su un nodo intermedio, si vuole determinare l'MTU pià piccola da considerare collo di bottiglia per ridurre la dimensione del pacchetto IP. La procedura di *Path MTU Discovery* è basata su messaggi ICMPv6 per segnalare pacchetti troppo grandi. Inviando pacchetti ICMPv6 progressivamente sul cammino per individuare il collo di bottiglia.

La funzionalità di *Neighbor Discovery* è simile all'ICMP redirect di ICMPv4. Per inviare un pacchetto senza conoscere l'indirizzo, in IPv4 il processo si ferma e lancia un processo di discovery di livello due per individuare l'indirizzo MAC del destinatario, con una richiesta ARP. Questo può esistere sulla rete, e quindi raggiungile direttamente, altrimenti si deve inviare la richiesta al router di default. Questo processo di neighbor discovery in IPv6 viene effettuato tramite ICMPv6. Questi pacchetti si occupano della risoluzione degli indirizzi. Sostituendo ARP request e reply con *neighbor solicitation* e *advertisement*, utilizzando gli indirizzi multicast all'interno di un link. I messaggi quindi non possono uscire dal link dove sono mandati, diminuendo il traffico sulla rete rispetto a broadcast di livello due. Il messaggio ICMPv6 redirect è simile alla controparte IPv4. Questa informa il router che è presente un router migliore sul link per raggiungere la destinazione, oppure che la destinazione si trova sullo stesso link. L'indirizzo solicited node si ottiene aggiungendo gli ultimi 24 bit dell'indirizzo al prefisso `ff02::1:ff00:0/104`, iscrivendo ogni interfaccia ai gruppi

multicast corrispondenti, rappresentati da opportuni gruppi multicast di livello due. La probabilità che due macchine abbiano questo stesso indirizzo multicast è molto bassa, poiché gli ultimi tre byte corrispondenti sono quelli dell'indirizzo MAC assegnati dal costruttore, quindi è improbabile siano uguali. Ogni interfaccia viene iscritta ai gruppi multicast corrispondenti a tutti gli indirizzi assegnati da essa. L'uso degli ultimi 24 bit dell'indirizzo IPv6 permette una riduzione delle collisioni, poiché dipendono dall'indirizzo MAC. L'indirizzo destinatario a livello due si può ottenere dall'ID del gruppo a cui è iscritta la macchina.

Chi deve risolvere l'indirizzo IPv6 in un indirizzo di livello MAC, invece di mandare un pacchetto broadcast, si calcola l'indirizzo multicast IPv6, da cui si ottiene l'indirizzo MAC di livello due per inviare il pacchetto sull'indirizzo multicast. Il ricevente risponde a questo messaggio inviando all'indirizzo mittente, un pacchetto di neighbor advertisement con il suo indirizzo IPv6 specificato nella porzione dati del pacchetto. In IPv6 c'è una cache che sostituisce l'ARP cache, una sorta di mappa che tiene conto della mutua raggiungibilità dei nodi vicini. Per i nodi vicini controlla che siano raggiungibili da neighbor solicitation, per i nodi remoti controlla che siano raggiungibili attraverso il router next-hop.

L'algoritmo di *Neighbor Unreachability Detection* (NUD) permette di individuare rapidamente cambiamenti di indirizzi fisici o guasti nella rete. Un'altra funzionalità interessante di IPv6 è l'auto-configurazione degli indirizzi. Permette ai singoli di nodi di generare un indirizzo IPv6 stabile in modo stateless, senza utilizzare server DHCP. La macchina può prendere l'EUI-64 dell'interfaccia come identificatore, mentre come prefisso di rete utilizza prefissi link-local. In questo modo può comunicare con il router per ottenere un prefisso global unicast. Tuttavia, il server DNS deve essere necessariamente specificato a mano, oppure tramite protocolli diversi.

Il meccanismo di *Router Advertisement* è un algoritmo dove i router inviano periodicamente su ogni link dei pacchetti di questo tipo, dove si identificano e specificano se siano disponibili agli host per diversi servizi. Specificano il loro indirizzo di livello due e indirizzo IPv6 link-local, indicano se sono disponili come default router. Specificano altri parametri come TTL, MTU, ed inoltrano una lista di prefissi utilizzabili sulla rete, assegnati al link, oppure globalmente unici. Quando un router comunica un prefisso, gli altri 64 bit si possono ottenere dall'indirizzo MAC, per comunicare con il resto del mondo.

Questi router advertisement vengono inviati ad intervalli regolari, c'è anche la possibilità di richiedere queste informazioni, tramite una *router solicitation*, inviato all'indirizzo multicast del link, quindi a tutti i router del link. Questa è una richiesta per un router advertisement, inviato in unicast all'indirizzo mittente del router solicitation Ci sono meccanismi di riconoscimento di indirizzi duplicati, a bordo della macchina si controlla se un'altra nodo ha già lo stesso indirizzo, specificando come indirizzo sorgente l'*unspecified address*, `::`. Se non riceve alcuna risposta, adotta quell'indirizzo.

I prefissi contenuti in un router advertisement sono associati ad un tempo di vita, un timer oltre il quale quel prefisso è da non considerarsi più valido, e non può essere riassegnato all'interfaccia. Questi tempi associati agli indirizzi sono *valid lifetime* e *preferred lifetime*, il primo è il tempo per cui un indirizzo può essere associato all'interfaccia, ed il secondo è il tempo per cui l'indirizzo può essere utilizzato per nuove connessioni. Questo permette il renumbering automatico degli host, tramite nuove router solicitation, e configurazioni automatiche.

Gli indirizzi e gli altri parametri di configurazione come i server DNS possono essere configurati

manualmente. È possibile effettuare una configurazione interamente manuale, con DHCPv6, su specifica dell'auto-configurazione stateless.

Le specifiche stateful nella stateless riguardano due flag dedicate nel router advertisement. Queste sono *managed address configuration*, che indica se l'host deve ottenere indirizzi da parte di DHCPv6, mentre un secondo campo indica se deve utilizzare quest'ultimo per ottenere altre informazioni di configurazione. Se è il primo è settato, lo anche questo.

4.6.2 Multihoming

In generale un nodo IPv6 può avere diversi indirizzi, di diverso ambito, link local, site local o global unicast, avendo almeno un indirizzo link local per interfaccia. Il router assegna tanti indirizzi, poiché ne riceve tanti dall'esterno e poiché una stessa macchina potrebbe avere più provider, ed ogni provider inietta sul router prefissi diversi. Quando si decide di parlare con un interlocutore remoto, è possibile entrambi abbiano un indirizzo globalmente unico, ma più spesso questo potrebbe avere diversi indirizzi IPv6. La scelta su quale indirizzo usare per la comunicazione impatta le prestazioni, influisce sulla rotta attraverso cui avviene la comunicazione. La scelta migliore può cambiare nel tempo, inoltre un provider potrebbe avere dei problemi e quindi bisognerebbe cambiare provide rotta attraverso un altro insieme di indirizzi IPv6. La scelta degli indirizzi di comunicazione è una parte critica nella comunicazione su IPv6.

Ci sono diverse scelte possibili per un ES per gli indirizzi mittente e destinazione di un flusso, il protocollo di scelta viene chiamato *source address selection*. Se si vuole connettere ad un nome di domino, ottiene dal server DNS una lista di possibili indirizzi. Tipicamente preferisce indirizzi IPv6, anche se è possibile andare ad IPv4, in base alla condizione del destinatario. Su IPv4, si provano gli indirizzi di destinazione uno per volta per valutare quale sia più appropriato, applicando una serie di regole. Questa scelta può essere sovrascritta dall'amministratore di sistema o dell'applicazione.

Il multihoming si basa su BGP, se si vuole comunicare tra due ISP diversi, attraverso BGP. Se le due zone sono separate e sono stati assegnati più indirizzi, BGP è costretto ad inoltrare tutti questi prefissi, detti *Provider Indipendente* (PI), non aggregabili. Altrimenti se l'AS di peering è una subnet si dice che gli indirizzi sono *Provider Aggregatable* (PA), ed è sufficiente annunciare la subnet. Per ogni indirizzo PI annunciato, si inserisce una entry al routing interdominio.

Gli effetti di questo fenomeno sono l'aumento esponenziale delle tabelle di instradamento a causa della frammentazione delle reti, e quindi dell'inoltro di queste nuove rotte.

In IPv6 avendo due indirizzi diversi, si ha un insieme di indirizzi, e si sceglie quale di questi utilizzare, così da mitigare questo problema.

Ma se non si parla BGP non si ha possibilità alternativa, tutte le connessioni che usano questo link potrebbero andare in fault. Inoltre, non si ha bilanciamento.

Un approccio proposto per risolvere questo è avere una scelta dinamica dell'indirizzo IP, sia per identificare la macchina sia per identificare il provider attraverso cui la si raggiunge. Separando queste due funzioni si separa il *locator*, la posizione del nodo nella topologia della rete, dall'*identifier*, per indirizzare il nodo agli strati superiori.

Si utilizza un ID per i livello superiori, *Upper-Layer ID* (ULID), per comunicare con protocolli di strato superiore, assegnandone uno per nodo. Questi indirizzi usati sono solamente identificativi, appartengono effettivamente ad un'interfaccia del destinatario, ma rappresenta solamente un ID

della macchina, che può essere per ogni nodo o per ogni processo all'interno del nodo. Questi sono effettivamente due indirizzi IPv6.

Il locator è l'indirizzo IP usato per instradare il pacchetto, identifica la posizione del nodo nella rete. L'idea è quella di usare il locator solamente al livello tre nella fase di instradamento, seguito da uno strato `shim6`, sempre al livello tre, per separare l'indirizzo utilizzato, locator, e l'indirizzo rappresentativo, ULID, delle due macchine nella comunicazione. Gli strati superiori sono convinti di utilizzare gli indirizzi ULID. Questo decide la coppia di indirizzi sorgente e destinazione da utilizzare anche rispetto alla disponibilità e le tempistiche dei pacchetti. Si riesce a modificare gli indirizzi identificativi dinamicamente, senza tagliare la connessione, accordandosi tra i peer su nuovi indirizzi.

La comunicazione si apre usando ULID, scelto da source address selection, se la connessione è persistente si inizializza shim6, per evitare overhead su connessioni brevi. Se il peer non risponde si continua senza, altrimenti negoziano tramite extension headers i *context tags*, che indica il contesto, il locator e l'ULID attualmente in uso. In caso di problemi i peer tentano gli altri locator, fino a trovare una coppia funzionante, modificando tutti i pacchetti in transito verso il peer, sostituendo dagli ULID ai locator in uso. Inoltre, inserisce nel context tag questi indirizzi per informare il peer la connessione corrispondente al pacchetto. Tutto questo mentre i protocolli superiori continuano a vedere l'indirizzo ULID come quello utilizzato.

Questo è un meccanismo simile al NAT, che lavora sugli host, quindi non rompe la convenzione end-to-end. Lo stesso si può usare con meccanismi sofisticati di SDN. Questa separazione non è stata introdotta nel protocollo quando è stato realizzato, poiché ha un effetto a cascata anche su livelli inferiori. Questo è facile risolvere utilizzando degli identificatori end-to-end, i *context tag* shim6 che identificano una connessione.

4.6.3 Transizione

Il livello di rete ha la caratteristica di essere uniformante, tutte le macchine intermedie devono avere lo stesso protocollo per poter parlare IPv4 o IPv6. Per cui sono necessari meccanici di transizione per passare in maniera definitiva da IPv4 a IPv6, poiché è estremamente difficile cambiare protocollo a questo livello.

IPv6 deve essere compatibile a IPv4 e deve permette il passaggio da IPv4 per la sua adozione. Adesso IPv6 è adottato in parallelo rispetto a IPv4 con apparati dual stack, ed il processo di transizione potrebbe durare decenni.

Il processo di transizione è diviso in tre fasi:

1. In un momento IPv6 componeva delle isole separate nell'Internet dove si poteva comunicare con IPv6, ma utilizza prevalentemente i servizi esistenti di IPv4.

2. Nella seconda fase i due protocolli coesistono nella diffusione globale, le varie isole di IPv6 si consolidano ed è possibile comunicare nella maggior parte di Internet solamente con IPv6.

3. Nella terza fase l'adozione di IPv6 supera la diffusione di IPv4, che rimane utilizzato solo su delle isole, e le reti IPv4 si appoggiano sull'infrastruttura IPv6.

Meccanismi per facilitare la transizione possono essere basati su apparecchiatura dual stack, con due pile protocollari, IPv4 e IPv6, i tunnel, in disuso, configurati manualmente per superare oceani in modo da comunicare solamente su uno stesso protocollo. Altri meccanismi possibili sono traduttori di protocollo, come SIIT e NAT64.

Il dual stack è l'approccio più semplice, ma non riduce il fabbisogno degli indirizzi IPv4, non ne riduce l'adozione. Le applicazioni che supportano IPv6 possono utilizzare entrambi i protocolli. Anche se è molto semplice, richiede una doppia gestione dell'infrastruttura e non integra la rete IPv6 con quella IPv4.

I client su di un host dual stack tentano di stabilire una connessione IPv6, e solo se non ci riesce stabilisce una connessione IPv4. Il comportamento è descritto da RFC 6555, il client tenta di instaurare due connessioni, una basata su IPv4 l'altra su IPv6, se il server risponde con un ack ad entrambi i protocolli, resetta una delle due connessioni.

Supponendo di essere in una fase avanzata, con una rete di backbone di IPv6, e parti periferiche con IPv4, le due reti non hanno apparecchiature dual stack, per cui bisogna usare dei protocolli di traduzione, che si comportano come un NAT, passando il traffico attraverso il nodo traduttore. Questo può provocare una serie di altri problemi di scalabilità, affidabilità, ecc.

Lavorando a livello tre ci sono due soluzioni, SIIT e NAT64. Queste mappano indirizzi IPv4 nello spazio IPv6, essendo molto più grande, come particolari indirizzi IPv6. Ci sono vari modi ufficiali e non ufficiali per effettuare questa traduzione. Un NAT normale è stateful, e questo ne limita le capacità.

Il primo meccanismo si chiama *Stateless IP/ICMP Translation* (SIIT), generale, che offre la possibilità di comunicare attraverso un traduttore.

Tutto lo spazio indirizzamento IPv4 è mappato dentro una porzione di IPv6. Gli indirizzi destinazione IPv4 sono mappati su indirizzi IPv6, dove gli ultimi 6 byte sono composti da `::ffff:a.b.c.d`, dove `a.b.c.d` è l'indirizzo IPv4. La macchina traduttrice si occupa di inoltrare questo indirizzo mappato. I nodi IPv6 ottengono indirizzi IPv4 temporanei che vengono mappati in indirizzi IPv6 IPv4-*translated* del tipo `::ffff:0:a.b.c.d` e usati come indirizzo sorgente. Il traduttore traduce i pacchetti in transito, con una traduzione stateless, non è necessario salvare informazioni aggiuntive. Inoltre, non è importante la macchina da cui entra o esce, poiché il processo è stateless. Questo richiede modifiche alle implementazioni IPv6, e richiede la presenza di assegnazione dinamica di indirizzi temporanei, questi devono essere inoltrati verso l'esterno come se fossero appartenenti alla rete. Richiede di gestire il routing per gli indirizzi IPv4-translated all'interno del site.

Il NAT64 è più elegante, ma richiede uno stato parziale. Si chiama così poiché ricorda il NAT, e fonde le soluzioni del SIIT con il NAT.

Il nodo traduttore dispone di un pool di indirizzi IPv4 che vengono assegnati ai nodi che lo utilizzano, ed il NAT64 mantiene lo stato delle associazioni. Gli indirizzi IPv4 vengono rappresentati in IPv6 aggiungendo i 32 bit dell'indirizzo IPv4 ad un prefisso di 96 bit di instradamento verso il traduttore. Mentre la mappatura da IPv6 a IPv4 è dinamica prendendo un indirizzo dal pool disponibile, per cui deve tenere traccia solo dell'associazione da IPv6 a IPv4 e non viceversa, poiché è deterministica.

La traduzione dei pacchetti in transito avviene come in SIIT. Inoltre, richiede uno speciale DNS chiamato DNS64, che aggiunge delle funzionalità, pensato ad un client IPv6 per connettersi ad un

servizio IPv4. Il DNS64 server consulta il DNS locale, cercando di capire gli indirizzi della macchina del servizio, poiché potrebbe avere indirizzi IPv6. Se ha solo indirizzi IPv4 nota il problema. Allora restituisce invece l'indirizzo IPv6 tradotto che codifica la destinazione.

Questa traduzione è trasparente ai nodi interessati e si può accoppiare con il PAT, usando un solo indirizzo IPv4 invece di una pool di indirizzi per molti client. Tuttavia presenta gli stessi problemi del NAT per IPv4, non è un meccanismo end-to-end. Nonostante questi problemi il NAT IPv4 è ampiamente adottato e molte applicazioni già supportano NAT.

5 Border Gateway Protocol: BGP

5.1 Introduzione al Routing Inter-Dominio

Tutti gli ISP utilizzano un protocollo di comunicazione intra-dominio, che sia IS-IS, OSPF o RIP, sotto c'è comunque un instradamento IP tradizionale, come per SDN.

Ogni organizzazione è formata da una collezione di router e LAN con una sola amministrazione. Un algoritmo di routing può essere utilizzato per aggiornare le tabelle di instradamento di questi apparati.

Quando più organizzazioni uniscono le proprie reti, si crea l'Internet, aggiungendo zone di demarcazione, le LAN aggiuntive per connettere diverse organizzazioni. Per avere una connettività globale è necessario che vengano passate informazioni di routing attraverso queste zone di demarcazione per permettere la comunicazione tra diverse organizzazioni. Si vorrebbe poter aggiornare le tabelle di instradamento di tutte le macchine.

Si potrebbe realizzare un singolo algoritmo di instradamento per gestire tutte le reti, ma non è realizzabile anche con un algoritmo scalabile come OSPF. Si potrebbero aggiornare le tabelle manualmente, ma questo non è affidabile. Altrimenti si possono usare EGP per aggiornare automaticamente le tabelle di instradamento.

Agli albori della rete si usava un unico protocollo e le configurazioni di ciascuna macchina influivano sulla configurazione complessiva della rete.

Questa soluzione è tecnicamente difficile, e lenta a convergere, sarebbe difficile da implementare e rilasciare su più organizzazioni con risorse vastamente diverse.

Aggiungere rotte statiche è un'opzione alternativa, ma sono difficili da aggiornare e mantenere.

L'unica ipotesi praticabile è utilizzare un protocollo diverso per comunicare tra diverse organizzazioni. Questo protocollo ignora l'interno e guarda solo alle zone di demarcazione. Sono le macchine di frontiera ad annunciare che al loro interno è presente una certa rete. È una tecnica di routing statico, ma nel senso opposto, in modo da indirizzare il traffico verso la propria zona. Se non viene trovato il destinatario all'interno, non è un problema poiché il traffico non avrebbe trovato altre rotte possibili.

La connessione tra i vari router di frontiera è gestita attraverso connessioni TCP, chiamate *peering*. In questo modo si semplifica l'instradamento considerando intere organizzazioni come singole destinazioni. Poiché è necessario comunicare tra i soli router di frontiera, se questi si trovano in una stessa organizzazione servono connessioni TCP interne. Si determina in questo grafo virtuale la soluzione al problema di routing. Dopo che la rotta per obiettivi remoti è stata decisa dai router di frontiera, questi iniettano le rotte esterne all'interno della loro organizzazione.

Il protocollo *Border Gateway Protocol* ha l'obiettivo di rendere disponili ed aggiornate le tabelle di instradamento con informazioni di routing inter-dominio. Questo è un protocollo tra i più complessi ed importanti. Può prendere in considerazione anche vincoli commerciali, preferenze locali, priorità, problemi legali, ecc. BGP è stato introdotto nell'89 ed adottato per la prima volta nel 94. Con IPv4 è stato aggiornato nel '94 e migliorato nel '98. La versione del 2006 di BGP include la compatibilità con il protocollo IPv6. È un protocollo complesso, non sono definite le scorciatoie e la prassi, serve esperienza per poterlo configurare senza errori.

BGP è usato da chi è connesso ad uno o più ISP, dai provider del transito, da ISP che scambiano traffico su IXP o NAP, o clienti con reti di grandi dimensioni. Generalmente organizzazioni comprano un pezzo dello spazio di indirizzamento sulla rete e lo annunciano tramite BGP.

BGP si fonda sul concetto si sistemi autonomi, *Autonomous System* (AS), questi rappresentano una rete sotto una singola amministrazione. Controllati da un unico gestore, amministratore. Sono società dichiarate per il protocollo e vengono identificati da un numero da 32 bit, aumentato dagli iniziali 2 byte. Questo identificativo viene chiamato *Autonomous System Number* (ASN). Alcuni di questi sono dedicati all'uso privato, poiché se si ha una rete molto grande su più continenti è utile fare routing in base a policy, come fossero policy commerciali, e quindi bisogna utilizzare BGP, organizzando la rete come se fosse composta da AS diversi. Gli identificatori da 0 a 2^{16} sono dedicati ad uso pubblico o privato.

Il numero 23456, indica una comunicazione con una macchina che ancora non conosce l'esterno, viene usato per convertire un numero da 16 bit a 32 bit, contenendo nel pacchetto regole per convertirlo.

L'organizzazione centrale IANA che gestisce l'assegnazione di indirizzi IP li concede ad organizzazioni regionali. Queste risorse vengono assegnate quando necessario. Questi registri regionali assegnano anche l'ASN globale ad un AS. Per comunicare con un certo provider con numero privato bisogna chiederlo direttamente all'ISP.

Non c'è nessun controllo di alcun tipo, eccetto controlli di indirizzi IP uguali.

BGP permette lo scambio di informazioni solo se è attiva una sessione di peering. Questa è una connessione TCP sulla porta 179, tra due router di frontiera. Poiché questi possono appartenere ad uno stesso AS, si distinguono in connessioni e-BGP e i-BGP, *external* e *internal* BGP, questi devono essere direttamente connessi in connessioni e-BGP, mentre può non essere diretta in i-BGP. La connessione TCP rimane viva nonostante i cambiamenti di routing e la perdita di connettività.

In e-BGP i pacchetti vengono inviati con un TTL pari ad uno. Invece in i-BGP è possibile propagare le informazioni di instradamento da un bordo al bordo opposto di un AS, avendo i router di frontiera di uno stesso AS in full mesh. Le connessioni di peering tra due AS potrebbero rimanere attive per anni.

BGP apprende l'esistenza di destinazioni remote attraverso e-BGP, con annunci che possono essere affermativi. Per ogni meta remota viene selezionato un percorso migliore tra tutte le alternativa, applicando le varie politiche di interesse. Questa viene indicata nella tabella BGP e solo questa viene propagata all'interno dell'AS e può essere propagata ad altri vicini BGP esterni.

La notifica delle LAN interne non è automatica, viene richiesta manualmente, e rappresenta sia una forza che un limite per BGP. Permette di annunciare solo LAN specifiche, tuttavia se si commettono degli errori è possibile impedire ad altri utenti di connettersi a certe LAN.

I pacchetti di annuncio contengono nel path una sorta di distance vector, dove sono presenti tutti gli AS attraversati, in questo modo gli echi vengono affrontati a priori. Se un AS riceve un pacchetto contenente il suo ASN, lo scarta per non creare rotte cicliche.

Gli annunci vanno in un verso, ed il traffico uscente va nel verso opposto, poiché gli annunci sono rotte statiche che vengono inoltrate all'interno.

5.2 Scalabilità di BGP

Su BGP il problema della rumorosità è risolto dal *route reflectors*, inoltre ci sono problemi di scalabilità per realizzare le policy, tramite route map. Si vorrebbe esprimere preferenze su ogni prefisso, ma questo non è possibile con le route map. Si può risolvere con le *communities*.

5.2.1 Route Refresh

Il route refresh è un meccanismo che risolve il problema di modifica delle policy. Quando viene modificata una policy, il router deve ricompilare tutte le rotte di instradamento. Il router potrebbe richiedere dai suoi vicini tutte le rotte, in un *hard BGP peer reset*, ma questa possibilità comporta un lavoro notevole e non garantisce la continuità del servizio. Altrimenti, quando si ricevono gli annunci dai vicini, si mantengono in memoria, ed avendo una copia di tutto in memoria, si può ricalcolare tutto, *soft reconfiguration*.

Entrambi hanno dei vantaggi e svantaggi. L'hard reset consiste nel buttare giù tutti i peering e riattivarli, e dopo aver ricalcolato le rotte, vengono ri-annunciate con effetti che riverberano sulla rete. Invece, per la soft reconfiguration si consumano risorse di memoria e non di calcolo.

La route refresh, descritta in RFC 2918, mantiene attivi i peering BGP, e chiede ai suoi vicini di ri-annunciare i prefissi, sfruttando la memoria dei vicini invece di usare memoria aggiuntiva propria. Impatta solo i prefissi affetti dai cambi di policy, e facilita il cambiamento di policy non distruttive. Inoltre, non sono richieste configurazioni aggiuntive, tutto ciò che viene negoziato dai router è automatico.

5.2.2 Next Hop e Recursive Lookup

In generale il next hop coincide con l'interfaccia di peering del router BGP mittente dell'annuncio. Quando la rotta viene annunciata con iBGP, il BGP canonico prevede che la rotta mantenga la stessa interfaccia remota. Quando viene annunciata all'esterno il next hop diventa di nuovo l'interfaccia di uscita. È un attributo che canonicamente ha sempre l'interfaccia di uscita dell'ultimo AS da cui è uscita.

Quando un router deve inserire una rotta BGP nel data plane verso un IP remoto, non appartenente allo stesso AS, il next hop è l'indirizzo su cui viene eseguito ARP. Ma possiede un next hop remoto, quindi deve determinare a quale interfaccia vicina inviare il pacchetto. Il router esegue un lookup sulla rete dov'è presente l'indirizzo del next hop. Per inserire la rotta sulla tabella di instradamento quindi inserisce il valore di next hop della rotta appena trovata.

L'iBGP che compila la tabella di instradamento, prima che BGP inietti le rotte di instradamento, deve iniettare tutte le rotte esterne, altrimenti non sarebbe possibile effettuare il recursive lookup. In generale queste non servono, poiché protocolli IGP non si interessano di queste, ma sono necessarie per iniettare rotte BGP. Per risolvere questo problema, si può inserire una clausola che impone di cambiare il next hop ad ogni passo, anche se ci si trova in uno stesso AS. In questo modo non bisogna portare in IGP tutte le rotte esterne:

```
neighbor x.x.x.x next-hop-self
```

Oppure si possono configurare peering iBGP su indirizzi di loopback. Questi possono essere aggiunti arbitrariamente, anche con indirizzi globalmente unici. Il pacchetto indirizzato all'interfaccia di loopback, risale al livello IP della macchina che lo riconosce come proprio e lo elabora.

Si può usare uno schema di enumerazione semplice nelle loopback per tutti i router. Questi diffondono sul protocollo IGP dei prefissi unici /32. Fino a che il router ha un'interfaccia attiva, è raggiungibile. Invece di usare un'interfaccia vera è possibile mantenere la connessione indipendentemente dallo stato delle sue interfacce. Questo è estremamente comodo anche in configurazione, le loopback vengono usate come identificatori di router. Tutti gli altri IP sono accidentali e non è necessario memorizzarli.

Quando si realizzano dei peering, non si realizzano tra due interfacce vere, ma tra interfacce di loopback, che non cadano a meno che non viene disattivato l'intero router. Le sessioni iBGP non dipendono neanche dallo stato della rete interna. Fino a che l'AS è connesso, possono comunicare tutte le macchine. Tuttavia, bisogna forzare l'indirizzo di peering, con quello di loopback, e non l'indirizzo dell'interfaccia da cui esce il pacchetto:

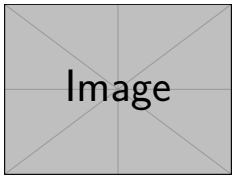
```
neighbor x.x.x.x update-source y.y.y.y
neighbor x.x.x.x update-source lo:z
```

Questo comando accetta un indirizzo IP o il nome di un'interfaccia.

5.2.3 Route Reflectors

Questo non toglie che il numero di peering BGP non scali, rimane quadratico, per mantenere la full mesh dentro un AS, in modo che non rimbalzino annunci BGP costantemente. I *Route Reflectors* sono una soluzione semplice, descritta in RFC 4456. Un route reflector è un router come gli altri, questo computa delle rotte, esattamente come un router BGP, e seleziona le sue best e le inoltra ai suoi vicini con una particolare policy.

Ogni router iBGP dell'AS è connesso a questo router, chiamato route reflector, questo quando riceve degli annunci da parte degli altri router, decide qual è il migliore e lo invia a tutti. Il comportamento da parte dei router di frontiera è buono, tutti ottengono la scelta migliore. Il route reflector si comporta come uno specchio che riflette rotte specifiche verso tutti gli altri router. Quindi è sufficiente che un router iBGP sia connesso al route reflector per conoscere le rotte inoltrate da tutti gli altri router di frontiera.



Esiste un route reflector che si comporta come server, una macchina BGP standard, con una serie di client che sono i router di frontiera. Alcuni peering iBGP diretti possono sopravvivere e possono diminuire la latenza in caso il route reflector sia troppo lontano.

Per sostenere la scalabilità si può creare una full mesh di route reflector, comunque più piccola dell'eventuale full mesh tra tutti i router, dove ogni annuncio compie un singolo rimbalzo. Tra loro sanno che stanno comunicando tra route reflector. Quando un route reflector riceve un annuncio da un client o un non client, seleziona la rotta migliore, se la riceve da un client, la rimanda a tutti i client ed ai suoi non client. Se invece viene da un non client, si suppone che tutti gli altri route reflector la sappiano già, e la manda verso i suoi soli client.

Si può scalare ulteriormente aggiungendo un altro livello di route reflector. Ciò che è importante è avere una full mesh solo al livello più alto. È possibile avere connessioni aggiuntive, per tollerare una ridondanza, un client potrebbe avere più server, ma non è necessario creare un albero.

Ad ogni livello gerarchico un route reflector serve un cluster della backbone, questo può avere due o più router.

Si crea un problema, poiché connessioni ridondanti potrebbero introdurre dei cicli. Per evitare il count to infinity in BGP si è introdotta la full mesh, e per evitare i problemi della full mesh si sono introdotti nuovamente dei cicli.

La soluzione è una specie di as-path, l'idea è quella di avere un attributo BGP che si memorizza l'ID di ogni router attraversato. Si vuole un attributo che porti tutti gli ID dei router attraversati, in modo che chi riceve un annuncio contenente il suo ID, lo scarta essendo un ciclo. Questa attività la devono realizzare i route reflector, non le macchine normali, poiché vedono questo come un peering BGP normale, devono partecipare a questo meccanismo senza saperlo. Ognuno si deve comportare in una certa maniera, ma ci sono degli attori inconsapevoli. Si realizza allo stesso modo di OSPF, impersonando quei router.

Si introducono dei campi aggiuntivi per questi annunci, il campo originator ID contiene l'ID del route reflector che lo ha generato, la cluster list contiene la lista dei cluster attraversati nella gerarchia dei route reflector. L'identificativo di un cluster viene aggiunto quando viene inviato un annuncio da ruote reflector. Se un router riceve una rotta iBGP con il suo ID contenuto nel campo originator ID lo scarta, se invece nell'attributo cluster list è contenuto l'ID del cluster lo scarta.

Queste connessioni TPC potrebbero non coincidere con le connessioni fisiche, ma si consiglia di seguire la topologia fisica della rete, in modo da non influenzare l'instradamento dei pacchetti. Tipicamente la PoP, *Point of Presence*, di un ISP ha due router principali, route reflector, con due cluster definiti sull'intera rete.

La metrica IGP viene considerata nel processo di selezione della rotta migliore. Il passaggio a cluster e route reflector è semplice, con una migrazione facile che non impatta i servizi, riducendo le ridondanze, dividendo la rete in cluster ed inserendo un route reflector per cluster.

5.2.4 Communities

Le *Communities* sono dei meccanismi di scalabilità per permettere ad un insieme di macchine di comportarsi in maniera omogenea. Questo viene realizzato raggruppando insiemi di prefissi in classi o community, sono delle etichette inserite in annunci. Ogni etichetta ha significati diversi. Si possono applicare ad annunci in iBGP o eBGP.

Venne introdotto l'attributo community in RFC 1998, dopo essere descritto in RFC 1997. Se queste etichette non vengono rimosse, rimangono nell'annunciano e vengono propagate in rete. È un intero a 32 bit, rappresentato da due numeri da 16 bit. I primi due byte sono il numero dell'AS,

si annuncia l'AS che ha deciso il significato dell'etichetta, non necessariamente l'AS che ha generato l'annuncio originario. Ogni rotta potrebbe essere membro di più comunità, si può trattare ogni comunità con un'unica policy, per aggregare più prefissi. Si possono realizzare route map su questo attributo etichetta.

Ci sono delle community abbastanza standard, con il prefisso `65553:x`:

- *No-export*, rotte da non annunciare ad alcun peer eBGP `:65281`
- *No-advertise*, rotte da non annunciare ad alcun peer BGP, `:65282`
- *No-export-subconfed*, rotte da non esportare fuori dall'AS locale, `:65283`
- *No-peer*, rotte da non annunciare a peer bilaterali, `:65284`

5.3 Gerarchia Internet

Nel mondo reale sono presenti più livelli di astrazione per BGP. Si possono considerare relazioni di tipo economico:

- Esista una gerarchia di AS, in modo che l'operatore che fornisce la connettività possa fornire il proprio servizio, attraverso provider di più alto livello.

  Il customer è un AS e va a pagare il provider, l'AS che fornisce connettività di più alto livello.

- Il secondo tipo di relazione è chiamata peer-to-peer, in questo tipo di relazione due provider circa dello stesso livello si mettono in accordo per scambiarsi traffico gratuitamente.

Normalmente i costi sono proporzionali alla quantità di traffico passato sul link. Se si riesce a mandare il traffico su un peer-to-peer l'unico costo è quello di manutenzione del link, diviso tra i due peer, e tutto ciò che passa sopra è gratis. Mentre è il customer a pagare il traffico sul link offerto dal provider, indipendentemente dalla direzione del traffico.

Da questa base logica si può arrivare al concetto della gerarchia di Internet. Quando si raggiunge la parte più alta della gerarchia, sono tutti in peer-to-peer in full mesh.

- Ad un provider conviene inviare traffico ai suoi customer, per cui un provider propaga traffico, diretto ai customer, se proveniente dal suo AS, da un suo customer o un peer, dove ci guadagna, oppure proveniente da un suo provider, supponendo i costi uguali a somma zero, poiché paga il link verso il suo provider. In discesa passa tutto ed è conveniente per il provider.
- In salita il customer invia il proprio traffico, ed il traffico proveniente da un suo customer, poiché è il motivo per cui si ha un contratto tra due AS, a somma zero, supponendo i link equamente costosi.
- Ricevendo traffico da un peer, gratis, se lo si inoltra verso l'alto, si ha una perdita quindi non viene inoltrato.
- Analogamente non si manda traffico proveniente ad un provider verso l'alto.

- Il traffico peer-to-peer non permette due hop, poiché si considera anche il costo dell'infrastruttura interna, se questa viene saturata, bisogna aumentarne le prestazioni. Quindi se si facesse passare il traffico peer-to-peer si avrebbe una spesa, quindi non lo si inoltra.

Il modello customer-provider di BGP viene chiamato *valley free*, poiché nel momento in cui si crea una valle, si realizza traffico vietato poiché comporta una perdita. Questa politica viene implementata da chi ci perde.

Fino a che si sale soltanto e si scende soltanto gli annunci BGP sono accettati. In qualche caso queste catene potrebbero essere mancanti. Non sono permessi peer-to-peer solo se dopo si scende, quindi non esistono percorsi con un peer-to-peer non alla massima altezza, poiché uno dei due peer ci perderebbe economicamente.

Le relazioni customer provider sono esclusivamente dei filtri. Il modello economico spesso si associa al modello *prefer-customer*. Se si deve scegliere dove inoltrare gli annunci, si preferisce verso il basso, o in orizzontale, dove non si perde. Mentre come ultima scelta si inoltra verso l'alto.

La configurazione vera di un router BGP è costituita da una serie di filtri, ed un insieme di route map per settare le preferenze tra i vari AS. L'Internet non è shortest path, nonostante BGP lo sia in assenza di policy, non sempre lo è nel mondo reale, poiché implementa policy legate anche a rapporti commerciali.

Una stima realistica degli AS è di circa 75 mila, ma è difficile da stabilire, poiché viene effettuata su quelli annunciati, non quelli assegnati, che non vengono riassegnati.

Gli AS in questa gerarchia si dividono in tre tipi, i provider di livello uno sono tutti in una full mesh tra tutti loro, sono tra i 7 ed i 12, non vengono pubblicati gli accordi commerciali, quindi possono solo essere dedotti dagli annunci BGP sulla rete.

Il customer cone di un AS è la parte dell'Internet che può raggiungere attraversando solamente connessioni di tipo customer-provider, l'insieme dei suoi customer diretti ed indiretti. Poiché possono essere disgiunti, dovendo offrire la piena raggiungibilità di Internet, bisogna realizzare una full mesh tra i coni dei vari provider.

Più in alto sono difficili da capire i rapporti, la rete è estremamente densa, con un numero di link molto alto rispetto al numero di nodi.

La rete di BGP è molto densa ed anche molto piatta, poiché una volta raggiunto tutto l'Internet, non è necessario aumentare i peer.

Il sito asrank è gestito da un'agenzia di ricerca CAIDA, per classificare AS. Questo esegue un algoritmo una volta al mese per determinare com'è strutturato l'Internet, realizzando un ranking degli AS per dimensione del cone size. RIPE offre molti strumenti, tra questi RIPEstat fornisce statistiche aggregando informazioni provenienti da BGP. Lo strumento chiamato *looking glass*, portali web pubblicati da certi AS, per vedere lo stato BGP del router, realizzando delle query. Si possono fare dei ping, ed eseguire comandi limitati. Questi sono convenienti per AS grandi, poiché BGP è una comunità.

5.4 Anomalie di Internet

BPG è stato progettato intorno al 1994, all'epoca il concetto di cybersecurity non era vivo quanto oggi. È noto che BGP è vulnerabile ad attacchi, e questi sono in molti casi totalmente invisibili

agli utenti. È estremamente facile modificare l'AS path di un annuncio, per indirizzare il traffico verso il proprio router ed analizzare il traffico, a seguito di operazioni malevole.

Non ci sono firme digitali o validazione degli annunci, mancano i principi basi delle politiche di sicurezza. Alcuni attacchi BGP sono estremamente banali, per cui CAIDA, per il dipartimento dell'Homeland Security, monitora BGP per settori critici di interesse.

5.4.1 BGP Leaks

Alcune anomalie sono BGP leaks, una leak è un prefisso che non dovrebbe uscire da un AS, ma viene annunciato verso l'esterno. Generalmente è un errore, e può trasformare un AS non di transito in un AS di transito.

Questo avviene in vari modi, alcuni prefissi viaggiano disaggregati per annunciare i prefissi in modo più specifico. Il numero di BGP leaks è stimato intorno a qualche centinaio al mese.

Nell'agosto del 2017 Google annunciò a Verizon rotte apprese tramite link peer-to-peer, diventando transito per 135 mila rotte causando un outage in Giappone.

Nel Gennaio 2017 l'azienda Telecom iraniana ha annunciato dei prefissi commerciali allo scopo di censurare dei siti web, generando dei leak.

5.4.2 BGP Hijkacking

Il BGP hijacking consiste nel rubare un prefisso, annunciandolo, che non si possiede realmente. Si possono realizzare hijack parziali rubando un sotto-prefisso più specifico, diffondendosi in modo più veloce. Oppure si può realizzare un hijack completo, aggiungendo determinati attributi per rappresentare la rotta migliore rispetto a quella fornita dalla vittima. Se l'attaccante inserisce nell'AS path l'AS effettivamente proprietario del prefisso come originator si parla di *upstream hijack*, poiché risulta che l'attaccante è un upstream della vittima.

L'effetto immediato può essere di deviare il traffico, rimandarlo all'origine, analizzarlo, ecc. Ci sono centinaia di attacchi di questo tipo al mese.

Nel 26 Aprile 2017, rotte appartenenti a più di una dozzina di altri servizi finanziari sono state annunciate da un AS russo.

Il 12 Dicembre 2017 molti prefissi americani sono stati annunciati da un AS russo generalmente silente.

5.4.3 BGP Interception

Un BGP interception è un evento in cui un AS annuncia di possedere un prefisso per attrarre uno specifico flusso di traffico, potendolo in caso dirigerlo verso il destinatario originale, comportandosi come un MitM, *Man in the Middle*. Si può mantenere la connessione verso il destinatario originale, inserendo nell'AS path gli AS lungo il cammino tra la vittima e l'attaccante.

Ci sono stati vari interception, i dati non sono molto chiari, poiché l'utente non se ne accorge, solo l'AS difficilmente, per cui l'incidenza è ignota.

5.4.4 Altre Anomalie

BGP outage è quando si perde visibilità di un certo prefisso, non necessariamente è un attacco, spesso è solo sparito. BGP è dimostrato che non è stabile, esistono configurazioni per cui la best oscilla in continuazione, si indica con BGP instability o *flapping* quando un prefisso viene annunciato e ritirato velocemente. Un BGP new upstream corrisponde ad un AS che annuncia per la prima volta un prefisso mettendosi nell'AS path prima dell'AS noto proprietario. Può essere sintomo di un hijack. Si ha un BGP newly routed prefix/sub-prefix, quando un prefisso non generalmente annunciato in rete viene annunciato per la prima volta, può essere sintomo di una leak. Si ha un BGP new AS origination, quando un AS origina per la prima volta un nuovo prefisso.

5.5 Multi-Protocol Label Switching e Virtual Private Networks: MPLS e VPN

Il *Multi-Protocol Label Switching* è una tecnologia usata dai data centers, che risponde ad esigenze di diversi attori.

Il cliente vorrebbe avere un filo diretto, mentre il provider vorrebbe utilizzare la rete che già possiede. I provider hanno una grande disponibilità di banda, ed è molto difficile saturarla, vorrebbero vendere connettività punto-punto ai loro utenti, attraverso fili virtuali.

Si può pensare ad una situazione dove le sedi periferiche di un'organizzazione si connettono alla sede centrale tramite il provider, per voler collegare punto-punto queste diversi sedi, con possibilmente gli stessi prefissi tre le varie sedi. Per cui il provider deve realizzare un filo virtuale mantenendo invariati gli indirizzi dell'organizzazione del customer. Vuole che questo traffico sia isolato rispetto al traffico di altri utenti, poiché potrebbe contenere informazioni sensibili, o ha esigenze di riservatezza. Almeno lo vuole isolare logicamente.

Il provider vuole realizzare una configurazione economica e scalabile, e non vorrebbe avere una riduzione delle prestazioni.

Per soddisfare i vincoli del provider e dei customer, si usano i VPN, *Virtual Private Network*, si comportano come una rete privata, ma è implementata solo virtualmente, e MPLS, *Multi-Protocol Label Switching*, un protocollo altamente scalabile per il trasporto di dati, utilizzato per implementare reti virtuali.

5.5.1 Label Switching

Il *label switching* è un meccanismo di instradamento dei pacchetti, si inserisce sopra ad ethernet, e sotto IP. Però mentre nella rete attraversano pacchetti MPLS, passano anche pacchetti IP.

Un nodo per inviare un pacchetto, si calcola il cammino, al pacchetto non lo si guarda più in base all'indirizzo, si inserisce in un altro pacchetto contenente un'etichetta. Le macchine successive instraderanno in base a questa etichetta. Ci sono tante etichette quanti sono i flussi che stanno attraversando la rete al momento, un numero molto minore rispetto allo spazio di indirizzamento. Quando si definisce questo flusso si possono fare considerazioni di banda, per ottimizzare il flusso dei dati.

Per ogni tratto si può usare un'etichetta diversa, in modo da non dover definire etichette univoche sull'intera rete, il calcolo deve essere effettuato dalla macchina che assegna l'etichetta. Prima

bisogna avere delle tabelle di etichette. All'inizio un qualche protocollo decide il cammino e determina per ogni tipo di flusso un tipo di etichetta. La macchina che determina se un pacchetto appartiene ad un dato flusso determina la sua etichetta, questo ad ogni hop del pacchetto. Lo inserisce in un circuito virtuale indirizzato su una strada già determinata, in base ai protocolli di instradamento del livello sottostante.

Il pacchetto MPLS incapsula i pacchetti con un header contenente uno o più etichette, in uno stack. Ogni stack ha quattro campi, il valore dell'etichetta, la classe del traffico, una flag che indica l'ultima etichetta ed il TTL. Molte volte il TTL non viene aggiornato, quindi non sono visibili i nodi intermedi con un traceroute.

Si potrebbe sovrapporre un certo numero di etichette in maniera arbitraria. Inoltre, si potrebbe creare un tunnel dentro ad un tunnel. Per instradare un pacchetto si guarda solo all'etichetta affiorante, confrontandola nella tabella di etichetta a quale flusso coincide, ed in caso si può rimuovere o sostituire con un'altra etichetta per indicare il flusso.

5.5.2 Implementazione

In generale i router dei provider o di backbone si indicano con P, i router di frontiera sui customer si indicano con PE, i router dei customer si indicano con CE. I PE rappresentano i PoP dell'ISP su una determinata zona.

Per implementarlo si potrebbero usare tunnel ed interfacce di loopback, assegnando un indirizzo IP di loopback ai PE, garantendo che siano connessi. Questi indirizzi di loopback vengono propagati attraverso un protocollo IGP. Si possono usare tunnel IP-in-IP per realizzare la VPN. Sulla macchina router si definisce un'interfaccia dove si vedono le destinazioni di questo traffico. Essendo un interfaccia virtuale, il traffico viene incartato in un altro pacchetto che possiede l'indirizzo destinazione effettivo dell'altro estremo del tunnel. Questo esce alla fine del tunnel, viene scartato e lo ripassa attraverso la tabella di instradamento e lo passa all'interfaccia connessa al customer.

In questo modo non si hanno problemi di sovrapposizione delle reti. Tuttavia, è difficile da configurare, e bisognerebbe configurare un numero quadratico di tunnel.

La seconda opzione consiste nell'usare BGP per annunciare un'opportunità di instradamento. Questo utilizza una variante di BGP, MP-BGP. Ogni PE instaura un peering iBGP con tutti gli altri PE nella rete del provider, condividendo informazioni di instradamento sulle reti del customer.

La terza opzione consiste nell'usare MPLS come tunnel, il pacchetto IP di un CE attraversa tutta la rete dopo essere incartato in un pacchetto MPLS dal PE, assegnando due etichette, indicando qual è il customer e qual è il PE di destinazione. Quando il PE destinazione rimuove la prima etichetta, e rimane solo l'etichetta che indica il customer mittente, la rimuove ed inoltra il pacchetto contenuto al destinatario corretto.

In realtà l'etichetta finale la toglie il penultimo così l'ultimo router, il PE, estrae direttamente il pacchetto effettivo e lo inoltra al destinatario.

L'etichetta più profonda che non viene usata per instradare viene usata per identificare quel tipo di traffico, indica la VPN, in maniera simile all'identificativo di una VLAN. Quella che emerge viene utilizzata per attraversare la rete.

Il protocollo che assegna queste tabelle di etichette è il LDP, *Label Distribution Protocol*. Questo genera i flussi, costruendo i *Label Switched Paths* per raggiungere le interfacce di loopback delle PE,

prendendo le informazioni dalla tabella di instradamento. Si suppone che un IGP abbia propagato le informazioni sulle interfacce di loopback dei vari PE.

Per realizzare un tunnel in questo modo bisogna utilizzare tutte e tre le opzioni. Si propagano le interfacce di loopback dei PE tramite un IGP. Si distribuiscono i prefissi VPN dei customer tra i PE con iBGP. Si costruiscono le LDP, verificando la presenza di LSP che connettono i PE.

I router PE devono distinguere le VPN, e mantenerle isolate, poiché potrebbero avere spazi di indirizzamento sovrapposti. Si usa il costrutto *Virtual Routing and Forwarding* (VRF), questo virtualizza il router, creando delle istanze di routing. Questo permette ad un router di avere diverse tabelle di instradamento, ogni tabella è un'istanza di VRF, sia il control plane che il data plane, che corrisponde ad una VPN.

Ogni PE mantiene più istanze di VRF ed ognuna contiene le rotte ricevute dai CE direttamente connessi, associati ad una specifica istanza di VRF, inoltre ogni VRF contiene le rotte propagate da iBGP dagli altri PE.

Per distinguere gli indirizzi IP si usa un *Route Distinguisher*, (RD), univoco tra le VPN. Questo converte indirizzi IP non univoci in indirizzi VPN unici. Questo permette di evitare conflitti se due customer hanno spazi di indirizzamento sovrapposti.

In MPLS i router si distinguono in LER, *Label Edge Router*, il primo router che incapsula il pacchetto dentro un MPLS LSP, corrisponde al PE connesso al CE. Mentre LSR, *Label Switching Router*, si occupa solamente di instradare i pacchetti MPLS seguendo un LSP, sono i vari P interni alla rete.

Un LSR può ricevere la stessa etichetta da più interfacce, in questi casi sono presenti dei meccanismi per evitare collisioni. Questo si basa sul label space usato, e dipende dall'implementazione ed a volte dall'interfaccia, non dalla configurazione. Può dipendere solo dall'etichetta o anche dall'interfaccia da cui la riceve.

A volte avere solamente un RD non è sufficiente, si vorrebbe poter avere VPN che non sono una mesh. In questi casi il RD indica solo una rete virtuale e non riesce a stabilire una topologia articolata.

Quindi si è introdotto un altro attributo *Route Target*, (RT), che specifica quali rotte devono essere importate o esportate all'interno di un VRF. RT è una comunità estesa, che supporta topologie complesse. Si assegna un RT ad ogni VPN, e si usa una singola VPN.

Oltre a specificare il RD, si indica anche il RT, che è una comunità di VPN.

Questo sistema di amministrazione interno di un AS permette di creare dei tunnel, con la massima flessibilità senza che il customer sia a conoscenza di quello che sta avvenendo dentro l'AS, vedendo la connessione come un'effettiva connessione via filo. La configurazione del provider inoltre è relativamente semplice.

5.6 Introduzione a RPKI

RPKI è un meccanismo abbastanza diffuso per la sicurezza di BGP, poiché è un protocollo non autenticato, dove sono possibili attacchi e situazioni per errori di configurazioni che possono rompere BGP.

5.6.1 Insicurezza BGP

Dato un annuncio BGP, un router BGP potrebbe chiedersi se quell'AS è un suo vicino, inoltre si potrebbe chiedere se è veramente il proprietario di quel prefisso, poiché su BGP non sono presenti controlli ed è possibile annunciare rotte arbitrarie. Non ci sono controlli di default, quindi tutti i router si fidano arbitrariamente.

Il problema è che BGP assume che tutti dicano la verità, ma in caso di utenti malevoli, è facile impersonare altri AS, ed annunciare prefissi arbitrariamente. Non necessariamente questi sono utenti malevoli, può essere semplicemente un errore di configurazione. Alcune situazioni possono generare un local hijack, dove solo la parte più locale viene instradata verso un finto prefisso. Oppure si potrebbe annunciare un prefisso più specifico, questi infatti verranno preferiti, causando un global hijack, direzionando l'intero traffico destinato al prefisso più specifico, diffondendosi in maniera autonoma sulla rete, poiché BGP effettua solo check esatti.

In questo caso per mitigare l'attacco si potrebbero annunciare due rotte più specifiche per coprire il prefisso annunciato dall'utente malevolo. Questo consiste nella propagazione di moltissime rotte aggiuntive, facendo crescere in modo incontrollato le tabelle di instradamento. Inoltre, si vorrebbe prevenire questo attacco, quindi risolverlo prima che l'attaccante possa annunciare questa rotta più specifica. Dato che c'è sempre un lasso di tempo dove l'attaccante può ottenere qualcosa.

Un altro tipo di global hijack consiste nell'annunciare una rotta inventandosi un AS path, annunciando lo stesso prefisso, in questo caso la vittima e l'attaccante si contendono la rotta.

Questo succede poiché gli annunci BGP sono annunciati senza alcuna validazione. I router verificano autonomamente che come AS affiorante sia presente il primo AS nell'AS path, questa è una politica di sicurezza già usata da tempo, in modo da garantire che gli AS vicini inoltrino solo rotte dove si sono inseriti. Questo si può disabilitare, in alcuni casi potrebbe essere utile.

In un IXP questa politica viene disabilitata, poiché l'IXP ha un suo AS, e dovendo connettere due AS diversi, gli annunci passano attraverso il route server interno dell'IXP. Concettualmente è un route reflector, ma è in un altro AS. Non si inserisce nell'AS path poiché il traffico non lo attraversa, non applica il processo decisionale, ma anche non va a modificare gli annunci.

Un attaccante può applicare censura, rubare cripto-valute, intercettare il traffico, realizzare black-holing, rubare delle credenziali, inviare spam, anche se sono presenti delle tecniche di cifratura sulle mail, ecc.

Non necessariamente si tratta di un attacco, potrebbe essere una configurazione errata a causa di typo, oppure a causa di differenze nei linguaggi usati dai router BGP. Per esempio il prepending nei router della Cisco si configura inserendo l'ASN ed il moltiplicatore, mentre nei router Juniper si inserisce l'ASN tutte le volte necessarie. Per questo è possibile individuare ASN come uno, due o tre, che appartengono a grandi Università americane non annunciati in internet, a causa di questi errori negli AS path. Un altro problema può essere causato dai configurazioni errate di filtri. Degli annunci interni se vengono passati esternamente con route leak, oltre a problemi economici possono saturare i link.

Modificare il protocollo BGP per aumentarne la sicurezza non è possibile, poiché molti di questi router sono vecchi con un tempo di riavvio critico. Per cui si punta a prevenire tutti gli attacchi non intenzionali.

5.6.2 Protocolli di Sicurezza

Gli IRR sono dei registri di Internet pubblicamente consultabili dove si indica la proprietà di un prefisso. Ma dato che è pubblico, bisogna comunque verificare le informazioni. Essendo scritte in maniera standard e pubbliche, queste informazioni possono essere utilizzate per creare automaticamente dei filtri BGP per AS, basati su informazioni verificabili.

Questi IRR non sono usati globalmente, non è presente un'autorità centrale, non necessariamente sono mantenuti ed accurati. Per cui utilizzando dei filtri automatici e stringenti, si blocca la maggior parte di Internet.

Una PKI, *Public Key Infrastructure* è un sistema pubblico di chiavi, un albero di certificazione. Si può generare un certificato firmandolo da solo, ma questo non è validato, si valida un certificato firmandolo da un certificato conosciuto, posseduto da organizzazioni riconosciute per la certificazione. Così si crea una catena di fiducia.

In questo modo si possono validare degli annunci BGP, associando gli ASN agli IP, creando un sistema di certificazione di certificati pubblici che accoppia la nozione che un certo prefisso è proprietà di un certo AS. Questo sta ignorando il problema di validazione di un path, si occupa solo dell'appartenenza di un certo prefisso.

RPKI aiuta a prevenire tutti quelli incidenti solitamente dovuti a configurazioni errate. RPKI deve fidarsi dei RIR come *Trust Anchor*, le cinque organizzazioni che si dividono lo spazio fisico del globo per assegnare indirizzi IP ai propri AS. I RIR hanno il controllo sull'accuratezza e saranno loro a firmare i certificati per garantire i proprietari di un prefisso.

Il proprietario di un prefisso non è detto che sia chi annuncia un certo prefisso, questi possono essere affittati.

RPKI funziona con due oggetti totalmente separati, uno si occupa di certificare ASN e prefissi IP, ed un altro validatore, esterno al protocollo, si occupa di inserire le informazioni relative nelle configurazioni BGP.

Ogni RIR ha il suo certificato self signed, che firmerà i certificati, questo lavoro potrebbe essere delegato ai LIR, o direttamente agli utenti finali. Il termine ROA indica un certificato verificato da una catena di fiducia fino ad un root certificate, posseduto da un RIR.

Ognuno può firmare certificati per qualcosa di più specifico, i cinque RIR hanno root certificate per tutte le risorse, `0`. L'idea del ROA è che sono firmati dal LIR ed il LIR si prende un pezzo dell'indirizzamento, fornito dal RIR, per garantire l'autenticità.

Un singolo ROA può contenere una lista di prefissi al suo interno, potenzialmente potrebbero essere contenuti in un unico ROA, in base a criteri determinati dal LIR. Questi ROA possono essere sovrapposti, contengono informazioni base, la data di inizio e fine validità, una delle condizioni per cui possono essere sovrapponibili.

Tra le informazioni base sono il prefisso, gli origin AS verificati, e la massima lunghezza del prefisso, per evitare di poter annunciare prefissi più specifici. In questo modo si va a certificare un prefisso.

L'idea è che esistono protocolli adatti, in modo che il validatore possa leggere le repository RPKI, gli AS dovranno avere un validatore oppure fidarsi di un validatore esterno. Si validano come si valida qualunque certificato, confrontando la firma digitale del ROA, salendo la catena di

fiducia controllando se è valida rispetto alla chiave pubblica del LIR che ha certificato il prefisso, analogamente si valida il certificato del LIR, da parte del RIR.

Una volta che si è verificata la validità, si installano sui router moderni, un protocollo moderno può interagire direttamente con queste repository, altrimenti software specifici li trasformano in route map da inserire direttamente nella configurazione BGP, per router più vecchi.

Per validare un annuncio, si controlla che il prefisso annunciato provenga da uno degli origin AS certificati, e si controlla che non sia più specifico della lunghezza massima presente nel certificato.

Questo sistema si scontra con la legacy degli AS, al giorno d'oggi molti prefissi hanno i ROA, ma pochi AS li verificano. Se arriva un ROA non verificato bisogna scegliere se accettarlo o scartarlo, dato che l'applicazione dei ROA non è uniformata su tutti gli AS.

Questo è un bilanciamento che si sta spostando nell'Internet, lentamente si filtrano gli indirizzi invalidi. In realtà invece di rifiutarli, si etichettano come una community, abbassando la loro local preference, se è valido si installa comunque, e se è invalido e non ha altro modo di arrivarci, viene installato comunque.

RPKI è favorito poiché non ha un costo ragionevole dal punto di vista dell'hardware, inoltre non modifica il protocollo esistente. Alternative diverse da RPKI per validare il completo AS path esistono da tempo ma non sono utilizzate, poiché è necessario un cambiamento considerevole.

Il principio chiave di RPKI è che non è necessario, si può mantenere la configurazione legacy di BGP oppure implementarlo indipendentemente dagli altri.

6 Transmission Control Protocol: TCP

La socket è vista dall'applicazione come un file descriptor, una sequenza di 16 bit, associato ad un'istanza di conversazione. Quando si vuole mandare un messaggio ad una macchina remota, l'applicazione scrive il messaggio su una di queste socket, con gli stessi comandi per la scrittura su di un file. Quando si scrive un blocco di dati, questi dati vengono messi in un buffer di uscita da quella socket, e sarà il livello quattro ad occuparsi di inoltrarli al livello quattro.

I pacchetti TPC si chiamano segment, e possono essere scomposti in diversi pacchetti, in base alla massima dimensione di un pacchetto MSS, *Maximum Segment Size*, stabilita durante il three-way handshake.

Il pacchetto TPC contiene la porta sorgente e mittente, queste sono due etichette che identificano il processo sulla macchina corrispondente relativa alla connessione. Un'etichetta serve solo a denotare un'istanza di comunicazione. La conversazione in sé viene definita sia dalla porta che dagli indirizzi.

La connessione dipende dallo stato di questi due host, su due flussi di comunicazione completamente indipendenti. Si potrebbe chiudere la conversazione in un verso solo, ma quando si chiude da un verso normalmente si chiude anche nell'altro, poiché non avrebbe senso continuare la conversazione.

6.1 Finestra di Controllo di Flusso

TCP è uno strato complesso, nella scelta delle tempistiche e delle strategie, mentre è semplice rispetto ai pacchetti. Per un periodo sembrava un sistema risolto, ma si sono scoperte nuove strategie per velocizzare la comunicazione.

Per rendere più efficiente il protocollo si vorrebbero mandare pacchetti con il massimo payload, essendo un protocollo per trasferire dati. Gli strati inferiori non vedono questa problematica. Quindi solo TCP può realizzare questa strategia.

Si vuole ridurre il numero di pacchetti inviati, quindi si vogliono inviare pacchetti quasi sempre di dimensione massima. Si può realizzare riducendo il numero di notifiche, ed evitando il numero di pacchetti perduti con varie strategie. In generale il livello IP è best effort, quando un router è congestionato, butta i pacchetti in maniera casuale, senza neanche mandare ICMP. IP non garantisce il servizio di consegna, invece è garantito da TCP. Questo è un sistema critico ed è necessario garantire che la sequenza dei byte sia uguale tra mittente e destinazione. È il primo livello a rilevare le congestioni e poter agire a riguardo.

Si possono implementare strategie di attesa, temporizzando i pacchetti. Le prestazioni dei servizi TCP cambiano in maniera significativa quando si cambiano queste policy.

Il TCP mittente può aspettare che i dati si accumulino nel buffer fino alla massima dimensione prima di inviarli, ad eccezione di dati urgenti, analogamente per il destinatario in lettura.

Si può utilizzare il campo window per comunicare il buffer disponibile in lettura o scrittura, questo indica la finestra di controllo di flusso *fwnd*. In questo modo avendo un mittente veloce ed un destinatario lento, questo metodo blocca il traffico eccessivo, fino a quando non sono inviati ulteriori messaggi dal destinatario che indicano che si è liberata la finestra. Può essere utilizzato in maniera strategica, quando un server instaura una connessione, mostra prima una finestra piccola,

per evitare possibili attacchi. Poi dopo che si fida del mittente può inviare una finestra più grande per rendere più efficiente la connessione.

6.1.1 Servizi Interattivi

Servizi interattivi, come telnet, prevedono uno scambio frequente di piccole quantità di dati, come l'invio di input su tastiera verso un terminale remoto.

Leggendo un carattere carattere da tastiera viene inviato un pacchetto, il server telnet lo riscontra, e dopo averlo letto invia un notifica di cambiamento di flusso, inviando 40 byte per pacchetto, con un solo byte significativo passato sulla rete, sui 121 scambiati in totale.

Per ovviare a questo problema si utilizzano riscontri ritardati, di solito le notifiche di variazione dei controlli di flusso e gli ack sono ritardati di qualche centinaio di millisecondi, per lasciar accumulare molti byte.

6.1.2 Algoritmo di Nagle

L'algoritmo di Nagle ha lo scopo di ridurre il numero di pacchetti che viaggiano dal produttore al consumatore del flusso. Questo algoritmo invia il primo troncone di dati ed i successivi si lasciano accumulare nel buffer. Il segment successivo è inviato solo se il precedente è stato riscontrato, oppure se è stata raggiunta la dimensione MSS per il segment in coda. Questo algoritmo è molto usato, ma disabilitato in applicazioni fortemente interattive, per esempio potrebbe provocare movimenti incontrollabili del cursore.

Gli effetti sono più visibili in reti geografiche dove i ritardi sono consistenti.

6.1.3 Algoritmo di Clark

Il problema simmetrico consiste in un'applicazione mittente che produce molti dati, mentre l'applicazione ricevente li usa un byte alla volta. Il problema è il buffer di ingresso del ricevente pieno, con la corrispondente finestra di controllo di flusso pari a zero. Leggendo un byte alla volta, invia una notifica ogni volta che un byte si libera, ed il mittente invia un singolo byte per volta. Questa problematica viene chiamata *silly window*.

L'algoritmo di Clark risolve questo problema a lato server. Se a lato server bisogna notificare la modifica della finestra, si invia solo se la finestra raggiunge la metà dell'ampiezza massima, oppure un MSS.

6.2 Controllo di Congestione

Durante una connessione la porzione di banda fornita a TCP può variare, è dinamica, il protocollo deve determinare quanti pacchetti può inviare sulla banda. Avendo un meccanismo di riscontro caratterizzato da un timeout, si ha un ritardo notevole nell'identificazione di pacchetti mancanti.

Quando un router è congestionato è costretto a scartare dei pacchetti se non può gestirli, prendendoli casualmente dalla coda di pacchetti. TCP rileva una perdita di un pacchetto, almeno, quando non viene riscontrato prima del suo timeout.

TCP usa una seconda finestra scorrevole per il controllo di congestione *cwnd*, con una limitazione arbitraria definita dal router. Questa finestra non viene condivisa nel pacchetto TCP, ma è unica per una entità TCP. Quando rileva che la rete è congestionata diminuisce la sua finestra di congestione. La finestra che viene usata per spedire dati è la più piccola tra le due, fwnd ricevuta dall'interlocutore e cwnd determinata dal router.

Può essere espressa in numero di segment, quando un pacchetto viene riscontrato, incrementa il valore della finestra, diminuisce quando ne invia uno.

6.2.1 Algoritmo di Slow Start e Congestion Avoidance

L'algoritmo di *slow start* inizializza le finestre di congestione alla dimensione del più grande segment utilizzabile. Ogni ack prima del timeout aumenta la finestra di un MSS, anche se il pacchetto non conteneva tutti i byte di un MSS. Se un segment viene perso, ritorna al valore iniziale. Se la trasmissione procede senza perdite, il numero di segment cresce esponenzialmente nel tempo, proporzionale al throughput, fino al limite della banda disponibile.

Prima dello slow start le macchine mandavano pacchetti fino all'esaurimento della finestra di controllo. Prima si usava slow start solo sulle reti geografiche. Non è ottimale, poiché riparte da un singolo pacchetto alla volta, per questo è inizialmente lenta.

L'obiettivo è stimare nel più breve tempo possibile la banda effettivamente disponibile, quindi si unisce ad un ulteriore parametro *ssthresh*, un valore inizialmente scelto supponendo sia ottimale come soglia per lo slow start Quando un segment è riscontrato ed aumenta il cwnd, la finestra di controllo di congestione, cresce e se è minore della soglia scelta aumenta di uno, mentre se è maggiore della soglia aumenta di un valore del quadrato di un MSS, diviso per il valore attuale della finestra, per la *congestion avoidance*.

$$\begin{cases} \text{cwnd} = \text{cwnd} + \text{MSS} & \text{cwnd} \leq \text{ssthresh} \\ \text{cwnd} = \text{cwnd} + \dfrac{\text{MSS}^2}{\text{cwnd}} & \text{cwnd} > \text{ssthresh} \end{cases}$$

Se si rileva una congestione la soglia ssthresh è portata alla metà del minimo tra la finestra di controllo di flusso e di congestione, ma deve valere almeno due volte MSS. E si porta `cwnd` ad un MSS. Questa crescita porta a stabilizzarsi in condizioni ottimali ad un intorno del valore ottimale.

Supponendo di essere nello stato di congestion avoidance e che la finestra valga cwnd. Si possono spedire tutti i pacchetti determinati dalla finestra di congestione. Supponendo che siano tutti riconosciuti, la finestra cresce globalmente di:

$$\frac{\text{cwnd}}{\text{mss}} \frac{\text{mss}^2}{\text{cwnd}} = \text{mss}$$

Quindi si ha una crescita costante lineare, anche se non propriamente è lineare poiché ogni pacchetto successivo ha una cwnd diversa. Il comportamento è esponenziale per il slow start fino a raggiungere la soglia, oltre il quale è lineare.

6.3 Timeout e Ritrasmissione

È molto più difficile stimare il tempo di timeout per decidere quando ritrasmettere un pacchetto. Se è troppo breve, si rischia di congestionare la rete inutilmente, se è troppo lungo si perde banda possibile. Su un singolo link il tempo di arrivo di un ack è facilmente stimabile, mentre su un intera rete non è costante, la sua varianza e valore atteso cambiano nel tempo, quindi il timeout TCP cambia costantemente.

6.3.1 Algoritmo di Jacobson

Il roundtrip time stimato per una connessione è la stima corrente, calcolato secondo l'algoritmo di Jacobson. Se si realizza una media bisogna tenere in considerazione gli ultimi dati, dandogli un peso maggiore.

Si vorrebbe scegliere un timeout pari al *Round Trip Time*, RTT. Nelle prime implementazioni una volta calcolato il RTT, si considerava il timeout un valore suo multiplo: $\beta \times$RTT. Ma questo non è efficacie poiché β non può essere costante dovrebbe dipendere dalla deviazione standard della distribuzione dei tempi di ack. Dato un ack prima del suo timeout, si ha il tempo di ritorno attuale per la rete M, si può aggiornare RTT come:

$$\text{RTT} = \alpha\text{RTT} + (1 - \alpha)M$$

Dove α è un parametro per indicare quanto tenere conto delle variazioni al tempo di ritorno, normalmente vale 7/8. Varie implementazioni di TCP usano un timeout pari a RTT+4D. Dove D è detta deviazione media, una stima velocemente calcolabile della deviazione standard:

$$D = \alpha D + (1 - \alpha)|\text{RTT} - M|$$

Dove α può essere diverso da quello usato per RTT.

6.3.2 Algoritmo di Karn

Quando un segment è ritrasmesso non è chiaro a quale istanza di trasmissione si riferisce il suo ack, questo influenza il RTT, avendo diversi M possibili. Quando si ha un fallimento di invio, a quello specifico segment si applica un timeout raddoppiato ogni volta che non riesce a passare, questa è una variante facile da implementare per l'algoritmo di Karn, che introduce un *exponential backoff*.

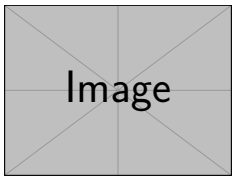
6.3.3 Fast Retransmit e Fast Recovery

Quando arriva un segmento fuori sequenza viene generato immediatamente un ack, duplicato, questo non arriva mai in ritardo. Può essere dovuto ad un inversione naturale dell'ordine dei segment, quindi si aspettano tre ack duplicati prima di ritrasmettere un segment, al quarto ack con lo stesso numero di sequenza. Si capisce di aver perso un solo pacchetto. Non ha il tempo di un timeout, ma di un roundtrip, nel caso peggiore. Poiché si ha una pipe di pacchetti, si notifica il mittente in modo immediato.

Una volta individuato questo pacchetto perso, il mittente effettua due operazioni. Il *fast retransmit* ed il *fast recovery*. Per il primo, quando rileva una perdita ritrasmette subito il pacchetto che

si suppone sia perso, senza attendere che scada il timeout, il ricevente invierà un ack cumulativo per l'intera sequenza, ora presente nel buffer. L'algoritmo di fast recovery non si porta al minimo *cwnd*, ma esegue un congestion avoidance, poiché si suppone si sia perso un solo segment e quindi è troppo drastico ripartire da zero con lo slow start. Questi due algoritmi vengono implementati insieme.

Se la connessione è stabile si impiega metà della banda effettiva, in un andamento a dente di sega, aumentando linearmente fino a quando non si raggiunge la soglia e ritornando giù fino a metà della soglia, utilizzando quindi tre quarti della banda disponibile.



Generalmente si usano tre ack duplicati, ma non sempre poiché dipende dalla crescita lineare dell'uso della banda, ma l'intuizione rimane l'uso ack duplicati per percepire le perdite.

6.4 Advanced TCP Topics

In una situazione dove il routing è stabile, TCP conosce tutti i dettagli del cammino in Internet fino alla destinazione, ed è l'unico traffico nella rete, si vuole determinare il miglior tempo per TCP per inviare i segmenti a destinazione.

Si considera un modello semplice, dove due macchine sono connessi da una serie di n link d_i, di banda μ_i, questa banda rappresenta la banda disponibile per la connessione, in una situazione reale questa non è costante, ma è variabile rispetto al traffico per quel link. Ad ogni router viene associato un buffer, di dimensione b_i. Inoltre, W è la dimensione della finestra di controllo di congestione cwnd della macchina mittente. E si suppone sia costante nel tempo. Si suppone che la finestra di controllo di flusso fwnd della seconda macchina è arbitrariamente grande.

Il flusso è limitato dalla minima banda possibile $\underline{\mu} = \min(\mu_i)$, solitamente questo collo di bottiglia non è della macchina stessa, poiché l'interfaccia direttamente collegata generalmente è più veloce dei link intermedi.

Poiché può mandare dati solamente alla velocità μ_1, se li inviasse continuamente, con $\underline{\mu} < \mu_1$ si saturerebbero velocemente i buffer dei router intermedi, quindi si distanziano i pacchetti tra di loro per avere una velocità di trasmissione media pari al collo di bottiglia. Dovendo inviare pacchetti da L bit, la macchina può inviare al massimo $\underline{\mu}/L$ pacchetti al secondo. In questo modo si occupa il meno possibile i buffer dei vari router, questi contengono al massimo un solo pacchetto.

Per potere inviare pacchetti ad intervalli regolari, bisognerebbe inserire un cronometro su TCP, ma è costoso dal punto di vista del sistema operativo.

Questi pacchetti arrivano con una latenza minima data da:

$$\frac{L}{\mu_1} + \cdots + \frac{L}{\mu_n} + 2d_1 + \cdots + 2d_n$$

Se la banda dell'ultimo link corrisponde a quello del collo di bottiglia, si potrebbe avere una trasmissione continua alla destinazione, ed un uso della banda con la massima efficienza possibile.

6.4.1 Self-Clocking

Se TCP non conosce la topologia della rete, invia pacchetti in base alla sua cwnd. Questi arrivano al router che li mette in coda e li invia in modo continuo. Quando ritorna un ack, viene immediatamente inviato un nuovo segment, così vengono automaticamente spaziati tra di loro dal tempo del collo di bottiglia. Da questo si può temporizzare il processo senza usare un clock interno. Quindi invia pacchetti subito dopo aver ricevuto un ack, per temporizzarli esattamente come vorrebbe il collo di bottiglia. Temporizzando in maniera ottimale, la coda di uscita dei router è vuota, mentre nella fase iniziale si stressa.

Questo è il meccanismo *self-clocking*, che permette alle macchine di auto-regolarsi per trasmettere pacchetti senza intasare la coda dei router. In questo modo non è neanche necessario conoscere la topologia della rete, avendo una temporizzazione esattamente sul collo di bottiglia.

6.4.2 Prodotto Banda Latenza

Questa finestra di ack di ritorno non permette di trasmettere continuamente, se è troppo corta e comunque ci impone di inviare pacchetti a blocchi. Se la finestra è troppo larga, mentre si stanno inviando i pacchetti potrebbe arrivare un ack precedente.

Si considera il tempo di immissione sul collo di bottiglia $t_{I\,\min}$, moltiplicato per il numero di pacchetti per la finestra di congestione W: $W/L \cdot t_{I\,\min}$. Si ha una finestra di controllo di congestione ideale quando questo prodotto corrisponde al tempo di round trip:

$$\begin{aligned} \frac{W}{L} \cdot \frac{L}{\underline{\mu}} &= \mathrm{RTT} \\ W &= \mathrm{RTT} \cdot \underline{\mu} \end{aligned} \tag{6.4.1}$$

Questo corrisponde si indica come prodotto banda-latenza. La dimensione della finestra di congestione dipende da questi due fattori,

6.4.3 Stima della Finestra di Congestione

Se il mittente non conosce i dettagli della connessione, non può calcolare direttamente la finestra di congestione. Può solo stimare RTT, ed ignora generalmente i valori della banda.

Per determinare il valore migliore per W si può incrementare linearmente ad ogni ack la complessità di cwnd

Ma in questo modo determinare il numero adatto di pacchetti è molto più lungo di quanto necessario. Slow-start ha uno scopo diverso, e per stimare più velocemente questa finestra effettua una ricerca dicotomica, raddoppiando ogni volta la finestra fino a quando non supera la banda massima, potendo facilmente stimare l'ordine di grandezza. Questa è un'esplorazione esponenziale dello spazio delle soluzioni.

6.4.4 Addictive Increase Multiplicative Decrease: AIMD

Il meccanismo di *Addictive-Increase Multiplicative-Decrease*, (AIMD), per TCP, vuole migliorare l'andamento a regime basato su fast-recovery e fast-retransmit. Quando la finestra di congestione aumenta, si ha una fase di congestion avoidance, ed aumenta di una quantità costante nel tempo, è un incremento additivo lineare. Ogni volta che la finestra diminuisce, questa viene dimezzata, quindi moltiplicata per 0.5. Questo comportamento di TCP ha un buon effetto sulla fairness, sull'uguale trattamento con cui la rete tratta diverse connessioni TCP, quando queste si contendono una banda, con AIMD se la dividono equamente. Nessuna delle connessioni ha precedenza, per esempio sul tempo da cui è attiva.

In questo modo, se la rete è stabile, la finestra cwnd oscilla tra due valori, la soglia e la metà della soglia, coprendo in media i tre quarti della banda disponibile alla connessione.

6.4.5 Bottleneck Bandwitdh and Roundtrip: BRR

Il controllo di congestione è basato sulla perdita dei pacchetti, si vorrebbe avere dei segnali o informazioni per poterlo determinare prima di raggiungere la perdita dei pacchetti. Nell'andamento a dente di sega, si aumenta il flusso di pacchetti linearmente rispetto al numero degli ack. Questi mettono dei pacchetti in coda al router più congestionato, un segno precoce di congestione è la dimensione del buffer del collo di bottiglia. Solo quando questo buffer è saturato, si cominciano a perdere i pacchetti. Per un lungo tempo si aumenta il numero di pacchetti nella finestra di controllo di congestione e si registra lo stesso RTT, in un caso stabile, mentre quando si satura, si registra un RTT maggiore. I tempi di immissione di uscita da quella coda aumentano, ogni pacchetto ci mette di più e quindi il RTT aumenta.

Controllando il RTT, nella fase di crescita lineare se un buffer si sta saturando il RTT cresce altrettanto. Quindi ci si può accorgere di star inserendo dei pacchetti in coda su un router.

La versione Google di TCP usa questo meccanismo di *Bottleneck Bandwidth and Roundtrip*, (BRR), per evitare le perdite, senza la necessità di un meccanismo di fast-recovery, rimanendo sospeso al livello di poco sotto l'ottimo. L'efficienza ottenuta corrisponde alla massima efficienza possibile, invece dei 3/4 dell'andamento a dente di sega.

Durante una connessione si prova ad aumentare la finestra per un breve periodo, se si rileva un cambiamento del RTT, si rallenta l'invio dei pacchetti, altrimenti si mantiene il valore aumentato.

Questo è più opportuno in una rete controllata, dove si conoscono tutte le caratteristiche. In Internet, si ha una situazione più caotica, e non è possibile determinare il motivo per l'aumento del RTT. Inoltre, non è un meccanismo equo nei confronti di altre connessioni TCP basate sulle perdite. Può essere considerato troppo greedy.

7 Livello Applicativo

7.1 Web Server

I servizi basati sul web sono cruciali ed è importante avere la possibilità di scalare, con esigenze stringenti in termini di prestazioni.

Con hit si intende la singola richiesta da parte del client, con page request si indica una richiesta che consiste di varie hit. Una sessione è la sequenza di page request consecutive da parte dello stesso client. Gli oggetti restituiti possono essere statici, dinamici, o sicuri. Ci si concentra principalmente su di oggetti statici e dinamici. La pagina stessa è dinamica poiché dipende dall'accesso, unendo dei dati appresi dal server. Il contenuto dinamico si vede in qualunque sito che ha una qualche forma di login.

Per rendere un sistema web più scalabile dal punto di vista del fornitore del servizio, si può analizzare il comportamento normale di una richiesta dell'utente. Una richiesta comporta l'interazione con un DNS per ottenere l'indirizzo del web server ed iniziare una connessione TCP per effettuare richieste HTTP. Una singola sessione richiede molte hit singole, ed ognuna di questa genera una nuova connessione, e viene terminata dopo molto poco, con un timeout che spreca risorse.

Per risolvere questo problema si utilizzano connessioni TCP persistenti con HTTP 1.1, in modo da effettuare un numero arbitrario di richieste per connessione, in modo che tutte le hit per una stessa page request viaggino su una stessa connessione TCP, risparmiando risorse sul client, sul servizio, e sulla rete, con un migliore uso della banda. Questo permette di effettuare slow-start TCP per ogni richiesta, con un migliore uso di banda.

C'è un compromesso da voler raggiungere, per decidere quando mantenere attiva una connessione, e quanto tempo mantenerla dopo la fine delle richieste. Un web server come Apache crea un nuovo processo con ogni nuova connessione, e tenendole attive possono causare uno spreco di risorse al lato server, quindi vorrebbe chiuderle il prima possibile. Su HTTP si può specificare un tempo per mantenere attiva la connessione, in genere di qualche minuto e poi il server butta giù la connessione del client.

Per effettuare più connessioni contemporaneamente un client può aprire connessioni multiple con lo stesso server, e si creano comunque meno connessioni di quante sono le hit. Se venisse permesso dal lato server si potrebbero creare un numero arbitrario di nuove connessioni, quindi vengono limitate.

Questi tipi di scalabilità sono al lato client. Per rendere più veloce questo processo si può effettuare *client-side caching*.

Il server ci avvisa tramite il contenuto cosa può essere salvato in cache e suggerisce al browser quanto tenerlo in cache.

Si usa una cache anche per il DNS locale, o il resolver locale, solitamente si fidano del TTL che gli viene fornito, anche se nella realtà non è così semplice.

Risorse statiche invece usano un ETag, negli header della richiesta, dov'è contenuto l'hash di questo file. In HTTP esiste un metodo HEAD per richiedere solamente l'header della pagina. Mettere l'ETag nell'header permette di mettere in cache la stessa richiesta. I browser moderni effettuano prima una HEAD, e se il valore dell'ETag è uguale a quello noto, prende il contenuto dalla cache, altrimenti effettua una nuova richiesta.

In un contesto aziendale si usano proxy web, anche se ormai quest'architettura non viene più utilizzata. Questo proxy ha una sua cache, ed in caso non è presente in cache effettua tutto il percorso di richiesta del client precedente, quando ottiene la richiesta, oltre ad inviarla al cliente, la salva in cache.

Quando si inserisce un layer di cifratura, questa lavora end-to-end, quindi non si può usare un proxy intermedio, altrimenti si dovrebbe rompere la cifratura.

Un servizio web scalabile è composto da vari server, un meccanismo per lo scheduling, per assegnare la richiesta al server migliore, determinato da un algoritmo di scheduling, ed un'entità che esegue l'algoritmo ed il relativo meccanismo di scheduling. Quest'entità può essere unica e fornire anche altri servizi, come CloudFlare.

- Si potrebbero avere meccanismi basati sulla risoluzione di nomi, gestito dal DNS, al livello di sessione.
- Si possono avere dei meccanismi anycast BGP, dove si annunciano di proposito due prefissi uguali da due punti diversi, in modo che automaticamente i router nei dintorni scelgano il server migliore. Si basa sull'idea che lo stesso indirizzo IP si può attribuire a più server. L'entità è a livello di routing, ed anche in questo caso si lavora a livello di sessione.
- Si possono avere meccanismi di HTTP redirection, con un web server a livello di page request, per un livello di granularità elevato.
- Per packet indirection localmente viene gestito da un load balancer, e ha un effetto a livello di sessione, page request o hit.

I web server possono avere una distribuzione locale con dei cluster, scheduling a livello di load balancer o/e redirection. Mentre con una distribuzione globale, si hanno server distribuiti globalmente, utilizzando tecniche di mirroring, redirection, DNS o anycast.

7.2 Distribuzione Locale

Per avere una scalabilità locale, un requisito è avere un cluster di server nella stessa locazione fisica, i server possono essere macchine fisiche o virtuali. Il mondo esterno vede un solo indirizzo virtuale, chiamato *Virtual IP* (VIP), a cui provvederà il load balancer a determinare a quale server inviare quella richiesta.

Si potrebbero specializzare i server, potrebbero essere per tipo di richiesta, così che il load balancer quando legge una richiesta la invia al server specifico in base al contenuto. Altre tecniche di load balancing sono Round Robin, oppure conoscendo lo stato dei server, inviando la richiesta al server meno carico.

Se si divide il carico in modo tale che le richieste vadano sullo stesso server, si può effettuare caching molto più facilmente.

7.2.1 Load Balancing a Livello di Trasporto

Si possono avere dei load balancer hardware oppure software su un normale sistema operativo. Possono realizzare un controllo sulle richieste, chiamati *Web Switch* (WS). Con dei load balancer di

livello quattro lo possono aprire fino al livello di trasporto. Ma non si può realizzare load balancing in base al contenuto della richiesta. Per realizzare questo si usano load balancer di livello 7 che possono leggere il tipo di richiesta o anche sui cookies, quindi sulle identità del client.

I load balancer di livello quattro lavorano al livello delle connessioni TCP, e deve avere una tabella per ricordarsi a quale server inviare quella connessione, poiché TCP è stateful il load balancer si deve ricordare a chi ha assegnato le connessioni. Per determinare una nuova connessione ed inserire nuovi elementi sulla tabella si guardano ai link di connessione. Se passa un FIN rimuove quella entry nella tabella.

Ci sono due possibili architetture, l'architettura a due vie, dove i pacchetti in arrivo ed in uscita attraversano sempre il load balancer. Questo però non è necessario, poiché quando arriva una richiesta ad un web server questo la può inviare direttamente a chi ha effettuato la richiesta. Tuttavia, questo non è semplice da realizzare.

L'architettura a due vie si implementa con un NAT, bisogna ricalcolare la checksum TCP poiché dipende anche dall'indirizzo IP. Mentre per i numeri di sequenza, anche se sono inizializzati random, il WS può inviare al server un SYN con lo stesso numero di sequenza per averli già sincronizzati.

A livello TCP la scelta avviene dal primo SYN, altrimenti bisognerebbe tenerlo in memoria. Si creano due connessioni TCP con gli stessi numeri di sequenza, in modo da non doverli modificare, tra il client ed il load balancer attraverso il VIP, e nella rete interna, tra il load balancer ed il server scelto.

L'architettura ad una via non ha problemi dal punto di vista di TCP poiché si usano gli stessi numeri di sequenza. L'indirizzo IP della connessione è del VIP, ma il server che risponde ha un altro indirizzo IP, quindi dovrebbe rispondere con un indirizzo che non è suo.

Sui server si può inserire sulla loopback o su di un'altra interfaccia il VIP. Non possono accettare connessioni da fuori su questo indirizzo, ma a questo punto sono in grado di creare un pacchetto con quell'indirizzo IP sorgente in uscita, come risposta al cliente.

Il forwarding del load balancer è effettuato al livello MAC, usando il MAC del server destinatario, non è necessario riscrivere il livello IP, e non serve ricalcolare la checksum. I load balancer ed i server devono appartenere alla stessa rete fisica per poter instradare a livello due.

Questo non è un comportamento standard, non è possibile realizzarlo su un computer general purpose, poiché quando ricevono un pacchetto lo aprono e lo inviano alla pila ISO-OSI.

Algoritmi di scheduling possono essere random, se uniformemente casuali, si comportano come round robin, per gli algoritmi statici. Si potrebbero realizzare algoritmi dinamici, basati su informazione dei clienti o lo stato del server, come un round robin pesato, o partizionando i client in base al loro indirizzo IP.

7.2.2 Load Balancing a Livello Applicativo

I load balancer di livello sette devono iniziare loro la connessone TCP, aprire il pacchetto HTTP e capire in base alle informazioni a quale web server inviarlo. Questo permette di realizzare politiche più efficienti. Dopo il three-way handshake con il client il LB effettua lo stesso con il server, ed al termine invia la sua risposta con i dati al client. Tuttavia, in questo modo si introduca una latenza alla risposta del client, che deve aspettare un altro three-way handshake.

Poiché sono connessioni diverse si possono pre-istanziare dal giorno in cui si accende il servizio, potenzialmente tenute aperte per sempre. In questo modo dopo che il LB riceve una richiesta, avendo una sessione già aperta con tutti i server, la risposta è diretta. La problematica diventa tradurre i numeri di sequenza, aumentando la latenza rispetto al load balancing a livello quattro.

Anche con un load balancer di livello sette si potrebbe realizzare un'architettura a due vie, ma non è efficiente per le capacità del load balancer.

È possibile cambiare server durante una sessione tra il client ed il WS, poiché sono indipendenti, per ri-allocare un server, dovendo comunque sostituire i numeri di sequenza tra le connessioni interne ed esterne.

Per contenere oggetti di grandi dimensioni si usano dei SAN, che contengono gli hard disk fisici, connessi virtualmente su richiesta ad un server, per evitare di perdere i dati in caso di guasto del server, ed è utile per condividere facilmente i dati e migrare facilmente macchine virtuali, avendo accesso alla stessa memoria di massa virtuale.

Per realizzare algoritmi di scheduling, a conoscenza delle informazioni del client, si potrebbero dividere le richieste in base al contenuto richiesto.

Si potrebbe identificare un utente per sessione, dividendo sessioni con lo stesso cookie, assegnate allo stesso server. Questi algoritmi non sono uno migliore dell'altro, dipendono dal loro utilizzo. Un altro algoritmo *locality-aware request distribution* assegna la prima richiesta di una certa risorsa al server meno carico, le richieste successive alla stessa risorsa, vengono assegnate allo stesso server.

Un'altra tecnica, *cache affinity*, si basa sul *cache manager* a cui è nota la situazione di cache del server, quindi può inoltrare la richiesta al server più adatto, che presenta la versione più recente della risposta in cache.

Nel momento in cui si va sul cloud, a seconda dei tipi di cloud si possono realizzare operazioni diverse. Un cluster elastico mette a disposizione un numero di server virtuali dinamico nel tempo, in base al carico delle richieste.

7.3 Distribuzione Globale

Invece di avere i server su uno stesso datacenter, si spargono in area geografiche diverse, per bilanciare la latenza media percepita dagli utenti, aumentando l'usabilità. Questi sono chiamati sistemi web distribuiti.

I meccanismi di distribuzione globale si possono comporre su più gerarchie, basandosi su meccanismi di distribuzione locale nelle varie PoP locali.

Se su di un DNS sono inseriti due o più record A per un nome, vengono restituiti RR, potenzialmente una distribuzione globale con questa tattica può avere prestazioni peggiori.

7.3.1 Siti Mirror

La distribuzione globale più semplice possibile sono i siti mirror, registrando nomi multipli con un indirizzo IP diverso per ogni server. Lo scheduling è lasciato all'utente che sceglie il server, normalmente da una pagina di partenza. Non è più utile su siti non statici, con pagine statiche era sufficiente scrivere la pagina statica sul server mirror.

Sono due siti separati che offrono lo stesso contenuto, ma con pagine dinamiche e database si vuole avere informazioni condivise su tutti i server. Inoltre, non c'è una replicazione visibile per l'utente ed il controllo della distribuzione del carico non possono essere implementati.

7.3.2 HTTP Redirection

Un'altra tecnica consiste nella ridirezione HTTP, utilizzando i codici di errore 301 e 302, codici che indicano che una risorsa è stata spostata ad un'altra posizione, in modo temporaneo, la prima e definitivo, la prima. Si può redirezionare la richiesta su un altro server, per assegnare la richiesta ad un server migliore. Dalla seconda connessione in poi è già noto il server giusto.

Si potrebbe utilizzare un sistema centralizzato, con un'entità che conosce lo stato dei server, per distribuire il carico, è un single point of failure, e deve conoscere lo stato di tutti i server. Oppure si può avere un sistema distribuito, in base a criteri come il sovraccarico di un server. Si può scegliere quali page request di cui effettuare il redirect, se sono di elevata dimensione, allora è efficiente scegliere un server con più prestazioni e meno latenza. Oppure si può effettuare per le sole pagine accedute di frequente.

È perfettamente compatible con ogni browser, e non introduce punti critici di vulnerabilità, distribuiti. Consente di ridirigere solo richieste HTTP, e può aumentare il traffico ed il tempo di risposta, il client instaura due connessioni HTTP per ogni richiesta. Questa tecnica ad oggi è poco utilizzata.

7.3.3 Scheduling Basato su DNS

Lo scheduling basato su DNS utilizza due meccanismi, senza ridirezione basato su record A o AAAA, e con ridirezione con CNAME. Si usano i record CNAME per diminuire il numero di record A o AAAA da modificare, poiché assegnando un alias bisogna solo cambiare il suo record.

Quando un client effettua il lookup di una risorsa chiede l'indirizzo IP corrispondente al nome del sito, riceve un opportuno indirizzo ed il TTL. La difficoltà sono i DNS server intermedi, che inseriscono questi indirizzi in cache per tutto il TTL, quindi non è possibile modificare il loro contenuto in caso si voglia effettuare load balancing.

Si può usare un TTL molto basso, ma la maggior parte dei server DNS moderni non si interessano del TTL dei record, allo stesso modo i browser o i sistemi operativi ignorano il TTL in cache. Oppure si può utilizzare un TTL adattivo, in base alla frequenza delle richieste o il carico del server.

Si impostano diversi DNS server sparsi per il mondo, configurati statici con un indirizzo IP diverso dall'altro. Si possono impostare diversi record A o AAAA per uno stesso nome, questi vengono permutati anche ad intervalli corti, producendo risposte diverse.

Si usano i CNAME per risolvere il problema della cache, quando un resolver riceve il CNAME, ricomincia il processo usando il nome canonico contenuto nel record. Questo se non ha ricevuto simultaneamente anche un record A o AAAA.

Questo permette di redirezionare il processo di risoluzione verso server secondari, in grado di effettuare uno scheduling locale.

La difficoltà principale è il TTL, ed il fatto di usare un certo numero di resolver intermedi, che spesso ignorano TTL molto piccolo.

Il DNS su cui viene adattata la risposta è il resolver, spesso è quello dell'operatore, con questo tipo di distribuzione è facile peggiorare l'arrivo dei contenuti, non potendo garantire il comportamento dei DNS intermedi.

7.3.4 IP Anycast

L'anycast è uno dei meccanismi di redirezione più famosi al mondo, questo lavora con BGP, uno stesso prefisso viene annunciato da più AS, che contiene la replica del servizio web. Il BGP di ogni AS in Internet sceglie l'annuncio che corrisponde alla metrica migliore per lui. Si porta nei punti di scambio di traffico importanti, IXP, un server ed un router che solamente annunciano in modo indipendente il loro prefisso. All'utente sembra che il traffico punta sempre allo stesso indirizzo IP, che in realtà è replicato, ognuna che si gestisce una porzione del globo. Un solo nome DNS, un solo IP, e BGP che effettua la redirezione.

Il conflitto di IP se configurato bene, funziona correttamente. BGP effettua sempre la sua scelta deterministica, e la mantiene senza altri cambiamenti. I conflitti di indirizzi IP creano problemi quando ci sta ambiguità, potendo inviare una parte della richiesta a server diversi.

7.3.5 Scheduling

Se non è possibile scegliere, quindi quando non sono note le scelte ottimali e si vuole una situazione semplice che funziona, si usa un sito mirror o IP anycast. Invece è possibile effettuare una scelta vera e propria con HTTP redirect, dopo aver inviato i pacchetti HTTP ed instaurato la connessione TCP, quindi tardi, oppure prima ancora di aver inviato la richiesta con DNS.

Le difficoltà dipendono da una buona allocazione del client al server, a causa del carico o per la prossimità tra i due.

Si vorrebbe prevedere il carico del server, per poter distribuire le richieste, in base al giorno ed ora, ai fusi orari e la distribuzione geografica degli utenti.

Algoritmi di scheduling utilizzati sono round robin o con funzioni di hash, dove la scelta è compresa interamente da questa funzione. Possono essere usati algoritmi dinamici che dipendono dalla conoscenza dello stato del client o server.

Tutti questi si basano sulla stima della distanza tra il client ed il server. Si potrebbe determinare dall'indirizzo IP del cliente, il numero di hop che separano client e server, ma questo comporta un ritardo sull'erogazione del servizio.

Alternativamente si può usare il TTL per determinare la distanza, una stima se nessuno ha modificato le impostazioni del sistema operativo. Windows e Linux hanno due default diversi, il TTL parte da 64 o 128. Anche scegliendo il minimo numero di hop non garantisce che si tratti del server a latenza minima. Il tempo di latenza roundtrip è largamente indipendente dalla prossimità geografica.

Informazioni da misure attive come il tempo di latenza del roundtrip, della richiesta, cambiano nel tempo, può anche dipendere dalla banda nota disponibile sui link. Hanno bisogno di tempo e traffico addizionale per la valutazione.

Lo scheduling generico può utilizzare più meccanismi in successione, a più livelli. Si può scegliere con uno scheduling basato su DNS, seguito da uno scheduling basato sull'HTTP redirect. Oppure a tre livelli utilizzando a monte uno scheduling basato su anycast.

Computer moderni in realtà realizzano molte query extra in parallelo per dare l'idea di andare più velocemente.

Si possono usare dei web cluster distribuiti per sommare le due tecnologie, avendo uno scheduling a livello globale per identificare quale specifico cluster da interrogare, ed in seguito si usa uno scheduling a livello locale per scegliere lo specifico server del cluster da richiedere.

7.4 Content Delivery Network: CDN

Il problema di condividere risorse è comune a molti provider, motori di ricerca, servizi di streaming, social networks, ecc. Tutti questi dovrebbero singolarmente risolvere un problema così difficile.

Da questo problema nasce il business dei *Content Delivery Networks*, (CDNs), aiutano a realizzare un sistema di distribuzione con le risorse a disposizione dei loro clienti. Realizzano un sistema di distribuzione globale, scegliendo opportuni punti, gestito da loro, ed ospitato da loro o ISP interessati.

I content provider sono i clienti che pagano, e firmano dei contratti con i CDN per distribuire i loro contenuti. Questi CDN formano una rete astratta superiore alla rete IP, per sincronizzare tutti i server, e si trova sotto alle applicazioni distribuite sul web.

Akamai è il più grande CDN, amministra centinaia di migliaia di router, ma pubblicizza di non avere risorse proprie, dice di usare, infatti, solamente gli apparati dei loro clienti. Altri CDN famosi sono CloudFlare CDN e Amazon CloudFront.

Akamai applica tutte le tecnologie trattate per i servizi scalabili per garantire una distribuzione locale e globale. Molti dei meccanismi adottati sono segreti industriali, altri sono stati rivelati per pubblicizzarli o da parte dei ricercatori di Akamai per favorire lo sviluppo e la ricerca.

Ogni provider di contenuti, vuole offrire ai suoi clienti sia un contenuto statico che dinamico, questo è il contenuto più difficile, poiché deve essere comunque richiesto alla fonte. Ma ci sono vari tempi per il dinamismo, alcuni contenuti devono essere riprodotti ad intervalli di tempo variabili.

La strategia di Akamai consiste nell'usare un singolo sistema di distribuzione globale basato su cluster in diversi punti di presenza ISP e proprietari di Akamai. Inoltrando le richieste di un utente al punto di accesso più vicino, la latenza sarà bassa, altrettanto per il traffico sulla rete.

Un altro meccanismo consiste nel classificare i contenuti in statico e dinamico, per gestirli entrambi sistematicamente. Per andare a prendere alla fonte contenuto dinamico, deve utilizzare dei cammini veloci, non quelli forniti da Internet stesso.

Hanno dei centri per misurare i tempi di percorrenza, monitorando la rete, per valutare le prestazioni di possibili richieste. Mettono insieme valori di latenza e di banda per riuscire a trasferire informazioni dinamiche dalla sorgente, per instradare all'utente lungo cammini veloci. Inoltre ottimizzano i parametri TCP per favorire il trasferimento veloce di questi dati.

Hanno dei linguaggi per descrivere le parti statiche e dinamiche all'interno di una pagina. I cammini veloci vengono usati per trasferire la parte dinamica. Ci sono altre tecnologie che utilizzano, ma non documentano per favorire questo scopo, questi permettono di forzare il traffico a passare per certi punti intermedi, quindi su rotte diverse da quelle annunciate dalla rete.

Le prestazioni sono migliorate assemblando frammenti dinamici e statici da repliche diverse. Si usa il linguaggio di markup *Edge Side Includes* (ESI), per definire i componenti dinamici per l'assemblaggio su una pagina web. Tutto questo è trasparente al livello del cliente, che riceve una

pagina web normale. I componenti ottenuti statici e dinamici vengono assemblati i una pagina web dalla replica, ogni componente ha il suo TTL. Questo aumenta le prestazioni del caching alle repliche.

I due meccanismi più comuni di distribuzione globale usati sono la ridirezione DNS, e l'URL rewriting. Questi permettono di indirizzare la richiesta di un cliente ad un particolare server. Offrono degli strumenti per realizzare al volo un rewrite dell'URL per puntare ai server del CDN, per raccogliere tipologie di traffico specificate.

Può essere usato per una piena redistribuzione del sito, o solo di una parte, attraverso certe repliche. Il server originale può quindi essere nascosto dal resto del mondo, ed accessibile solamente dalle repliche. Alternativamente è possibile renderlo accessibile dal resto del mondo, e mantenere specifici URL o risorse su delle repliche specifiche, per rendere più veloce queste richieste, possibili colli di bottiglia.

Ci sono degli script forniti dal CDN che in maniera trasparente si passano le pagine e traducono tutti gli indirizzi di un certo tipo per puntare ai server replica di Akamai. Tutte queste operazioni sono trasparenti dal punto di vista del browser.

La prima DNS hit può avvenire direttamente dal content provider, oppure per un server replica di Akamai. Questa ridirezione può avvenire a più livelli, possono inoltrare le richieste a server DNS Akamai, a cui si manda una prima richiesta di risoluzione. Quel DNS che riceve la richiesta, utilizza un meccanismo di delega per capire la posizione dell'utente che richiede la risorsa, per avvicinarlo a dei server per gestire meglio la sua richiesta. Dentro le stringhe DNS che si vedono girare quando usano il CNAME, vi è codificato dentro anche il customer. La scelta del server è basata sul customer, la sua posizione, ed il tipo di contenuto richiesto.

Se qualcosa cambia dinamicamente, rimane comunque invariato nella versione in cache, quindi Akamai gioca sui TTL, modificandoli e diminuendolo all'aumentare della granularità del DNS server da cui ha ottenuto la richiesta.

Alcuni CDN funzionano come singoli AS, con una rete globale per raggiungere punti strategici nel territorio, come Akamai, di ASN 20940. Questi instaurano peering BGP con AS *eyeball networks*, con degli utenti che principalmente guardano all'Internet per raggiungere contenuti.

Altre CDN hanno più AS e PoP diversi, il più possibile sparsi sul mondo. Akamai offre sia single-ISP e multi-ISP CDN.

Le CDN possono instaurare peering tra di loro per espandere l'area di influenza reciproca, si formano delle federazioni sostanzialmente per offrire contenuti a prestazioni migliori e costi minori rispetto a singoli CDN. Questa rappresenta la tendenza attuale del mercato.

8 Data Center

8.1 Multi-Path

L'idea di multi-path consiste nella presenza di più rotte per uno stesso prefisso, quindi avere vari next-hop possibili. Dal momento in cui sono più di uno diventa multi-path. BGP invece di scegliere una best sceglie due rotte di qualità uguale, e quindi le accetta entrambe, scrivendole in tabella. Anche alcuni IGP possono effettuarlo.

Serve il supporto kernel per poter avere più next-hop in tabella di instradamento, ma nel momento in cui si va su un pezzo di hardware si ha bisogno di hardware support per queste capacità.

Molte politiche non si possono usare, quindi si applicano politiche molto semplici come round robin.

Il concetto più importante è di inviare pacchetti dello stesso flusso sullo stesso percorso. Un flusso viene identificato a livello quattro, la quadrupla indirizzo e porta, sorgente e destinazione. Per una comunicazione bidirezionale si avranno due flussi, scambiando gli indirizzi e porte. Si effettua poiché riordinare i pacchetti è computazionalmente pesante.

Dentro un data center si vuole poter sfruttare tutto fino all'estremo. Quando si mandano dei pacchetti, TCP può ricevere pacchetti fuori sequenza e riordinarli, inviando ack duplicati, mantenendoli nel buffer. In un data center non si ha la dimensione di buffer in grado di gestire l'enorme quantità di dati che passano al suo interno. Si usano link da almeno 100 gigabit.

Se si manda un pacchetto fuori ordine per poterlo riordinare il pacchetto deve inviare un ack duplicato ed un interrupt, e si perde il vantaggio del multi-path. Si vuole immaginare una politica che i pacchetti dello stesso flusso prendano lo stesso percorso.

Ci si concentra sul tipo di multi-path più usato, *Equal-Cost Multi-Path* (ECMP), accettando varie alternative solo se hanno lo stesso costo.

Spesso vengono usati nei *Fat-Tree* dei data centers. Le policy più famose sono basate su una funzione di hash in base al flusso livello 3 e 4, anche se tecnicamente più complesso può utilizzare al meglio il multi-path. Oppure si può usare algoritmo round robin, anche se non è così efficiente come si crede. Queste policy permettono di parallelizzare la banda.

I data center qua trattati avranno multi-path, questi non funzionano su Windows, poiché il suo kernel non permette multi-path, mentre questo è disponibile su Linux e Mac.

8.2 Hyper-scale Data Centers

Quando si parlano di data centers si parla di hyper-scale data centers. Bisogna conoscere la differenza tra i due, questi sono data center che scalano tanto, i data center di AWS, Azure, Google Cloud.

Il Namex usa una struttura ridotta di questa architettura, ma è comunque dello stesso tipo.

Un data center deve avere connessioni ad Internet, sono multiple, ridondanti, ad alta velocità ed in fibra ottica. Per fabric si intende l'insieme di router e switch, l'infrastruttura dedicata a collegare i server all'Internet, ci si concentra principalmente sulla fabric. Il cuore del data center è la server farm, contenente le applicazioni ed i servizi.

Lo standard vuole che siano rappresentati verticalmente dall'alto verso il basso con Internet, fabric e server farm. Per cui si parlerà di link verso nord e verso sud, quindi verso l'alto e verso il basso, universalmente usata per i data center.

I link sono tutti in fibra ottica, poiché bisogna andare velocemente ed avere bassa latenza, e tutte le qualità che non sono garantite da cavi in rame. Ogni connessione ha link da almeno 100 gigabit al secondo. Uno switch ha almeno 64 porte, si parla di circa 5000 oggetti di rete e 60000 server.

In un anno si suppone che migliaia di macchine ed hard drive si rompano, è alta la probabilità di avere almeno un fallimento del trasformatore dedicato, e si avrà una completa sostituzione di tutte le fibre ottiche del server.

Ci sono tre principali service-models, il primo *on-premise* è costruito ed operato da una stessa società, ospitato in un edificio dell'organizzazione. Questa strategia con la moda del cloud sta venendo sempre più accantonata. I *colocation* data centers comprano un edificio e gestisce l'infrastruttura, ed altre compagni o organizzazione pagano l'affitto per portare là la propria macchina, comprando parte dell'infrastruttura e della connessione. In questo caso ci sono dei server non di proprietà della compagnia che possiede l'infrastruttura. L'ultima opzione sono i *cloud* data centers di proprietà di grande compagnie, generalmente OTT, a cui si può pagare per utilizzare macchine e risorse virtuali.

Nel cloud, chi possiede il data center affitta le sue risorse. Nell'on-premise chi le possiede le usa.

Si possono identificare due tipi di flussi di traffico in un data center, i flussi *north-south*, entranti o uscenti dal data center per l'Internet. Oppure i flussi *east-west* per permettere ai server di interagire tra di loro, soprattutto per architetture a microservizi e distribuite.

Le macchine all'interno di un data-center si muovono in continuazione, per qualunque coppia di server si vuole avere un'elevata banda di comunicazione. Dal punto di vista di un utente, le macchine virtuali rimangono sempre disponibili, trasparenti alle modifiche dell'hardware sottostanti.

La rete deve essere in grado di scalare in modo efficiente e resiliente, poiché non sono molto adatti all'uomo, poiché sono estremamente rumorosi e freddi, con illuminazione opzionale.

I nodi interni di un data center possono essere connessi con hardware specializzato, creato appositamente per comunicazione ad alta velocità tra cluster di migliaia di nodi, questa è la scelta adottata da Google, ed è la scelta più costosa. L'idea di vendor lock-in è l'idea di comprare apparati dello stesso vendor, e si sarà costretti a comprare solo apparati di quella marca, altrimenti non sarebbe compatibile con la l'infrastruttura. L'hardware specializzato rappresenta vendor lock-in, e si vuole evitare. L'altra opzione consiste di usare modelli normali, non specializzati, più economico, ed anche meno affidabili. Questa sarà la decisione che verrà usata andando avanti.

La topologia deve essere scalabile, deve poter essere ingrandita senza ricostruire l'intera infrastruttura o spegnere l'intero data center, in modo trasparente dal punto di vista degli utenti e del data center, e senza avere una spesa eccessiva. Inoltre, si vorrebbe che gli host della fabric siano in grado di comunicare tutti con tutti, utilizzando ciascuno la piena banda della sua NIC. Questo requisito di banda sarà una guida fondamentale nelle decisioni successive.

Bisogna andare su topologie di rete dedicate a realizzare data center. A livello di letteratura o altri data center OTT, si usano altre topologie, oltre a Fat-Tree, come Clos...

La Clos è stata progettata nel 1938 per gestire la scalabilità in reti telefoniche, formalizzata da Charles Clos nel 1953. Il problema che la Clos vuole risolvere è consentire di avere N linee di input e N linee di output in contemporanea, senza usare un singolo grande switch.

I Fat-Tree sono stati creati nel nel 1985, di certo non progettati per i data center. Sono un caso speciale della Clos, e consiste in un'architettura a tre livelli di nodi ad alta ridondanza. Consente una bisezione costante della banda disponibile. Tutti possono parlare con tutti usando tutta la banda. I nodi hanno le stesse caratteristiche, per immagazzinare più facilmente i ricambi.

La topologia Fat-Tree è modulare, facilmente scalabile. Viene definito da due soli parametri, il radix del nodo, un valore pari rappresentato con $2K$, indica il numero di porte del singolo apparato di rete. Mentre il fattore di ridondanza R, compreso tra 1 e K, più è alto avrà più ridondanza, più è basso più è grande la topologia. Dato un apparato si avranno K porte verso nord e K porte verso sud. Secondo la definizione del Fat-Tree si potrebbero usare diverse scelte, non analizzate in questo corso.

Le *Leaf* sono apparati connessi direttamente alla server farm, le *Spine* connettono leaf ai nodi ToF. I *Top of Fabric* (ToF) sono i nodi che forniscono comunicazione tra i PoD. I *Point of Delivery* (PoD) son oun insieme di spine in full mesh con tutte le leaf. Non sono presenti link orizzontali, bisogna sempre salire e poi scendere per attraversare orizzontalmente.

Il fattore R determina il massimo numero di link tra un nodo ToF ad una PoD. Diminuendo la ridondanza si aumentano le connessioni tra un nodo ToF a più PoD.

Anche aumentando la dimensione di K, lo schema matematico rimane invariato. Quando $K \neq R$ il fat-tree si chiama multi-plane. I nodi ToF si possono separare in insiemi chiamati piani, si distinguono perché tutti i nodi di un plane si connettono alla i-esima posizione nella spine. Rappresentazione tridimensionale.

Per collegare il Fat-Tree all'Internet si possono avere due metodi principali. Si potrebbe inserire nella ToF usando apparati con più porte, rompendo il vincolo di avere apparati solo uguali, ma si ha Internet su porte dedicate sui piani. Un'altra opzione consiste nell'usare un PoD collegandolo ad Internet, qua si sacrifica la capacità di attaccare qualcuno dei server al data center, ma si rimane rispettosi delle regole base dei Fat-Tree.

9 Kathará

Una rete è composta da diversi apparati, computer, router, switch, ecc., ognuno con differenti interfacce, diversi protocolli attivi sulla rete, diverse connessioni fisiche che generano una topologia complessa. L'Internet è una rete best effort, non è stabile, ma si voglio garantire le sue prestazioni. I router sono automi a stati finiti, non si ha il tempo di computazione per gestire tutti i pacchetti nel modo per garantire le migliori funzionalità e prestazioni. Quindi per garantire queste caratteristiche bisogna lavorare dal punto di vista della struttura della rete e le loro interazioni. Ma effettuare sperimentazioni di questo tipo usando tecnologie dal vivo è molto costoso, per cui si vorrebbe realizzare ciò tramite dei modelli locali, o principalmente localizzati per diminuire la necessità di apparati costosi. Per creare modelli di rete si può scegliere di emulare o di simulare la rete. Una simulazione riproduce le prestazioni, mentre l'emulazione riproduce le funzionalità. Non si può emulare tutto, un processore general purpose non ha le caratteristiche necessarie per emulare milioni di pacchetti come quelli di un router, che effettuano queste operazioni dal lato hardware. O si toglie il processamento di pacchetto e quindi la funzionalità, oppure si affronta la logica e si mantengono le prestazioni.

Kathará è un sistema per emulare le reti. Quindi ne rappresenta le funzionalità, su di un qualsiasi calcolatore general purpose, senza rappresentare le effettive prestazioni della rete. Viene utilizzato per valutare la loro struttura, e le qualità derivanti delle funzionalità descritte della rete. Ogni apparato di rete è in un suo container docker separato, e può avere il ruolo di un qualsiasi apparato nella rete, un calcolatore, un router, un DNS, un web server, ecc. Queste funzionalità sono disponibili in base all'immagine del container.

Ogni dispositivo emulato ha una sua console, una memoria, un filesystem, e può avere, o anche non avere, un certo numero di interfacce di rete, per comunicare con altri gli altri dispositivi emulati. Queste connessioni avvengono tramite domini di collisione emulati, a cui possono essere connesse le interfacce, ognuna connessa ad al massimo un singolo dominio.

Per usare Kathará: appunti introduttivi dal corso di Reti di Calcolatori.

Per visualizzare le informazioni sui pacchetti si può utilizzare `tcpdump`, specificando varie flag:

```
tcpdump {flags}
```

- `-e`: include gli headers del livello link, per osservare gli indirizzi MAC, se presenti
- `-i {interfaccia}`: analizza i pacchetti solo su una data interfaccia
- `-n`: non converte indirizzi con DNS
- `-t`: non stampa un timestamp per ogni pacchetto
- `-v`: stampa un output prolisso, contenente opzioni e vari campi per i singoli pacchetti

Altri comandi utili sono `traceroute`, `ping`, ecc.

9.1 Configurare un Lab

Un modello di rete si chiama *lab* in Kathará. Questo viene descritto da un file di configurazione `lab.conf`, situato alla radice del lab. In questo file possono essere presenti una serie di assegnazioni del tipo:

```
machine[arg] = value
```

Per indicare parametri o opzioni da attivare per un certo dispositivo emulato nel progetto. Si possono condividere file tra il filesystem interno dei dispositivi e quello esterno del calcolatore con le cartelle `/shared`, abilitato di default, e `/hostname`, per ogni dispositivo, disattivato di default. Tutto ciò che è contenuto in sottocartelle in quest'ultimo viene copiato e non riflesso anche nel filesystem esterno. Nella stessa directory può essere inserito un file bash `{hostname}.startup` che indica i comandi da eseguire all'avvio del progetto. Si usano per configurare la tabella di instradamento statica e gli indirizzi per le interfacce della macchina:

```
ip address add {indirizzo-ip}/{netmask} dev {interfaccia}
ip route add {indirizzo-dest}/{netmask} via {indirizzo-src} dev {interfaccia}
```

Per aggiungere delle rotte di default si usa l'opzione `default` per l'indirizzo di destinazione

Inoltre si può usare per avviare servizi, come web server, routing frr, ecc. Queste macchine sono basate su linux e `systemd` come gestore dei servizi. Per avviare un servizio bisogna inserire in questo file:

```
systemctl start {servizio}
```

Esistono strumenti per comporre file `lab.conf` in maniera grafica come, accessibili dal GitHub di Kathará

Esistono tre livelli per configurare una macchina, si può interagire da utente, dove si possono effettuare comandi come `ping` e `traceroute` per verificare la struttura e la raggiungibilità della rete, e controllare gli altri utenti che la stanno utilizzando. Elevando i privilegi ad amministratore si possono interrogare direttamente le tabelle di instradamento del router. Per modificare la configurazione stessa bisogna elevare ancora i privilegi per poter modificare la configurazione corrente. Le configurazioni che vengono applicate a questo stadio non sono immediate, solo dopo aver confermato, tutte le configurazioni aggiunte o modificate verranno attivate. Questo è rilevante, poiché se si è connessi in SSH alla macchina, è possibile che il canale di comunicazione venga tagliato, e quindi è necessario fisicamente resettare il router per poter connettersi nuovamente.

9.2 FRRouting

Un router ha tante interfacce e per gestire i pacchetti tra tutte queste e scegliere a quale inviarli. Questo viene effettuato grazie ad una tabella di routing. Una rete però non è una struttura statica ed è necessario modificare ed aggiornare questa tabella di routing dinamicamente. I protocolli di routing hanno questo scopo. Su Kathará ogni macchina che esegue un protocollo di routing identifica un router per la rete.

La suite di protocolli *FRRouting* (FRR) è open-source e disponibile per piattaforme Linux e Unix. Questo implementa vari protocolli di routing, RIP, OSPF, IS-IS, BPG, ecc. Ha le sue

radici nel progetto quagga, non più supportato, ma dovendo rimanere compatibile con macchine esistenti sulla rete sistemi e protocolli obsoleti, come IPv4, devono continuare ad essere supportati per mantenere la rete funzionante. Quagga a sua volta si basava su un progetto chiamato *Zebra*, che l'ha superato, ed è ancora in uso.

Il protocollo FRR gestisce la condizione di informazioni di routing tra vari protocolli di rete diversi, utilizzando Zebra per unire i loro risultati. Ogni suo protocollo ha un demone indipendente, che utilizza un suo file di configurazione sulla macchina.

Zebra deve unire varie tabelle di routing ottenute dai demoni di BGP, RIP, OSPF, ed ogni altro protocollo abilitato per quella macchina. Genera una tabella generale da iniettare al kernel aggiungendola alla tabella statica. Gli indirizzi IP delle interfacce di rete impostate generano automaticamente queste entry statiche. Questa tabella viene chiamata tabella di instradamento o del data-plane. Le tabelle di instradamento in Zebra sono virtuali, si trovano nel control-plane, su un software.

Per configurare FRR su Kathará bisogna creare una cartella `/etc/frr` nel filesystem degli host predisposti come router, contenente i vari file di configurazione. Il primo file, `daemons`, determina i demoni dei vari protocolli di routing da avviare, nel formato `{daemon}={'no'/yes}`. Quando vengono etichettati, vengono avviati da Zebra, questi demoni si trovano su di una specifica interfaccia, a cui viene connessa la macchina tramite loopback. I vari servizi si trovano in `/etc/service`. La configurazione per FRR è operativa e non dichiarativa, non si può definire il comportamento del router, ma solo usare dei comandi per avviare il router. Inoltre non si possono rimuovere comandi, ma per ometterli bisogna inserire la stringa `no` prima del comando. Invece di creare file di configurazioni separati per ogni protocollo, si può abilitare `vtysh` per gestire allo stesso modo questi protocolli. Per abilitarlo, o disabilitarlo, si inserisce nel file `vtysh.conf` la seguente: `{ /no} service integrated-vtysh-config`. In questo modo si può usare un unico file `frr.conf` per gestire tutte le configurazioni dei vari demoni. Per osservare la configurazione attuale, e verificare quali comandi sono stati accettati o rifiutati si può usare `show running-config`. Si può effettuare un primo debugging osservando le differenze tra l'output di questo comando, e la configurazione inserita nei file di avvio. Il linguaggio di questa shell non è *context free*, dipende dal contesto i cui ci si trova, e quindi dai comandi precedenti. Generalmente non si configura manualmente in questo modo interattivo su di una shell, è più efficiente modificare il file di configurazione e riavviare il router.

Per avviare FRR, bisogna inserire nel file di avvio della macchina `systemctl start frr`, per gestire i vari protocolli si può usare la shell `vtysh`, se abilitata, che gestisce tutti i protocolli attivi di routing, compreso Zebra. Per visualizzare le tabelle di instradamento composte dai singoli protocolli si può usare `show ip {protocollo} {opzioni}`, dentro a questa shell. Per visualizzare la tabella di instradamento complessiva si usa `show ip route`, quest è diversa dalla tabella iniettata sul kernel, poiché alcune rotte possono essere ripetuto, e Zebra seleziona quale tra queste rotte iniettare. Per configurare un protocollo si usa il comando `configure`, per scegliere che tipo di protocollo, in questo caso di routing, si usa `routing {protocollo}`. Per aggiungere una rotta statica ad uno di questi protocollo si usa `route {prefisso}/{netmask}`. Accanto ad ogni rotta, viene inserita la sua origine, può essere del kernel segnata con K, oppure può essere ottenuta da uno devi vari protocolli di rete. Su `vtysh`, si può usare il carattere `?` per mostrare le possibili alternative per completare un comando. Modificando un file di configurazione modificato su una macchina virtuale, è necessario

copiarlo su una cartella per condividerlo con il filesystem esterno, altrimenti alla terminazione di Kathará la configurazione sarà persa.

Oltre a questa modalità di interazione, si può invocare la shell con la flag `-e` per eseguire un comando senza avviare la shell interattiva: `vtysh -e "{comando}"` Si può reindirizzare con per scrivere l'output su di un file, che può essere utilizzato da altri programmi `... > {file}`, oppure indirizzandolo ad altri comandi sulla shell con `... | ...`. Per cercare un dato utile sul database, è utile il comando `grep`, che seleziona solo le linee contenenti una certa stringa o espressione regolare passata come parametro.

9.3 RIP

RIP è un protocollo di routing distance vector, che può essere gestito da FRR. Nel file di configurazione di FRR bisogna avviare RIP ed impostare la rete su cui può mandare pacchetti multicast:

```
router rip
network {prefisso}/{netmask}
```

All'avvio di FRR quindi la macchina si comporta autonomamente come un router, utilizzando il protocollo RIP ed inviando periodicamente i suo distance vector.

9.4 OSPF

OSPF è un protocollo di routing link state packet, che può essere gestito da FRR. Per funzionare invia in multicast pacchetti contenti il suo stato attuale, interfacce e reti raggiungibili. Ed ogni router si genera il suo database per calcolarsi un albero dai cammini minimi.

Alcuni comandi utili per OSPF, da eseguire direttamente sulla shell della macchina o su `vtysh` sono:

```
show ip ospf {route/interface/neighbor/database}
```

Per ogni rete un'interfaccia viene promossa un'interfaccia come router designato (DR), per gestire il database. Questo può essere scelto in diversi modi, solitamente per priorità si sceglie anche uno o più router di backup, tra quelli a priorità più alta. Viene cambiato DR ogni volta che viene modificata la topologia della rete, identificato dall'invio di altri pacchetti di stato. Per questo cambiano raramente. Questo DR viene scelto per ogni sotto-rete, e per ogni rete il suo indirizzo è quello del suo DR. Si può osservare dal comando `... database`, questo è uguale per ogni router. Per scegliere un DR ogni router invia un pacchetto di "presentazione" sulla LAN, l'interfaccia con l'indirizzo più alto che invia questi pacchetti diventa il DR. Per gli spareggi si utilizza l'ID del router a cui appartiene l'interfaccia considerata, e si sceglie in base all'ID più alto tra questi. Inoltre si può configurare una priorità, per indicare quanto un router sia propenso a diventare il DR.

Aree, router e LAN si etichettano con ID, che coincidono con certi indirizzi IP, per i router questo è l'indirizzo IP di una loro interfaccia, per le LAN è l'indirizzo del DR associato. Questi ID possono coincidere. Vengono scelti per un router come l'indirizzo dell'interfaccia più alta che ha, mentre la LAN lo sceglie come l'interfaccia con indirizzo più alto che vede.

Con il comando `... route` si possono vedere tutte le LAN raggiungibili e la loro area di appartenenza, tutte le reti presenti nell'area di backbone dovrebbero essere presenti, altrimenti la configurazione è errata.

Per configurare questo protocollo nel file di configurazione di FRR, si inserisce `router ospf` per indicare che si utilizza il protocollo, ed in seguito si possono definire le interfacce che appartengono ad una determinata area, tutte quelle che ricadono nel prefisso specificato:

```
network {prefisso}/{netmask} area {area-id}
```

Per selezionare ogni interfaccia si usa `0.0.0.0/0`, per router con interfacce su più aree è necessario specificare singolarmente o a gruppi le interfacce. Inoltre per definire il tipo di un'area backbone, stub o transit si usa la notazione:

```
area {area-id} {area-type}
```

Questo non è richiesto per la backbone. Da questa configurazione, all'avvio del lab, i router invieranno gli opportuni pacchetti link state e realizzeranno la loro tabella di instradamento.

Aree di tipo `stub` non vengono usate per il transito, questi sono connetti ad un solo router, e visibili specificando l'opzione `router` al comando per visualizzare il database OSPF. Al livello di controllo non c'è una rotta di default, si assume che tutti i router negli stub abbiano una rotta di default che punta all'esterno della rete. Ai router nelle aree stub viene offerta una rotta di default, che manca ai router nella backbone. Queste rotte puntano alla backbone, che deve gestire i collegamenti tra le varie aree. Quando vengono iniettate le rotte verso altre aree, con pacchetti link state, questi vengono non vengono inseriti nella tabella di instradamento per aree stub, sono presenti solamente i router direttamente connessi e la rotta di default.

Quando vengono iniettate rotte dall'esterno, il costo della rotta è in base al costo specificato dal protocollo esterno e configurabile manualmente, a questo viene aggiunto il costo di default per OSPF. A parità di costo per spareggiare si usa il costo dell'OSPF.

9.5 BGP

Per configurare BGP, si utilizza sempre `frr.conf`, per cominciare la configurazione si utilizza `router bgp {ASN}`. Inserendo un comando non di BGP si esce automaticamente all'ambiente BGP. Si dichiara subito l'ASN, a differenza degli OSPF. Per impostare i vicini si utilizza la sintassi:

```
neighbor {ip-vicino} remote-as {asn-vicino}
```

Si può includere anche una descrizione, per aiutare a riconoscere i vari ASN

```
neighbor {ip-vicino} description {descrizione}
```

La connessione tra due router BGP è simmetrica, se i numeri IP non combinano non si alza la connessione.

Nella configurazione `daemons` devono essere abilitati sia Zebra che BGP.

Le rotte statiche si inseriscono con un comando di tipo `network`:

```
network {indirizzo-ip} mask {netmask}
network {indirizzo-ip}/{netmask}
```

BGP effettua un controllo su questa rotta, controllando se l'indirizzo è nella tabella di instradamento, per evitare di condividere una rotta che non può soddisfare. Se non è presente la raggiungibilità nel data plane, non si possono eseguire queste operazioni nel control plane. Questo è abilitato per sicurezza, per disabilitarlo e testare altre configurazioni si può usare:

```
no bpg network import-check
```

Disabilitando questa configurazione si possono trasmettere rotte non locali o non esistenti.

Un altro comando, di default abilitato è:

```
no bpg egbp-requires-policy
```

Questa configurazione quando attiva, impedisce di passare informazioni ad un vicino, se non sono presenti policy.

Il comando `network` non modifica le rotte nel kernel.

Nei log di Zebra non vengono inseriti i pacchetti scartati, mentre si possono vedere con FRR.

Per debuggare problemi di BGP si comincia dal comando `route`, per controllare le tabelle di instradamento, se non sono presenti, si prova a connettersi con un `ping` o `traceroute`. Per indagare questi errori bisogna interrogare con il demone per controllare se sono presenti le rotte BGP che ci si aspetta di aver appreso con `show ip route`. Per controllare perché non ci sono si controllano le configurazioni BGP con `show ip bgp`. Se ci sono le rotte le alternative corrette allora il problema risiede in un altro punto. Se mancano alcune rotte, si può vedere nel `summary` che mancano delle rotte.

Il router BGP per capire quando deve cambiare nexthop, prende un annuncio da un'interfaccia e lo deve rimandare indietro dalla stessa interfaccia. `s-path` ed il percorso del traffico non sono sempre la stessa cosa.

Nel momento in cui si lavora in iBGP, l'attributo `next-hop` non cambia in uno stesso autonomous system. In iBGP i peering devono essere in full mesh, poiché sono tunnel TCP, non si considera la topologia interna dell'AS. Il nexthop coincide con l'interfaccia da cui si sta facendo uscire l'annuncio, poiché il peering è eBGP. Quando questo annuncio viene ricevuto da un router BGP in un AS, questo viene propagato ed il campo nexthop rimane invariato. In AS quando un router iBPG riceve un annuncio, per inserirlo nel data plane, deve conoscere il prefisso, il nexthop e l'interfaccia. Il nexthop di questa entry, ricerca nella sua tabella di instradamento come raggiungere il prefisso, tramite un processo di *recursive lookup*. Se non lo può raggiungere non può usare quella rotta, mentre se lo può raggiungere dalla rotta ottenuta inserisce il nexthop risultante. Non sempre è lo stesso che viene installato nella rotta del kernel, uno ottenuto dall'annuncio nel control plane, ed un altro nel data plane dalla tabella di instradamento.

Il campo `origin` descrive l'origine dell'annuncio, può essere `igp`, dichiarato interno dall'AS di origin; `bgp`, se è iniettato su BGP da parte di EGP, per retrocompatibilità, ormai inutilizzato; `incomplete` se è generato da una ridistribuzione di un protocollo IGP.

L'attributo `aggregator` si manifesta in situazioni particolari, quando vengono aggregati due prefissi insieme, inserisce la sua firma, ed indica che è stato lui ad effettuare questa aggregazione. Questa firma corrisponde al BGP router ID.

Per scegliere la best route, un router BGP riceve un numero arbitrario di rotte per raggiungere quel prefisso. Non può applicare un criterio casuale, ma usare un protocollo deterministico. La

migliore rotta è l'unica che verrà propagata ai vicini. Questi annunci devono essere relativi allo stesso prefisso.

Il processo decisionale di un router per determinare la rotta segua una preferenza specifica, ed il processo passa a passi sotto-stanti in caso di parimerito:

1. *Largest Local Preference*: dato che vive solamente dentro un AS, questo dipende dal costo delle reti connesse, assegnando una preferenza maggiore a costi minori
2. *Locally Originated*: per determinare la rotta per un prefisso, si usa il prefisso annunciato dallo stesso router, invece di usare rotte annunciate da altri router
3. *Shortest* `as-path` *Length*: vince l'annuncio con il `as-path` più corto. Se si vuole abbassare la preferenza di un annuncio che si passa fuori è possibile appendere il numero AS del router più volte, allungando virtualmente l'`as-path`
4. *Lowest Origin*: secondo l'ordine `igp`<`egp`<`incomplete`, questo è un attributo che non può essere determinato nel momento in cui viene inoltrato l'annuncio
5. *Lowest Multi-Exit-Discriminator* (MED): solamente per gli stessi AS vicini, vince chi è meno penalizzato. Questo attributo non è transitivo. Si appende ad un annuncio quando si viene inviato fuori da un AS. La MED è un attributo facoltativo, quindi non necessariamente è possibile confrontarli. Lo scopo principale è il link principale ed i link di backup, si scelgono anche in base alla banda, e definiscono quale link in uscita si preferisce usare
6. *Prefer eBGP over iBGP*: si preferisce un routing esterno, poiché mantenere un AS è un costoso, e si preferisce di avere un traffico basso. Si vuole fare bella figura con l'esterno, e mantenere la rete interna efficiente
7. *Lowest IGP Metric*: ad un certo punto bisogna avere dei fattori deterministici, per cui si decide il nexthop in base alla metrica del protocollo usato, che sia RIP o OSPF, e si sceglie l'annuncio con la metrica più bassa
8. *Lowest Router ID*: questo ha l'unico scopo di dividere la perfetta parità, il router più basso è puramente arbitrario, si è scelto come normativa, non è possibile avviare BGP su una rete con due router dallo stesso ID

Per i router Cisco, esiste un altro parametro *weight*, proprietario, che viene inserito per primo per decidere quale rotta usare.

Le policy di input permettono di filtrare gli annunci in input, prima che entrino nel processo decisionale BGP. Per ogni coppia di annunci si che sono passati si esegue il processo decisionale, e si iniettano le rotte nel kernel. A questo punto si applicano i filtri in uscita e si inoltrano le rotte agli altri AS.

9.5.1 AS-path filtering

Si possono filtrare gli annunci per vari motivi, si può filtrare l'`as-path` per motivi politici. I pacchetti passanti per la rete non sono filtrati, a meno che non siano cifrati end-to-end. Per cui è possibile

sniffare del traffico, e anche cifrato per estrarre informazioni cruciali. Delle potenze ostili tra di loro quindi vogliono evitare che informazioni critiche passino attraverso le altre. Gli USA quando è nato Internet l'hanno accomunato al servizio postale; se la posta ha come indirizzo destinazione e sorgente un indirizzo statunitense, non può passare per suolo straniero. Quindi non si può passare legalmente per AS stranieri quando gli indirizzi sorgenti e destinazione sono entrambi sul suolo americano. Per queste ed altre decisioni politiche si utilizzano dei filtri per evitare che il traffico passi per certi AS. Per realizzare questi filtri si usano elementi delle normali espressioni regolari, `regexp`.

Tutte queste politiche hanno alla fine un `deny all` implicito.

```
neighbor {indirizzo} filter-list {nome-lista} in
bgp as-path access-list {nome-lista} permit {regexp}
```

Si possono configurare le `route-map` come:

```
neighbor {indirizzo} route-map {nome-map} {in/out}
route-map {nome-map} {permit/deny} {seq-number}
    match {proprietà annuncio}
    set   {opzione attributo}
```

Si può effettuare un match su qualsiasi proprietà dell'annuncio, ed eventualmente con `set` si può modificare arbitrariamente l'annuncio, anche realizzando annunci non validi secondo BGP. Si può alterare completamente l'annuncio. Anche il `match` è opzionale, in questo caso si comporta come un filtro, applicando le operazioni della `set` a tutti i pacchetti. Pià `match` di fila si comportano come un or logico.

È conveniente utilizzare delle `access-list` o `prefix-list` per realizzare operazioni su più indirizzi contemporaneamente Le access list matchano il contenuto di tutti i prefissi possibili contenuti, con le prefix list si fanno match esatti per prefisso, mentre con access si matchano anche tutti i suoi sotto-prefissi.

9.5.2 Stub AS

In un mercato di acquisto vendita, gli ISP salgono e scendono da provider a customer, in base alle loro richieste ed offerte. Questi accordi commerciali definiscono le connessioni peer-to-peer tra i vari provider. Queste vengono mantenute segrete dai provider, e per motivi politici ed economici possono cambiare. Si possono stimare studiando lo storico degli annunci BGP, cercando di capire i provider ai vari livelli di gerarchia, divisi in cinque livelli.

C'è una sorta di grafo completo di AS che ricopre la rete con connessioni peer-to-peer.

Nella gerarchia da partire da un certo livello non si ha la default, gli AS di più alto livello non hanno un upstream a cui consegnare tutto, sono loro l'upstream. Per determinare la default si considera un prefisso che non è annunciato in rete. Se non è annunciato allora facendo un traceroute su questo prefisso, che non ha macchine, si passa solo per la default, il primo router che invia una risposta che l'indirizzo non è raggiungibile si capisce che si è raggiunto un router senza la default, quindi appartiene ad un livello alto.

Uno stub AS è una zona con un unico link con altri AS, ed a priori potrebbero non usare BGP, poiché comunicano con un solo provider. In queste condizioni generalmente si usa il routing statico in single peering.

In questo gioco di ISP chi è in più tempo in esercizio ha un vantaggio, ed ha un numero di AS maggiore, ed è salito di più sulla gerarchia. Un AS con un numero basso è più vecchio di uno con numero elevato.

Uno stub AS si configura anche senza una policy eBGP, queste rotte vengono considerate con il comando `network`. Si assume che tutti i prefissi vengano considerati da questi comandi, ma non verificano che la LAN indicata sia posseduta da questo AS, non è certificato. Questo può quindi essere annunciato ad un router di frontiera, bisogna verificarlo alla fonte con un certificato di proprietà, ma solo nel momento in cui si utilizza. Ma questi certificati possono essere duplicati e modificati, per modificare anche la distanza. Questo porta a problemi di sicurezza per BGP.

```
router bgp {id-stub}
no bgp ebgp-requires-policy
network {prefisso-stub}/{netmask-stub}
neighbor {indirizzo-isp} remote-as {id-isp}
```

I router di frontiera connessi con questi stub AS devono avere dei filtri e configurazioni di sicurezza, per garantire che riceve messaggi da indirizzi contenuti nel prefisso annunciato dallo stub AS, questo non può essere controllato automaticamente e si utilizzano dei filtri per realizzarlo. Disattivando l'opzione `bgp network import-check`, si possono annunciare anche rotte non presenti nel kernel, poiché definendo la rotta di default con `network`, questa non viene iniettata nel kernel. Si mostrano questi filtri, realizzando due prefix list `customerIn` per ali annunci in entrata proveniente dallo stub, e `defaultOut` per gli annunci in uscita dal provider, contenente la rotta di default. Si assegna la default per mostrare il funzionamento, ma ben pochi router possono avere questa configurazione con la default.

```
router bgp {id-isp}
no bgp network import-check

network {prefisso-isp}/{netmask-isp}
network 0.0.0.0/0

neighbor {indirizzo-stub} remote-as {id-stub}
neighbor {indirizzo-stub} default-originate
neighbor {indirizzo-stub} prefix-list customerIn in
neighbor {indirizzo-stub} prefix-list defaultOut out

ip prefix-list customerIn permit {prefisso-stub}/{netmask-stub}
ip prefix-list defaultOut permit 0.0.0.0/0
```

Una prefix list è composta da una serie di comandi permit e deny, con sottinteso `deny any` alla fine. Gli ultimi due comandi di configurazione filtrano il traffico per la LAN che è effettivamente presente. La default non viene annunciata direttamente verso il basso, poiché mostrerebbe

nell'`as-path` i vari AS contenuti, e dato che si vogliono mantenere segreti, si propaga, identificandosi come origine del default. Nel momento in cui si propaga la default quindi si taglia l'`as-path`. Per rapporti di tipo commerciale questo è usato per non far conoscere da dove proviene.

La rotta di default dovrebbe essere dichiarata alla radice della gerarchia, e propagata verso i livelli sottostanti, mentre i livelli intermedi non dovrebbero utilizzare il comando `network`, altrimenti i router preferirebbero la rotta generata localmente, e si rimuoverebbe la rotta annunciata da fuori. Il comando `default-originate` è un comando accessorio che non inserisce la default nella tabella locale del BGP, né nel kernel. Tuttavia se si ha la rotta di default, si comporta come se si abbia utilizzato il comando `network`, nascondendo l'`as-path`. Questo comando ha lo stesso effetto di `network` e può generare la default anche quando non viene usato il comando `network`, e quando è presente sovrascrive l'effetto di una default dichiarata localmente, sempre senza inserirla nel kernel, dato che le annuncia. Mentre nel cliente che le ha ricevute, vengono inserite nel kernel.

Se un router intermedio perde il link all'upstream contenente la rotta di default, continua ad annunciare la rotta di default con `default-originate`, ma non può effettivamente instradare il traffico alla rotta adi default, poiché ha perso il collegamento.

Il secondo comando per influenzare le policy è `route-map`, oltre ad essere è un filtro si comporta come modificatore, per esprimere una preferenza. La rotta di default viene trattata come ogni altra rotta se è generata da `network`, mentre con `default-originate` è trattata da un'altra `route-map`:

```
neighbor {indirizzo} default-originate route-map {nome} in
```

Non si possono attaccare due `route-map` ad uno stesso prefisso, come per le `prefix-list`.

Con il comando `network` le policy sono applicate sulle route `in` e `out`, mentre applicando policy speciali su `default-originate`, si hanno anche le rotte specifiche per questa.

9.5.3 Stub AS Static

In queste situazioni semplici potrebbe essere più conveniente configurare rotte statiche verso e dallo stub. È sufficiente una default statica che punta al provider come next hop, per cui non è necessario utilizzare BGP:

```
route add default gw {indirizzo-isp}
```

In questo modo non si lancia un protocollo di instradamento nel customer, almeno se non serve per comunicare con i suoi vicini interni.

9.5.4 Multi-Homed Stub AS

Un altro tipo di AS si chiama *multi-homed*, presente più link con uno o più provider. Per cui non si può filtrare direttamente il traffico, poiché uno dei due link potrebbe cadere. In generale questo avviene in grosse organizzazioni, che chiedono al provider che questi link viaggino per due strade fisiche diverse, in modo da non concentrare i guasti in una stessa zona geografica.

Ma il fatto che possano guastarsi singolarmente implica che deve essere possibile utilizzare uno solo dei due per mantenere tutto il traffico. Quindi non si può usare una `prefix-list`, poiché vincolano il traffico ad un solo link, bisogna usare delle policy per effettuare load balancing o altre

operazioni simili, quando i link sono verso lo stesso provider. Quando questi due link sono verso due provider diversi, è improbabile che uno sia il backup dell'altro, quindi si usano in parallelo, si vuole bilanciare il traffico verso i due, sia in entrata che in uscita. Nell'eventualità che un link cada, si vuole che il link verso l'altro provider regga tutto il resto del traffico. Inoltre, non si vuole che il provider lo stub come tramite per comunicare con l'altro provider connesso. Quindi non si vuole che gli annuncino vengono propagati dallo stub verso l'esterno, tramite speciali filtri per annunci che non devono realizzare dei percorsi. Questa è quindi una situazione più difficile.

Lo stub è responsabile delle macchine offerta dall'ISP, questi due link possono essere realizzati da un singolo router, o due router, per mitigare i guasti del router. Tutto ciò che entra nello stub ha due modi per entrare nello stub, e quindi bisogna scegliere quale link usare. Se si usasse IGP, si sceglie quale link usare in base alla distanza e non si possono implementare policy, quindi è possibile che il traffico tra i due router dell'ISP passi attraverso questo router dello stub. Invece con rotte statiche non sarebbe possibile configurare tutto manualmente ed aggiornarlo in caso di eventuali guasti.

Per gestirlo come primario e secondario si utilizza BGP, annunciando un prefisso aggregato su ogni link, il primario effettua l'annuncio normale, mentre il secondario aumenta la metrica negli annunci esterni, per ridurre la preferenza locale, secondo la gerarchia per la scelta delle best da parte di BGP. Se uno dei due link fallisce, l'annuncio del prefisso aggregato permette di mantenere la connessione tra provider e stub. Si può valutare rispetto alla metrica solo se l'annuncio arriva dallo stesso AS. In base a questi valori di MED, lo stub probabilmente sceglierà il primario con la metrica più bassa. Il customer assegna una preferenza minore al link di backup, questo influisce sugli annunci in entrata ed il traffico in uscita, preferendo link con valori maggiori.

```
router bgp {id-stub}

neighbor {indirizzo-isp-primario} remote-as {id-isp}
neighbor {indirizzo-isp-backup} remote-as {id-isp}

network {prefisso-stub}/{netmask-stub}
```

Non si possono inserire due route map sullo stesso peering, o due prefix list. Il nome delle route map e delle prefix list è case sensitive, ed in caso sia sbagliato, nel loro utilizzo, non le applica. Per cui bisogna fare particolarmente attenzione al nome di queste etichette.

```
neighbor {indirizzo-isp-primario} prefix-list mineOutOnly out
neighbor {indirizzo-isp-primario} prefix-list defaultIn in
neighbor {indirizzo-isp-backup} prefix-list mineOutOnly out
neighbor {indirizzo-isp-backup} route-map metricOut out
neighbor {indirizzo-isp-backup} prefix-list defaultIn in
neighbor {indirizzo-isp-backup} route-map localPrefIn in
```

Si usa una route map che fa uso di un aggregato che viene definito da un access list:

```
access-list myAggregate permit {prefisso-stub}/{netmask-stub}
```

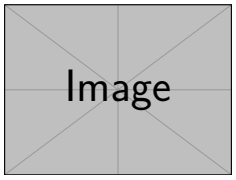
```
ip prefix-list mineOutOnly permit {prefisso-stub}/{netmask-stub}
ip prefix-list defaultIn permit {prefisso-stub}/{netmask-stub}

route-map metricOut permit 10
match ip address myAggregate
set metric 10

route-map localPrefIn permit 10
set local-preference 90
```

Verso il provider invia solo la propria LAN, mentre dal provider prende solo la default, quindi non può rimbalzarla, quindi non può essere utilizzato per farsi attraversare.

Usando la route map `metricOut`, si inviano annunci uscenti verso il link di backup aumentano la metrica. Mentre usando la mappa `localPrefIn`, si diminuisce la preferenza locale in ingresso per il link di backup. Queste si attaccano sullo stesso peering sullo stesso link di backup.

Tutti questi scenari sono configurati in modo sia determinabile con quale interfaccia si sita facendo peering.

Si rappresenta ora un laboratorio più grande sommando le varie tecnologie trattate in precedenza. Questo lab è composto da tre AS: AS20, AS100, e AS200, quest'ultimi sono due stub, ovvero non hanno a loro volta customer.



Viene usato RIP sia in AS20 che in AS100, bisognerebbe configurare RIP in questi due AS, in AS20 è semplice avendo due router, mentre in AS100 con tre router, RIP ha uno scopo molto più forte. Questi router vengono configurati con RIP e per distribuire le rotte annunciate da BGP, con RIP:

```
router rip
network {prefisso}/{netmask}
redistribute bgp
```

Questo per i router di frontiera, per i router interni che non devono parlare BGP si ha invece:

```
router rip
network {prefisso}/{netmask}
redistribute connected
```

Bisogna ignorare il `network import-check`, poiché si usano dei prefissi aggregati per gli indirizzi delle connessioni, e quindi non sono presenti nella tabella di instradamento. Uno dei due peering viene definito primario, e l'altro secondario, tra i router di AS20. Si ha una politica semplice per il neighbor primario ed una più complessa per il neighbor secondario.

```
debug bgp keepalives
debug bgp updates in
debug bgp updates out

router bgp 100
no bgp network import-check

neighbor {indirizzo-isp-primario} remote-as 20
neighbor {indirizzo-isp-backup} remote-as 20
```

Si vuole evitare che il traffico per un link diretto a AS100 rimbalzi e ritorni ad AS20:

```
neighbor {indirizzo-isp-primario} prefix-list mineOutOnly out
neighbor {indirizzo-isp-backup} prefix-list mineOutOnly out

ip prefix-list mineOutOnly permit {prefisso-stub}/{netmask-stub}
```

Poiché le tabelle di instradamento possono essere di elevate dimensioni, per AS piccoli è conveniente avere un unica rotta di default in entrata ai router di frontiera, per delegare l'instradamento esclusivamente ai router superiori. Questo si effettua creando un'altra prefix list in entrata:

```
neighbor {indirizzo-isp-primario} prefix-list defaultIn in
neighbor {indirizzo-isp-backup} prefix-list defaultIn in

ip prefix-list defaultIn permit 0.0.0.0/0
```

Per impostare i link primari e secondari bisogna impostare la MED e la preferenza locale del link scelto come secondario. La default che arriva dal link primario si vuole preferire, quindi la local preference in arrivo sul link primario deve essere più alta. Si potrebbe alzare, oppure, come in questo caso, si diminuisce nel link di backup. Il default è 100.

```
neighbor {indirizzo-isp-backup} route-map localPrefIn in

route-map localPrefIn permit 10
set local-preference 90
```

Gli annunci in uscita invece devono passare esclusivamente sul link primario, quindi bisogna modificare la scelta che faranno i router dell'AS superiore, per preferire il link primario. Questo si effettua modificando la MED, diminuendo la MED del link primario o aumentando la MED del link secondario. Il default è 0. Questa si inserisce solo sugli annunci sugli indirizzi della propria LAN.

```
neighbor {indirizzo-isp-backup} route-map metricOut out
access-list myAggregate permit {prefisso-stub}/{netmask-stub}

route-map metricOut permit 10
match ip address myAggregate
set metric 10
```

BGP spesso su configurazioni realistiche ha una serie di controlli ulteriori per evitare errori di tipo fat finger.

L'unico router di AS200 deve avere la flag disattivata di `ebgp-requires-policy`, per permettere alle rotte annunciate da BGP di essere inserite come fossero definite da `network`. I peering iBGP sono obbligatori, altrimenti BGP non funziona, mentre i peering eBGP sono opzionali, in full mesh, ovvero se sono presenti tre router, bisogna realizzare tre link, tutti connessi con tutti, con un albero ricoprente completo.

I router di AS20 hanno quindi peering tra di loro ed anche con i router di frontiera dei router stub. Si considera il router connesso sia ad AS100 che AS200:

```
router bgp 20
no bgp network import-check

neighbor {indirizzo-stub-as100} remote-as 100
neighbor {indirizzo-stub-as200} remote-as 200
neighbor {indirizzo-as20} remote-as 20
```

Questi router devono dichiarare le LAN interne, e le LAN di peering. Queste nella vita reale non vengono mai annunciate, poiché potrebbe essere possibile tagliare la connessione tra due router BGP, poiché viaggia su TCP, sarebbe possibile tagliarlo con tcp-reset. Per motivi di sicurezza le LAN di peering non sono annunciate, sono solo disponibili direttamente connesse.

```
network {LAN-as20}
network {LAN-peering-as100}
network {LAN-peering-as200}
network 0.0.0.0/0
```

Si definiscono le liste di prefissi per poter annunciare permettere alle LAN stub di poter annunciare solo i loro prefissi:

```
neighbor {indirizzo-as200} default-originate
neighbor {indirizzo-as200} prefix-list as200In in
neighbor {indirizzo-as200} prefix-list defaultOut out
neighbor {indirizzo-as100} default-originate
neighbor {indirizzo-as100} prefix-list as100In in
neighbor {indirizzo-as100} prefix-list defaultOut out

ip prefix-list as200In permit {LAN-as200}
ip prefix-list as100In permit {LAN-as100}
ip prefix-list defaultOut permit 0.0.0.0/0
```

Poiché annuncia la rotta di default deve disabilitare il controllo sulle rotte del kernel.

Con `sh ip bgp` in `vtysh` si possono vedere le rotte inserite, quelle denotate con `i` sono interne, quindi annunciate da router interni alla LAN. Questi prefissi sono arrivati da iBGP, mentre se il campo path ha una `i` l'attributo origin del pacchetto è igp. Le rotte indicate da `>` sono quelle

installate nel kernel, come preferite. Il campo local preference se è vuoto ha il valore di default, ma non era contenuto nel pacchetto di annuncio.

Per resistere ai guasti un AS stub potrebbe usare due provider contemporaneamente, potendoli usare come primario e backup, oppure in modo più complesso con load balancing.

Per impostare i due link in entrata si usa la local preference, mentre per gli annunci in uscita dallo stub, non si può usare la MED, poiché si utilizza solamente quando due link partono dallo stesso AS. Generalmente questi due link si realizzano su due router diversi, altrimenti si introduce nuovamente un single point of failure.

Si vuole evitare la situazione dove un pacchetto rimbalza e attraversa il customer per passare tra i due provider, analogamente si vuole evitare che un pacchetto per passare tra due host dell'AS stub passi attraverso i due provider.

Per gestire questi due link si considera l'idea del load sharing, una policy del tipo active-active, tenendo accesi entrambi i link, poiché essendo provenienti da provider diversi vengono pagati entrambi sempre. In questo modo si inoltra il traffico sui due link. Si potrebbe realizzare dividendo la LAN in due, ed attribuendo queste metà ai due link, senza avere un hardware apposito non è possibile realizzare una divisione accurata. Questa divisione si effettua solo su BGP. Data una $/n$, si ottengono due $/n+1$ annunciate sui due link. La prima metà passa per il primo link, e la seconda metà passa per il secondo link. Inoltre si manda la $/n$ su entrambi, meno specifica quindi non viene usata per instradare, nel momento in cui una delle $/n+1$ non viene tagliata la $/n$ che aggrega tutto si prende il traffico e quindi si ottiene piena connettività.

Si considera un laboratorio con un AS radice AS1, con due AS di transito AS30 e AS40, entrambi servono l'AS stub AS300.

I router di AS300 devono avere in uscita solamente le proprie LAN, ed in entrata solo l'AS30 o l'AS40. Entrambi questi router di frontiera annunciano la propria metà $/n + 1$ ed il prefisso aggregato $/n$:

```
neighbor {indirizzo-as30/as40} prefix-list mineOutOnly out
neighbor {indirizzo-as30/as40} prefix-list defaultIn in

ip prefix-list mineOutOnly permit {prefisso-n}/{n}
ip prefix-list mineOutOnly permit {prefisso-n+1}/{n+1}
ip prefix-list defaultIn permit 0.0.0.0/0
```

Applicando queste politiche su entrambi i router di frontiera, nel momento in cui cade uno di questi due link, l'altro link ed il rispettivo provider si prende il carico del traffico totale.

I router interni ad AS300 non usano BGP, usano solamente RIP, e dato che parlando direttamente sul prefisso $/n$, non sono necessari ridistribuzioni del kernel, con `redistribute connected`.

Avendo due router BGP è necessario un peering iBGP, ma in questo caso non è presente, poiché è necessario un pezzo di BGP ancora da trattare, che gestisce race condition, in questi casi potrebbe creare problemi, in situazioni come questa vengono specificato in maniera chiara.

Per i router di transito verso l'alto, in uscita, non vengono inserite politiche, solo un filtro in ingresso, per comunicare direttamente tutto quello che gli passa il customer. Manda verso l'alto tutti i prefissi propri, ed i prefissi ottenuti dai propri customer.

9.5.5 Pitfalls

Per ogni neighbor, per applicare comportamenti complessi bisogna utilizzare una sola route map o prefix list, poiché dichiarazioni successive sullo stesso neighbor sovrascrivono le precedenti. Alcuni software di router BGP permettono di scrivere codice, con la stessa flessibilità di un linguaggio di programmazione, FRR è flessibile, ma non è Turing completo, senza l'utilizzo di cicli.

FRR non genera errori, silenziosamente sovrascrive la prima configurazione con la seconda. Per vedere quale configurazione si sta utilizzando, si utilizza il comando `show running-config`, nella shell `vtysh`.

Le liste di prefissi si attaccano ad un neighbor, si possono agganciare in ingresso o in uscita. Vanno attaccate, e poi vanno definite. Non si possono definire in mezzo alle righe di neighbor, altrimenti si perdono, poiché si esce dalla parte di router BGP. Guardando la configurazione da `vtysh` vengono espresse in modo esplicito le `exit`, quindi si notano questi fallimenti totalmente nascosti, scartando le righe successive.

L'indentazione è pura estetica, viene usata solo per migliorare la leggibilità. Ciò che conta è l'ordine dei comandi. Prima bisogna dichiarare un neighbor, e poi vengono modificati i vari campi.

I comandi `network` devono essere inseriti dentro BGP, quindi prima di comandi per le access list e prefix list. La prefix list può essere composta da più statement, che hanno ciascuno un sequence number:

```
ip prefix-list {nome} seq {numero} permit {prefisso}/{netmask}
```

Questi statement vengono eseguiti in base al sequenze number.

Prefix list si usa per matchare prefissi esatti, per identificare indirizzi più specifici si usano le access list. Una prefix list negativa si realizza con un deny seguito dai prefissi da rifiutare, seguita da un `permit any`, altrimenti rifiuterebbe tutti gli indirizzi della lista, avendo un `deny any` implicito finale. Il primo prefisso matchato fa uscire dalla prefix list. Se si attacca ad un neighbor una prefix list che non esiste, automaticamente si ha un `deny any`, come implicito finale.

Una route map può avere zero o più azioni di match, e zero o più azioni di set. Analogamente alle prefix list hanno un numero di sequenza, il primo matchato esce dalla sequenza.

Una route map con un solo match è implicitamente un filtro, senza un secondo blocco di route map che non match niente, ha un `deny any` implicito.

Se già si è inserita con un sequence number più basso una che fa un match con tutti gli indirizzi, effettivamente non filtra niente.

Condizioni tipiche di set sono `prepend`, per aggiungere in modo virtuale lo stesso numero di AS nell'as path, tuttavia è possibile inserire AS fittizi nell'AS path, e gli annunci potrebbero avere le loro conseguenze.

Tutte le match sono in and logico tra di loro, tuttavia più condizioni sullo stesso oggetto risultano in un or logico, per esempio matchando una tra più prefix list.

Si usano le access list anche per matchare le rotte più specifiche, si può scrivere sulle access list `exact-match` per avere un match esatto dei prefissi.

9.5.6 Transit AS

Un AS di transito si occupa di propagare tutti gli annunci ed il traffico ad altri AS. Si può distribuire la tabella di instradamento BGP sul protocollo IGP, ma questo causa un aumento delle dimensioni delle tabelle IGP, e gli aggiornamenti di BGP hanno ripercussioni anche su IGP. Si può realizzare incapsulando il traffico in altri pacchetti, creando un tunnel, che comunica solamente tra i router di frontiera, oppure senza usare link intermedi, quindi avendo un link diretto tra i router di frontiera.

Avendo un AS di transito, ci sono degli AS periferici con cui comunica, attraverso i router di frontiera. Se viene distribuito BGP dentro IGP, i router di frontiera si comportano come magneti poiché si comportano come destinazione delle rotte che distribuiscono ed attraggono tutto il traffico verso quelle rotte. È possibile che un router di frontiera quando propaga le rotte su IGP il router next hop punti al router di frontiera che ha inviato il pacchetto, provocando un ciclo.

Può essere che le macchine non distribuiscano quello che hanno appresso con eBGP e non iBGP, tuttavia non esiste il comando per propagare eBGP, ma per iBGP. Di default per una macchina CISCO la propagazione di iBGP è disattivata.

Per distribuire solo eBGP, evitando il problema dei rimbalzi, si crea una route map che permette solo rotte avendo come next hop le altre LAN di peering:

```
ip prefix-list myNeighbors permit {peering-LAN} le 32
route-map eBGP permit 10
    match ip next-hop prefix-list myNeighbors
router rip
    network {LAN}
    redistribute connected
    redistribute bgp route-map eBGP
```

Quando si usano interfacce di loopback per gli indirizzi di peering BGP su un AS, bisogna settare il next hop come self, in modo da poter realizzare recursive lookup sull'indirizzo della macchina. In questo modo la connessione TCP rimane attiva nonostante la topologia della rete AS sottostante possa cambiare.

9.6 WebServer

Utilizzare un oggetto di livello applicativo è molto utile, poiché è molto più semplice testare regole BGP su oggetti di livello applicativo. In questo corso si vedranno dal punto di vista applicativo i WebServer, e verranno utilizzati per simulare dei servizi all'interno del datacenter.

Il server di default è `apache2`, il client usa `links`, un browser da linea di comando. Per avviare il webserver, si avvia il servizio di Apache con:

```
systemctl start apache2
```

Una macchina webserver non è un router, solitamente, quindi non ha bisogno di un demon di routing in esecuzione, quindi ha bisogno di avere una tabella di instradamento popolata. Ha bisogno del default gateway. Si inserisce la pagina HTML su `/var/www/html/index.html`, sovrascrivendo la pagina di default di Apache. È sufficiente inserire testo per distinguere un webserver da un altro:

```
<html>
    <body>
        <h1>Hello!</h1>
    </body>
</html>
```

Questo è necessario per valutare i load balancer per differenziare su quale server si è stati indirizzati.

Si possono monitorare gli accessi al webserver con:

```
tail -f /var/log/apache2/access.log
```

Inoltre, si può controllare il file `error.log` sulla stessa directory. Si può osservare la configurazione con `apache2 -l`.

Apache è estremamente configurabile con file `.htaccess`, ma proprio per questo è molto più lento di Ngnix. Poiché Apache controlla su ogni directory se è presente questo file, fino alla radice del filesystem, mentre su Ngnix il file di configurazione è unico.

9.7 Data Center Routing

Nel routing all'interno di data center, ci si trova in uno stesso AS, gestito dalla stessa organizzazione, nonostante questo verrà usato BGP al loro interno.

Di solito all'interno di una rete locale si effettua instradamento attaccando varie VLAN, solamente usando switch. Questa configurazione è più semplice, ma non è efficiente, poiché in un Fat-Tree si massimizza la banda, quindi la maggior parte delle porte verranno spente. Si potrebbe effettuare load balancing utilizzando VLAN, ma è estremamente complicato. LAN più grandi di barra 16 non sono effettivamente utilizzabili, a causa del protocollo ARP tutti i dispositivi collegati inviano pacchetti broadcast per ARP. Nei data center si vuole evitare il più possibile, poiché si sprecherebbe una porzione di banda solo per ARP request. Se si occupa la rete occupando qualcosa che non dovrebbe passare, è banda sprecata, utilizzo sprecato.

Per risolvere questi problemi, si lavora al livello tre, parlando di multipath, servono protocolli di routing. Queste configurazioni sono complesse, e si vuole rendere automatica, per evitare che errori umani compromettano la rete.

Si potrebbero usare protocolli di rete IGP classici, OSPF o IS-IS, invece di RIP. Si potrebbe usare BGP, usato in data center hyper-scale.

Il protocollo RIFT, *Routing In Fat-Trees*, standardizzato dall'IETF, è open source e ha prestazioni eccellenti sui Fat-Tree, ma viene implementato solo da alcuni vendor, per cui si potrebbero avere situazioni di vendor lock. Altri protocolli com OpenFabric, una variante di IS-IS per topologie Clos, o SDN.

BGP è uno dei protocolli che genera meno flooding di tutti, è stabile, la maggior parte dei problemi è stata sistemata dopo trent'anni di utilizzo. BGP ha un processo decisionale complesso, e può generare ECMP, *Equal-Cost Multi-Path* a costo zero, semplicemente interrompendo il processo decisionale.

iBGP non fa routing, ma condivide rotte, è necessario un protocollo sottostante per il routing, quindi bisogna usare eBGP. In iBGP il multi-path è estremamente complesso da impostare e realizzare.

La modalità di routing inter-dominio è diversa rispetto alla modalità di routing nei data centers. In Internet la connettività è molto sparsa, mentre nei data center la connettività è molto densa. In Internet si preferisce la stabilità rispetto ai tempi di convergenza, se un router BGP riceve una rotta simile ad una già presente in tabella, la mantiene invariata, si preferisce essere stabile. Mentre in un data center guasti non devono avvenire, quindi bisogna trovare alternative immediatamente a scapito della stabilità. In BGP una nuova rotta per convergere ci mette una decina di minuti, questo non è possibile in un data center. Lo scopo di BGP è computare una singola best-path per una destinazione, nei data center si vuole computare tante rotte possibili per una stessa destinazione, per massimizzare la banda. Questo non si effettua sull'Internet pubblico, il multi-path funziona se tutte le macchine sono della stessa organizzazione.

Esiste uno standard su come usare BGP all'interno di un data center, descritto da RFC 7938. Si assegnano numeri di AS, si modificano le policy per ECMP, e bisogna modificare i timer interni del protocollo.

Per usare eBGP si hanno diversi AS che parlano tra di loro. Non bisogna usare numeri di AS noti, poiché gli operatori li associano ai nomi e riportarli nei data center può risultare in fraintendimenti. Inoltre, è pericoloso in caso ci siano delle leak di annunci BGP, che potrebbero causare interruzioni sulla rete, o rivelare informazioni interne al data center, che si vogliono mantenere segrete.

Si utilizza il range di ASN privati, solo i 1023 AS nell'intervallo 64512 e 65534 sono privati. Se un router riceve annunci con questi numeri di AS, li scarta, se non ne appartiene. BGP è stato aggiornato per avere 4 byte per i numeri degli AS, supportando quasi un cento milioni di AS privati, sufficienti per configurare BGP in un data center.

L'assegnazione più semplice comprenderebbe di associare ogni nodo con un AS diverso, per il problema di BGP *path exploration*. Nel momento in cui cade un pezzo della rete, va a ricerca di percorsi alternativi, essendo un algoritmo distribuito ogni router implementa una sua alternativa, comportando uno stato transitorio, prima di stabilizzarsi.

Per indicare multipath FRR indica le rotte migliori con |*—, e la prima viene indicata con |¿—, le altre rotte equivalenti vengono marcate con |=—, per dire che sono equivalenti. Questo propaga la rotta incrociata, che non viene scelta poiché ha un AS path più lungo

Se un router si spegne a causa di un guasto, i suoi peer se ne accorgono dato che l'interfaccia va giù, e la connessione TCP per il peering cade. Questi annunciano ai loro vicini di rimuovere i prefissi precedentemente annunciati che passano al router ora spento. Ma in caso di multi-path, le rotte dirette vengono rimosse, ma quelle incrociate non vengono rimosse, quindi è possibile che il traffico rimbalzi in maniera incontrollata tra i router presenti, è possibile che venga creato un loop di rete. Ed in un data center il quantitativo di traffico è presente a link saturo, quindi anche pochi secondi di loop possono realizzare problemi gravi.

Per assegnare gli AS allora si utilizza una diversa alternativa, ogni foglia di un PoD ha un suo AS, i nodi spine di una stessa PoD hanno lo stesso AS. Mentre tutti i nodi dei ToF sullo stesso piano appartengono allo stesso AS. Qui non si hanno peering iBGP, poiché i nodi nei ToF e nelle spine dei PoD non sono direttamente collegati. Inoltre, serve siano sullo stesso AS solamente per l'algoritmo di loop avoidance di BGP, si blocca automaticamente senza rout-map le rotte incrociate. Automaticamente sono valley free, senza inserire politiche.

Si ottiene la scalabilità originale dei Fat-Tree, unita al loop avoidance, ed in caso di guasti prolungati evita di realizzare rotte anomale. Questo permette di avere più percorsi tra due foglie

nello stesso PoD. Per foglie tra due PoD diversi invece il numero di multi-path è esponenziale all'aumento di K. Questi passano per l'AS dello spine del proprio PoD, l'AS del ToF, e l'AS dello spine del PoD dove si trova la foglia. Essendo questi percorsi di stessa lunghezza, si può usare ECMP, essendo tutti dello stesso costo.

In realtà BGP se si abilita ECMP di default, considera due rotte uguali se hanno tutte le metriche dell'algoritmo decisionale, eccetto per il router-ID, ma questo implica che i due annunci siano uguali, confrontando l'AS path elemento per elemento. Nelle multi-plane gli AS path non necessariamente saranno identici, quindi alcuni annunci avranno un percorso e per altri il contenuto dell'AS path sarà diverso.

Si utilizza l'opzione di FRR, ed in tutti i maggiori vendor, `as-path multipath-relax`, per usare la versione rilassata e controllare solamente gli AS path per la loro lunghezza. Questo non si può usare in BGP vero, poiché si rischiano situazioni in cui il traffico è degradato per l'attivazione del multipath.

Il primo timer da modificare è l'*advertisement interval timer*, di default di 30 secondi, in un data center si imposta a zero secondi, per inviare appena possibile un nuovo annuncio. Poiché è necessario un algoritmo di instradamento rapido.

I timer di keepalive sono come per TCP, mentre hold indica quanti keepalive da aspettare prima di buttare giù il peering. Di default in BGP il keepalive viene mandato ogni minuto, mentre l'hold sono tre minuti, il tempo di tre keepalive, prima di tagliare il peering. Nei data center si impone un keepalive di tre secondi, ed un hold di 9 secondi. Il connect timer, è il tempo prima di provare a riconnettersi ad un peer, nei data center si tiene a 10 secondi, di default è un minuto.

Bisogna automatizzare la configurazione, si ha necessità di renderla flessibile. Si usano due configurazioni la prima *unnumbered interfaces* e la seconda *peer groups*. Per evitare di realizzare tutti i peer singolarmente, utilizzando unnumbered interfaces non si inserisce l'indirizzo IP su quelle interfacce. Inoltre, si usa l'indirizzo IPv6, che autonomamente si crea un indirizzo link-local. Grazie a questo si può indicare di realizzare un peering su chi si trova all'altro lato di un'interfaccia. Si possono creare configurazioni boilerplate, permettendo di scalar tanto.

Invece di applicare le policy direttamente ai vicini, uno ad uno, si può applicare una policy ad un gruppo, ed aggiungere tutti i vicini interessati a quel gruppo.

In questa versione per connettersi ai server, si assegnano dei prefissi, poiché nel cloud computing, si affittano VM o container all'interno di un server. Per accedere a questi container o macchine virtuali, si utilizza parte del prefisso assegnato per indirizzare i container o le VM contenuto nel server. Sarà la foglia ad assegnare il prefisso del server.

Bisogna abilitare IPv6 su tutti i router, specificando nel `lab.conf` per tutti le macchine:

```
{host}[ipv6]=True
```

La nomenclatura in questi laboratori segua la convenzione: `{tipo}_x_y_z`. Il tipo può essere `tof`, `leaf`, `spine` e `server`. Per i ToF i numeri `x`, `y` e `z` sono rispettivamente il numero del piano, il livello, sempre due, ed il numero di ToF. Per una foglia si ha il numero di PoD, il livello, sempre zero, ed il numero della fogli.a. Per Spine si ha il numero di PoD, il livello, sempre uno, ed il numero di spine. Per un server, il primo è il numero di PoD, il numero di foglia corrispondente ed il numero del server.

Inoltre bisogna abilitare la multipath policy, si usa l'opzione `sysctl` per impostare i parametri nel kernel:

```
{host}[sysctl]="net.ipv4.fib_multipath_hash_policy=1"
```

Il multipath con quest'opzione si basa anche al livello quattro.

Per i tof la configurazione di `tof_x_y_z.startup` è sempre e solo avviare FRR:

```
systemctl start frr
```

Nella configurazione bisogna impostare il BGP router ID, rappresentato come un'indirizzo IP, di default BGP si prende il numero di una delle sue interfacce, ma non avendo IPv4, avrebbe `0.0.0.0`, quindi bisogna specificarlo. Questo è un campo di 32 byte che si rappresenta per facilità come un indirizzo IPv4.

```
router bgp {ASN}
    timers bgp 3 9
    bgp router-id {indirizzo-ID}
    no bgp ebgp-requires-policy
    bgp bestpath as-path multipath-relax
```

Tendenzialmente i router sono collegati ad una LAN di management, dedicata all'amministratore per entrare su di una macchina e controllarne lo stato, potenzialmente queste interfacce avranno come indirizzo quello corrispondente al router-ID, quindi si inseriscono indirizzi IP privati univoci. Gli operatori di tipo wireless, hanno torri radio sparse sul territorio, connesse tramite fibra, con una sim 4G dedicata per poter sempre connettersi su un canale *out-of-band*, che in caso di guasto può essere usato dall'operatore per gestire il guasto. In questi lab, la LAN di gestione non viene rappresentata.

Si creano dei peer group come:

```
neighbor {nome-gruppo} peer-group
neighbor {nome-gruppo} remote-as external
neighbor {nome-gruppo} advertisement-interval 0
neighbor {nome-gruppo} timers connect 10
```

Questi gruppi possono essere `fabric`, per indicare che vadano verso l'alto, o `TOR`, *Top Of the Rack* che vanno verso il basso. Si chiamano così poiché vengono messe in cima ad uno stesso rack, Per indicare di effettuare peering con un'interfaccia si usa:

```
neighbor {interfaccia} interface peer-group {nome-gruppo}
```

Inoltre bisogna abilitare IPv4, poiché gli indirizzi usati sono IPv6, bisogna attivare la modalità per annunciare indirizzi IPv4:

```
address-family ipv4 unicast
    neighbor fabric activate
    maximum-paths 64
exit-address-family
```

Su un peering IPv6, scambia prefissi IPv4, e quando installa questi prefissi, avrà come next hop un indirizzo IPv6. Inoltre limita le rotte multipath a 64, questa è l'opzione che effettivamente attiva multipath.

Una spine ha due peer group, per differenziare i link che vanno verso l'alto e verso il basso.

La leaf ha un solo peer group, ed inoltre annuncia i prefissi dei suoi server direttamente collegati:

```
address-family ipv4 unicast
    neighbor TOR activate
    network {prefisso-1}/{netmask}
    network {prefisso-1}/{netmask}
    maximum-paths 64
exit-address-family
```

Bisogna indicare negli startup delle leaf i punto uno dei prefissi a cui sono collegate, inoltre ai server si assegna un indirizzo IP, per esempio il due, e si inserisce una rotta di default verso la sua leaf. Tutti gli altri 252 indirizzi possono essere usati per indirizzare le macchine virtuali dentro i server. Queste si possono re-serializzare in altro modo, si possono assegnare prefissi più piccoli ai server in base alle necessità.

Normalmente l'instradamento avviene al livello due, quindi il router usa l'indirizzo IPv4 o IPv6 del next-hop per ottenere o con ARP, o con solicited node, un indirizzo MAC, per instradare il pacchetto. Per cui si possono instradare pacchetti IPv6 o IPv4 con next-hop IPv4 o IPv6.

Quando si usa il multipath, il traceroute perde completamente di valore, poiché mostra un solo percorso tra i tanti possibili. Per calcolare più strade bisogna modificare l'hash, che dipende dall'indirizzo e la porta. Sono stati inventati due nuovi strumenti `paris-traceroute` e `dublin-traceroute`, questi effettuano un traceroute con pacchetti di livello quattro, per alterare il contenuto e modificare l'hash, per poter mostrare in output il multipath. Non è detto che usando questi strumenti sia possibile vedere l'output come un normale traceroute, ma viene fornito come file JSON abbastanza complessi, esistono strumenti online per visualizzare questi risultati.

Usando traceroute, quando un router riceve un ICMP per time exceeded devono mandare un pacchetto ICMP per avvisare il router mittente, ma non hanno un IPv4 configurato. Quindi esiste un indirizzo IPv4 apposito da utilizzare quando non si ha un indirizzo IPv4 configurato, `192.0.0.8`.

Facendo routing il traffico broadcast non esiste, avendo solamente connessioni via cavo tra due macchine, un pacchetto è sempre unicast a livello pratico, azzerando il broadcast, se non nella LAN dei server. Copiando una configurazione, bisogna modificare solamente il router ID, e l'ASN.

9.7.1 VXLAN

In una configurazione normale di un Fat Tree ogni server è legato solamente ad una leaf, se questa guasta, tutti i container su quel server vanno offline. Un orchestratore quando vede che un set di container è andato offline, prova a crearne altri, per mantenere il numero di container prestabilito. Oltre a questo scopo di creare container, ha lo scopo di bilanciare il carico spostandoli tra i vari server possibili, per avere un'esperienza utente migliore possibile. Lo scopo dell'orchestratore deve verificare che tutto è funzionante, quindi se dei container cadono prova a riavviarli, se si accorge che il server è guasto li re-istanzia sugli altri server.

Gli orchestratori possono effettuare queste operazioni anche su richiesta dell'utente. Poiché questi container hanno un indirizzo IP, questo è legato al server, se vengono spostati su un altro server4, cambia l'indirizzo IP. All'interno di un datacenter i container vengono migrati spesso nella loro vita, per cui questo non può rappresentare un rallentamento per il data center. Non si possono reindirizzare i container ogni volta. Per aspettare che le cache di DNS si allineino ci sarebbe un down time considerevole. Nel momento in cui un datacenter diventa di un cloud provider, i container in un server appartengono ad utenti diversi, e si vedono tutti uno vicino all'altro perdendo l'isolamento tra server.

Non si ha modo di muovere i container senza ri-mappare gli indirizzi IP, gli utenti utilizzatori di un servizio cloud saranno costretti ad usare gli indirizzi IP del data center. Questi vorrebbero usare una LAN privata per comunicare con altre macchine dell'organizzazione. Oppure un utente potrebbe essere un AS, e vorrebbe indirizzare le richieste alle proprie macchine.

Per gestire più utenti, bisogna capire l'architettura dei server. I server sono gestiti a tre livelli, il livello bare metal, l'hardware su cui operano le virtual machine, in queste macchine virtuali vengono creati i container offerti ai propri clienti. Si creano le VM per poter separare le risorse in modo più preciso. In questo corso, si parlerà di due livelli soltanto, quindi si tratterà in maniera mischiata container o VM, a livello della rete non si hanno differenze. Si deve permettere lo spostamento delle VM o container tra i vari server. Quando si vuole istanziate un orchestratore, per lavorare la meglio richiede una serie di qualità. Quando si sposta un container vuole avere un downtime minimo, e poter portare appresso l'indirizzo MAC del dispositivo. Tecnicamente è fattibile, poiché gli indirizzi MAC di VM e container sono virtuali. Dal punto di vista delle prestazioni di rete si vogliono per evitare di effettuare nuovamente richieste ARP.

Quando si ha una VM, il suo indirizzo MAC è scritto all'interno della VM stessa, altrimenti si avrebbero problemi a migrare tra VM. Si vuole avere uno spostamento rapido, e non si possono toccare tutte le macchine del data center per modificarne la cache.

Ulteriori requisiti dipendono dal tenant, o cliente, vorrebbero poter creare la propria LAN, e mantenerla privata, senza che altre VM o container sullo stesso server possano sniffare il traffico. SI vuole isolare il traffico tra quello di altri clienti, ed isolare il traffico da quello del data center.

Questo è un requisito fondamentale del traffico, inoltre, ogni cliente vuole poter gestire il proprio indirizzo IP, come se la VM o container nel cloud fosse all'interno della rete privata, con IP privato, o all'interno dell'organizzazione del cliente.

Si vuole poter gestire il proprio spazio di indirizzamento. Quando si parla di IP privati, chiunque può usarli, quindi bisogna stare attenti a tenere le reti virtuali dei vari clienti isolate, altrimenti si potrebbero mischiare al traffico di altri utenti, che potrebbero essere malevoli.

Per rispettare questi requisiti si usano dei tunnel, dei canali di traffico che usano un'incapsulamento aggiuntivo, non usano il solito stack protocollare, ma è presente una duplicazione di parte dello stack protocollare. Si creano dei tunnel tra tutti i container dello stesso tenant, per isolarlo dal traffico del data center e dagli altri tenant.

Il problema consiste nell'utilizzare un protocollo di tunnel opportuno, si vuole avere un tunnel incapsulato all'interno del livello quattro, per utilizzare a pieno il multi-path, il pacchetto deve essere almeno di livello tre, per poter attraversare i router. Inoltre, si vuole attraversare l'Internet, tra due data center dello stesso cloud provider, in modo trasparente dal punto di vista dell'utente. Se si usasse una stessa tecnologia per questi requisiti sarebbe molto favorevole dal punto di vista

del data center. I pacchetti di livello tre hanno dei problemi a passare sulla rete, poiché la maggior parte dei firewall usati in rete sono di livello quattro, per cui è necessario incapsularli a livello quattro, per evitare di essere scartati.

I protocolli di tunnel conosciuti sono VLAN, tecnicamente sono protocolli di tunnel, sono di livello due, quindi non possono essere usati nei data center di livello tre. MPLS può essere una scelta, poiché per ogni nuovo tenant bisogna creare una nuova etichetta e quindi riconfigurare ogni router, per effettuare label swapping correttamente, analogamente all'aggiunta di un container. La configurazione è pesante, e non è facile farlo passare via Internet, poiché è necessario avere accordi con ISP per inoltrarli sulla base dell'etichetta.

Si usa un nuovo protocollo chiamato VXLAN, creato ad-hox per risolvere problemi simili. *Virtual eXtensible Local Area Network*, standardizzato in un RFC 7348. È stato creato apposta per gestire più tenant all'interno dello stesso data center. Incapsula pacchetti di livello due, in realtà al livello due si parla di frame non di pacchetti, in pacchetti UDP. Questo permette di migrare anche gli indirizzi MAC, per non alterare il pacchetto, non si ha bisogno di ricalcolare il checksum, che nel caso del pacchetto di livello due si chiama FCS, in coda al pacchetto.

Si è ottenuto isolamento, incapsulamento degli indirizzi MAC, attraversamento dei router e dell'Internet.

Dentro l'header UDP si inserisce un piccolo header VXLAN. Si hanno 24 bit per il campo VNI, *VXLAN Network Identifier*, dedicato all'identificativo del tunnel, ben oltre le possibili VLAN, purtroppo si hanno 32 bit sprecati, sono bit riservati, per usi futuri, ma si è capito fossero completamente inutili. Inoltre si hanno altri 8 bit per il VTEP, *VXLAN Tunnel End Point*, il dispositivo della rete che si occupa dell'incapsulamento e de-incapsulamento dei pacchetti.

La porta di destinazione standard per VXLAN è definita dallo standard 4789. La porta sorgente è un campo che in realtà non serve a molto, solo quando si fanno delle connessioni TCP per avere dei pacchetti di ritorno. Si usa il campo sorgente di VXLAN ad un numero casuale in modo che ogni flusso di VXLAN ha un numero che cambia. Pensando al flusso tra due container, si avrà un outer IP uguale, così come la porta di destinazione, se non si variasse la porta sorgente, verrebbe scelto sempre lo stesso percorso, per cui si prende casuale per poter sfruttare al meglio il multi path.

Bisogna essere completamente trasparente dal punto di vista dell'utente, non deve avere visibilità di nulla oltre al container. Bisogna incapsulare tutto ciò che passa attraverso il data-center e decapsularlo quando esce. La porta sorgente è volutamente casuale per poter utilizzare il multi-path.

All'interno di un data center, si ha una rete privata, quindi si può aumentare la MTU usando un'opzione di ethernet chiamata jumbo-frame, che porta il frame a 9000 byte, quindi un overhead di 50 byte per l'incapsulamento VXLAN non è considerevole.

Il problema di base è che in un data center si hanno tante VTEP, ma si vorrebbe che il pacchetto avendo un MAC di destinazione arrivasse alla VTEP giusta. La VTEP di una macchina quando incapsula un pacchetto, deve sapere a chi inviarlo, a quale VTEP sulla stessa VNI. Usa la tabella MAC-to-VTEP, altrimenti dovrebbe inviare il pacchetto in broadcast. Questa tabella tiene delle coppie MAC ed IP, dove il MAC è l'indirizzo della macchina interna alla VTEP, mentre l'indirizzo IP è relativo alla VTEP associata. Lavorano esattamente come le tabelle degli switch, quando passa un pacchetto attraverso una VTEP, si segna dove sono presenti le macchine, associando l'IP sorgente del pacchetto, ed il MAC interno aprendo il pacchetto. Queste entry hanno una scadenza,

esattamente come la tabella degli switch. Questo lega un indirizzo MAC ad un indirizzo IP, a differenza di uno switch che lega un indirizzo MAC ad una porta dello switch.

Per poter comunicare tra le due macchine, è necessario inviare un messaggio ARP, con un indirizzo MAC destinazione broadcast, e questi pacchetti sono diversi dal broadcast di una VTEP che non conosce l'indirizzo MAC destinazione. Gli apparati di rete moltiplicano il pacchetto, la macchina ha mandato un solo pacchetto. In questa situazione, non si può mandare veramente in broadcast, ma si manda una copia del pacchetto per ogni VTEP che esiste. Si usa il multicast, associando ad ogni VNI un IP multicast della rete fisica, in modo che si evita di mandare copie del pacchetto, ma uno solo destinato al gruppo multicast e saranno i dispositivi di rete a sdoppiarlo per i vari indirizzi. Quando una VTEP appartiene ad una VNI, si iscrive anche al gruppo multicast relativo a quella VNI. Questa è una soluzione molto più scalabile, ma usare multicast in rete richiede di usare molti più protocolli, IGMP, IGMP Snooping, PIM, ecc. Sono tre protocolli abbastanza ostici da configurare, e devono essere configurati abbastanza bene, altrimenti può diventare una broadcast storm. Deve funzionare bene anche il sistema delle iscrizioni, IGMP Snooping, gli switch devono ricordarsi le iscrizioni, IGMP, e PIM, nella sua versione sparse, deve gestire queste connessioni. Poiché non vengono usati quasi mai, non è nota quell'esperienza necessaria per poterli configurare adeguatamente.

Invece di disabilitare il traffico multicast, si utilizza la tecnica del proxy ARP, questo è il protocollo che causa il più traffico broadcast. Quando ad una VTEP arriva una ARP request, si potrebbe realizzare un proxy ARP su una VTEP, in modo che se un pacchetto ARP passa per un apparato che conosce la risposta, blocca il broadcast e risponde come se fosse il vero utente, invece di lasciar continuare la richiesta broadcast, creando un pacchetto finto, spacciandosi per l'utente finale e rispondendo.

Invece di questa soluzione, si usa EVPN-BGP, standardizzato da RFC, *Ethernet VPN*, (EVPN). Utilizza MP-BGP, *Multi-Protocol BGP*, questa è la capacità di BGP di trasportare dati di protocolli diversi. Questo è quello usato per andare oltre BGP classico, come IPv6, poiché BGP è nato con IPv4. L'idea è che esistono degli identificatori chiamati AFI, *Address Family Identifier*, e SAFI, *Subsequent AFI*, che rappresentano una famiglia di indirizzi. Vengono usati da BGP per trasmettere indirizzi IPv6, next-hop IPv6. Usando AFI pari a 25, e SAFI pari a 70, BGP, protocollo livello tre, annuncia indirizzi MAC. Una VTP apprende automaticamente gli indirizzi MAC locali e gli annuncia alle altre VTEP della stessa VNI. Inoltre, le VTEP fanno da proxy ARP per limitare il traffico broadcast.

Due VTEP instaurano un peering eBGP multi-hop. All'accensione le macchine si mandano sempre un pacchetto, per cui una VTEP impara subito il MAC di una macchina locale, e lo passa subito annunciandolo ai suoi VTEP peer.

Implementare VXLAN, si ha un overhead di 50 byte, per cui nello startup delle macchine connesse alle VTEP, bisogna modificare la MTU:

```
ip link set dev {interfaccia} mtu 1450
```

Inoltre, nella VTEP corrispondente bisogna impostare BGP:

```
router bgp {asn}
    no bgp ebgp-requires-policy
```

```
neighbor {indirizzo-peer} remote-as {asn-peer}

address-family l2vpn evpn
    neighbor {indirizzo-peer} activate
    advertise-all-vni
exit-address-family
```

Bisogna attivare l'address family giusta, in questo caso `l2vpn` e `evpn`, questi sono standardizzati dallo IANA, bisogna attivarli sul vicino giusto, ovvero gli altri VTEP. Inoltre, bisogna far passare i pacchetti per tutte le VNI. Bisogna permettere cambiamenti nella configurazione del data center, senza dover modificare la configurazione, per cui non è noto a priori quale VNI passeranno.

Lo startup dei VTEP è più complesso, manca l'interfaccia `eth0`, poiché è quella connessa verso le macchine da connettere. La VTEP è al livello due, non può avere un indirizzo IP. Bisogna aggiungere un'interfaccia VXLAN, specificando un nuovo nome, il VNI corrispondente, la porta e l'indirizzo di destinazione, e si specifica di non usare multicast con l'opzione `nolearning`. Bisogna inserire un companion bridge, necessario per una VTEP. Si indica di non generare indirizzi su questo nuovo bridge, questo lavora solo a livello due e bisogna specificarlo, altrimenti Linux genererà per lui un indirizzo livello tre. Attaccare un'interfaccia ad un bridge su Linux si chiama *enslaving*, si specifica di attaccare questo bridge all'interfaccia VTEP creata. Si attacca l'interfaccia `eth0` al bridge, e si accendono queste interfacce. L'ultima istruzione è accendere FRR, che si connette in automatico al bridge creato, per connettersi alla MAC-to-VTEP table e trasmetterla tramite BGP.

```
ip address add {indirizzo} dev eth1

ip link add {nome-vtep} type vxlan id {vni} dev eth1 dstport {porta} local
↪  {indirizzo} nolearning

ip link add {nome-bridge} type bridge
ip link add {nome-bridge} addrgenmode none

ip link set dev {nome-vtep} master {nome-bridge} addrgenmode none
ip link set {nome-vtep} type bridge_slave neigh_suppress on learning off
ip link set dev eth0 master {nome-bridge}

ip link set up dev {nome-vtep}
ip link set up dev {nome-bridge}

systemctl start frr
```

Si usa un bridge poiché bisogna salvare questa tabella da MAC a VTEP, e verrà salvata dentro ad un bridge. Questo diventa un bridge normalissimo quando si attacca alla VTEP. FRR si connette solamente al bridge per annunciare ed inserire le informazioni che passano su eBGP.

Avviando il lab si possono verificare molti indirizzi con `ip link`.

Con `sh bgp sum`, si vedono i peering instaurati con L2VPN. Si usa il comando `sh evpn mac vni all`, sempre dentro vtysh, per vedere la tabella MAC-to-VTEP. Questo indica il numero di macchine remote e locali, ad ogni entry della tabella è inserito l'indirizzo MAC, ed indica se sono remoti o locali, con l'indirizzo a cui è connesso. Per vedere la tabella di instradamento del bridge, si usa il comando `brctl showmacs` seguito dal nome del bridge. Per ogni entry è indicato se è locale, ed il suo ageing timer. Gli indirizzi MAC locali possono apparire duplicati, ma non causa problemi.

9.7.2 Configurazione

In una struttura di Fat-Tree in un data center, bisogna determinare l'apparato che si occupa di essere la VTEP. Non può essere il server, poiché deve poter parlare BGP, ed un server non può usare questo protocollo. Una soluzione migliore è usare una leaf, poiché già parla BGP, il problema è che tra la leaf ed il sever è presente un unico cavo, e quindi estraendo i pacchetti il server non può distinguere a quale tenant appartengono, e quindi un utente malevolo potrebbero leggere tutto il traffico. Si usa una VLAN all'interno del singolo link tra la leaf ed il server per distinguere tra loro i vari tenant. Tuttavia, c'è un problema sulla scalabilità avendo al massimo 4096 VLAN, questo è abbastanza per un server. Si mappano le VNI con unu set di VLAN, si potrebbero mappare tecnicamente per ogni server. Nei lab si usa una versione semplificata dove il mapping è univoco in tutto il data center, ma è impossibile mappare un dominio di 4 milioni su un codominio di 4 mila. La lef estrae i pacchetti VXLAN, e li incapsula in un pacchetto VLAN, il server sa gestire la VLAN a livello hardware direttamente, e quindi è sufficiente rimappare queste VLAN ai tenant. Dalla leaf in sù si parla VXLAN, sulla leaf si parla anche VLAN con il sever, ed all'interno del server non ci incapsulamenti. Sono necessari due mapping, uno nella leaf per mappare la VLAN ad un VNI, ed uno nel server per mappare dalla VLAN al tenant, quindi i suoi container. I container si parlano senza incapsulare il pacchetto e credono di essere sulla stessa LAN virtuale, non vedono hop tra di loro, è completamente trasparente.

Quando si sposta un container, ed appena manda un pacchetto, la leaf corrispondente manda un annuncio BGP, in modo che tutto il traffico arrivi correttamente al container, senza problemi.

Ogni tenant ha il suo VNI, VLAN ID e prefisso. Un indirizzo IP lo hanno solamente i container e le leaf, attraverso interfacce di loopback. Nei lab i container sono macchine a parte connesse alle macchine server, questi non hanno una logica applicativa relativa al server, poiché a livello pratico si comportano come uno switch, si toglie quanto più possible carico di lavoro ai server, per aumentarne le loro risorse computazionali. Bisogna inserire un IP alla leaf della macchina, per mantenere la raggiungibilità. Le loopback verranno annunciate su BGP, quindi le uniche rotte sulla fabric saranno verso questi prefissi di loopback, essendo delle barra 32 sono essenzialmente degli indirizzi.

Si potrebbero sovrapporre i prefissi dei tenant, e funzionerebbe comunque, poiché sono distinti dalla VLAN e VXLAN, ma si tengono divisi nei lab per facilitare la comprensione.

Per ogni leaf, allo startup si inserisce l'indirizzo di loopback e si crea forzatamente una rotta in tabella di routing con quella loopback. Questa rotta serve poiché verrà usata dopo. Verranno create le VTEP corrispondenti a tutti i possibili tenant, preconfigurando tutte le leaf con tutte le VTEP, in modo che ovunque si sposti il container la leaf è già pronta.

```
ip address add {indirizzo-lo}/32 dev lo:1
```

```
ip route add {indirizzo-lo}/32 dev lo:1

ip link add {vtep-vni-1} type vxlan id {vni-1} dev lo dstport 4789 local
↪  {indirizzo-lo} nolearning
...
ip link add {vtep-vni-n} type vxlan id {vni-n} dev lo dstport 4789 local
↪  {indirizzo-lo} nolearning

ip link add {nome-br} type bridge
ip link set {nome-br} addrgenmode none
```

Il bridge creato sarà unico tra le due interfacce in uscita e con tutte le VTEP in entrata. Si attaccano tutti questi al bridge, e si abilita la VLAN su questo bridge:

```
ip link set dev {vtep-vni-1} master {nome-br} addrgenmode none
ip link set {vtep-vni-1} type bridge_slave neigh_suppress on learning off
...
ip link set dev {vtep-vni-n} master {nome-br} addrgenmode none
ip link set {vtep-vni-n} type bridge_slave neigh_suppress on learning off
ip link set dev {interfaccia-verso-server} master {nome-br}


ip link set dev {nome-br} type bridge vlan_filtering 1
bridge vlan add vid {vlan-id-1} dev {vtep-vni-1} pvid untagged
...
bridge vlan add vid {vlan-id-n} dev {vtep-vni-n} pvid untagged
bridge vlan add vid {vlan-id-1} dev {interfaccia-verso-server}
...
bridge vlan add vid {vlan-id-n} dev {interfaccia-verso-server}
```

Aggiungere un nuovo tenant implicherebbe riconfigurare tutte le leaf, si potrebbero pre-configurare con un approccio telescopico, ma in ambiente reale si un kernel proprietario, degli stessi produttori di FRR, che permette di associare una VTEP ad un intervallo di VNI, per non dover riconfigurarlo all'aggiunta di una nuova VNI. Il link tra il bridge e le VTEP non sono etichettati, poiché nel mondo reale si ha un altro mapping tra VNI e VLAN, essendo molte di meno. Sono etichettati solamente i pacchetti in uscita verso il server.

Alla fine bisogna attivare le interfacce ed il bridge:

```
ip link set up dev {vtep-vni-1}
...
ip link set up dev {vtep-vni-n}
ip link set up dev {nome-br}

systemctl start frr
```

Nella configurazione BGP, si usa come router ID l'indirizzo della loopback associata alla leaf.

```
router bgp {ASN}
    timers bgp 3 9
    bgp router-id {indirizzo-lo}
    no bgp ebgp-requires-policy
    bgp bestpath as-path multipath-relax

    neighbor TOR peer-group
        neighbor TOR remote-as external
        neighbor TOR advertisement-interval 0
        neighbor TOR timers connect 10
        neighbor eth0 interface peer-group TOR
        neighbor eth1 interface peer-group TOR
```

La differenza è inserire un redistribute dentro BGP, è un errore gravissimo inserirlo dentro BGP. Si usa una route-map per matchare un'interfaccia, rendendo la configurazione BGP identica per tutte le leaf, e qualunque IP posto alla loopback questo li annuncia. Per questo server la rotta statica, inserita nel server.

```
address-family ipv4 unicast
    neighbor TOR activate
    redistribute connected route-map LOOPBACKS
    maximum-path 64
exit-address-family

address-family l2vpn evpn
    neighbor TOR activate
    advertise-all-vni
exit-address-family

route-map LOOPBACKS permit 10
    match interface lo
```

Sulla spine, si attiva l'AFI/SAFI corretto per avviare BGP. Sulla parte alta della fabric non è cambiato niente, solamente questa configurazione aggiuntiva:

```
address-family l2vpn even
    neighbor fabric activate
    neighbor TOR activate
exit-address-family
```

Si aggiunge alle spine, e non alle ToF.

Si crea nel server un bridge per aprire i pacchetti VLAN in modo intelligente.

```
ip link add {nome-br} type bridge

ip link set dev eth0 master {nome-br}
```

```
...
ip link set dev eth{n} master {nome-br}


ip link set dev {nome-br} type bridge vlan_filtering 1
bridge vlan add vid {vlan-id-1} dev eth1 pvid untagged
...
bridge vlan add vid {vlan-id-n} dev eth{n} pvid untagged
bridge vlan add vid {vlan-id-1} dev eth0
...
bridge vlan add vid {vlan-id-n} dev eth0

ip link set up dev {nome-br}
```

Il server è non etichettato verso i propri tenant, mentre è etichettato verso la leaf, poiché utilizza la VLAN

Per la comunicazione tra due container di uno stesso tenant, un pacchetto non etichettato passa al server. Questo lo etichetta con la VLAN corrispondete al tenant e lo passa al companion bridge della leaf. Questa lo apre, legge la VLAN ID e lo manda alla VTEP corrispondente. Questa legge dalla MAC-to-VTEP table dove mandarlo per raggiungere il MAC destinazione, da questo conosce l'indirizzo loopback destinazione. Crea il pacchetto VXLAN relativo a questa VTEP, ci inserisce il pacchetto di livello due ricevuto, senza il tag VLAN, verso l'indirizzo di loopback della leaf destinazione. Questo arriva con multipath, e lo apre, guarda alla VXLAN, ed in base alla VNI determina la VTEP a cui inviarlo. Lo manda alla VTEP, questa lo passa al companion bridge, che lo associa alla VLAN corrispondente. Da questo companion bridge viene inviato al server che legge il suo VLAN ID e lo invia al tenant associato ospitato.

Ci sono delle primitive kernel con cui un protocollo come eBPF può leggere le informazioni quando passano il kernel.

Avendo due possibilità, per ogni pochetto che si riceve si mandano due pacchetti dalla ToF.

9.7.3 Bonding

Il lab senza la parte di no bond non è stabile, poiché si utilizza il kernel Linux pubblico e quindi non funzionano correttamente pezzi di tecnologia necessari. Il link tra la leaf ed il server è unico, se si rompe il server si buttano già tutti i link di quella lef. Il problema si risolve utilizzano la tecnologia chiamata bonding, di livello due. Tecnologie di livello più alto non si possono usare. Questo aggrega più interfacce di rete per creare un'unica interfaccia virtuale. Si possono usare diverse policy, mantenendo una attiva ed una di backup, oppure mantenendo tutte le interfacce attive. Per questa modalità bisogna bilanciare il traffico, si può usare round robin, oppure usando lo xor per implementare una forma di hashing. Lo standard ethernet 802.3ad realizza un bilanciamento guardando anche all'utilizzo dei singoli link.

Si ha un problema, poiché il dual-attached è progettato per avere due link verso lo stesso switch, ma in un Fat Tree si hanno due link a due leaf diverse, da parte di un server. Questa tecnologia si chiama invece MLAG, *Multi-Chassis Link Aggregation*, l'idea di avere un bond con due interfacce

che punta a due switch diversi. Il kernel classico di Linux non supporta questa tecnologia. Questo permette di avere ridondanza, usando due switch su due porte diverse come se appartenessero ad un singolo switch logico. Ogni server dovrebbe essere legato ad entrambe le leaf.

Se un link cade per un server, bisognerebbe che la leaf corrispondente invii all'altra leaf i pacchetti destinati a questo server. Il problema è che leaf di un PoD hanno lo stesso indirizzo anycast, quindi i pacchetti possono arrivare in maniera completamente indifferente ad una leaf o all'altra.

Ci sono compagnie e società che valutano l'uso di una topologia diversa. Il Fat-Tree più comunemente utilizzato è a tre livelli, per data center più piccoli sono sufficienti topologie a due livelli.

Si potrebbero aggiungere più leaf, o rimuovere il ToF, semplicemente modificando i peering, si possono aggiungere altri server o container, ma bisogna modificare molte più configurazioni.

Gli step relativi al debugging è analogo per gli altri lab, bisogna verificare che le leaf sono raggiungibili tra di loro. Si verifica se i peering sono funzionanti, sia delle ToF che delle leaf, e si controllano gli as-path, si segue la propagazione dei prefissi, poiché deve essere presente su ogni apparato della fabric. Per le VTEP è più complesso, ed è importante inserire gli indirizzi giusti. Sulle VLAN si può inserire uno sniffer con wireshark, oppure si può usare `tcpdump`.