

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 L^AT_EX ed ultima versione del testo disponibile al link:

<https://github.com/00Darxk/Internet-and-Data-Centers/>

Indice

1	Introduzione: ISPs ed IXPs	1
1.1	PSN: Polo Strategico Nazionale	3
1.2	SPC: Sistema Pubblico di Connettività	3
2	Algoritmi di Instradamento	5
2.1	Algoritmi Distance Vector	7
2.2	Algoritmi Link State Packet	9
2.3	Protocolli di Routing	11
2.3.1	IGP	11
2.3.2	OPSF	11
2.3.3	Coesistenza di Protocolli Diversi	12
2.3.4	BGP	12
2.4	Spanning Tree Algorithm: STP	13
2.5	Software Defined Network	15
3	Kathará	19
3.1	Configurare un Lab	20
3.1.1	FRRouting	20
3.1.2	RIP	22
3.1.3	OSPF	22

1 Introduzione: ISPs ed IXPs

Gli *Internet Service Provider* ISP o *Autonomous System* sono gestori autonomi della rete che hanno una responsabilità su un certo territorio. Su queste reti indipendenti viaggiano i pacchetti, gli ISP sono infatti responsabili di inoltrare i pacchetti che passano al loro interno per raggiungere la destinazione. In totale sono presenti circa 170000 ISP al mondo, alcuni di questi sono pubblicizzate e pubbliche per offrire servizi ad utenti comuni, altri offrono servizi a grandi compagni o altri ISP. Questi ISP formano una gerarchia, non dichiarata, che si può inferire, non essendo questi ISP governati da una struttura o organizzazione sovrastante. Alcuni di questi ISP sono privati e non si mostrano, per cui è possibile solamente inferire la gerarchia considerando questi ISP privati. Ogni ISP si cura solamente dei suoi interessi specifici, attraverso i loro clienti, permettono di connettersi ad un prezzo economico con altri servizi o utenti nel loro territorio. Questi ISP pagano altri ISP più grandi per raggiungere obiettivi sempre più remoti, in questo modo si può determinare una gerarchia che parte dagli ISP più piccoli ai più grandi.

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 varie 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 pagano entrambi insieme il costo del PNI.

Generalmente si paga sul 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. Questo si chiama *Internet eXchange Points IXP*. Questi sono governati da associazioni che mettono a servizio dei propri consorziati questi punti di scambio in modo sicuro. Quando si offre un servizio 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, gratuiti, 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 dove l'energia costa poco o è facilmente procurabile, oppure in zone dove la temperatura è pressoché bassa, per diminuire il costo di raffreddamento. In questi data center si brucia una quantità molto elevata di energia. Per l'importanza di questi data center, sono costantemente sorvegliati e protetti. Ogni macchina ha una procedura di recupero

e sono pubbliche e quindi si assume che chi è in grado di avvicinarsi ad una macchina può effettuare queste procedure di recovery ed entrare in possesso della macchina. 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 ma viene procurato dal cliente. Altri centri invece hanno un solo cliente che 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 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.

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, risposta rapida e misurazione accurata.

I servizi offerti da piattaforme cloud sono 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*, il software che viene eseguito su queste macchine virtuali può essere comprato o affittato direttamente da queste piattaforme cloud. Altri servizi specializzati possono offrire calcolo massivi. 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 regioni, in questo caso 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 sparso non necessariamente su uno stesso data center, poiché per 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 facilmente. 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 per garantire servizi specificando un modello con garanzia di controllo sulla locazione delle macchine e la strada che percorrono i dati.

In Italia l'Agenzia per la Cybersicurezza Nazionale ha fondato un cloud marketplace per offrire alla pubblica amministrazione di acquistare risorse cloud da parte di provider qualificati. La CON-SIP 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 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.

1.1 PSN: Polo Strategico Nazionale

Questo progetto ha 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.

I public cloud esistenti che parteciperebbero a questo progetto sono 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 utilizzando 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 messi a disposizione da parte del PSN su una piattaforma per erogare elementi applicativi e middleware, astruendo 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 dei tunnel per inviare il flusso di dati all'infrastruttura del PSN.

I quattro data centers sono collegati su due regioni, tra queste regioni il traffico viene gestito tramite il protocollo *Multi Protocol Label Switching* MPLS, sul backbone IP di TIM.

Tra DC 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 nelle varie LAN.

1.2 SPC: Sistema Pubblico di Connettività

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.

Questa 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, il jitter è 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 determinate 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, 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, connessi da reti fisse.

Esistono due tipi principali di algoritmi di instradamento, *distance vector* e *state packet*. In TCP si utilizza una variante del *distance vector*. Queste sono 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 una tabella, ma con strutture ad albero specializzate per avere un accesso il più veloce possibile. Si vuole mantenere la maggior quantità possibile di computazione sull'hardware, inoltre la dimensione dei pacchetti è piccola, e dipende dalla dimensione dei pacchetti ethernet di 1500 byte, definita dal primo consorzio NIX. Inoltre le risorse computazionali dei router non sono necessariamente sufficienti per poter gestire la complessità della rete. Questi pacchetti sono piccoli, poiché essendo in competizione dovevano realizzare schede di rete più economiche dei loro competitori. Utilizzando meno memoria si hanno schede di rete più economiche, e questo rappresenta un errore da parte del consorzio NIX, anche se si è affermato come lo standard per le comunicazioni via filo, è necessario avere operazioni di inoltramento 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'altra risorsa è l'utilizzo delle linee, avendo una banda limitata, non può mandare 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 riscontrano malfunzionamenti, errori di configurazione da parte di amministrazioni di rete, etc. Nello 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.

Si è realizzato un protocollo inter-dominio che oscilla in continuazione, dato che il suo effetto non era completamente compreso. Per cui è possibile realizzare esperimenti dove la rete diventa instabile.

Il traffico 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.

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 topologia 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. 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'inoltrare ogni pacchetto a tutte le interfacce connesse al router.

Gli algoritmi dinamici possono essere successivamente divisi in routing isolato, dove ogni router decide senza comunicare con altri router; il routing centralizzato, dove un router centrale determina la scelta migliore, questa strategia è rimasta per molto tempo sottovalutata; il routing distribuito, questa è la strategia corrente della rete, dove ogni router informa i propri vicini le informazioni note. 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.

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.

Per comunicare sulla rete è richiesto un indirizzo di rete associato ad una scheda di rete, ed una netmask associata. Una netmask indica da un indirizzo qual è la parte di rete e quale di host. Per mandare il pacchetto apparentemente non serve, ma è necessaria per determinare la rete dentro cui la macchina è presente e può raggiungere direttamente.

Si controlla prima se la macchina destinazione è direttamente raggiungibile, inviando un pacchetto MAC di livello due, poiché non è necessario un pacchetto di livello 3.

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 gli 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, queste 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 consiste l'amministratore fornisce 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, oppure verso il next hop. 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 solo su un insieme di linee selezionato, scartando pacchetti troppo vecchi, scartando un pacchetto al suo secondo passaggio per un nodo.

Nel routing isolato ogni *intermediate system* 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* dai pacchetti in ingresso vengono determinate le macchine raggiungibili su quella linea, IEEE 802.1D a livello due, 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, accoppiandolo ad un algoritmo per il calcolo dello spanning tree. Può essere raffinato aggiungendo un campo che specifica il costo di un cammino, incrementandolo ad ogni hop. Si possono mantenere alternative ordinate. Quando la destinazione è ignota si effettua flooding. Per la sua necessità di avere una struttura ad albero non è presente in internet, dato che non è possibile tagliare 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, questo spesso non è realistico. È rimasta per molto tempo teorica, ma recentemente sono apparse esigenze di trasferire un grade volume di traffico tra due macchine. Quindi un'autorità centrale tramite dei protocolli, *Software Defined Networking* (SDN), 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 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 queste tabelle dei propri vicini ricalcolano la tabella di instradamento integrando queste informazioni.

Le destinazioni vengono apprese con un meccanismo di passa-parola. 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 nel control plane ed inviate ai suoi vicini. Questi effettuano lo stesso inviando i propri indirizzi direttamente connessi.

Ogni destinazione nella tabella è corredata dal costo del cammino, dalla destinazione all'is stesso. Questo vengono calcolati dal distance vector ottenuto dai vicini, sommando i costo della linea di ingresso da cui è stato ricevuto.

Molte aziende e produttori cercando di vendere di più aggiungono caratteristiche e modificano questi protocolli. Il protocollo di distance vector originario appartiene a Cisco.

Per passare l'informazione bisogna inviare i vari distance vector, 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, fino a decine di secondi, un tempo estremamente lungo per ingegneria delle reti. Non si può intasare la rete inviando distance vector molto grandi, con 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. I costi delle linee vengono sommati, e vengono inseriti nella tabella di instradamento dell'is le connessioni alle reti di costo minore, per ogni LAN determinata.

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 vengono sfasati di una certa quantità. Sono presenti inoltre meccanismi per garantire che questo processo non sia sincro.

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, questa propaga il suo distance vector e l'informazione viene propagata sull'intera in un tempo relativamente breve.

Se questa connessione viene tagliata, il vicino non può aggiornare la sua tabella di instradamento. Queste macchina avrebbero raggiunto la rete ora disconnessa passando per macchine altre macchine adiacenti. Poiché non possono eliminare questa entry nella tabella di instradamento, ad ogni nuovo distance vector la distanza alla rete disconnessa incrementa continuamente, tendendo all'infinito. Si crea una specie di eco.

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 più uno.

Se un componente ha inviato la rotta al vicino, e questo gliela rimanda, non dovrebbe considerarla, questo fenomeno si chiama *split horizon*, ma non sempre è efficace, in reti circolari, l'eco propaga in un'unica direzione perpetuamente provocando il fenomeno del *count to infinity*.

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, ed al massimo prende tutti i router presenti. Non tutte le reti ammettono un cammino che preda tutte le reti.

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

Se 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. Negli distanze condivise tra i router ci sono due elementi, le buone notizie e le cattive notizie. 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 brutta 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 interno al router. Ogni tabella ha al più $O(n)$ righe, in ciascuno dei passi il router spende tempo $O(mn)$, 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 mappa si calcola l'instradamento ottimale per ogni rotta. Usano un algoritmo di calcolo di costo minimo, in questo caso usa Dijkstra. Il router compone da queste informazioni una tabella di instradamento per il valore del next hop per ogni rotta. La mappa della rete viene costituita usando questi pacchetti speciali, qui sorge un problema, poiché i link non sono componenti attivi, al massimo un router può descrivere le macchine presenti intorno all'IS. Questi pacchetti vengono mandati ai vicini e si suppone che i vicini li propaghino in selective flooding da ogni IS ad ogni altro IS della rete.

Se viene modificata la topologia, aggiorna la sua mappa e la propaga nella rete. Ogni IS conserva in un database il lsp più recente, per aggiornare la topologia della rete, questi link hanno dei costi e per questo si utilizza un algoritmo come Dijkstra invece di un BFS. Questo database è importato sia identico tra tutti gli IS, le informazioni devono essere propagate, tenendo solo l'ultima copia del lsp. Se la topologia non cambia non è necessario inviare periodicamente l'intero database. Si tratta di un protocollo molto più silenzioso, 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.

Si considera un insieme $V = \{1, \dots, n\}$ 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 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 finché $V \setminus S$ non è vuoto scegliendo un nodo in questo insieme tale che $d[i]$ sia minimo, aggiungendolo ad S , e per ogni nodo j adiacente ad i , si pone $d[j] = \min(d[j], d[i] + a_{ij})$.

$$S = \{1\}$$

$$\forall i \in \{2, \dots, n\} \rightarrow d[i] = a_{1i}$$

se $V \setminus S \neq \emptyset$, $i \in V \setminus S$ 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.

$d[k] = m(k)$, ovvero il cammino è ottimo poiché è calcolato da $d[j]$, per ipotesi $d[j] = m(j)$ ed l'arco (j, k) è utilizzato da p .

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. 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 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.

Poiché i link sono inerti, non è possibile comunicare direttamente con questi. Esiste un protocollo di *neighbor greetings*. Questo si utilizza per mantenere aggiornate la adiacenze, inviato periodicamente su tutte le interfacce. Quando un IS rileva una modifica della topologia invia un lsp per propagare la modifica.

Ogni pacchetto lsp ha un numero di versione relativo al router, il pacchetto non cambia 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 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 al mittente, per allinearli con i 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. 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 termine 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. I protocolli usati per trasferire informazioni di routing tra diversi domini amministrativi passano per BGP, il protocollo standard esterno.

2.3.1 IGP

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 permette pari a 15, sono reti piccole. Non vengono considerate altre metriche, tuttavia è compatibile 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.

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.

2.3.2 OSPF

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 molto. È organizzato gerarchicamente, dividendo la rete in aree connesse dell'intera rete, 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 contenuti da più di un'area, per fare da ponte tra queste aree. I router che si trovano sulla backbone hanno una visione di tutto il resto del mondo, 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 interfaccia 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, ed inietta sulla sua rete locale queste rotte prendendosi la responsabilità di inoltrarli eventuali messaggi alle altre 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 esser eletto e gestito da questi. 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 fa specifico ad uno specifico protocollo di rete.

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

Queste macchine devono permettere a più protocolli di condividere. 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. Le applicazioni devono scegliere quale pila da usare.

Gli IS sono in grado di instradare pacchetti relativi ad entrambi i protocolli di rete.

La filosofia è "navi nella nebbia", i diversi protocolli convivono in modo totalmente isolato. Per questi motivi 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 distance vector sono invisibili l'uno dall'altro. Tutte queste tabelle di instradamento vengono a determinare un mix di tutte le tabelle di instradamento. Ma questi protocolli utilizzano metriche tra di loro non confrontabili, tipicamente l'amministratore di rete decide in che modo gestire questi 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 ricevute come rotte statiche. Iniettare dentro un altro è un'operazione delicata, se si inserisce un protocollo che contiene tutte l'internet, la sa solo dall'iniezione, ma pubblicizza di poter raggiungere tutto l'internet creando un buco nero, se viene diffusa fuori.

Due ISP che comunicano usando due protocolli di routing differenti, utilizzano dei router appartenenti ad entrambi i loro domini. Questi possono comunicare attraverso router contenenti rotte iniettate da entrambi i protocolli. Questo può pubblicizzare le rotte apprese da entrambi i protocolli. Il fatto che non ci sia una chiara divisione delle competenze su questo router è un problema di natura gestionale. Se la rotta attraversa diversi domini, questo processo di iniettare rotte ignorando le metriche, è un processo non ottimale se attraversa molti domini.

2.3.4 BGP

Per evitare di avere questi problemi di carattere amministrativo, invece di co-gestire una macchina e separare le competenze, ci si inventa una struttura dove tra le due reti si condivide un filo e la

cogestione di un filo è più semplice.

Il protocollo EGP è adottato tra domini amministrativi diversi. Su queste connessioni viene adottato un solo protocollo. Può prediligere le rotte che attraversano meno domini amministrativi. 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 a motivi economici come accordi tra ISP. 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 passare attraverso router altri router di frontiera. Queste informazioni sono condivise agli altri router di frontiera dello stesso dominio amministrativo tramite connessioni TCP.

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. spf

Usa lo strato di trasporto connesso per scambiarsi informazioni tra strati BGP, sulla porta 179. Tipicamente prova a connettersi in maniera incrociata, e si prova a fare 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 conoscere. Richiede una conoscenza molto profonda della rete, le configurazioni su questo protocollo è molto costoso.

2.4 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 che sono state rimosse tutti i superflui.

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

Lo standard chiama LAN il dominio dove vengono collegati gli switch, un insieme di domini di collisione. Si assume che ognuna 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, senza chiudere nessuna LAN. Ogni bridge ha un identificatore, gli spareggi si fanno sull'id più basso. Anche le porte hanno un identificatore, analogamente più basso meglio è. Hanno un valore di costo per ogni LAN, definito dalla velocità della LAN. Per cui lo standard viene aggiornato per aggiornare questa tabella dei costi quando vengono rilasciate nuove tecnologie da 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 viene 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 questo costo, in ingresso.

Questo algoritmo è diviso in fasi, che possono essere realizzate in un singolo pacchetto. Nella prima fase viene eletto un bridge da radice, si identifica per ogni bridge una porta 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 conatterà 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.

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 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 si crede 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.

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.

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. 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.

2.5 Software Defined Network

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 una 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 inoltra 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, queste 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 in 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 che chiamano southbound diretto verso gli elementi della rete, un'altra comunicazione chiamata northbound è orientata 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 seguente entry 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 al 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 una o più tabelle 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 indirizzi o dati contenuti nel pacchetto. L'azione da

svolgere 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 distribuite con due tempi di timeout, *idle* e *hard*. Se c'è un flusso che non si presenta per un certo periodo questo viene cancellato in tempo *idle*. Mentre l'*hard* timeout è indipendente, impone di scartare il flusso dopo che è passato.

Il controller è un'entità software che deve conoscere l'intera rete in modo dinamico, producendo flow entries ed inviandole alle macchine. 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 essere inoltrato per intero al controller, che può guardarlo completamente, oppure più semplicemente prende l'header, 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, usato dal controller per istruire allo switch l'invio di un singolo pacchetto. Non necessariamente sono inviati in risposta ad un *PacketIn*. Quando sono in risposta indicano ad uno switch il flusso da usare, altrimenti l'azione è specificata dall'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.

Il messaggio *FlowMod* invia una flow entry, può essere in risposta ad un datapath. Può avere una priorità, quindi le tabelle di flussi potrebbero essere ordinate per priorità creando una gerarchia, per ordinare meglio il traffico. Per un certo periodo c'è stata una grande ricerca per inserire le flow entry e creare una tabella più piccola con meno flow entry per avere più prestazioni.

Se la regola è già presente, allora la nuova regola sovrascrive la vecchia. Con questo pacchetto si possono modificare, rimuovere flow entries, oppure notificare al controller l'istante di eventuale rimozione, o può richiedere una verifica.

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, questo protocollo già esistente, mantiene questi pacchetti all'interno della rete. Questi sono spediti periodicamente per esplorare la rete. Il controller fa 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, tutti questi pacchetti ricevuti da uno switch devono essere rimandati dentro ad un *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 specifica porta. Uno switch diverso che riceve il pacchetto LLDP, inviato da uno altro switch su una certa porta, lo spedisce al controller in un pacchetto *PacketIn*. Questi pacchetti non possono passare attraverso eventuali 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 identificare la topologia. Nei pacchetti *PacketIn* viene specificata la porta di ricezione.

In base a quanto detto finora ogni rete ha un solo controller, ma ci sono soluzioni distribuiti con controller organizzati in maniera gerarchica, per gestire il traffico in maniera distribuita, dove ogni controller controlla un numero limitato di switch, a cui delega di distribuire regole ad altri switch.

Anche negli switch ci sono limitazioni, non hanno un buffer illimitato, né tabelle abbastanza grandi. È comunque presente una rete sottostante con i loro protocolli di instradamento, su queste è possibile costruire una rete SDN.

Questo garantisce grande libertà per la gestione di una rete.

3 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.

3.1 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 fr, 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.

3.1.1 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, BGP, 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'}`. 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 `vysh` per gestire allo stesso modo questi protocolli. Per abilitarlo, o disabilitarlo, si inserisce nel file `vysh.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 in 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 `vysh`, 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 ripetute, 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 dei vari protocolli di rete. Su `vysh`, 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 dividerlo 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: `vttysh -e "{comando}"` Si può reindirizzare con `> {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.

3.1.2 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 suoi distance vector.

3.1.3 OSPF

OSPF è un protocollo di routing link state packet, che può essere gestito da FRR. Per funzionare invia in multicast pacchetti contenenti 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 `vttysh` 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 transit, questi sono connessi 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 sparteggiare si usa il costo dell'OSPF.