Università Politecnica delle Marche

FACOLTÀ DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Biomedica

Uso del software Huawei eNSP per la simulazione di un'architettura di rete *enterprise*

Use of the Huawei eNSP software to simulate an enterprise network architecture

Relatore:
Prof. Ennio Gambi

Candidato:
Matteo Paolini

Correlatore:
Prof. Adelmo De Santis

Anno Accademico 2019–2020

Alla cara memoria di Guerrino Paolini
1939–2013
e Graziella Riccitelli
1939–2018

Ringraziamenti

Ringrazio Francesco per avermi aiutato nei momenti più difficili della mia carriera universitaria, e per avermi spronato a dare il massimo quando sentivo di non farcela, e Simone, con cui ho condiviso diverse ore di esercizi sui *router*.

Ringrazio il Prof. De Santis, il cui senso dell'umorismo ha ispirato i titoli di alcune sezioni di questo elaborato, e il Prof. Gambi, che ha offerto a me e ad altri studenti l'opportunità di frequentare il corso di *networking* da cui ha origine questa tesi.

Ringrazio infine Huawei Technologies Co., Ltd. per aver organizzato il corso e per aver fornito il materiale didattico per lo studio e le esercitazioni.

Indice

Elenco delle figure

Elenco delle tabelle

Glossario

BitTorrent protocollo di comunicazione *peer-to-peer* per la condivisione di file.

Bolla delle *dot-com* bolla speculativa causata da un eccessivo investimento in società basate su Internet durante la fine degli anni 1990 e l'inizio degli anni 2000.

Checksum valore utilizzato per verificare l'integrità di un dato.

Client calcolatore che usufruisce dei servizi offerti dal *server*.

Convergenza stato in cui tutti i *router* appartenenti ad una rete possono comunicare con successo.

Cracker criminale informatico che si introduce all'interno di un calcolatore o di una rete di calcolatori per scopi illegali.

Downtime tempo in cui un sistema non è in funzione.

End-to-end principio secondo il quale è più facile ottenere informazioni sull'affidabilità di una connessione attraverso processi operanti ai capi piuttosto che in nodi intermedi di questa.

Ethernet tecnologia che permette la connessione tra dispositivi all'interno di una rete locale.

Full duplex tipo di comunicazione tra due dispositivi simultanea e bidirezionale su mezzi trasmissivi indipendenti che non formano un dominio di collisione.

Gateway *router* che instrada pacchetti Internet verso reti diverse dalla rete locale.

Half duplex tipo di comunicazione tra due dispositivi su un mezzo trasmissivo condiviso che determina l'esistenza di un dominio di collisione.

Host dispositivo connesso ad una rete che scambia dati all'interno di questa.

Hub strumento capace di connettere fisicamente più dispositivi Ethernet.

Load balancing pratica che permette di distribuire il carico di lavoro equamente tra più componenti analoghe di un sistema..

NAT box *router* su cui è implementata NAT.

Overhead dati aggiuntivi non presenti nel messaggio originale ma necessari per la corretta comunicazione tra dispositivi.

Payload dato da trasmettere all'interno di una rete.

Peer-to-peer architettura di rete in cui gli *host* scambiano dati tra loro senza comunicare con un *server* centrale.

Porta strumento che permette al *transport layer* di inoltrare un segmento o datagramma al processo corretto.

Rete commutata rete composta solamente da *host* e *switch*.

Router dispositivo di livello 3 che separa i domini di *broadcast* all'interno di una rete.

Routing hole sottospazio degli indirizzi non raggiungibile a causa di instradamenti scorretti da parte di un *router* dovuti ad una divisione in sottoreti poco attenta.

Script kiddie termine dispregiativo rivolto a individui inesperti che utilizzano codice malevolo scritto da altri senza conoscerne il reale funzionamento.

Server calcolatore che fornisce servizi ad altri dispositivi detti *client*.

Social engineering pratica usata per scopi fraudolenti che utilizza tecniche ingannevoli per indurre un utente a divulgare informazioni riservate.

***Stack* protocollare** Implementazione ordinata dei protocolli che rendono possibile la comunicazione tra calcolatori.

Switch dispositivo di livello 2 (*data link*) che separa i domini di collisione all'interno di una rete.

Telnet protocollo di rete che permette di interagire con un altro dispositivo attraverso una interfaccia a riga di comando.

Elenco degli acronimi

ACL Access Control List.

ARP Address Resolution Protocol.

ARPANET Advanced Research Projects Agency NETwork.

CIDR Classless Inter-Domain Routing.

CSMA/CD Carrier Sense Multiple Access with Collision Detection.

DARPA Defense Advanced Research Projects Agency.

FIB Forwarding Information Base.

FLSM Fixed Length Subnet Masking.

FTP File Transfer Protocol.

GRE Generic Routing Encapsulation.

IANA Internet Assigned Numbers Authority.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IP Internet Protocol.

IPX Internetwork Packet Exchange.

ISO International Organization for Standardization.

LAN Local Area Network.

MAC Media Access Control.

NAPT Network Address Port Translation.

NAT Network Address Translation.

OSI Open Systems Interconnection.

OSIE Open Systems Interconnection Environment.

OSPF Open Shortest Path First.

OUI Organization Unique Identifier.

PDU Protocol Data Unit.

PVID Port VLAN ID.

RFC Request for Comments.

RIP Routing Information Protocol.

SSH Secure Shell.

STP Spanning Tree Protocol.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

VLAN Virtual Local Area Network.

VLSM Variable Length Subnet Masking.

VoIP Voice over IP.

VPN Virtual Private Network.

Capitolo 1

It's a kind of magic

L'epidemia di COVID-19 del 2020 ha reso chiaro come Internet sia diventato uno strumento necessario alla vita di tutti i giorni. Ad esempio, grazie ad applicazioni Voice over IP (VoIP) gli studenti hanno potuto continuare a frequentare le lezioni, molti lavoratori hanno sperimentato per la prima volta lo *smart working* e in generale le persone sono rimaste in contatto durante il periodo di isolamento.

Una ricerca su Google Trends del termine «Cloud» mostra come, da gennaio 2004 a settembre 2020, l'interesse per il *cloud computing* sia cresciuto nel tempo. Occorre ricordare che queste tecnologie sono supportate da una complessa rete di dispositivi tra cui *router*, *switch* e *server* che comunicano tra loro tramite protocolli standardizzati.

Lo scopo di questa tesi è simulare, su piccola scala, una rete *enterprise* attraverso il software eNSP di Huawei. Poiché questo programma emula i dispositivi di questo produttore, le considerazioni fatte sulle tecnologie utilizzate si baseranno sull'implementazione effettuata da Huawei.

1.1 Requisiti

Per simulare il funzionamento della topologia di rete mostrata in figura 1.1 è necessario soddisfare i seguenti requisiti:

- AR5 deve essere il *gateway* predefinito e NAT per la VLAN 50;
- AR6 deve essere il *gateway* predefinito e NAT per la VLAN 60;
- *client* 1 deve poter accedere al *server* 1 ma non al 2;
- *client* 2 deve poter accedere al *server* 2 ma non al 1;
- *server* 1 e 2 devono poter comunicare direttamente attraverso un tunnel GRE tra AR4 ed AR1.

Sebbene sia buona pratica progettare una rete cercando di prevedere anche espansioni future, per semplicità considereremo questa topologia come immutabile e compiremo le scelte progettuali di conseguenza.

1.2 Reti *enterprise*

Partiamo dunque col definire il concetto di rete *enterprise*. Huawei ne fornisce una definizione interessante:

> The enterprise network originally represents the *interconnection of systems belonging to a given functional group or organization* to primarily enable the sharing of resources such as printers and file servers, communication support through means such as email, and the evolution towards applications that enable collaboration between users [12].

In passato una rete era confinata in luoghi geograficamente ristretti come ad esempio l'edificio di un'azienda, per cui la creazione, lo sviluppo e il *troubleshooting* di questa avvenivano principalmente in un ambiente controllato e potevano essere garantite sicurezza e riservatezza. Con lo sviluppo dell'Internet pubblico il concetto di rete *enterprise* si è espanso oltre i confini dell'edificio e ora può comprendere un insieme di reti in luoghi geograficamente dispersi facenti parte di una singola organizzazione: pensiamo alle filiali che fanno capo a una sede principale, oppure a impiegati in regime di *smart working* da casa o che operano all'interno delle sedi dei clienti.

Quest'evoluzione delle reti *enterprise* pone sfide impegnative per quanto riguarda l'affidabilità della connessione (la qualità potrebbe variare drasticamente da regione a regione), la sicurezza e la riservatezza dei dati (essendo instradate all'interno dell'Internet globale, le informazioni potrebbero attraversare delle reti compromesse ed essere catturate da agenti indesiderati). Nemmeno una rete all'interno di un singolo edificio è tuttavia al sicuro: oltre a doversi difendere da attacchi al suo interno (ad esempio dovuti al *social engineering* contro gli impiegati [6]), la rete, affacciandosi a Internet, è in balia di *crackers* finanziati da stati avversari e *script kiddies* che tentano di fare breccia al suo interno.[1] Per risolvere queste problematiche sono state sviluppate tecnologie come le Virtual Private Network (VPN) per garantire connessioni criptate e IEEE 802.1X per l'autenticazione degli utenti all'interno di una rete aziendale [10].

Oltre alle necessità di sicurezza e riservatezza, una rete *enteprise* deve offrire ridondanza, la quale impedisce che un guasto a un dispositivo connesso possa compromettere il funzionamento dell'intera rete, e scalabilità, per cui è possibile far crescere la rete in base alle esigenze dell'impresa e far sì che funzioni adeguatamente. La topologia in figura 1.2 rappresenta, in modo estremamente semplificato, una tipica rete aziendale.

[1] Clamoroso è stato il caso del sedicenne che si è intrufolato nella rete di Apple e si è impossessato di oltre 90 GB di informazioni riservate [18].

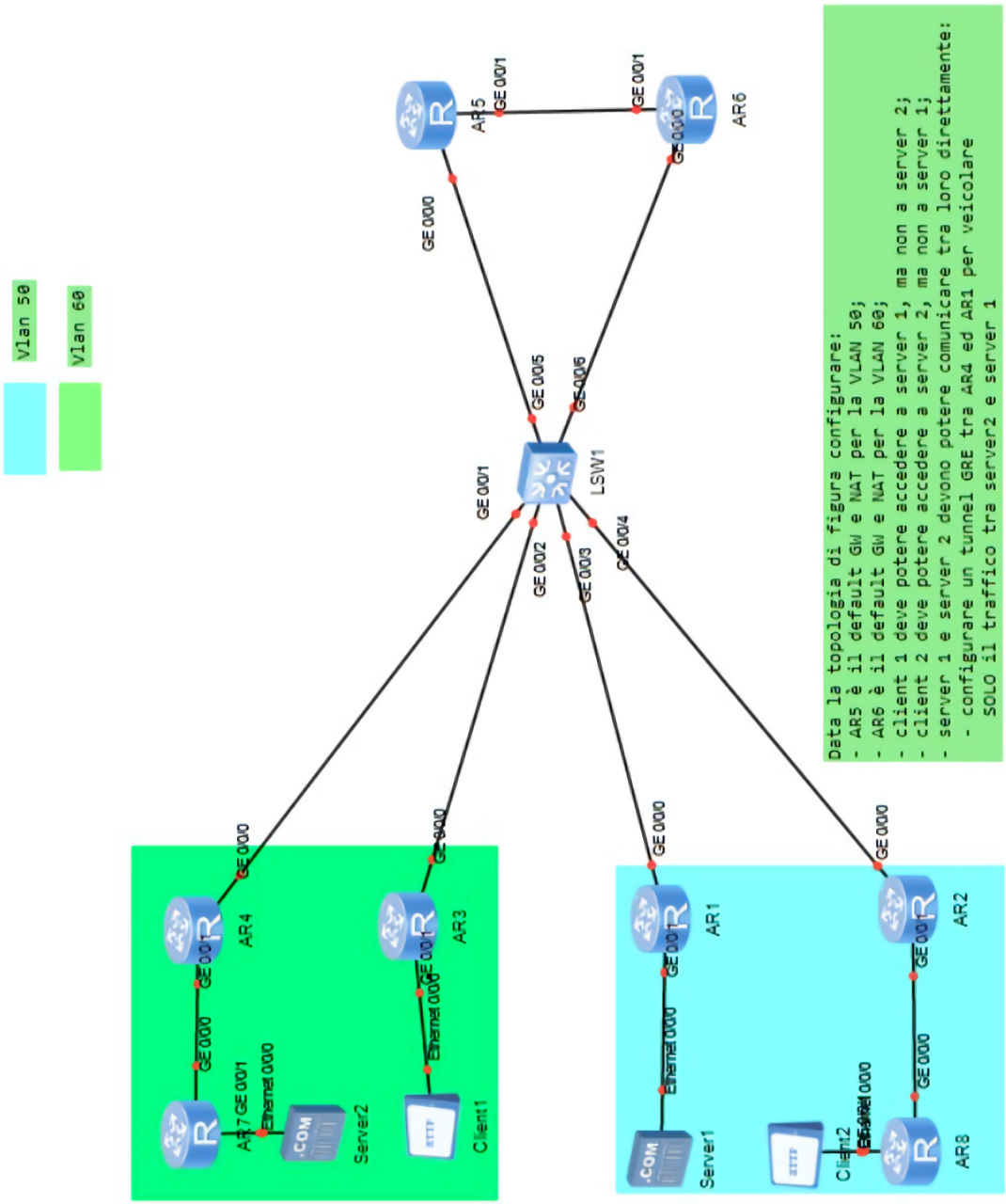


Figura 1.1: Topologia di partenza.

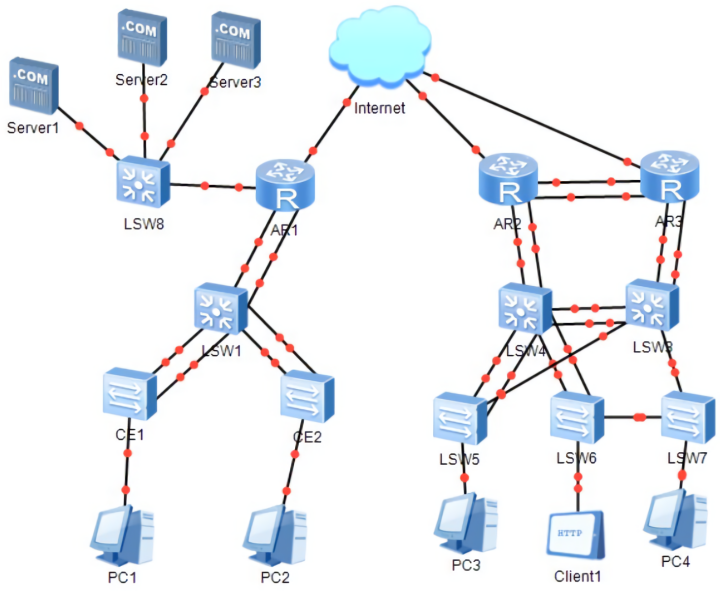


Figura 1.2: Topologia semplificata di una rete aziendale. L'immagine mette in evidenza il complesso schema di connessioni che garantisce ridondanza e affidabilità.

In base alle dimensioni dell'azienda, possiamo distinguere due tipologie di rete [12]:

Small enterprise in genere limitata geograficamente, ha un design piatto e ha poca scalabilità per via del ridotto numero di utenti che vi è connesso. Le risorse limitate dell'azienda fanno sì che non ci sia ridondanza.

Large enterprise copre sedi distribuite all'interno di un paese o in tutto il mondo, per cui sono necessarie grande scalabilità e affidabilità attraverso un design gerarchico, a più strati e modulare tale da minimizzare il *downtime* e permettere all'azienda di espandere la rete con minimo sforzo; fondamentale è anche il controllo dell'accesso alla rete da parte degli utenti.

Le grandi imprese sono quelle più affette dalle limitazioni dovute alle tecnologie di rete usate correntemente, poiché queste si basano su modelli e protocolli pensati per connettere un esiguo numero di utenti in luoghi circoscritti, come ad esempio le università, ma che oggi hanno il compito di supportare comunicazioni a livello globale.

Negli anni sono state sviluppate molte soluzioni per la comunicazione tra calcolatori, ma oggi quelle più utilizzate sono il modello TCP/IP e il modello ISO/OSI. A prescindere dal modello utilizzato, un calcolatore deve

essere fisicamente connesso agli altri affinché possa scambiare dati attraverso segnali elettrici od ottici; per queste connessioni gli standard più utilizzati sono gli IEEE 802, tra cui lo standard Ethernet 802.3.

1.3 ARPANET si evolve: il modello TCP/IP

Lo standard TCP/IP è stato ideato negli anni 1970 da Vint Cerf e Bob Kahn [23], scienziati della Defense Advanced Research Projects Agency (DARPA) responsabili dello sviluppo di Advanced Research Projects Agency NETwork (ARPANET). Kahn, lavorando a una rete satellitare per la trasmissione di pacchetti, definì 4 principi per quello che diventerà poi lo standard Transmission Control Protocol (TCP):

- ogni rete può connettersi a un'altra attraverso un *gateway*;
- non esiste un'amministrazione centralizzata;
- i pacchetti persi vengono ritrasmessi;
- una rete non deve cambiare la sua struttura interna per connettersi a un'altra rete.

Cerf e Kahn, durante lo sviluppo del protocollo, mutuarono da CYCLADES, una rete di ricerca francese, alcune funzionalità, tra cui quella di rendere gli *host* responsabili per l'affidabilità della trasmissione. La prima versione di TCP risale al 1973, ma venne documentata nel RFC 675 [4] del 1974. La versione di TCP/IP più usata oggi è la 4, descritta nel RFC 1122 [1], per cui lo *stack* protocollare è diviso in 4 livelli:

Application layer supporta protocolli che permettono agli utenti e al sistema di interagire con la rete come Telnet, Secure Shell (SSH) e File Transfer Protocol (FTP).

Transport layer offre servizi di comunicazione *end-to-end* alle applicazioni, principalmente attraverso i protocolli TCP e User Datagram Protocol (UDP).

Internet layer permette di scambiare dati tra *host* utilizzando il protocollo Internet Protocol (IP), il quale tuttavia non controlla l'integrità dei datagrammi, lasciando questa responsabilità ai livelli superiori.

Link layer permette ai dispositivi connessi a una rete di interfacciarsi con la stessa.

Sebbene TCP/IPv4 sia ancora ampiamente utilizzato, l'esplosione dell'Internet globale ha messo in luce le sue limitazioni, le quali vengono superate dal protocollo IPv6, presentato nel 1998 [7] ma ancora limitatamente adottato.

In quegli anni pionieristici anche IBM e varie compagnie telefoniche europee stavano sviluppando le loro reti proprietarie. La crescente frammentazione mise in luce la necessità di sviluppare uno standard internazionale, e dopo vari fallimenti sembrò profilarsi una soluzione che prenderà il nome di Open Systems Interconnection (OSI).

1.4 Scontro fra titani: il modello ISO/OSI

Il modello ISO/OSI, proposto per la prima volta da Charles Bachman [14] e sviluppato da un comitato internazionale, ha il pregio di dividere tutte le azioni necessarie alla comunicazione tra calcolatori in 7 livelli distinti.[2] Un'altra caratteristica importante è quella di considerare i calcolatori connessi come pari, al contrario di architetture di rete del passato come quella di IBM, che proponeva una gerarchia basata sul terminale e sul calcolatore [20].

A differenza del modello TCP/IP, il modello ISO/OSI permette di distinguere chiaramente il comportamento dei livelli inferiori dello *stack* e aggiunge altri livelli al di sotto dell'*application layer*. Lo *stack* protocollare completo è il seguente [13]:

Application layer è il livello più alto e permette alle applicazioni di interagire con l'Open Systems Interconnection Environment (OSIE); in particolare contiene tutte le funzioni per le operazioni di comunicazione che non sono state già eseguite dai livelli sottostanti. Queste funzioni possono essere svolte da programmi oppure da esseri umani.

Presentation layer individua e negozia una sintassi di trasferimento e traduce dalla sintassi comprensibile all'*application layer* a quella di trasferimento e viceversa (un esempio è dato dalla compressione/decompressione di dati).

Session layer provvede alla sincronizzazione della comunicazione tra dispositivi e gestisce lo scambio dati affinché avvenga in maniera ordinata, assegnando ad ogni comunicazione un indirizzo di sessione.

Transport layer si occupa dell'efficacia e dell'economicità della comunicazione tra dispositivi offrendo le prestazioni necessarie ai livelli superiori al minimo costo, tenendo sempre conto della mole delle loro richieste e della qualità e affidabilità del servizio offerto dal livello inferiore; i protocolli a questo livello operano solamente alle estremità della rete.

Network layer il suo compito è instradare dati e connettere tra loro dispositivi ubicati in luoghi geograficamente distanti, identificati da un

[2] Questo modello è ritenuto antiquato da Tom Nolle, il quale propone come alternativa un modello a 3 livelli [16].

indirizzo di rete, e negozia la qualità del servizio con il *Transport layer*, la quale sarà la stessa a entrambi i capi della connessione.

Data link layer organizza la trasmissione dei dati sul mezzo trasmissivo identificando ogni dispositivo connesso alla rete locale attraverso un indirizzo Media Access Control (MAC) e può correggere eventuali errori di trasmissione avvenuti nel *Physical layer*.

Physical layer permette la trasmissione del flusso di bit tra dispositivi connessi attraverso vari mezzi trasmissivi; può inoltre stabilire se una connessione può avvenire in modalità *half duplex* o *full duplex*.

Occorre tenere presente che ciascun livello non ha consapevolezza delle operazioni effettuate dai livelli adiacenti, poiché ognuno elabora le istruzioni di sua competenza e vede il resto del contenuto come dati generici. Questa peculiarità è dovuta al fatto che, attraversando lo *stack* protocollare, i dati subiscono un processo di *incapsulamento* in invio, oppure di *decapsulamento* in ricezione.

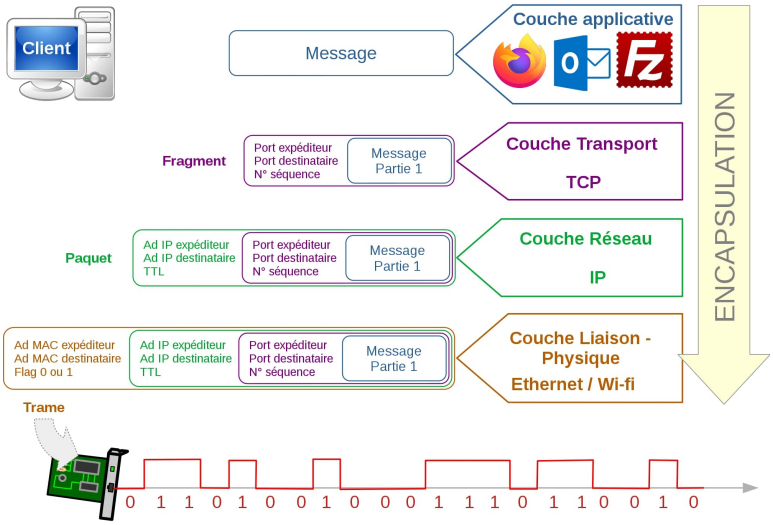
A cavallo tra gli anni 1980 e 1990 sembrava che OSI, forte dell'appoggio entusiasta di governi e aziende di tutto il mondo, si sarebbe imposto come standard per le comunicazioni tra calcolatori. Tuttavia ogni sostenitore tentava di portare avanti i propri interessi, costringendo i vari membri del comitato a scendere a scomodi compromessi che aggiunsero complessità al modello originale e rallentarono lo sviluppo dello standard [20]. Nel frattempo TCP/IP stava guadagnando popolarità grazie al fatto di essere semplice e gratuito (l'esatto contrario di OSI) e di essere guidato dalla *leadership* mirata di DARPA e della Defense Communications Agency. Nella metà degli anni 1990 OSI era considerato irrilevante e il protocollo di Cerf e Kahn la faceva da padrone.

1.5 Incapsulamento e decapsulamento

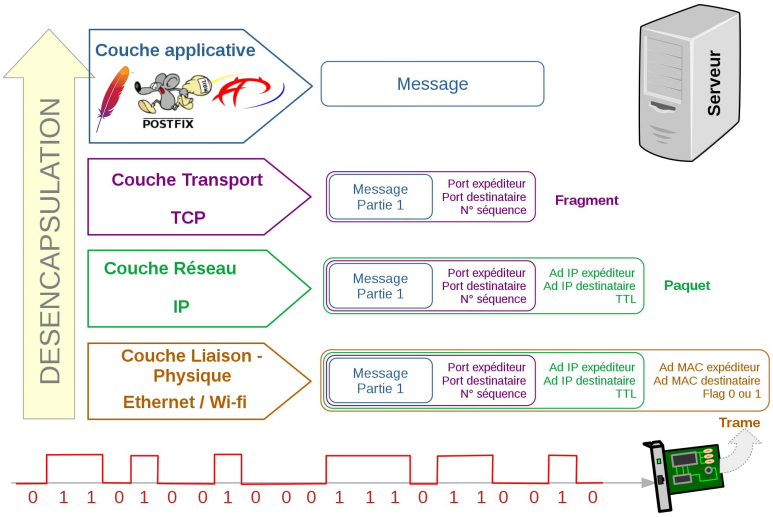
Con *incapsulamento* si intende quel processo per cui al dato originale, che attraversa i livelli del modello di riferimento, vengono posposti e preposti altri dati (detti *header* e *trailer*) necessari per la corretta trasmissione dell'informazione [12]; il *decapsulamento* è invece il processo opposto. Durante questi processi il dato viene generalmente chiamato Protocol Data Unit (PDU), ma siccome ciascun livello compie operazioni distinte, questo viene chiamato con nomi differenti a seconda del *layer* in cui viene elaborato.

L'incapsulamento avviene in questo modo [11]:

1. un calcolatore chiede di inviare dati attraverso la rete;
2. il dato passa dall'*application layer* al *transport layer* attraverso una porta TCP, per cui il dato è chiamato *segmento*, o UDP, per cui il dato viene chiamato *datagramma*;



(a) Incapsulamento. © Jean-Michel HAROUY. Schema_encapsulation. CC BY-SA 4.0. Convertita in formato JPEG.



(b) Decapsulamento. © Jean-Michel HAROUY. Schema_desencapsulation. CC BY-SA 4.0. Convertita in formato JPEG.

Figura 1.3: Operazioni di incapsulamento e decapsulamento.

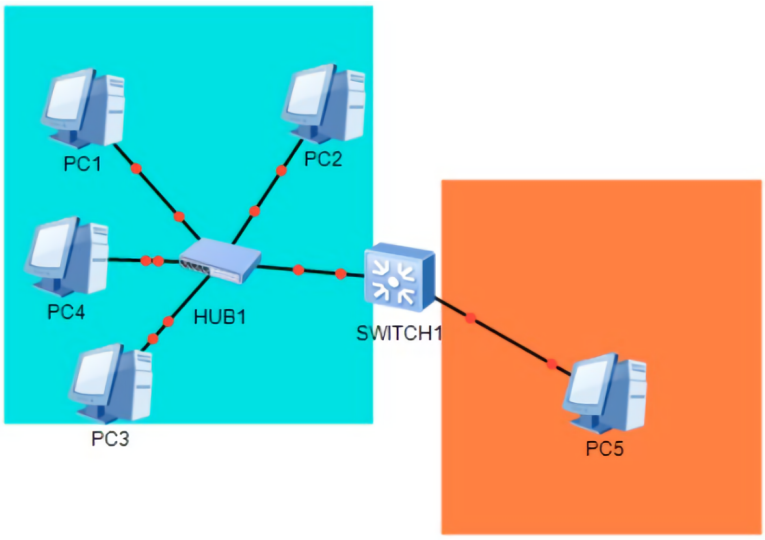


Figura 1.4: Domini di collisione.

3. il segmento o datagramma passa all'*Internet layer* dove gli vengono aggiunti i dati necessari affinché possa raggiungere calcolatori oltre la rete locale, divenendo un *pacchetto*;

4. il pacchetto viene inoltrato al *link layer* dove ottiene le informazioni necessarie per essere trasmesso con successo in un segmento di rete, divenendo un *frame*.

Durante il decapsulamento il calcolatore destinatario elimina passo dopo passo *header* e *trailer* e dopo queste operazioni, se non sono presenti errori, il dato può finalmente essere letto.

Notiamo nella figura 1.3a l'effetto collaterale dell'incapsulamento: il *payload* è più grande rispetto al messaggio originale a causa dell'*overhead* che si aggiunge ad ogni livello.

1.6 Come si suddivide una rete locale

Si definisce *dominio di collisione* l'insieme di dispositivi connessi a un unico canale trasmissivo [12]. La figura 1.4 mostra alcuni *host* collegati a un *hub*, creando così un unico dominio di collisione (indicato dall'area azzurra). Notiamo anche che un dominio di collisione è spezzato da uno *switch*, per cui PC5 si trova in un dominio a sé (area arancione).

Una collisione avviene quando più dispositivi inviano, sullo stesso mezzo trasmissivo, un segnale contemporaneamente, per cui si viene a creare un conflitto elettrico che lo distorce. Per risolvere questo problema si utilizza la tecnologia Carrier Sense Multiple Access with Collision Detection

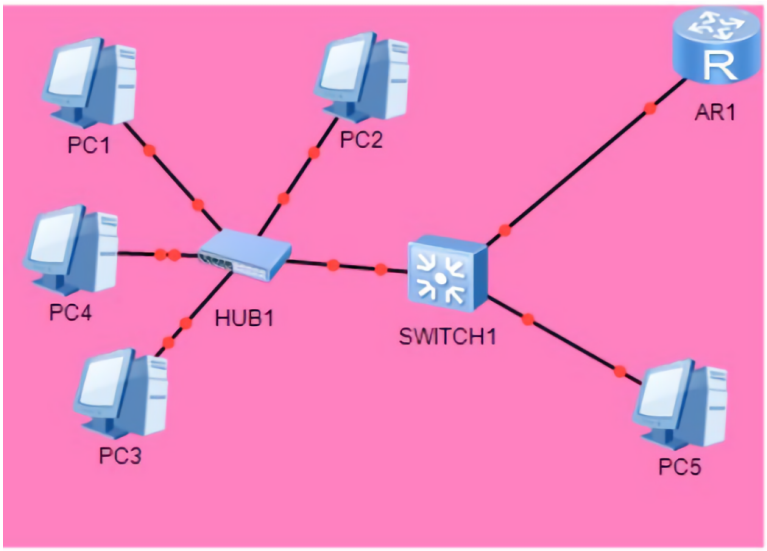


Figura 1.5: Dominio di *broadcast*.

(CSMA/CD), la quale permette ai calcolatori di organizzare la trasmissione di segnali e di riportare ordine in caso di collisioni. Occorre notare che questa problematica è presente sono nelle connessioni *half duplex* e non nelle *full duplex* [12].

Come visto nel paragrafo 1.3, uno dei dispositivi più importanti all'interno di una rete è il *gateway*. Ecco dunque che possiamo definire un *dominio di* broadcast come l'insieme di tutti i dispositivi che possono ricevere un *frame broadcast* (vedremo questo elemento più in dettaglio nel paragrafo 2.1) e delimitato da un *gateway* (come vediamo nella figura 1.5, in cui il dominio di *broadcast* è delimitato dall'area rosa), che ha il compito di instradare dati oltre i confini della rete locale comunicando con altri *router* attraverso protocolli IPv4 o IPv6 [12].

Capitolo 2

I livelli *data link* e *network*, *routing* e VLAN

Nel capitolo 1 abbiamo presentato alcuni concetti teorici generali per capire il funzionamento di una rete di calcolatori. Ora è giunto il momento di analizzare in dettaglio il funzionamento dei livelli *data link* e *network* del modello TCP/IP. In seguito connetteremo gli spazi degli indirizzi avvalendoci della capacità dei *router* di instradare pacchetti tra diverse Local Area Networks (LANs). Vedremo infine come unire logicamente queste sottoreti in VLANs.

2.1 Comunicazione a livello *data link*

Affinché i dati possano essere trasmessi sul mezzo è necessario che questi siano all'interno di un *frame* Ethernet, il quale contiene le informazioni necessarie, tra cui gli indirizzi MAC del mittente e del destinatario, per la comunicazione tra due *host* all'interno di una rete locale o LAN [12].

A livello *data link* due *host* possono comunicare attraverso l'indirizzo MAC, un codice unico e immutabile che identifica una specifica interfaccia di rete di un dispositivo.[1] Ogni indirizzo MAC è formato da 48 bit divisi in due campi da 24 bit ciascuno: il primo campo è detto Organization Unique Identifier (OUI) e identifica il produttore della scheda di rete, mentre il secondo campo contiene un valore incrementale assegnato ad ogni prodotto [12].

La trasmissione di *frame* può avvenire in tre modi differenti [3]:

Unicast un *host* invia all'interno della LAN un *frame* che viene ricevuto da tutti gli altri *host*, ma che viene ulteriormente elaborato solo dall'effettivo destinatario e scartato dagli altri; Castelli lo paragona a

[1] Non totalmente esatto poiché esiste la tecnica del MAC *spoofing* che consente di modificare via *software* l'indirizzo MAC della propria interfaccia di rete, permettendo di compiere anche atti di dubbia legalità…

una riunione post-lavoro in cui ogni impiegato è diretto verso un'unica destinazione, ma per arrivarci usa la propria auto.

Broadcast un *host* invia un *frame* a tutti i dispositivi presenti nella LAN e tutti lo elaborano; questo tipo di *frame* viene utilizzato dall'Address Resolution Protocol (ARP), ad esempio quando un nuovo dispositivo entra a far parte della rete locale e vuole annunciare questo agli altri dispositivi.

Multicast è una forma più intelligente ed economica di comunicazione tra più *host* allo stesso tempo, per cui un unico *frame* è inviato a uno *switch* che lo divide e lo inoltra agli effettivi destinatari; Castelli lo paragona al *carpooling*.

È facile distinguere un *frame broadcast* dagli altri poiché questo specifica come destinatario l'indirizzo FF:FF:FF:FF:FF:FF, mentre è possibile distinguere gli tipi di *frame* osservando l'ottavo bit del primo byte dell'indirizzo MAC: se questo è uguale a 0 la trasmissione è di tipo *unicast*, viceversa se questo è uguale a 1 la trasmissione è di tipo *multicast*.

A prescindere da queste modalità, ogni *host* invia un *frame* ad ogni altro dispositivo connesso alla LAN, creando una notevole quantità di traffico.

2.2 Oltre i confini della LAN: indirizzamento IP

Così come per inviare una lettera a una persona dobbiamo conoscere dove essa si trovi (e viceversa), due calcolatori che vogliano comunicare devono conoscere reciprocamente i loro indirizzi IP. Un *host* infatti, prima di inoltrare il pacchetto al livello *data link*, confronta il proprio indirizzo IP con quello del destinatario: se questo appartiene alla stessa rete del mittente, il pacchetto può essere incapsulato in un *frame* ed essere inviato sul mezzo trasmissivo, altrimenti il primo viene inviato a un dispositivo (detto *gateway* predefinito) che si suppone sappia dove si trovi il destinatario, oppure il pacchetto viene scartato prima ancora di raggiungere il livello inferiore.

Per semplicità, d'ora in avanti si illustrerà la teoria alla base dell'instradamento di pacchetti usando il protocollo IPv4, per cui gli indirizzi IP verranno espressi in *dotted decimal notation* ovvero una stringa di quattro valori decimali separata da punti fermi.

Un indirizzo IP è formato da 32 bit suddivisi in 4 byte distribuibili tra il campo *network* (il «quartiere» dove si trova il calcolatore) e il campo *host* (la «casa» dove vive il calcolatore). Ogni byte può assumere un valore decimale da 0 a 255 ma, per il campo *host*, solo i valori da 1 a 254 sono validi poiché gli altri hanno un significato speciale: se tutti i bit assumono il valore 0, allora abbiamo un indirizzo che identifica una rete, mentre se tutti i bit assumono il valore 1 (cioè il byte assume il valore decimale 255) l'indirizzo è detto

Tabella 2.1: Classi di reti e i relativi intervalli di indirizzi, numero di reti e *host* per rete disponibili [12].

Classe	Intervallo	Reti	*Host*
A	1.0.0.0-126.255.255.255	126	16 777 214
B	128.0.0.0-191.255.255.255	16 384	65 534
C	192.0.0.0-223.255.255.255	2 097 152	254
D	224.0.0.0-239.255.255.255	-	-
E	240.0.0.0-255.255.255.255	-	-

di *broadcast*, per cui un pacchetto che lo specifica come destinazione viene inviato a tutti i dispositivi all'interno della rete locale [12]. La lunghezza del campo *network* e quella del campo *host* determinano la *classe* di una rete, identificata da una delle cinque lettere dell'alfabeto latino tra A ed E: ad esempio, la classe A riserva 1 byte al campo *network* e 3 byte al campo *host*, la classe B riserva 2 byte al campo *network* e 2 byte al campo *host*, infine la classe C riserva 3 byte al campo *network* e 1 byte al campo *host*; gli indirizzi di classe D sono riservati per le comunicazioni *multicast*, mentre quelli di classe E sono sperimentali.[2]

La tabella 2.1 mostra a quale classe appartengono gli intervalli di indirizzi. Osservandola attentamente si può notare che vi è una lacuna tra l'intervallo della classe A e quello della classe B; questo è dovuto al fatto che gli indirizzi da 127.0.0.0 a 127.255.255.255 sono riservati per operazioni diagnostiche. All'interno di questi intervalli vengono riservati degli indirizzi per uso privato (tabella 2.2 a pagina seguente), poiché non è possibile assegnare ad ogni dispositivo connesso a una rete un indirizzo IP pubblico a causa della mole di dispositivi dotati di connettività Internet in circolazione e del limitato intervallo totale di indirizzi specificato da IPv4. Questi indirizzi privati sono assegnabili liberamente all'interno di una LAN e sono unici solamente all'interno di questa, al contrario di quelli pubblici che devono essere assegnati da enti come la Internet Assigned Numbers Authority (IANA) [19]. Esistono infine altri due indirizzi speciali, 0.0.0.0 che identifica «qualsiasi rete» e 255.255.255.255 che ha la funzione di indirizzo di *broadcast* per qualsiasi rete IPv4.

Tenendo a mente i valori nella quarta colonna della tabella 2.1, mettiamo in luce alcuni dei limiti più grandi delle reti *classful*: dovendo creare una LAN con 260 *host* si sprecherebbero molti indirizzi preziosi (più di 60 000!), di contro non sarebbe nemmeno auspicabile sfruttare al massimo una rete di classe B poiché il traffico generato rischierebbe di sopraffare i dispositivi di rete (si dice comunemente che «Ethernet non scala»).

[2]Nel 2008 alcuni ingegneri Cisco avevano proposto di rendere disponibili al pubblico gli indirizzi di questa classe, tuttavia i problemi tecnici che sarebbero sorti e l'esiguo guadagno di nuovi indirizzi hanno stroncato quest'idea sul nascere [22].

Tabella 2.2: Intervalli di indirizzi privati appartenenti ad ogni classe e numero di reti disponibili [19].

Classe	Intervallo	Reti
A	10.0.0.0-10.255.255.255	1
B	172.16.0.0-172.31.255.255	16
C	192.168.0.0-192.168.255.255	255

Per rimediare a questi inconvenienti portati da questo approccio, detto *classful*, si può ricorrere alla creazione di *subnet*, ovvero reti più piccole ricavate da una rete di una certa classe [12]: A partire da una rete non partizionata, si può privare il campo *host* di alcuni bit, i quali vengono allocati in un campo *subnet* [17]. Notiamo che una *subnet* è una rete a tutti gli effetti, per cui possiede anch'essa un indirizzo *network* e un indirizzo *broadcast* che tolgono due indirizzi assegnabili agli *host*. Con questo approccio, detto *classless*, non possiamo però distinguere facilmente in quale rete si trovi un particolare *host*, per cui dobbiamo introdurre il concetto di *subnet mask*, una stringa di valori analoga a un indirizzo IP, ma i cui valori dei bit assumono un significato particolare: se tutti i bit di un ottetto hanno valore 1, allora l'ottetto dell'indirizzo IP appartiene al campo *network*, se i bit di un ottetto assumono tutti il valore 0, l'ottetto dell'indirizzo IP appartiene al campo *host*, mentre se solo alcuni bit dell'ottetto assumono il valore 1, quell'ottetto dell'indirizzo IP appartiene al campo *subnet*. Ad ogni classe di indirizzi IP è associata una *subnet mask* che specifica la dimensione della rete; in questo caso gli ottetti, in base decimale, possono assumere solo i valori 255 e 0. La tabella 2.3 mostra le maschere associate a ciascuna classe.

L'approccio classico al *subnetting*, detto Fixed Length Subnet Masking (FLSM), prevede che una rete debba essere partizionata in parti uguali, per cui il problema dello spreco di indirizzi rimane: se avessimo a disposizione una rete di classe B che vogliamo partizionare in tante reti di classe C, sprecheremmo molti indirizzi nelle aree a cui sono connessi pochi utenti o nei collegamenti punto-punto tra *router*. Per risolvere questa problematica è stata introdotta la tecnica del Variable Length Subnet Masking (VLSM) che permette di dimensionare una sottorete in base alle reali esigenze degli utenti permettendo di manipolare a piacere la *subnet mask* [12]. Grazie a questa tecnica possiamo ad esempio assegnare a un collegamento punto-punto tra *router* due indirizzi appartenenti a una rete con maschera 255.255.255.252, la quale supporta solamente due *host*. [3]

Sebbene poter sfruttare al massimo uno spazio degli indirizzi sia molto vantaggioso, bisogna avere cura di non sovraccaricare i *router* presenti in

[3] Ricordiamo che il numero di *host* che può far parte di una rete è dato dall'espressione $2^h - 2$, dove h indica il numero di bit appartenenti al campo *host* di un indirizzo IP, poiché gli indirizzi *network* e *broadcast* non possono essere assegnati.

Tabella 2.3: Classi di indirizzi e relative *subnet mask* [12].

Classe	*Subnet mask*
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

una rete: questi, come vedremo più avanti, compiono le decisioni di instradamento attraverso un *database* detto *tabella di* routing, che può raggiungere dimensioni notevoli, affaticando così il dispositivo. La convenzione Classless Inter-Domain Routing (CIDR) permette di aggregare tante reti sotto un unico prefisso ed introduce una notazione molto comoda detta *prefix length* ed espressa come `/<valore>`, dove il valore indica la lunghezza in bit del prefisso. Ad esempio possiamo indicare la rete 192.168.16.0 con maschera 255.255.255.0 come 192.168.16.0/24 e un insieme di reti con indirizzi da 192.168.16.0/24 a 192.168.31.0/24 possono essere aggregate come 192.168.16/20.

2.3 Assegnazione degli indirizzi

Con i concetti illustrati fino ad ora è possibile iniziare a lavorare sulla topologia. Prendiamo come punto di partenza la rete privata di classe A 10.0.0.0 che possiamo dividere in *subnet* più piccole alla bisogna. Definiamo questa convenzione:

- il valore del secondo byte dell'indirizzo indica la VLAN a cui appartiene il dispositivo;
- i valori del terzo e del quarto byte sono arbitrari.

Osserviamo che i collegamenti dei dispositivi nelle aree colorate sono punto-punto, per cui sarebbe opportuno assegnare indirizzi appartenenti a spazi /30, ma per evitare *routing holes* lavoreremo solo con spazi degli indirizzi /24.

Dopo aver assegnato gli indirizzi IP alle rispettive interfacce (tabella 2.4), ci accertiamo che le connessioni funzionino a dovere effettuando un test di raggiungibilità con il comando `ping`: AR1, AR2 e AR5 possono comunicare, così come AR3, AR4 e AR6. Notiamo tuttavia alcuni fenomeni interessanti ma prevedibili: sebbene AR1, AR2, AR3, AR4, AR5, AR6 si trovino nello stesso dominio di *broadcast*, essi non possono comunicare fra loro direttamente poiché si trovano in spazi degli indirizzi differenti; notiamo inoltre che i *client* e i *server* possono comunicare con i *router* direttamente adiacenti, ma non con gli altri dispositivi che appartengono alla stessa VLAN poiché non abbiamo ancora configurato l'instradamento dei pacchetti all'interno della rete e gli *host* non hanno un *gateway* predefinito.

Tabella 2.4: Indirizzi IP assegnati alle interfacce dei dispositivi emulati.

Dispositivo	Interfaccia	Indirizzo
server 1	ethernet 0/0/0	10.50.1.1/24
AR1	gigabit ethernet 0/0/1	10.50.1.2/24
	gigabit ethernet 0/0/0	10.50.2.1/24
AR5	gigabit ethernet 0/0/0	10.50.2.2/24
	gigabit ethernet 0/0/1	10.56.1.1/24
AR2	gigabit ethernet 0/0/0	10.50.2.3/24
	gigabit ethernet 0/0/1	10.50.3.1/24
AR8	gigabit ethernet 0/0/0	10.50.3.2/24
	gigabit ethernet 0/0/1	10.50.4.2/24
client 2	ethernet 0/0/0	10.50.4.1/24
client 1	ethernet 0/0/0	10.60.1.1/24
AR3	gigabit ethernet 0/0/1	10.60.1.2/24
	gigabit ethernet 0/0/0	10.60.2.1/24
AR6	gigabit ethernet 0/0/0	10.60.2.2/24
	gigabit ethernet 0/0/1	10.56.1.2/24
AR4	gigabit ethernet 0/0/0	10.60.2.3/24
	gigabit ethernet 0/0/1	10.60.3.1/24
AR7	gigabit ethernet 0/0/0	10.60.3.2/24
	gigabit ethernet 0/0/1	10.60.4.2/24
server 2	ethernet 0/0/0	10.60.4.1/24

2.4 I principi del *routing*

Come visto nel paragrafo 1.6, una LAN è delimitata da un *gateway* che ha il compito di instradare pacchetti destinati a *host* che si trovano in reti diverse da quella locale. Egli infatti analizza il *frame* in arrivo e, se questo è *broadcast* o indirizzato al *gateway* stesso, lo trasferisce a livello 3 (*network*), estrae l'indirizzo IP di destinazione e lo confronta con gli spazi degli indirizzi da lui conosciuti: se il *gateway* conosce la destinazione, allora il pacchetto viene instradato di conseguenza, altrimenti questo è inviato a un altro *router*, che si spera conosca cosa fare, oppure viene semplicemente scartato [12]. Le decisioni di instradamento vengono prese confrontando l'indirizzo di destinazione con le rotte presenti nella sua Forwarding Information Base (FIB), le quali *entry* contengono gli spazi degli indirizzi ricavati dalla tabella di *routing* e le interfacce attraverso cui inoltrare i dati, attraverso il principio del *longest match*, cioè compiendo una operazione logica AND tra gli indirizzi e selezionando la rotta per cui si ha il maggior numero di risultati positivi [12]. Una volta scelta la rotta adeguata, il pacchetto viene inoltrato al *next hop*, ovvero il dispositivo successivo che si interpone tra l'origine e la destinazione del pacchetto.

Una tabella di *routing* può essere popolata attraverso l'uso di vari protocolli di *routing* a cui è associato un valore di preferenza: se più rotte conducono alla stessa destinazione, viene scelta quella con valore minore. Ad esempio, se una stessa destinazione può essere raggiunta da due rotte differenti, una definita dal protocollo Open Shortest Path First (OSPF), l'altra definita dal Routing Information Protocol (RIP), verrà scelta la prima poiché a OSPF è associato un valore di preferenza 10, mentre a RIP è associato un valore 100 [12]. Spesso alcune rotte possono avere un valore di preferenza uguale, dunque per prendere una decisione si utilizza il valore di costo, basato sulla lunghezza e sulla larghezza di banda della rotta. Analogamente alla preferenza, il costo più basso determina la rotta su cui instradare un pacchetto; se i costi fossero equivalenti, alle rotte verrebbe assegnato un valore bilanciato in modo da poter effettuare *load balancing* [12].

Osservando una tabella di *routing*, notiamo che le varie rotte si possono dividere in tre categorie in base a come il *router* ne sia venuto a conoscenza:

Dirette il *router* e il dispositivo di destinazione possono comunicare senza l'aiuto di un intermediario [2].

Statiche la via da seguire è definita manualmente dall'amministratore.

Dinamiche i *router* condividono indipendentemente le loro informazioni di instradamento tramite protocolli come OSPF o RIP.

Notiamo subito che nella maggior parte delle situazioni è preferibile definire rotte dinamiche poiché queste sono più resilienti nei confronti dei

```
[ar5]disp ip routing-table
Route Flags: R - relay, D - download to fib
------------------------------------------------------------------------------
Routing Tables: Public
         Destinations : 8        Routes : 8

Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

      10.50.1.0/24  Static  60   0           RD   10.50.2.1       GigabitEthernet
0/0/0
      10.50.2.0/24  Direct  0    0            D   10.50.2.2       GigabitEthernet
0/0/0
      10.50.2.2/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
    10.50.2.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
      127.0.0.0/8   Direct  0    0            D   127.0.0.1       InLoopBack0
      127.0.0.1/32  Direct  0    0            D   127.0.0.1       InLoopBack0
127.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
255.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
```

Figura 2.1: Tabella di *routing* di AR5 dopo aver impostato la rotta statica verso lo spazio degli indirizzi 10.50.1.0/24.

cambi improvvisi di topologia (si dice che la rete arriva più in fretta alla convergenza).

2.5 Comunicazione all'interno della rete locale

Scegliamo di mettere in comunicazione gli spazi degli indirizzi all'interno delle rispettive LAN configurando sui *router* le appropriate rotte statiche poiché la relativa semplicità della topologia non richiede l'utilizzo di protocolli di *routing* più complessi come OSPF o RIP. Ad esempio su AR5 vanno configurati come *next hop* gli indirizzi 10.50.2.1 e 10.50.2.3 in modo da poter raggiungere rispettivamente gli spazi 10.50.1.0/24 e 10.50.3.0/24.

Analizziamo il dettaglio la configurazione della VLAN 50, ponendo l'attenzione sulla figura 2.1: configurando semplicemente la rotta verso lo spazio 10.50.1.0/24, AR5 può comunicare con tutti gli *host* al suo interno, ma questi ultimi non possono fare altrimenti poiché non saprebbero a chi rispondere (vedi la figura 2.2, in cui si osserva che la *Echo request* fatta da `ping` non ha ricevuto risposta). In questo caso possiamo impostare su AR1 una *default static route*, una sorta di ultima spiaggia, per cui tutti i pacchetti la cui destinazione sia ignota vengono inoltrati a un *next hop* specifico, in questo caso l'interfaccia gigabit ethernet 0/0/0 di AR5. Osservando la figura 2.3, notiamo che nella tabella di *routing* di AR1 compare l'indirizzo 0.0.0.0 (il cui significato è spiegato nel paragrafo 2.2) come destinazione e 10.50.2.2 come *next hop*; quella particolare riga della tabella può essere tradotta in: «instrada ogni pacchetto la cui destinazione sia sconosciuta all'indirizzo 10.50.2.2». Infine impostiamo come *gateway* predefinito di *server* 1 l'indirizzo 10.50.1.2 per permettere al primo di comunicare al di fuori del suo spazio degli indirizzi.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
2	2.000000	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
3	4.000000	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
4	6.031000	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
5	8.047000	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request

Figura 2.2: Cattura dei dati eseguita sull'interfaccia gigabit ethernet 0/0/1 di AR1.

```
Routing Tables: Public
         Destinations : 11       Routes : 11

Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

        0.0.0.0/0   Static  60   0           RD   10.50.2.2       GigabitEthernet
0/0/0
      10.50.1.0/24  Direct  0    0           D    10.50.1.2       GigabitEthernet
0/0/1
      10.50.1.2/32  Direct  0    0           D    127.0.0.1       GigabitEthernet
0/0/1
    10.50.1.255/32  Direct  0    0           D    127.0.0.1       GigabitEthernet
0/0/1
      10.50.2.0/24  Direct  0    0           D    10.50.2.1       GigabitEthernet
0/0/0
      10.50.2.1/32  Direct  0    0           D    127.0.0.1       GigabitEthernet
0/0/0
    10.50.2.255/32  Direct  0    0           D    127.0.0.1       GigabitEthernet
0/0/0
      127.0.0.0/8   Direct  0    0           D    127.0.0.1       InLoopBack0
      127.0.0.1/32  Direct  0    0           D    127.0.0.1       InLoopBack0
127.255.255.255/32  Direct  0    0           D    127.0.0.1       InLoopBack0
255.255.255.255/32  Direct  0    0           D    127.0.0.1       InLoopBack0
```

Figura 2.3: tabella di *routing* di AR1 dopo aver impostato la *default route* verso 10.50.2.2.

Ora, come si può vedere dalla figura 2.4, *server* 1 e AR5 comunicano con successo poiché il primo è in grado di rispondere alla *Echo request* del secondo con una *Echo reply*.

Volgendo la nostra attenzione alla porzione di rete più lunga, notiamo che tra AR5 e *client* 2 ora si interpongono 2 *router*. Impostiamo dunque la rotta statica che permette ad AR5 di raggiungere lo spazio 10.50.3.0/24 attraverso il *next hop* 10.50.2.3. Anche in questo caso vediamo che AR5 riesce a comunicare con AR2 all'indirizzo 10.50.3.1, ma non con AR8 poiché quest'ultimo non saprebbe a chi rispondere. Configuriamo allora le *default routes* su AR8 e AR2 (attenzione al fatto che i due *router* avranno *next hop* differenti, come si può vedere dalle figure 2.5 e 2.6)

Terminiamo l'opera aggiungendo su AR5 e AR2 una rotta statica che permette loro di raggiungere lo spazio 10.50.4.0/24 e impostiamo 10.50.4.2 come *gateway* predefinito di *client* 2. Come si può vedere dalla figura 2.7, siamo riusciti a creare con successo una linea di comunicazione tra *server* 1

No.	Time	Source	Destination	Protocol	Length	Info
42	4578.0…	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
43	4578.0…	10.50.1.1	10.50.2.2	ICMP	98	Echo (ping) reply
44	4578.5…	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
45	4578.5…	10.50.1.1	10.50.2.2	ICMP	98	Echo (ping) reply
46	4579.0…	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
47	4579.0…	10.50.1.1	10.50.2.2	ICMP	98	Echo (ping) reply
48	4579.4…	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
49	4579.4…	10.50.1.1	10.50.2.2	ICMP	98	Echo (ping) reply
50	4579.9…	10.50.2.2	10.50.1.1	ICMP	98	Echo (ping) request
51	4579.9…	10.50.1.1	10.50.2.2	ICMP	98	Echo (ping) reply

Figura 2.4: Una seconda cattura dei dati eseguita sull'interfaccia gigabit ethernet 0/0/1 di AR1.

e *client* 2.

Siccome la VLAN 60 ha una struttura speculare alla 50, utilizzeremo gli stessi procedimenti visti in precedenza per mettere in comunicazione i dispositivi all'interno della prima.

2.6 Creare reti locali in maniera logica

Protocolli di rete come Spanning Tree Protocol (STP) o ARP sono piuttosto «chiacchieroni», cioè inviano periodicamente dei dati all'interno di un dominio di *broadcast* per saggiare le condizioni della rete. Se un dominio contiene molti dispositivi, questo tipo di traffico congestiona in maniera significativa la rete, degradandone le prestazioni. Per dividere un dominio di *broadcast* è necessario un *router*, ma abbiamo visto nel paragrafo 2.4 che le operazioni compiute da questo dispositivo sono piuttosto dispendiose, il che porterebbe a un ulteriore calo di prestazioni. Può inoltre essere necessario, per questioni organizzative o limiti di spazio, raggruppare dei calcolatori in maniera più flessibile rispetto alle divisioni nette offerte dai concetti di dominio di collisione e dominio di *broadcast*.

Per risolvere questi problemi è stato introdotto il concetto di VLAN, descritto per la prima volta nel RFC 3069 [15], per cui *host* appartenenti alla stessa rete commutata possono risiedere in reti logiche differenti. In questo modo si possono dividere i domini di *broadcast* utilizzando dispositivi veloci come gli *switch*.

La divisione in VLAN viene effettuata etichettando i *frame* in transito secondo lo standard IEEE 802.1Q, per cui un'etichetta contiene un particolare VLAN ID da 12 bit che indica allo *switch* su quali VLAN può transitare il dato. Questi frame possono transitare in collegamenti *access*, che connettono tra loro *host* e *switch*, oppure *trunk*, che connettono tra loro gli *switch* [12]. Questi collegamenti fanno capo alle porte dello *switch*, le quali possiedono

```
<ar2>dis ip routing-table
Route Flags: R - relay, D - download to fib
------------------------------------------------------------------------------
Routing Tables: Public
         Destinations : 11       Routes : 11

Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

        0.0.0.0/0   Static  60   0           RD   10.50.2.2       GigabitEthernet
0/0/0
      10.50.2.0/24  Direct  0    0            D   10.50.2.3       GigabitEthernet
0/0/0
      10.50.2.3/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
    10.50.2.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
      10.50.3.0/24  Direct  0    0            D   10.50.3.1       GigabitEthernet
0/0/1
      10.50.3.1/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/1
    10.50.3.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/1
      127.0.0.0/8   Direct  0    0            D   127.0.0.1       InLoopBack0
      127.0.0.1/32  Direct  0    0            D   127.0.0.1       InLoopBack0
127.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
255.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
```

Figura 2.5: Tabella di *routing* di AR2.

```
<ar8>dis ip routing-table
Route Flags: R - relay, D - download to fib
------------------------------------------------------------------------------
Routing Tables: Public
         Destinations : 11       Routes : 11

Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

        0.0.0.0/0   Static  60   0           RD   10.50.3.1       GigabitEthernet
0/0/0
      10.50.3.0/24  Direct  0    0            D   10.50.3.2       GigabitEthernet
0/0/0
      10.50.3.2/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
    10.50.3.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
      10.50.4.0/24  Direct  0    0            D   10.50.4.2       GigabitEthernet
0/0/1
      10.50.4.2/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/1
    10.50.4.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/1
      127.0.0.0/8   Direct  0    0            D   127.0.0.1       InLoopBack0
      127.0.0.1/32  Direct  0    0            D   127.0.0.1       InLoopBack0
127.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
255.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
```

Figura 2.6: Tabella di *routing* di AR8.

No.	Time	Source	Destination	Protocol	Length	Info
20	40.391…	10.50.4.1	10.50.1.1	ICMP	74	Echo (ping) request
21	40.391…	10.50.4.1	10.50.1.1	ICMP	74	Echo (ping) request
22	40.438…	10.50.1.1	10.50.4.1	ICMP	74	Echo (ping) reply
23	40.438…	10.50.1.1	10.50.4.1	ICMP	74	Echo (ping) reply
24	40.485…	10.50.4.1	10.50.1.1	ICMP	74	Echo (ping) request
25	40.500…	10.50.4.1	10.50.1.1	ICMP	74	Echo (ping) request
26	40.531…	10.50.1.1	10.50.4.1	ICMP	74	Echo (ping) reply
27	40.547…	10.50.1.1	10.50.4.1	ICMP	74	Echo (ping) reply

Figura 2.7: Cattura dei dati sull'interfaccia gigabit ethernet 0/0/0 di AR5 che mostra la comunicazione tra *server* 1 e *client* 2.

un Port VLAN ID (PVID) che determina il comportamento del dispositivo nei confronti dei *frame* in transito [12]. Una porta può essere di tre tipi:

Access associata a un collegamento *access*, riceve dagli *host frame* non etichettati e li etichetta in base al proprio PVID. Quando riceve un *frame* già etichettato, questa lo inoltra e gli toglie l'etichetta solo se il VLAN ID e il PVID corrispondono, altrimenti lo scarta.

Trunk associata a un collegamento *trunk*, decide se un *frame* possa viaggiare etichettato o non in base alla corrispondenza tra il il VLAN ID e il PVID associato a questa. Un *frame* il cui VLAN ID fosse diverso dal PVID viene fatto transitare solo se ammesso dalla porta.

Hybrid può assumere un comportamento *access* o *trunk* a seconda che venga impostata come *untagged* o *tagged*. Le porte *untagged* etichettano i *frame* in entrata e filtrano quelli in uscita, ma hanno anche la capacità di togliere l'etichetta a un *frame* in uscita per farlo transitare in una VLAN dove normalmente non potrebbe. Le porte *tagged* filtrano i *frame* da uno *switch* all'altro in base alle impostazioni della porta stessa, ma questi viaggiano sempre etichettati.

Un dispositivo può essere assegnato a una VLAN con i seguenti metodi:

1. questo è connesso a una particolare porta dello *switch*, e a un *frame* non etichettato viene assegnato un VLAN ID corrispondente al PVID della porta;

2. l'indirizzo MAC della interfaccia di rete del dispositivo viene associato a un certo VLAN ID, e un *frame* viene etichettato in base a questa corrispondenza;

3. il dispositivo fa parte di una particolare sottorete a cui corrisponde un VLAN ID, per cui l'etichettatura avviene in modo simile al secondo caso;

Tabella 2.5: VLANs presenti nella topologia e le relative interfacce gigabit ethernet dello *switch*.

VLAN	Interfacce
50	0/0/3, 0/0/4, 0/0/5
60	0/0/1, 0/0/2, 0/0/3

4. il *frame* in transito trasporta un certo protocollo associato a un VLAN ID, e l'etichettatura avviene in modo simile ai due casi precedenti.

Il primo metodo è quello più semplice ed è utilizzato di *default*, mentre i restanti sono più laboriosi perché l'amministratore di rete deve mappare gli elementi a mano, ma non costringono i dispositivi a restare connessi a una particolare porta. Siccome gli *switch* sono dispositivi di livello 2, i metodi 3 e 4 sono più dispendiosi dal punto di vista computazionale poiché il dispositivo deve decapsulare il *frame* e analizzare i relativi campi.

Dopo questa esposizione teorica sulle VLANs, andiamo a dividere logicamente la nostra topologia secondo i requisiti.

2.7 Dividiamo la rete in VLANs

In questo caso la creazione delle VLANs richiede che vengano configurate solo porte di tipo *access*, poiché nella rete è presente un solo *switch*. Per semplicità usiamo il metodo di etichettatura basato sulle porte. Creiamo le VLANs non contigue 50 e 60 usando l'opzione «batch», assegniamo il tipo di porta desiderato alle interfacce, e infine raggruppiamo queste alle giuste VLANs, come vediamo dalla tabella 2.5.

Abbiamo raggiunto un ottimo traguardo, cioè far comunicare con successo *client* e *server* all'interno delle rispettive VLANs, e ora è giunto il momento di connetterle tra loro, facendo in modo che nessuna delle due conosca la struttura interna dell'altra, avvalendoci della NAT.

Capitolo 3

NAT, ACL e GRE

Nel capitolo 2 abbiamo diviso gli indirizzi IP in pubblici e privati, sottolineando che i secondi possono essere utilizzati liberamente all'interno di una rete locale e che diverse reti locali possono utilizzare lo stesso spazio degli indirizzi. Sorge allora la domanda: come possono due *host*, appartenenti a LAN differenti ma con lo stesso indirizzo privato, comunicare tra loro? Occorre in questo caso avvalersi di un artificio che permetta ai dispositivi all'interno di una rete locale di avere un indirizzo che li identifichi univocamente all'interno della rete globale; a questo proposito è stata sviluppata NAT, che permette ad un dispositivo (spesso il *gateway*) di *tradurre* vari indirizzi privati in un indirizzo pubblico e viceversa. Oggi, come in passato, questo metodo è oggetto di forti polemiche poiché si ritiene che stia ritardando l'adozione di massa di IPv6, il quale dovrebbe risolvere l'esaurimento di indirizzi IPv4, e inoltre stravolge uno dei principi fondamentali di Internet, cioè quello secondo cui i dispositivi connessi a una rete possono comunicare affinché ci sia una connessione fisica tra loro.

Vedremo poi come poter intercettare e filtrare il traffico in base a dei criteri specifici grazie alle ACLs e come poter connettere direttamente dei dispositivi appartenenti a LAN differenti attraverso un tunnel GRE.

3.1 Breve storia di NAT

Nei primi anni 1990 Internet stava conoscendo uno sviluppo inaspettatamente veloce: sempre più utenti privati, grazie alla mercificazione dei calcolatori, si stavano connettendo alla rete globale e iniziavano ad avere accesso a una quantità di informazioni (spesso di dubbia qualità) mai vista nella storia dell'umanità. Qualche anno più tardi gli imprenditori, osservando dove volgeva in quel momento l'interesse dei consumatori, iniziarono a investire pesantemente nella loro presenza sul *web* e crearono le cosiddette *dot-com companies*, seminando il vento che avrebbe portato la tempesta dello scoppio della bolla delle *dot-com* nei primi anni 2000.

Ricordiamo che Internet era nato per connettere un numero limitato di utenti, e questa democratizzazione della rete rendeva sempre più concreto il timore che gli indirizzi IP sarebbero un giorno terminati. Nel 1991 l'RFC 1287 [5] suonava il primo campanello d'allarme e delineava alcune soluzioni a breve termine, tra cui l'idea di rendere un indirizzo IP unico all'interno di uno spazio ristretto piuttosto che in tutto il mondo [25]. Negli anni seguenti vennero proposte nuove soluzioni tra cui CIDR e NAT, mentre la Internet Engineering Task Force (IETF) lavorava a IPng, che in seguito sarebbe diventato IPv6 [25].

Purtroppo la IETF ha peccato di ottimismo, prevedendo che l'adozione di IPv6 sarebbe stata molto rapida e poco costosa, ed è stata troppo inflessibile nell'attenersi ai dettami originari del protocollo IP [25], per cui non si è curata di standardizzare NAT, che è stata sviluppata autonomamente dai vari produttori e largamente adottata dagli utenti poiché offre numerosi vantaggi immediati, ma di contro ha molti svantaggi a lungo termine [25] (vedremo entrambi quando parleremo del funzionamento di questa tecnologia). Oggi NAT è una tecnologia fondamentale per la comunicazione via Internet, e secondo Zhang forse non ci abbandonerà mai [25].

3.2 *A packet, by any other address...*

Studiamo ora il funzionamento della tecnologia NAT così come descritta nel RFC 3022 [21]. Una *NAT box* funge da «confine» che separa le reti *interne private* da quelle *esterne pubbliche* [12], per cui questa è di solito il *gateway* predefinito di una rete. La funzione di NAT è di modificare gli indirizzi IP di origine e di destinazione in modo che una rete interna possa essere raggiunta dalle reti esterne attraverso un unico indirizzo. Per fare ciò il *router* deve avere in memoria una *tabella di* mapping che contenga le corrispondenze tra indirizzi privati, indirizzi pubblici e, in alcuni casi, porte private e pubbliche, altrimenti sarebbe impossibile per due dispositivi all'interno di reti differenti comunicare.

Osserviamo la figura 3.1: supponiamo che Host, connesso ad una *NAT box*, voglia comunicare con Server. Il primo invia un pacchetto che ha come origine il suo indirizzo privato, e come destinazione l'indirizzo pubblico del secondo. Il *router*, essendo un *gateway* predefinito, intercetta il pacchetto e modifica l'indirizzo di origine, inserendo al posto dell'indirizzo privato un indirizzo pubblico in base alla tabella di *mapping* che il dispositivo possiede. Server riceve il pacchetto e invia una risposta che ha come destinazione l'indirizzo pubblico assegnato al *gateway*. Ancora una volta quest'ultimo intercetta il pacchetto e modifica l'indirizzo di destinazione pubblico con quello privato di Host, in modo che il pacchetto possa raggiungere il destinatario corretto. Le operazioni svolte da NAT appaiono semplici al primo sguardo, ma in realtà

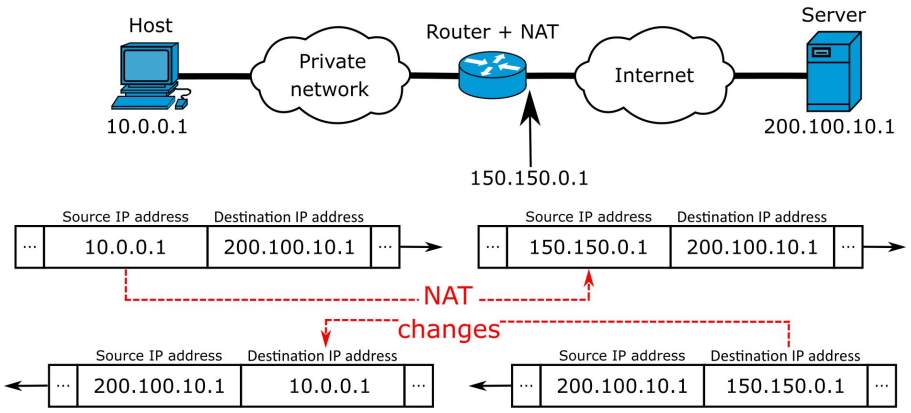


Figura 3.1: Funzionamento di NAT. © Michel Bakni. NAT_Concept-en. CC BY-SA 4.0. Convertita in formato JPEG.

sono estremamente complesse dal punto di vista logico e computazionale e possono introdurre latenze non trascurabili [12].

Essendo NAT un metodo estremamente diffuso per la comunicazione tra reti, ne esistono varie implementazioni con lo scopo di soddisfare le possibili esigenze degli utenti [12]:

***Static* NAT** permette di mappare un indirizzo IP privato a *uno e un solo* indirizzo pubblico. Questa tecnica non permette di risparmiare indirizzi pubblici, ma è utile affinché dispositivi come *server* possano essere raggiunti tramite un unico indirizzo IP pubblico.

***Dynamic* NAT** permette di mappare un numero limitato di *host* a un numero limitato di indirizzi pubblici presi da un *pool* di indirizzi. Questo è possibile grazie al fatto che non tutti gli *host* di una rete locale comunicano con l'esterno allo stesso tempo, e la grandezza del *pool* è stabilita in base al numero di dispositivi che accedono a reti esterne durante i picchi di traffico.

Network Address Port Translation (NAPT) permette di mappare vari indirizzi IP e varie porte a un solo indirizzo e altrettante porte. In questo modo gli *host* all'interno della LAN vengono distinti in base alla porta attraverso cui passa il dato, invece che in base all'indirizzo IP associato all'interfaccia del dispositivo.

***Easy* IP** funziona in modo analogo a NAPT, ma viene utilizzata per LAN di piccola scala che vogliono connettersi a Internet tramite un indirizzo IP temporaneo.

NAT *internal server* simile a NAT statico, ma protegge l'*host* all'interno della rete privata, il cui compito è di offrire un servizio specifico, specificando un *mapping* preciso tra indirizzi IP e porte.

Dopo aver visto le varie implementazioni di NAT, possiamo osservare come questa tecnica abbia vantaggi e svantaggi tra cui:

- un amministratore di rete può disegnare una topologia con estrema libertà e di modificarla a piacimento poiché questa, qualunque siano le sue dimensioni e la sua struttura, è visibile all'esterno attraverso un unico indirizzo IP;
- le tabelle di *routing* dei *router* nelle reti esterne rimangono di piccole dimensioni poiché queste contengono solo gli indirizzi pubblici delle *NAT box*, ed è compito di queste effettuare la traduzione tra indirizzi privati e pubblici e viceversa;
- la topologia di una LAN rimane nascosta ai dispositivi esterni a questa, per cui si crede che ci sia un maggior livello di sicurezza;
- non è possibile la comunicazione *peer-to-peer*, per cui protocolli come BitTorrent devono implementare tecniche per aggirare le restrizioni imposte da NAT, in modo da poter funzionare correttamente.
- l'unione di reti dovuta alla fusione di più aziende può risultare problematica se queste utilizzano uno stesso spazio degli indirizzi poiché si generano conflitti e *routing holes*;
- la manipolazione degli indirizzi IP rende necessario il ricalcolo del *checksum* del pacchetto, aumentando il carico computazionale e introducendo latenze.

Terminata questa presentazione teorica di NAT, non ci resta che implementare questa tecnologia su AR5 e AR6 in modo da connettere VLAN 50 e VLAN 60.

3.3 Configurazione di NAT

Avendo a che fare con una topologia di piccola scala, per semplicità usiamo una configurazione statica sia per i *client* che per i *server*. Per prima cosa assegniamo gli indirizzi IP mancanti alle interfacce gigabit ethernet 0/0/1 di AR5 e AR6 secondo la tabella 2.4 a pagina 16. Occorre poi creare un *mapping* tra gli indirizzi privati dei dispositivi, utilizzati per comunicare all'interno della VLAN, e quelli pubblici utilizzati per la comunicazione tra VLAN, come mostrato dalla tabella 3.1.

Eseguiamo alcuni test di raggiungibilità con il comando `ping` su tutti i dispositivi, e osserviamo come la configurazione di NAT sia andata a buon fine

Tabella 3.1: Corrispondenza tra dispositivi, indirizzi privati e indirizzi pubblici.

Dispositivo	Indirizzo privato	Indirizzo pubblico
client 1	10.60.1.1	10.56.1.61
client 2	10.50.4.1	10.56.1.54
server 1	10.50.1.1	10.56.1.51
server 2	10.60.4.1	10.56.1.64

poiché tutti gli *host* comunicano tra loro. Per vedere concretamente l'effetto di NAT sulla comunicazione, osserviamo la figura 3.2: una *Echo request* viene inviata da *client* 1 con origine 10.60.1.1 e destinazione 10.56.1.51 (figura 3.2a). Il pacchetto viene instradato senza essere alterato fino ad AR6 (come previsto dalla configurazione mostrata nel paragrafo 2.5). AR6 non possiede l'indirizzo di destinazione nella sua tabella di *routing*, ma per il principio di *longest match* lo passa, dopo aver scambiato l'indirizzo di origine privato con quello pubblico, ad AR5, che si presume sappia cosa farsene. Quest'ultimo conosce l'indirizzo di destinazione e scambia l'indirizzo pubblico con quello privato 10.50.1.1 (figura 3.2b), e infine invia il pacchetto a destinazione. Un discorso analogo vale per la *Echo reply* e per le successive comunicazioni.

Ora che abbiamo soddisfatto le prime due richieste del problema, occorre filtrare il traffico in modo che i *client* comunichino con i *server* aventi lo stesso numero, e per farlo useremo le ACLs.

3.4 ACL

In un contesto *enterprise* è altamente desiderabile monitorare il traffico all'interno di una rete in modo da poter operare su di esso. Per poter fare ciò si può ricorrere a un'ACL, una lista di regole che hanno il compito di *filtrare* il traffico passante attraverso una interfaccia di rete, *classificarlo* in base a criteri più o meno stringenti, e infine *selezionarlo* compiendo su di esso operazioni particolari [12]. Illustriamo gli usi principali delle ACLs con alcuni esempi: supponiamo di avere due aree di un'azienda, A e B, a cui è associato uno spazio degli indirizzi ciascuna. Solo A può accedere a Internet, per cui sul *gateway* che connette la rete aziendale e l'esterno è possibile intercettare il traffico che proviene dall'area B e bloccarlo. Supponiamo inoltre che alcuni *host* appartenenti all'area A gestiscano dei dati sensibili che vengono regolarmente scambiati con l'esterno, per cui il *gateway* menzionato in precedenza può captare il traffico sensibile e instradarlo attraverso una connessione criptata.

In base ai criteri con cui il traffico può essere classificato, un'ACL può essere di tre tipi, a cui è associato un intervallo di valori [12]:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.60.1.1	10.56.1.51	ICMP	74	Echo (ping) request
2	0.110000	10.56.1.51	10.60.1.1	ICMP	74	Echo (ping) reply
3	0.125000	10.60.1.1	10.56.1.51	ICMP	74	Echo (ping) request
4	0.219000	10.56.1.51	10.60.1.1	ICMP	74	Echo (ping) reply
5	0.219000	10.60.1.1	10.56.1.51	ICMP	74	Echo (ping) request
6	0.297000	10.56.1.51	10.60.1.1	ICMP	74	Echo (ping) reply
7	0.313000	10.60.1.1	10.56.1.51	ICMP	74	Echo (ping) request
8	0.391000	10.56.1.51	10.60.1.1	ICMP	74	Echo (ping) reply

(a) Cattura sull'interfaccia gigabit ethernet 0/0/1 di AR3.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.56.1.61	10.50.1.1	ICMP	74	Echo (ping) request
2	0.000000	10.50.1.1	10.56.1.61	ICMP	74	Echo (ping) reply
3	0.109000	10.56.1.61	10.50.1.1	ICMP	74	Echo (ping) request
4	0.109000	10.50.1.1	10.56.1.61	ICMP	74	Echo (ping) reply
5	0.203000	10.56.1.61	10.50.1.1	ICMP	74	Echo (ping) request
6	0.203000	10.50.1.1	10.56.1.61	ICMP	74	Echo (ping) reply
7	0.281000	10.56.1.61	10.50.1.1	ICMP	74	Echo (ping) request
8	0.281000	10.50.1.1	10.56.1.61	ICMP	74	Echo (ping) reply

(b) Cattura sull'interfaccia gigabit ethernet 0/0/1 di AR1.

Figura 3.2: Due catture di pacchetti sulle interfacce di *router* appartenenti a VLAN differenti mostrano lo scambio di dati tra *client* 1 e *server* 1.

Basic i pacchetti in transito vengono riconosciuti solamente in base all' indirizzo IP di origine. Una ACL di questo tipo possiede un valore da 2000-2999.

Advanced i pacchetti vengono riconosciuti in base al protocollo, agli indirizzi IP e alle porte di origine e destinazione. Una regola ACL di questo tipo possiede un valore da 3000-3999.

Layer 2 i pacchetti vengono riconosciuti in base alle informazioni di livello *data link*, ovvero gli indirizzi MAC di origine e destinazione. Una regola ACL di questo tipo possiede un valore da 4000-4999.

Le regole contenute in una ACL possono essere valutate in ordine di configurazione, cioè in base a un rule ID in ordine ascendente che può essere immesso manualmente o, se non specificato, assegnato automaticamente dal software, oppure in ordine automatico, per cui si valutano prima le regole più stringenti e poi quelle più lasche [12]. L'ordine di configurazione è utilizzato di *default* poiché la priorità imposta dai rule ID permette di risolvere eventuali conflitti causati da regole che si sovrappongono.

All'arrivo di un pacchetto, il sistema valuta tutte le regole impostate ed esegue solo quella per cui si verificano le giuste condizioni. Nel caso in cui nessuna condizione si verifichi, vengono applicate delle regole implicite che variano in base al tipo di operazione compiuta dall'ACL: se vogliamo filtrare il traffico, la regola implicita permette il passaggio di qualsiasi pacchetto, mentre se vogliamo classificarlo (ad esempio quando si utilizza NAT dinamico), la regola implicita blocca tutti i pacchetti [12].

Una particolarità delle regole basate sul confronto di indirizzi IP è quella di utilizzare una *wildcard mask*, una controparte della *subnet mask*, per cui i bit uguali a 0 indicano la parte di un indirizzo da esaminare, mentre i bit uguali a 1 indicano quella da ignorare; ad esempio, se in una regola compare l'indirizzo 10.1.1.0 con *wildcard mask* 0.0.0.255, significa che tutti gli indirizzi i cui primi 24 bit sono uguali a quello dell'indirizzo specificato nella regola soddisfano la condizione.

3.5 Configurazione delle ACLs

Per via dei semplici requisiti e per ridurre al minimo il carico computazionale, configuriamo due ACLs di tipo *basic* che contengano ciascuna solo due regole per la connessione ai *server*:

1. blocca il traffico proveniente dal *client* appartenente alla stessa VLAN del *server*;

2. permetti tutto il resto del traffico.

No.	Time	Source	Destination	Protocol	Length	Info
18	35.563…	10.56.1.61	10.50.1.1	ICMP	74	Echo (ping) request
19	35.563…	10.50.1.1	10.56.1.61	ICMP	74	Echo (ping) reply
24	44.547…	10.50.4.1	10.50.1.1	ICMP	74	Echo (ping) request
29	54.891…	10.56.1.64	10.50.1.1	ICMP	74	Echo (ping) request
30	54.906…	10.50.1.1	10.56.1.64	ICMP	74	Echo (ping) reply

(a) Cattura sull'interfaccia gigabit ethernet 0/0/0.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.56.1.61	10.50.1.1	ICMP	74	Echo (ping) request
2	0.000000	10.50.1.1	10.56.1.61	ICMP	74	Echo (ping) reply
3	19.343…	10.56.1.64	10.50.1.1	ICMP	74	Echo (ping) request
4	19.343…	10.50.1.1	10.56.1.64	ICMP	74	Echo (ping) reply

(b) Cattura sull'interfaccia gigabit ethernet 0/0/1.

Figura 3.3: Due catture di pacchetti sulle interfacce di AR1 mostrano l'effetto che le regole ACL esercitano sul traffico.

Per avere il massimo controllo sull'esecuzione delle regole, utilizziamo l'ordine di valutazione di *default*, ovvero quello di configurazione, specificando sempre il rule ID. È buona pratica configurare una *basic* ACL il più vicino possibile alla destinazione, per cui operiamo sulle interfacce gigabit ethernet 0/0/1 di AR1 e di AR7. Siccome è necessario bloccare l'accesso di un solo *client* la *wildcard mask* della prima regola è 0.0.0.0, il che significa che l'indirizzo di origine del pacchetto e quello presente nella regola dovranno corrispondere completamente. La seconda regola (detta *catchall rule* poiché si applica a tutti i casi non specificati in precedenza) ha come ID il valore 2999, in modo che sia sempre eseguita per ultima, e specifica «*any*» come sorgente affinché possa permettere tutto il traffico proveniente da origini non specificate in precedenza. Poiché il traffico *parte* dall'interfaccia del *router* e *arriva* a quella del *server*, specifichiamo l'opzione «*outbound*» quando configuriamo il filtraggio del traffico sull'interfaccia.

La figura 3.3 conferma che la configurazione delle regole è corretta, poiché *server* 2, *client* 1 e *client* 2 inviano ciascuno una *Echo request* a *server* 1 (figura 3.3a), ma quest'ultimo non riceve mai quella di *client* 2 poiché il traffico proveniente da quel *client* viene bloccato dall'interfaccia gigabit ethernet 0/0/1 (figura 3.3b). Il risultato ottenuto è analogo per *server* 2.

Ci avviciniamo alla risoluzione della topologia poiché ora bisogna soddisfare l'ultimo requisito, ovvero creare un collegamento diretto tra i due *server* avvalendoci del Generic Routing Encapsulation (GRE).

3.6 GRE

Come visto nel paragrafo 1.2, una rete aziendale oggi si estende ben oltre le mura di un edificio e può connettere regioni anche molto distanti tra loro. Per questa ragione la condivisione di informazioni aziendali avviene attraverso la rete pubblica, il che rende di primaria importanza l'utilizzo di tecniche e protocolli per mantenere la confidenzialità dei dati in transito.

Il protocollo IP originale prevede che i dati vengano inviati in chiaro, per cui è stata sviluppata la *suite* di protocolli IPsec, la quale permette di stabilire connessioni criptate tra dispositivi [24]. IPsec non permette tuttavia di trasmettere dati relativi a protocolli di *routing* dinamico come OSPF o RIP, per i quali viene utilizzato il protocollo GRE, sviluppato da Cisco e nato per trasmettere protocolli come Internetwork Packet Exchange (IPX) attraverso reti IP [12], e descritto per la prima volta in RFC 1701 [8] e RFC 1702 [9]. Questo protocollo permette di creare tunnel virtuali tra due *router*, in modo che il *payload* resti ignaro del fatto di essere trasportato all'interno di una rete più grande.

Il trasporto del *payload* attraverso un tunnel GRE avviene in questo modo [12]:

1. il *router* riceve sulla interfaccia connessa alla rete privata un pacchetto destinato a un *host* che risiede in una LAN differente;

2. il modulo del protocollo *network* del *router* analizza l'indirizzo di destinazione, e se in base alla tabella di *routing* il *next hop* specificato è un'interfaccia tunnel, il pacchetto viene inviato al modulo tunnel;

3. il modulo tunnel incapsula il pacchetto aggiungendo prima informazioni sul protocollo di questo e sui parametri di *checksum* del tunnel, poi gli indirizzi IP di origine e destinazione specificati come capi del tunnel;

4. il nuovo pacchetto è inviato al modulo IP che lo instrada attraverso la rete pubblica;

5. all'uscita dal tunnel, il *router* di destinazione analizza il pacchetto e riconosce che questo deve essere inviato al modulo GRE per iniziare il decapsulamento;

6. il modulo GRE riceve il pacchetto, rimuove le informazioni inserite dal modulo tunnel, e analizza il protocollo del *payload*;

7. il *payload* è inviato al livello dello *stack* protocollare appropriato, per essere elaborato in modo adeguato.

GRE non supporta la crittografia dei dati in transito nativamente, ma questo protocollo può viaggiare attraverso un tunnel IPsec. L'unione di questi due protocolli permette dunque il passaggio sicuro dei protocolli di

Tabella 3.2: Configurazioni iniziali delle interfacce tunnel di AR1 e AR4.

Dispositivo	Interfaccia	Indirizzo	Sorgente	Destinazione
AR1	tunnel 0/0/1	10.156.1.1/24	10.50.2.1	10.60.2.3
AR4	tunnel 0/0/1	10.156.1.2/24	10.60.2.3	10.50.2.1

routing dinamico, in modo che agenti esterni non possano carpire informazioni sulla struttura interna di una rete aziendale [12]. RIP ottiene un ulteriore beneficio, poiché è possibile aggirare la limitazione per cui possono essere specificati al massimo 15 *next hop* prima che sopraggiunga il problema *count-to-infinity* [12], per cui i *router* connessi si scambiano informazioni sbagliate a proposito dei passi necessari per raggiungere ad una destinazione, che possono arrivare a infinito.

Sebbene GRE non sia nativamente sicuro, egli supporta alcune funzionalità interessanti che permettono di monitorare l'integrità del tunnel e di impedire la connessione in caso di anomalie: l'autenticazione per mezzo di una chiave e rilevamento *keepalive*.

Per evitare di elaborare per errore dei pacchetti estranei, due dispositivi connessi da un tunnel possono essere configurati per aggiungere una chiave di 4 B ad un pacchetto durante l'incapsulamento compiuto dal modulo tunnel, la quale identifica i pacchetti appartenenti allo stesso flusso di traffico. Un pacchetto viene decapsulato con successo solo se la sua chiave è coerente con i valori impostati ai capi del tunnel, altrimenti questo viene scartato [12].

Il rilevamento *keepalive*, che deve essere impostato almeno su uno dei due *router*, permette di rilevare se un capo del tunnel è raggiungibile. Il *router* su cui è attiva questa funzionalità invia periodicamente pacchetti *keepalive* all'altro nodo, incrementando ogni volta un contatore di 1. Se non viene ricevuta una risposta prima che il contatore raggiunga il limite massimo, il nodo viene considerato non funzionante e la connessione viene recisa.

3.7 Configurazione di GRE

Per semplicità scegliamo di non configurare né l'autenticazione né il rilevamento *keepalive*. Per prima cosa configuriamo le interfacce tunnel su AR1 e AR4 come indicato nella tabella 3.2. Specifichiamo «gre» come protocollo tunnel, e inseriamo gli indirizzi di origine e di destinazione relativi alle interfacce fisiche gigabit ethernet 0/0/0 dei *router*, poiché quelle sono connesse con l'esterno. Infine configuriamo una rotta statica su ciascun *router* che permetta di instradare i pacchetti provenienti dai *server* verso le rispettive interfacce tunnel; questa dovrà specificare come destinazione lo spazio degli indirizzi a cui appartiene il *server* e come *next hop* l'interfaccia tunnel del *router* appartenente alla stessa VLAN del *server*.

No.	Time	Source	Destination	Protocol	Length	Info
26	54.031…	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) request
29	57.063…	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) request
31	60.063…	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) request
33	63.078…	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) request
2…	579.89…	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) request

(a) Cattura sull'interfaccia gigabit ethernet 0/0/0 di AR5.

No.	Time	Source	Destination	Protocol	Length	Info

(b) Cattura sull'interfaccia gigabit ethernet 0/0/1 di AR5.

Figura 3.4: Due catture di pacchetti sulle interfacce di AR5.

Tabella 3.3: Indirizzi pubblici associati alle interfacce ai capi del tunnel GRE.

Dispositivo	Interfaccia	Indirizzo
AR1	gigabit ethernet 0/0/0	10.56.1.52
AR4	gigabit ethernet 0/0/0	10.56.1.62

Un test di raggiungibilità rivela tuttavia che questa configurazione non permette ai due *server* di comunicare. Eseguendo il comando `ping` da *server* 1 e catturando il traffico sulle interfacce gigabit ethernet 0/0/0 e gigabit ethernet 0/0/1 di AR5 (figura 3.4), vediamo come il pacchetto arrivi con successo al *router* (figura 3.4a), ma non venga instradato verso la destinazione (figura 3.4b). Questo accade perché, come possiamo vedere dalla figura 3.5, AR5 non possiede nessuna informazione nella sua tabella di *routing* circa lo spazio degli indirizzi di destinazione, e dunque scarta il pacchetto. Testando la connessione attraverso *server* 2 abbiamo un risultato analogo poiché nemmeno AR6 sa come raggiungere AR1.

Ricordiamo che AR5 e AR6 sono delle *NAT box*, per cui si può configurare una NAT statica che traduca gli indirizzi privati delle interfacce gigabit ethernet 0/0/0 di AR1 e AR4 in indirizzi pubblici, cosicché queste possano essere raggiunte dall'esterno. La tabella 3.3 mostra gli indirizzi pubblici associati ai capi del tunnel GRE.

L'ultimo passo consiste nel riconfigurare le destinazioni delle interfacce tunnel (tabella 3.4), in modo che gli indirizzi associati ai capi siano quelli pubblici e non quelli privati. Con quest'ultimo accorgimento notiamo che il

```
<ar5>dis ip routing-table
Route Flags: R - relay, D - download to fib
------------------------------------------------------------------------------
Routing Tables: Public
         Destinations : 15       Routes : 15

Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

      10.50.1.0/24  Static  60   0           RD   10.50.2.1       GigabitEthernet
0/0/0
      10.50.2.0/24  Direct  0    0            D   10.50.2.2       GigabitEthernet
0/0/0
      10.50.2.2/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
    10.50.2.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/0
      10.50.3.0/24  Static  60   0           RD   10.50.2.3       GigabitEthernet
0/0/0
      10.50.4.0/24  Static  60   0           RD   10.50.2.3       GigabitEthernet
0/0/0
      10.56.1.0/24  Direct  0    0            D   10.56.1.1       GigabitEthernet
0/0/1
      10.56.1.1/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/1
     10.56.1.51/32  Unr     64   0            D   127.0.0.1       InLoopBack0
     10.56.1.54/32  Unr     64   0            D   127.0.0.1       InLoopBack0
    10.56.1.255/32  Direct  0    0            D   127.0.0.1       GigabitEthernet
0/0/1
      127.0.0.0/8   Direct  0    0            D   127.0.0.1       InLoopBack0
      127.0.0.1/32  Direct  0    0            D   127.0.0.1       InLoopBack0
127.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
255.255.255.255/32  Direct  0    0            D   127.0.0.1       InLoopBack0
```

Figura 3.5: Tabella di *routing* di AR5 prima della corretta configurazione del tunnel GRE.

Tabella 3.4: Configurazioni delle corrette destinazioni per le interfacce tunnel.

Dispositivo	Interfaccia	Destinazione
AR1	tunnel 0/0/1	10.56.1.62
AR4	tunnel 0/0/1	10.56.1.52

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.60.4.1	10.50.1.1	ICMP	98	Echo (ping) request id=0x0100, seq=11520/45, ttl=253
2	0.063000	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) reply id=0x0100, seq=11520/45, ttl=254
3	0.110000	10.60.4.1	10.50.1.1	ICMP	98	Echo (ping) request id=0x0100, seq=11776/46, ttl=253
4	0.141000	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) reply id=0x0100, seq=11776/46, ttl=254
5	0.203000	10.60.4.1	10.50.1.1	ICMP	98	Echo (ping) request id=0x0100, seq=12032/47, ttl=253
6	0.250000	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) reply id=0x0100, seq=12032/47, ttl=254
7	0.297000	10.60.4.1	10.50.1.1	ICMP	98	Echo (ping) request id=0x0100, seq=12288/48, ttl=253
8	0.360000	10.50.1.1	10.60.4.1	ICMP	98	Echo (ping) reply id=0x0100, seq=12288/48, ttl=254

```
> Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface -, id 0
> Ethernet II, Src: HuaweiTe_c4:22:68 (00:e0:fc:c4:22:68), Dst: HuaweiTe_13:48:e8 (00:e0:fc:13:48:e8)
> Internet Protocol Version 4, Src: 10.56.1.62, Dst: 10.56.1.52
> Generic Routing Encapsulation (IP)
> Internet Protocol Version 4, Src: 10.60.4.1, Dst: 10.50.1.1
> Internet Control Message Protocol
```

Figura 3.6: Test di raggiungibilità eseguito sull'interfaccia gigabit ethernet 0/0/1 di AR6. Notare in basso il doppio incapsulamento causato da GRE.

tunnel tra AR1 e AR4 è perfettamente funzionante, come si può vedere dalla figura 3.6.

Capitolo 4

It's not a kind of magic

Con il completamento dell'ultimo compito giungono al termine la configurazione della topologia di rete e questa tesi. Nel corso di questo elaborato abbiamo potuto apprezzare come una rete di calcolatori moderna si basi su un insieme di tecnologie complesse, molte delle quali nate per sopperire alle mancanze dei protocolli di rete originali. Anche l'interazione di queste tecnologie risulta complessa, poiché una comprensione non completa di queste, causata principalmente dall'inesperienza, porta a configurare i dispositivi in modo sbagliato, con il risultato di avere innumerevoli grattacapi (l'autore ha perso una quantità di tempo non indifferente durante la configurazione del tunnel GRE).

Sebbene la rete funzioni come richiesto, le scelte compiute sono piuttosto ingenue. L'utilizzo esclusivo di rotte statiche, ad esempio, garantisce il massimo controllo ma rende l'espansione di una rete molto laboriosa. Inoltre la configurazione statica di NAT non fa nulla per ridurre il consumo di indirizzi IP, ma in questo caso serve solamente ad «appiattire» la topologia, cioè rendere i *router* esterni ignari delle sottoreti a cui in realtà sono connessi gli *host*. Infine sarebbe stato più opportuno configurare un filtraggio del traffico più generico, in modo da garantire la protezione del *server* anche da connessioni effettuate da *host* sconosciuti.

La comunicazione ad alta velocità e a grande distanza anche su dispositivi mobili è un miracolo (altri lo considerano un flagello) di questo secolo, e il suo funzionamento può apparire magico; questa tuttavia, come mostrato in precedenza, è tutt'altro che magia.

Bibliografia

[1] R. Braden, Ed. *Requirements for Internet Host — Communication Layers*. RFC 1122. RFC Editor, ott. 1989. URL: https://www.rfc-editor.org/rfc/rfc1122.txt.

[2] Caldera International, Inc. *Direct versus indirect routing*. URL: https://web.archive.org/web/20200929130410/http://osr5doc.xinuos.com/en/NetAdminG/iproutingC.direct_indirect.html.

[3] Matthew Castelli. *LAN switching first-step*. Cisco Press, 8 lug. 2004. ISBN: 9781587054198. URL: http://proquest.safaribooksonline.com/1587201003.

[4] V. Cerf, Y. Dalal e C. Sunshine. *Specification of Internet Transmission Control Program*. RFC 675. RFC Editor, dic. 1974. URL: https://www.rfc-editor.org/rfc/rfc675.txt.

[5] D. Clark et al. *Towards the Future Internet Architecture*. RFC 1287. RFC Editor, dic. 1991. URL: https://www.rfc-editor.org/rfc/rfc1287.txt.

[6] Kate Conger e Nathaniel Popper. *Florida Teenager Is Charged as 'Mastermind' of Twitter Hack*. 31 Lug. 2020. URL: https://www.nytimes.com/2020/07/31/technology/twitter-hack-arrest.html.

[7] S. Deering et al. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460. RFC Editor, dic. 1998. URL: https://www.rfc-editor.org/rfc/rfc2460.txt.

[8] S. Hanks et al. *Generic Routing Encapsulation (GRE)*. RFC 1701. RFC Editor, ott. 1994. URL: https://www.rfc-editor.org/rfc/rfc1701.txt.

[9] S. Hanks et al. *Generic Routing Encapsulation over IPv4 networks*. RFC 1702. RFC Editor, ott. 1994. URL: https://www.rfc-editor.org/rfc/rfc1702.txt.

[10] Steve Hargis. «User Authentication in the Enterprise Network». URL: http://www.bu.edu/tech/files/2010/01/sc02_enterasys.pdf.

[11] *How Encapsulation Works Within the TCP/IP Model*. 27 Gen. 2008. URL: https://web.archive.org/web/20120807024653/http://learn-networking.com/tcp-ip/how-encapsulation-works-within-the-tcpip-model.

[12] Huawei Technologies CO., LTD. «Materiale del corso di preparazione alla certificazione HUAWEI Routing e Switching v2.5». 2019.

[13] *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*. Standard ISO/IEC 7498-1. Geneva, CH: International Organization for Standardization, 15 nov. 1994.

[14] J.A.N. Lee. *Computer Pioneers — Charles Bachman*. URL: https://history.computer.org/pioneers/bachman.html.

[15] D. McPherson e B. Dykes. *VLAN Aggregation for Efficient IP Address Allocation*. RFC 3069. RFC Editor, feb. 2001. URL: https://www.rfc-editor.org/rfc/rfc3069.txt.

[16] Jim Metzler e Steve Taylor. *Why it's time to let the OSI model die*. Set. 2008. URL: https://www.networkworld.com/article/2276158/why-it-s-time-to-let-the-osi-model-die.html.

[17] J.C. Mogul e J. Postel. *Internet Standard Subnetting Procedure*. RFC 950. RFC Editor, ago. 1985. URL: https://www.rfc-editor.org/rfc/rfc950.txt.

[18] Mike Murphy. *A teenager reportedly hacked Apple and stole tons of information*. 16 Ago. 2018. URL: https://qz.com/1360782/apple-was-reportedly-hacked-by-a-teenager-who-stole-90-gb-of-information/.

[19] Y. Rekhter et al. *Address Allocation for Private Internets*. RFC 1918. RFC Editor, feb. 1996. URL: https://www.rfc-editor.org/rfc/rfc1918.txt.

[20] Andrew L. Russell. *OSI: The Internet That Wasn't*. 30 Lug. 2013. URL: https://spectrum.ieee.org/tech-history/cyberspace/osi-the-internet-that-wasnt.

[21] P. Srisuresh e K. Egevang. *Traditional IP Network Address Translator (Traditional NAT)*. RFC 3022. RFC Editor, gen. 2001. URL: https://www.rfc-editor.org/rfc/rfc3022.txt.

[22] stretch. *IPv4 Exhaustion: What About Class E Addresses?* 14 Ott. 2010. URL: https://packetlife.net/blog/2010/oct/14/ipv4-exhaustion-what-about-class-e-addresses/.

[23] *TCP/IP — Complete History of the TCP/IP Protocol Suite*. URL: https://history-computer.com/Internet/Maturing/TCPIP.html.

[24] *What is IPsec?/How IPsec VPNs work*. URL: https://www.cloudflare.com/learning/network-layer/what-is-ipsec/.

[25] Lixia Zhang. *A Retrospective View of NAT*. 7 Ott. 2007. URL: https://www.ietfjournal.org/a-retrospective-view-of-nat/.