ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea in Ingegneria e Scienze Informatiche

DoS and DDoS attacks in Named Data Networking

Tesi di laurea in
Programmazione Di Reti

Relatore
Prof. Giovanni Pau

Candidato
Fabrizio Margotta

III Sessione di Laurea
Anno Accademico 2019-2020

Sommario

La nascita di Internet ha portato benefici senza precedenti in termini sociali ed economici. Le tecnologie e i protocolli inizialmente adottati andavano a risolvere i problemi causati dall'inaffidabilità della preesistente linea telefonica e nel corso dei vari decenni hanno garantito il funzionamento delle comunicazioni globali. Tuttavia ci si pone delle domande sulla necessità di un nuovo cambio di paradigma che risponda alle nuove esigenze riscontrate nell'evoluzione di Internet stesso e dei servizi offerti. Named Data Networking, come progetto della famiglia delle *Future Internet Architectures*, propone una visione di Internet rinnovata in cui l'entità fondamentale scambiata è il *Nome*. L'attenzione viene posta sul *cosa* i nodi richiedono alla rete e non sul *dove* reperiscono le informazioni. Come nuova architettura, NDN propone approcci innovativi sui fronti dell'inoltro e dell'instradamento dei pacchetti nella rete e garantisce l'implementazione di diversi aspetti di sicurezza *by design*. Ciò non è però sufficiente a proteggere da tutte le minacce della rete, tra cui gli attacchi DDoS che costituiscono ancora oggi un rischio importante per i fornitori di servizi in Internet. Sono quindi necessari ulteriori sforzi per partorire un'architettura che risponda a requisiti di efficienza e sicurezza.

Keywords: Future Internet Architectures, Information-Centric Networking, Named Data Networking, NS-3, ndnSIM, Network Security, DDoS Attacks, Interest Flooding Attack

Alla mia famiglia
A Giovanna

Indice

Elenco delle figure

Elenco dei listati

Capitolo 1

Introduzione

Al fine di comprendere al meglio tutti i dettagli della tecnologia presa in esame e degli approfondimenti effettuati occorre chiarire le ragioni dietro la sua nascita nonché il ruolo chiave della sicurezza informatica nei processi produttivi e le conseguenze su ampia scala a cui possono condurre gli attacchi informatici.

1.1 Nascita di Internet

L'infrastruttura di comunicazione che oggi permette lo scambio di informazioni a livello globale a fini commerciali, finanziari, multimediali e legati ai servizi è frutto di un'evoluzione complessa e articolata, sia dal punto di vista tecnologico che da quello funzionale.

Internet, infatti, nasce per esigenze militari espresse dal Dipartimento della Difesa statunitense: durante la guerra fredda vi era la necessità di una rete di comando e controllo in grado di resistere ad uno scontro nucleare. L'esistente rete telefonica, per la sua costruzione gerarchica, non avrebbe consentito una comunicazione affidabile in caso di guasti alle centrali di commutazione di alto livello.

A partire dagli anni 60 venne quindi commissionata la progettazione di una rete alternativa. Dopo le prime proposte rifiutate dalla AT&T (che ai tempi monopolizzava la rete telefonica USA) l'organizzazione ARPA (Advanced Research Projects Agency) riuscì a progettare una rete a commutazione di pacchetto, ARPANET, che venne costruita e fatta operare nel 1969 in quattro nodi tra università e centri di ricerca statunitensi. Nei tre anni successivi erano più di trenta i nodi collegati alla rete ARPANET.

Dalla fine degli anni 70 alla fine degli anni 80 venne invece realizzata, a cura della NSF (National Science Foundation), una rete parallela per favorire la comunicazione tra enti di ricerca che non avevano un contratto con il ministero della

difesa (questi ultimi, infatti, erano già collegati ad ARPANET). Tale infrastruttura prese il nome di NSFNET e venne presto collegata ad ARPANET a causa del suo immediato successo.

A partire dagli anni 90 si osservò la graduale uscita degli enti governativi americani come soggetti finanziatori di tali progetti e il conseguente ingresso di organizzazioni commerciali che potessero regolare l'accesso alla rete (di fatto i precursori dei moderni Internet Service Provider)[21].

1.2 Evoluzione della rete e dei servizi

Ben presto la rete di ricercatori inizialmente realizzata si ampliò in termini di numero di utenze. Tale espansione fu dovuta anche all'implementazione ed adozione del modello di riferimento TCP/IP che garantisce interoperabilità tra reti tecnologicamente diverse, realizzando di fatto il concetto di *internetworking*, ossia una rete di reti.

I servizi inizialmente fruibili erano strettamente legati alla comunicazione (*e-mail* e *newsgroups*) o al controllo dei computer (*terminale remoto* e *trasferimento file*).[21] L'avvento del World Wide Web [22], dell'applicativo *browser* e del concetto di *ipertesto* ha favorito un'esplosione dell'utilizzo di Internet da parte di utenti non accademici, nonché la nascita di servizi multimediali e di intrattenimento, commerciali, finanziari, gestionali, logistici e molto altro ancora.[23]

Contestualmente alla creazione di reti negli USA, anche in Europa e nel resto del pianeta furono realizzate infrastrutture simili e successivamente collegate tra loro. Ciò ha consentito la rapida crescita del numero di utenti, passando da diverse centinaia di milioni a più di tre miliardi di persone connesse ad Internet (fig. 1.1).

Le previsioni effettuate in [2] mostrano un andamento in crescita sia nel numero di utenti sia nella varietà dei dispositivi coinvolti (fig. 1.2).

Le connessioni effettuate, infatti, continuano a riguardare computer e smartphone, tuttavia l'impiego massivo di tecnologie legate all'Internet of Things evidenzia (fig. 1.3) una forte presenza di dispositivi diversi (sensori installati in abitazioni, automobili collegate in rete e così via) che scambiano informazioni tra loro (connessioni M2M: *Machine-to-Machine*).

1.3 Verso l'Information-Centric Networking

Come descritto nelle sezioni precedenti, la nascita di Internet è legata alla necessità di mettere in comunicazione dei computer, aspetto ereditato dalla preesistente infrastruttura telefonica.

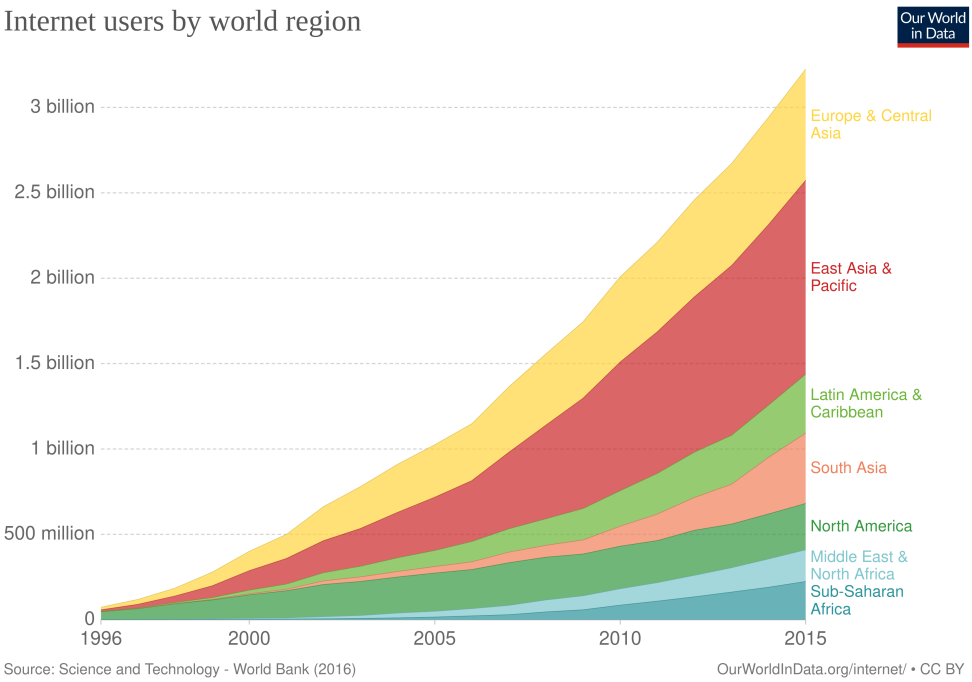


Figura 1.1: Numero di utenti Internet dal 1996 al 2015 diviso per regione geografica. [1]

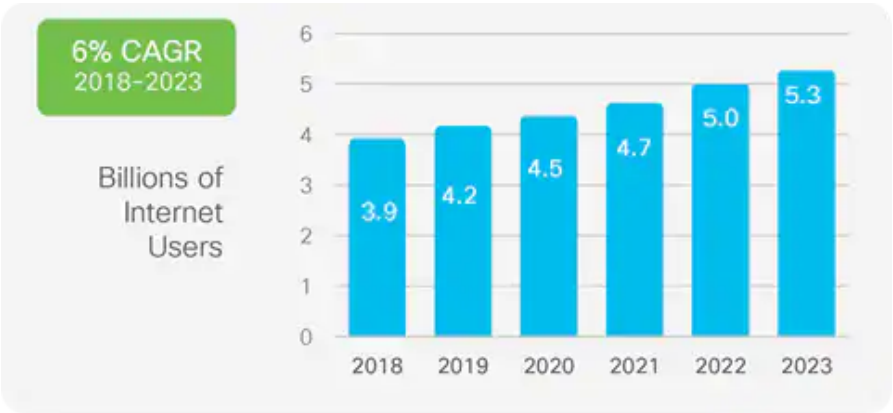


Figura 1.2: Numero di utenti Internet (in miliardi) previsto fino all'anno 2023. [2]

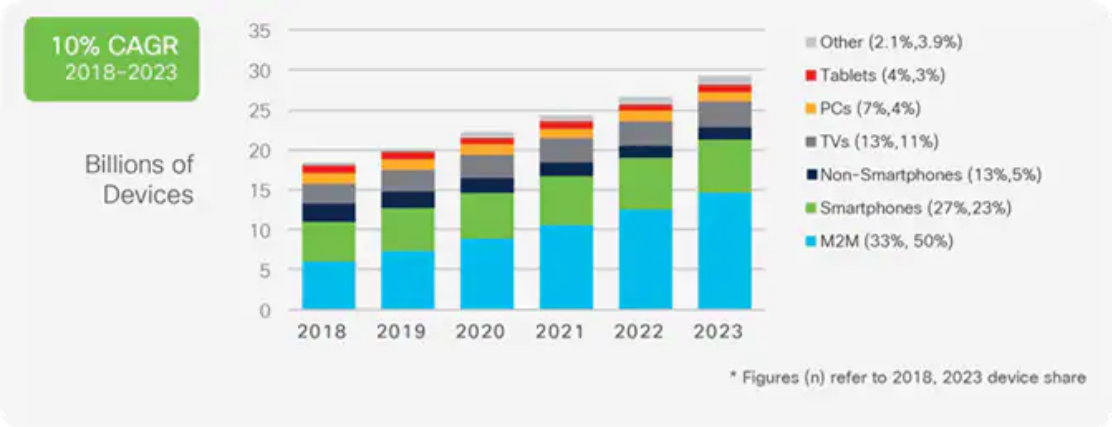


Figura 1.3: Previsione fino all'anno 2023 del numero di dispositivi connessi ad Internet diviso per tipologia. [2]

L'evoluzione dei servizi erogati (dal commercio alle piattaforme multimediali) ha evidenziato una differenza sostanziale nel modo in cui Internet stesso viene utilizzato. Se nei decenni passati veniva concepito un modello di comunicazione che poneva come entità principale gli *host* identificati da indirizzi, oggi si può pensare a un modello di distribuzione dei contenuti a più ampia scala e che generalizzi il precedente. Tale astrazione riguarda i nomi delle entità coinvolte nei pacchetti scambiati dal protocollo: un modello orientato alla distribuzione digitale può infatti adottare una qualsiasi informazione come entità principale nei pacchetti (gli indirizzi dei dispositivi, porzioni di video, azioni richiamabili su un oggetto IoT...) anziché i riferimenti agli *endpoint* utilizzati nel modello corrente.

Questa teoria è stata inizialmente presentata in un "Google Tech Talk" nel 2006 a cura di Van Jacobson [24] in cui il ricercatore motivava l'inadeguatezza di IP nel diffondere contenuti, nel meccanismo di *broadcasting* e nel fornire appropriati meccanismi di sicurezza ai dati piuttosto che ai canali di comunicazione. Lo stesso Jacobson in [25] introdusse una proposta con il nome di "Content-Centric Networking" in cui illustrava il modello ideato e i dettagli architetturali in merito a instradamento, scalabilità e sicurezza.

A partire dal 2010 il termine "Information-Centric Networking" è stato introdotto negli articoli accademici e discusso nelle varie conferenze, ispirandosi proprio alla presentazione di Jacobson. Nello stesso anno la National Science Foundation ha fondato quattro progetti come parte del programma Future Internet Architecture coinvolgendo la comunità di ricercatori con l'obiettivo di esplorare e sviluppare soluzioni architetturali innovative per Internet che prendessero in considerazione l'evoluzione dell'infrastruttura esistente e delle sue implicazioni sociali, economiche e legali.[26] A far parte del progetto vi è NDN (Named Data Networking) [12]

ovvero una proposta di architettura per Internet che segue il modello Information-Centric Networking e realizza una serie di implementazioni *software* del protocollo basandosi sul lavoro precedentemente pubblicato da Jacobson.[25]

Tra il 2011 e il 2012 l'Internet Research Task Force (IRTF) accoglie l'ingresso del gruppo di lavoro dedicato a ICN (ICNRG: ICN Research Group) che, come tutti i gruppi dell'IRTF, concentra in un unico canale la comunità di ricercatori e la relativa documentazione.[27]

Il progetto NDN viene poi finanziato dalla National Science Foundation tra il 2011 e il 2016 per una cifra di 13.5 milioni di dollari e vede negli anni la graduale partecipazione consorziale di decine di enti tra università e centri di ricerca aziendali.

Le attività portate avanti dal gruppo ICNRG e in particolare da quello NDN riguardano: (i) la redazione di RFC per la formalizzazione degli aspetti salienti del modello proposto (ii) l'esplorazione in letteratura di soluzioni efficienti per le caratteristiche architetturali principali (routing, forwarding, caching, sicurezza e nomi) e l'impatto sulle tecnologie più innovative (IoT, vehicular networks, edge computing, big data) (iii) lo sviluppo di opportune librerie e software a supporto degli esperimenti da condurre in scenari reali o simulati.

1.4 La sicurezza in Internet

La crescita del numero di utenti e l'evoluzione dei servizi disponibili su Internet ha portato notevoli benefici in termini economici e sociali: basti pensare alla creazione di nuovi mercati e opportunità professionali, all'estrema facilità con cui è possibile accedere alle informazioni o alla disponibilità di piattaforme di aggregazione digitale che permettono di comunicare con chiunque, nel mondo, in qualsiasi istante e in modo immediato.

Una tale opportunità porta con sé anche dei pericoli per le aziende che forniscono servizi digitali mediante Internet e per gli utenti stessi. Ogni settore tecnologico può essere soggetto a rischi e minacce differenti (fig. 1.4), tra cui malware, phishing, DDoS e attacchi legati a cyber-attivisti (fenomeno dell'*hacktivism*) o gruppi spesso sponsorizzati dagli Stati (APT: *Advanced Persistent Threat*).

Il principale vettore di attacco in una violazione di un'azienda o di un utente Internet è la email, che viene tipicamente usata per ingannare la vittima al fine di compiere un furto di credenziali o installare software malevolo per la compromissione di altri servizi o sistemi.

La minaccia principale osservata dai fornitori di servizi Internet [2] è rappresentata dagli attacchi DDoS (*Distributed Denial of Service*), che consistono nel recapitare, attraverso un'estesa flotta di dispositivi compromessi (tipicamente chiamati *zombie* o *bot*), un'ingente quantità di richieste verso i server della vittima. Il risul-

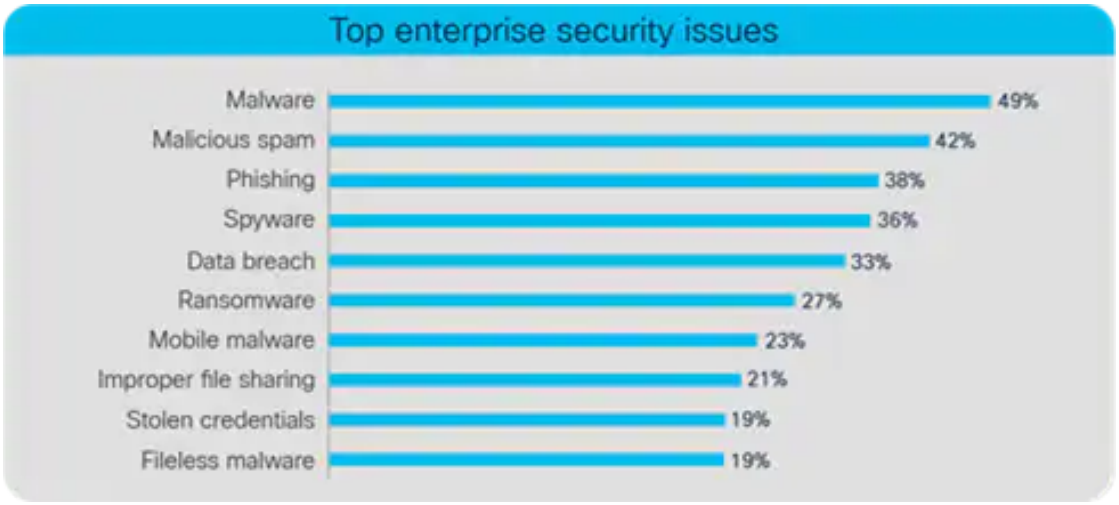


Figura 1.4: Principali problemi di sicurezza riscontrati nelle aziende. [2]

tato di tale operazione rappresenta un problema per le organizzazioni in quanto le macchine attaccate vedono le risorse esaurirsi al punto tale da non poter più erogare agli utenti i servizi esposti su Internet.

Il più grande attacco DDoS conosciuto ad oggi è stato osservato da Google nel 2017 e reso noto il 16 ottobre 2020 [28]: l'azienda statunitense ha mitigato del traffico malevolo della portata di 2.5 Tbps proveniente da centinaia di migliaia di macchine infette, il doppio di quanto registrato nell'attacco del 2018 ai danni di GitHub.[29]

L'aumento del numero di dispositivi IoT già precedentemente discusso ha forti legami con i temi di *cybersecurity*. L'adozione globale di soluzioni *smart* porta con sé numerosi vantaggi per gli utenti ma nasconde insidie legate alla loro implementazione. I *software* per i sistemi *embedded* e i relativi *webservices* realizzati dai fornitori di servizi IoT non sono infatti risultati sicuri. Questi dispositivi, una volta compromessi, sono stati utilizzati per compiere attacchi DDoS formando *botnet* di grandi dimensioni come la popolare Mirai [30].

Come mostrato in fig. 1.5 il fenomeno è in forte crescita, gli studi indicano che il numero di attacchi DDoS previsti nel 2023 può raddoppiare rispetto a quanto registrato nel 2018.

Pertanto è fondamentale che vengano concentrati gli sforzi delle comunità produttive e di ricerca affinché si promuovano processi di sviluppo sicuri e misure di mitigazione efficaci per gli attacchi.

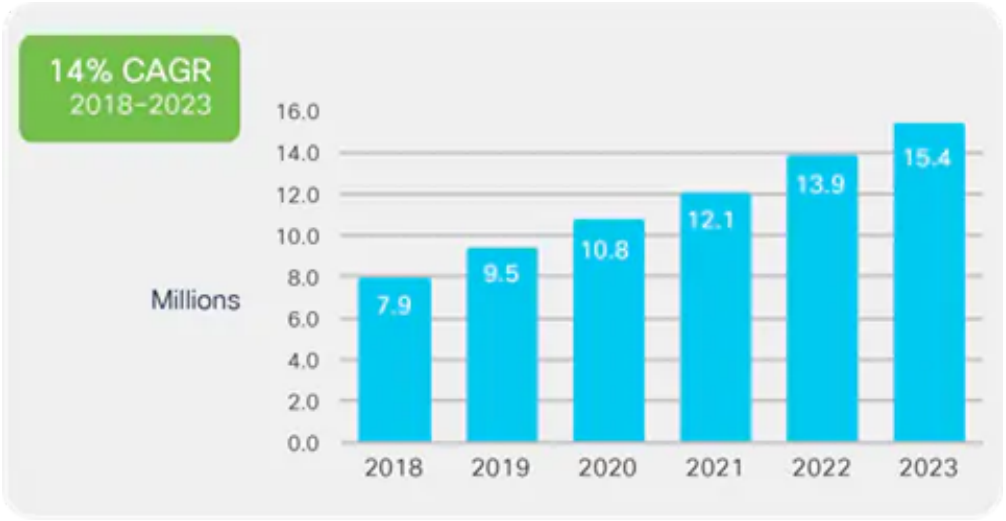


Figura 1.5: Numero (in milioni) di attacchi DDoS previsti fino all'anno 2023. [2]

1.5 Struttura della tesi

Con le premesse effettuate in questo capitolo è possibile comprendere il percorso proposto in questa sede. Nel capitolo 2 viene mostrato il protocollo IP e il modello architetturale di riferimento al fine di presentare nel capitolo 3 NDN, le sue caratteristiche e le differenze con IP. Il capitolo 4 descrive gli aspetti principali della negazione del servizio in NDN (DoS) con un'attenzione particolare all'attacco *Interest Flooding*. Nel capitolo 5 viene presentato il progetto sviluppato dal candidato, con riferimento all'attacco mostrato nel capitolo precedente. In ultimo si dà spazio alle conclusioni e alle proposte di lavoro future.

Capitolo 2

Internet Protocol

Di seguito vengono descritte le caratteristiche principali del protocollo fondamentale che permette ai dispositivi di comunicare in Internet. Si tralasciano per semplicità alcuni aspetti non necessari all'approfondimento proposto nel presente elaborato di tesi.

2.1 Modello architetturale di riferimento

2.1.1 Architetture e protocolli di rete

La rete Internet nel suo complesso è formata da dispositivi tecnologicamente diversi, così come le applicazioni e i servizi forniti possono avere necessità differenti in termini temporali e di risorse: una videochiamata richiede una latenza nettamente inferiore rispetto all'invio di una email. Risulta quindi chiaro che non tutte le comunicazioni, seppur effettuate sulla stessa rete, debbano essere operate nello stesso modo. Nonostante ciò è fondamentale che tutte le interazioni tra gli utenti avvengano senza errori e che le differenze appena descritte non siano in alcun modo evidenti. É possibile fare un altro esempio: due utenti che scambiano messaggi possono essere connessi a mezzi trasmissivi differenti come la rete cablata o wireless e ciò non deve avere alcuna conseguenza sulla trasmissione, che deve essere un processo trasparente agli utenti stessi.

Negli anni in cui stava nascendo Internet sono stati quindi progettati dei modelli architetturali che permettessero di concepire la trasmissione di informazioni in modo altamente strutturato e gerarchico. É il caso dei modelli ISO/OSI e TCP/IP, che prevedono più livelli di astrazione, indipendenti tra loro e ognuno dedicato ad uno specifico scopo (fig. 2.1).

Non verranno analizzate le differenze tra i due modelli, si consideri perciò che TCP/IP è l'architettura di riferimento per l'attuale rete Internet. Il suo successo è

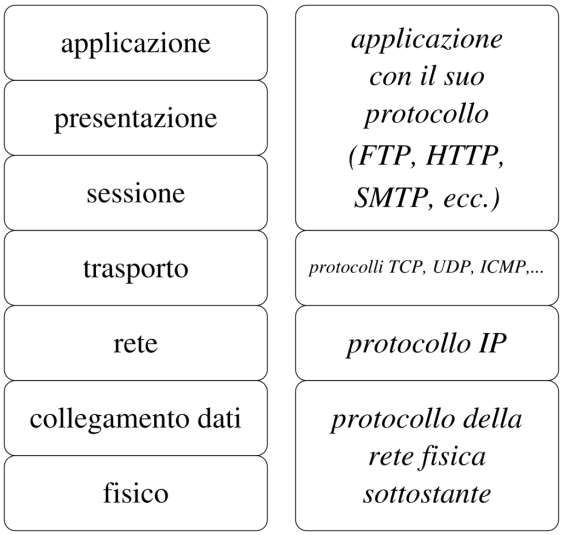


Figura 2.1: Modello a livelli ISO/OSI (sinistra) e TCP/IP (destra). [3]

dovuto all'attenzione particolare posta ai livelli di trasporto e di rete (descritti in seguito) e all'interoperabilità tra i protocolli usati nelle reti che venivano collegate ad ARPANET, aspetto fondamentale per la presenza di tecnologie di trasmissione differenti.

Ogni livello dell'architettura, quindi, fornisce un preciso servizio allo strato superiore e all'occorrenza risolve i problemi del livello sottostante.

La comunicazione con i livelli adiacenti avviene mediante opportune interfacce che, da un lato descrivono i servizi offerti allo strato soprastante, dall'altro realizzano un vero e proprio incapsulamento dell'implementazione del livello corrente: ciò garantisce l'interoperabilità descritta in precedenza poiché, se usate le stesse interfacce ed implementato lo stesso servizio, i livelli sono intercambiabili tra loro (cosa che di fatto avviene con i protocolli di trasporto TCP e UDP).

Affinché una comunicazione sia coerente, è necessario che le entità di pari livello dei due dispositivi adottino un protocollo, ossia un medesimo insieme di regole e lo stesso formalismo nell'interpretazione dei messaggi e delle informazioni di controllo (fig. 2.2).

La comunicazione tra due dispositivi parte logicamente dal livello più alto del mittente, che invia (tramite le interfacce) un messaggio allo strato sottostante il quale lo arricchisce con delle informazioni di controllo. Questo passaggio viene iterato fino al livello più basso (fig. 2.3). L'informazione viene quindi inviata sul canale trasmissivo fino al destinatario in cui, dal livello più basso a quello applicativo, vengono opportunamente interpretate le informazioni di controllo trasmesse "virtualmente" dallo strato paritario del mittente. Queste ultime possono riguardare la dimensione massima dei messaggi o il numero di sequenza di un cer-

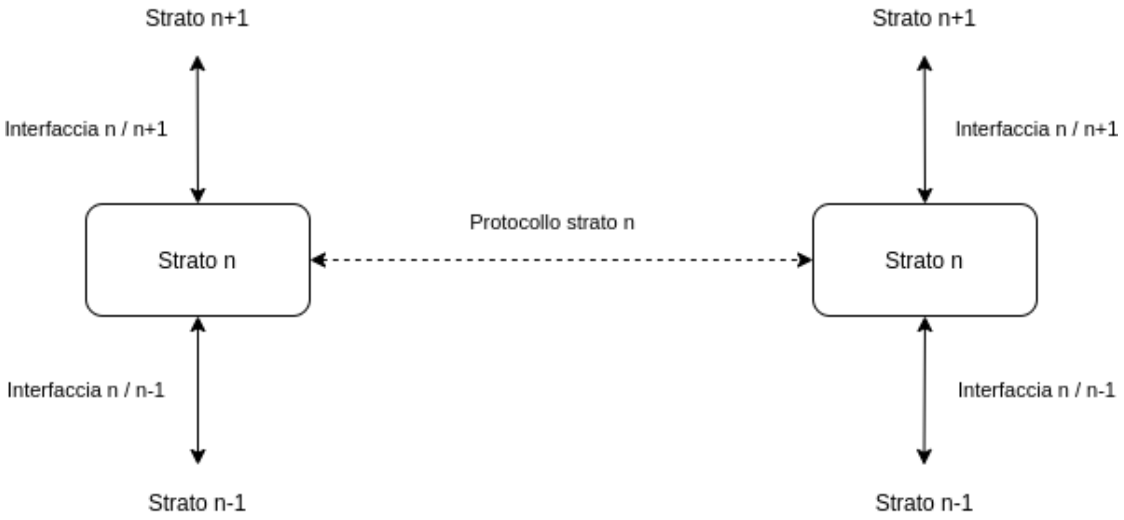


Figura 2.2: Interfacciamento dei livelli adiacenti e protocollo di comunicazione tra entità di pari livello (liberamente estratto da "Computer Networks" di Andrew S. Tanenbaum).

to pacchetto (utilizzato affinché si possa ricostruire la trasmissione nell'ordine di invio) e possono dipendere dal protocollo utilizzato.[31]

2.1.2 Servizi forniti dai livelli di TCP/IP

Nella sezione 2.1.1 è stato introdotto il modello a livelli per le reti e sono state descritte le caratteristiche fondamentali di tali architetture, come l'uso di opportuni messaggi corredati da intestazioni (*headers*) e il flusso che questi ultimi compiono dal mittente al destinatario.

Segue ora un elenco dei servizi offerti da ciascun livello dell'architettura TCP/IP [32], utile a fornire un contesto generale su tutto lo *stack* protocollare per poi effettuare un approfondimento del livello di rete.

Il livello più alto consiste nelle applicazioni a cui l'utente si interfaccia direttamente: è infatti l'unico strato di cui quest'ultimo ha la reale consapevolezza. Le applicazioni forniscono quindi all'utente dei servizi, quali ad esempio l'accesso al World Wide Web, la consultazione della posta elettronica, il trasferimento di file, il controllo di dispositivi *smart* (nel contesto dell'*Internet of Things*) o l'accesso remoto ad altre macchine. Ogni applicazione comunica con la controparte attraverso un protocollo: tutti i servizi citati dispongono di uno o più protocolli, come HTTP per il World Wide Web o FTP per il trasferimento di file.

I protocolli del livello di trasporto si occupano di gestire il trasferimento dei dati ricevuti dallo strato applicativo dal mittente ai destinatari (*end-to-end*). Sono due

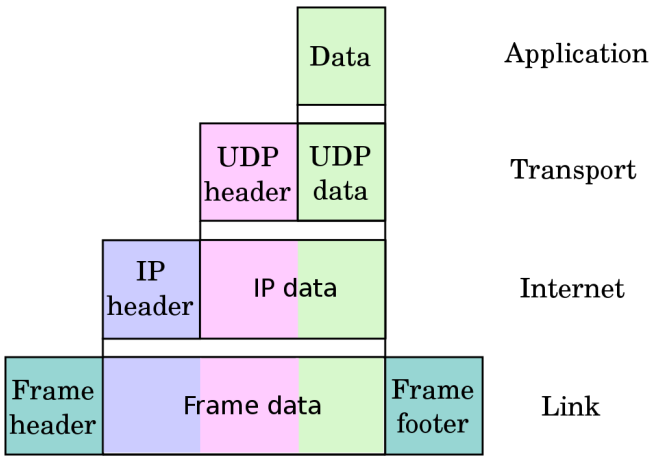


Figura 2.3: Incapsulamento dei dati nei messaggi di ogni livello, con relative informazioni di controllo. [4]

i protocolli definiti ed usati in questo strato: TCP (*Transmission Control Protocol*) [33] che fornisce un servizio orientato alla connessione e privo di errori e UDP (*User Datagram Protocol*) [34] basato sull'invio delle informazioni senza l'instaurazione di una connessione tra mittente e destinatario (*connectionless*).

Il livello di rete consente ai dispositivi di inviare pacchetti in una qualsiasi rete, indipendentemente dal loro ordine (sarà infatti il livello superiore a doversene occupare, se necessario) attraverso il percorso più breve disponibile da un host all'altro. Ulteriori dettagli verranno forniti nella sezione 2.2 con un approfondimento al protocollo IP appartenente a tale strato.

Infine il modello in esame non pone, volontariamente, particolare attenzione alla parte fisica di una rete: vengono unicamente definite le modalità con cui gli indirizzi IP debbano essere tradotti in indirizzi delle interfacce fisiche di rete (mediante il protocollo ARP: *Address Resolution Protocol*).[35]

2.2 Caratteristiche di IP e dello strato di rete

Internet Protocol (IP) è il protocollo usato nello strato di rete del modello di riferimento TCP/IP. Il suo obiettivo è quello di inviare pacchetti dall'*host* mittente a quello di destinazione attraversando un sistema di reti interconnesse. Un tipico esempio di collegamento tra reti diverse è presente in figura 2.4, ove è possibile osservare la presenza di più reti private (*LAN: Local Area Network*) connesse tra loro mediante router. Questi ultimi consistono in dispositivi ad elevate prestazioni il cui obiettivo è quello di salvare i pacchetti ricevuti e inoltrarli verso l'interfaccia

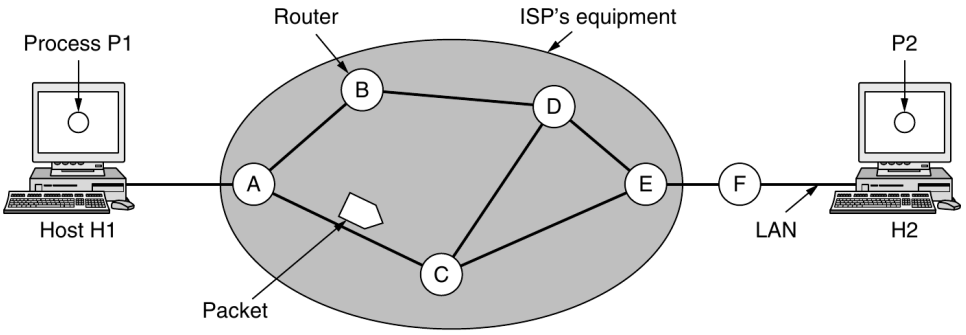


Figura 2.4: Schema di rappresentazione del collegamento tra due host in LAN diverse connesse ad Internet mediante dei router.[5]

di rete connessa al prossimo nodo in direzione della destinazione (meccanismo *store and forward*). Tale operazione è possibile grazie al fatto che i router implementano l'architettura TCP/IP fino al livello di rete (fig. 2.5).

Tutti gli host (*client*, *server* e router) sono identificati attraverso un indirizzo logico di lunghezza fissa, che verrà descritto in sezione 2.2.2. IP è nato con il semplice scopo di consegnare messaggi: meccanismi per garantire affidabilità o gestione del flusso sono demandati ai protocolli dei livelli superiori.[36]

2.2.1 Formato dei pacchetti

Ciò che IP riceve dal livello superiore è un segmento (o datagramma nel caso di UDP) che viene (compreso il suo *header*) incapsulato nella sezione dedicata ai dati di un pacchetto IP.

L'intestazione di un pacchetto IP versione 4 è rappresentata in fig. 2.6.

I campi specificati nell'intestazione [36] descrivono delle indicazioni di cui i nodi che instradano il pacchetto dovranno considerare:

- *Version*: versione del protocollo.
- *Internet Header Length*: numero di *dword* (32 bit) per la lunghezza dell'intestazione (valore minimo 5)
- *Type of Service*: maschera di singoli bit che specificano la qualità del servizio intesa come precedenza, ritardo, portata e affidabilità
- *Total Length*: dimensione totale del pacchetto IP, comprensiva di intestazione e sezione dati
- *Identification*: identificatore del pacchetto se soggetto a frammentazione

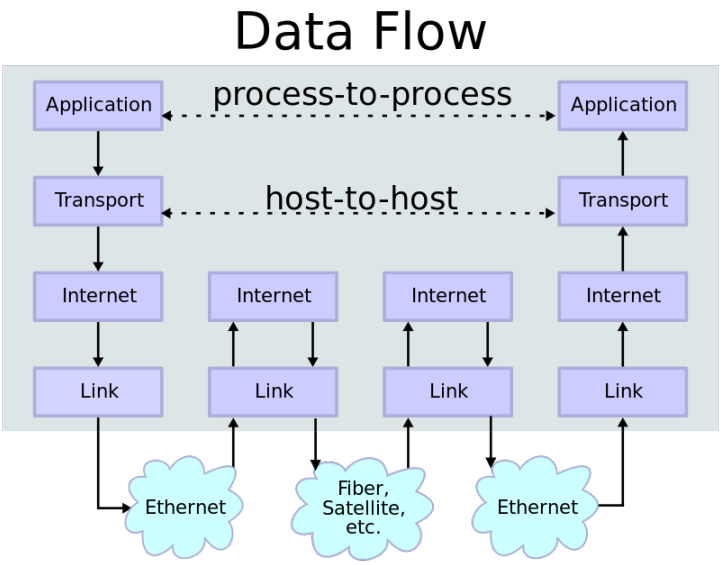


Figura 2.5: Rappresentazione del percorso effettuato dai pacchetti tra i livelli del modello TCP/IP.[6]

- *Flags*: 3 bit di controllo (di cui uno inutilizzato e lasciato al valore 0) relativi alla frammentazione. Indicano la possibilità di sfruttare la frammentazione, la presenza di ulteriori frammenti o l'indicazione che il pacchetto è l'ultimo frammento

- *Fragment Offset*: indica la posizione del pacchetto all'interno di un pacchetto originale frammentato

- *Time to Live*: tempo di vita di un pacchetto all'interno della rete Internet. Viene decrementato ad ogni passaggio da un nodo all'altro e se raggiunge il valore 0, il pacchetto viene scartato indicando che non può essere consegnato a destinazione

- *Protocol*: specifica il protocollo dello strato superiore a cui verrà consegnata la sezione dati del pacchetto

- *Header Checksum*: sequenza di bit usata per il controllo di integrità dell'intestazione

- *Source Address*: indirizzo del dispositivo mittente (si consulti la sezione 2.2.2 per dettagli)

- *Destination Address*: indirizzo del dispositivo di destinazione

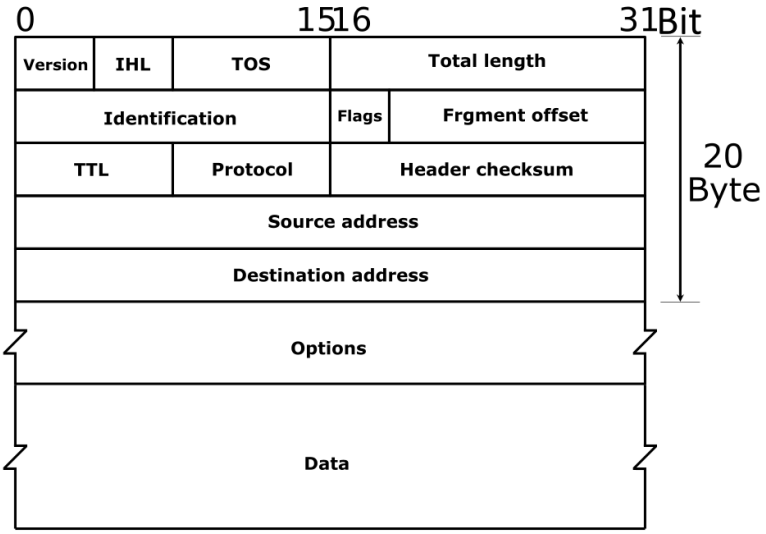


Figura 2.6: Intestazione di un pacchetto IPv4.[7]

- *Options*: campo dell'intestazione a lunghezza variabile e opzionale. Può contenere indicazioni che i router devono o possono rispettare circa i percorsi che i pacchetti dovranno effettuare oppure informazioni di diagnostica come la lista dei router che il pacchetto ha visitato e in quale momento.

2.2.2 Indirizzamento

Ogni *host* nella rete deve essere opportunamente identificato affinché possa inviare e ricevere pacchetti. IP si occupa di tale aspetto attraverso l'indirizzamento, che è una delle caratteristiche fondamentali del protocollo e costituisce una delle principali differenze con NDN.

Gli indirizzi IP sono dei numeri di lunghezza 32 bit e fanno riferimento, oltre al dispositivo, anche alla rete a cui esso appartiene. Un host che è collegato a più reti (attraverso più interfacce fisiche) sarà quindi dotato di più indirizzi IP.

Il protocollo alla sua nascita prevedeva [36] un'organizzazione a classi dello spazio di indirizzamento, ognuna contraddistinta dal numero di reti e host che poteva rappresentare. Tale suddivisione è stata ampiamente usata nel corso degli anni, tuttavia la crescita esponenziale del numero di dispositivi ha richiesto l'adozione di nuovi approcci e tecniche atte a ridurre o mascherare il numero di dispositivi univocamente identificati in Internet.

La classificazione inizialmente proposta prevedeva cinque spazi di indirizzamento diversi di cui uno riservato per utilizzi futuri. Le prime tre classi, A, B e C, sfruttano la divisione dei 32 bit in parti a multipli di 8 bit per distinguere la

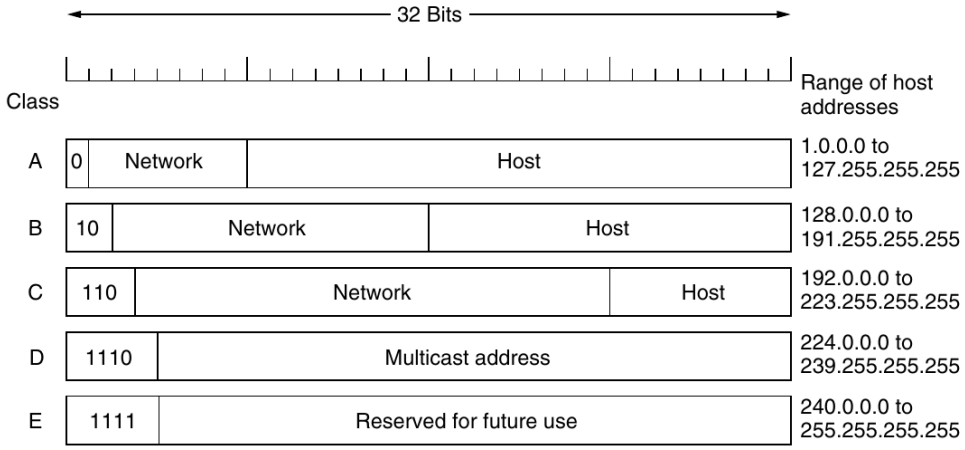


Figura 2.7: Indirizzamento basato sulle classi in IPv4.[8]

numerazione relativa alla rete e quella relativa agli host. La prima porzione del numero indica sempre la rete ed è preceduta da una sequenza di bit che specifica la classe stessa. In fig. 2.7 è fornita una rappresentazione visuale delle classi e delle loro differenze.

La classe D è dedicata al multicast, mentre la classe E è riservata per usi futuri.

Gli indirizzi vengono per semplicità rappresentati in notazione decimale puntata, ad esempio `192.168.0.1` corrisponde in binario alla sequenza `11000000 10101000 00000000 00000001`.

Le classi A, B e C prevedono [37] inoltre dei sottospazi di indirizzamento riservati e utilizzati nel meccasismo di NAT (*Network Address Translation*). Al fine di ridurre il numero di indirizzi IP utilizzati in Internet è stata infatti adottata una tecnica che permette di mascherare un'intera rete attraverso un singolo nodo di connessione ad Internet (*gateway*).

Con CIDR (*Classless InterDomain Routing*) [38] si è passati ad una gestione dell'indirizzamento priva di classi: la divisione tra sottoparte dell'indirizzo relativa alla rete e all'*host* non avviene più a multipli di 8 bit ma può essere applicata arbitrariamente in un qualsiasi punto dell'indirizzo stesso.

2.2.3 Instradamento

Il principale compito del livello di rete consiste nel consegnare i pacchetti contenenti i dati del livello superiore dal mittente al destinatario. Tale operazione è chiamata instradamento e prevede che i pacchetti vengano fatti transitare attraverso i router della rete Internet seguendo, se possibile, il percorso più breve nel rispetto dei canali congestionati e delle direttive del protocollo utilizzato.

I router sono dotati di più interfacce, il che li rende connessi a più reti: ciò permette loro di inoltrare (*forwarding*) i pacchetti sull'interfaccia di ingresso su diverse interfacce di uscita sulla base di quanto stabilito da opportuni algoritmi e protocolli di routing.

Gli algoritmi di routing [39] hanno quindi l'obiettivo di fornire, conoscendo la topologia della rete (in parte o tutta), il percorso migliore per il pacchetto da instradare. Se si pensa ad una rete come un grafo in cui i router sono i nodi e gli archi sono le linee di trasmissione è possibile ricondursi all'operazione di ricerca del cammino minimo in un grafo per definire il "percorso migliore".

Gli algoritmi di routing possono essere statici o dinamici: nel primo caso i percorsi più brevi tra tutte le coppie di nodi vengono calcolati a priori e scaricati dai router al loro avvio mentre nel routing dinamico i percorsi vengono adattati qunado la rete è già operativa sulla base delle variazioni alla topologia o sulle stime di traffico tra i collegamenti.

I due algoritmi di routing più popolari sono dinamici e prendono il nome di *distance vector* e *link state*. Il primo è stato originariamente adottato in ARPANET e consiste nella memorizzazione, in ogni router, di una tabella in cui ogni voce descrive la linea di trasmissione da utilizzare per una certa destinazione e la distanza usando una determinata metrica (ad esempio il numero di salti, detti *hop*). L'algoritmo di routing *link state* prevede invece che ogni router scopra i propri "vicini" e misuri la distanza da essi per poi distribuire tutte le informazioni raccolte a questi ultimi. Così facendo tutti i router riescono ad ottenere la topologia completa della rete e a calcolare il percorso più breve verso tutte le possibili destinazioni.

Per ragioni di scalabilità Internet è organizzata come una rete di reti interconnesse che presentano caratteristiche differenti, a partire dalle prestazioni dei router e dei collegamenti fino agli algoritmi implementati. Inoltre, ogni sotto-rete (che prende il nome di *Autonomous System*) è gestita da organizzazioni differenti, le quali possono adottare politiche diverse per le proprie infrastrutture. Nonostante ciò, gli AS devono continuare a poter comunicare tra loro, ragion per cui sono stati definiti dei protocolli che regolano il transito delle informazioni internamente ed esternamente agli *Autonomous Systems*.

OSPF (*Open Shortest Path First*)[40] è il protocollo di routing intra-AS più popolare e si basa su algoritmi *link state*. In un AS gestito da OSPF i messaggi sono autenticati per ragioni di sicurezza, è nativamente previsto il multicast e una gestione gerarchica dei router nei casi in cui gli AS siano di grandi dimensioni. Per la comunicazione tra AS è invece comunemente utilizzato BGP (*Border Gateway Protocol*) [41] che consente ai router di confine di scambiare informazioni sugli AS che "rappresentano" al fine di distribuire internamente le indicazioni sulle rotte disponibili verso le altre reti.

2.3 Attacchi in TCP/IP

I protocolli che servono i livelli del modello architetturale TCP/IP sono di fondamentale importanza per il funzionamento di Internet e ciò li rende potenziali vittime di attacchi informatici. É necessario inoltre considerare che, quando tali protocolli furono implementati, non venne concessa particolare attenzione ad aspetti di sicurezza.

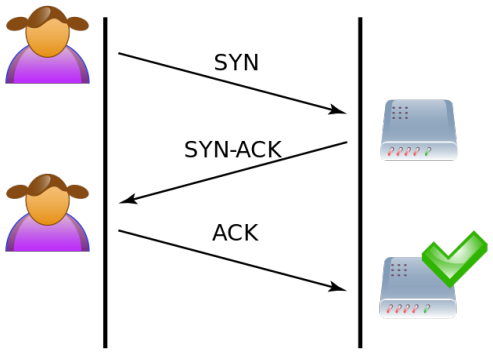
Per ciò che concerne il livello di rete, nel 1998 è stata introdotta ed adottata la piattaforma IPsec (*IP security*) con l'obiettivo di fornire diverse soluzioni di sicurezza indipendenti tra loro e dagli algoritmi (anch'essi soggetti a vulnerabilità). I servizi offerti da IPsec consistono in segretezza e integrità delle informazioni, autenticità del mittente e protezione da attacchi *replay*. L'adozione di tali misure viene principalmente effettuata mediante delle nuove intestazioni da aggiungere ai pacchetti IP.[42]

La presenza di IPsec o di integrazioni di sicurezza su altri protocolli tuttavia non protegge da ulteriori possibili attacchi che sfruttano, ad esempio, delle vulnerabilità logiche nelle connessioni o nella comunicazione tra *host*.[43, 44]

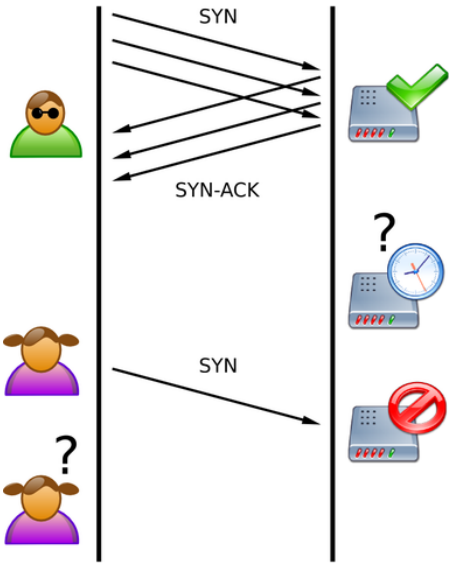
Nella sezione 1.4 sono stati citati gli attacchi DDoS come una delle principali minacce presenti in Internet. Questi vengono definiti come degli scenari in cui uno o più dispositivi tentano di impedire che l'*host* vittima svolga regolarmente il proprio compito, comportando quindi una negazione del servizio (*Denial of Service*) nei confronti degli utenti. Si parla di DDoS (*Distributed Denial of Service*) nel momento in cui l'attaccante usufruisce della potenza di calcolo e della banda disponibile di centinaia o migliaia di dispositivi già compromessi.

Le vittime possono essere dispositivi semplici, server o router: in ogni caso l'obiettivo è quello di ridurre o azzerare le risorse di rete o computazionali disponibili così da impedire che le richieste legittime non possano più essere servite.

Le tipologie di attacchi DDoS più popolari sono legate al protocollo TCP e al flusso di pacchetti scambiati durante la connessione tra mittente e destinatario. *Transmission Control Protocol* [33] è un protocollo dello strato trasporto che si occupa di instaurare la connessione e gestire una comunicazione affidabile tra gli applicativi degli *host*. L'affidabilità del protocollo è da intendersi come l'applicazione di metodi risolutivi in caso di errori al livello sottostante, come la ritrasmissione di pacchetti non consegnati o riordinare la sequenza dei datagrammi ricevuti. La connessione effettuata in TCP prevede che i due interlocutori scambino dei messaggi di controllo prima (e dopo) della reale trasmissione delle informazioni (processo di *handshake*, ovvero "stretta di mano"). In tali messaggi vengono utilizzati dei bit specifici dell'intestazione del pacchetto TCP per instaurare e chiudere la connessione (rispettivamente *SYN* e *FIN*) o per confermare la ricezione dei pacchetti stessi (*ACK*). Le connessioni istanziate vengono registrate dai server e richiedono quindi un'occupazione di memoria che può essere massimizzata abusando di questo



(a) Instaurazione di una regolare connessione TCP mediante *handshake* a tre vie.[47]



(b) Esecuzione dell'attacco *SYN flood*.[48]

Figura 2.8: Differenze nell'invio di pacchetti TCP tra l'instaurazione della connessione mediante *handshake* e l'attacco *SYN flood*.

fenomeno. L'attacco *SYN-flooding* va proprio in questa direzione: l'instaurazione della connessione prevede uno scambio di messaggi come quello in fig. 2.8a quando in realtà l'attaccante lascia la connessione aperta "per metà", non rispondendo con il pacchetto contenente l'*ACK* (fig. 2.8b).[45, 46]

Il fenomeno può assumere dimensioni ben maggiori se l'attaccante dispone di un numero elevato di macchine già compromesse che possono contattare direttamente la vittima o coinvolgere altri dispositivi mediante la tecnica di amplificazione. Nel primo caso i pacchetti inviati alle macchine compromesse avranno nel campo *Source Address* un indirizzo casuale (così da non poter risalire all'attaccante) e come *Destination Address* l'IP della vittima.

Nel secondo scenario si parla di attacco amplificato [49] e avviene in due fasi. Inizialmente viene inviato un pacchetto ai dispositivi compromessi con *Source Address* l'indirizzo della vittima (tecnica di *IP spoofing*) e come IP destinazione l'indirizzo dei riflettori. A queste ultime i *bot* invieranno pacchetto con mittente casuale e come *Destination Address* l'indirizzo della vittima, portando così a compimento l'attacco.[46] In fig. 2.9 viene rappresentato in modo semplificato un attacco amplificato.

Un ulteriore tipo di attacco DoS può essere realizzato nei confronti di un intero AS: è possibile infatti, mediante dei router compromessi e l'utilizzo del protocollo BGP, informare l'AS vittima dell'esistenza di route più vantaggiose per una specifica destinazione (*black-holing* mediante *prefix hijacking*). Così facendo l'at-

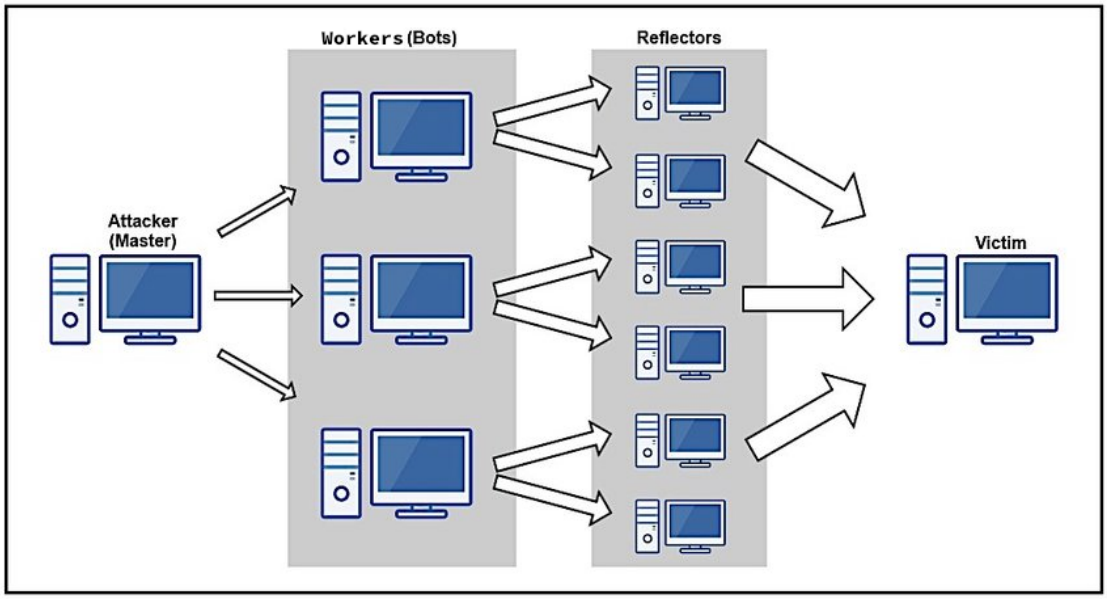


Figura 2.9: Rappresentazione di un attacco amplificato.[9]

taccante è in grado di dirottare il traffico dell'AS vittima verso il proprio AS malevolo al fine di intercettarlo o scartarlo, causando così un'interruzione del servizio (fig. 2.10).[46, 50]

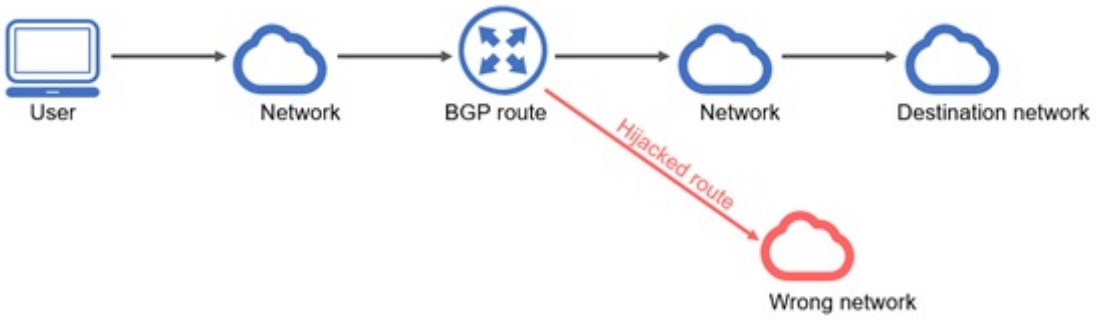


Figura 2.10: Rappresentazione dell'attacco di BGP hijacking.[10]

Capitolo 3

Named Data Networking

L'evoluzione di Internet in termini di nuovi servizi proposti agli utenti è stata discussa nel capitolo 1. Allo stesso modo è stato evidenziato il cambio di paradigma da una rete di comunicazione ad una rete di distribuzione di contenuti, opportunamente teorizzata e formalizzata da Van Jacobson e i gruppi di ricerca operanti nel contesto dell'*Information-Centric Networking* e *Future Internet Architectures*.

L'attuale rete Internet si basa su un modello di comunicazione che trova spazio nello strato di rete dell'architettura TCP/IP. Il protocollo fondamentale in questo contesto è *Internet Protocol* che, come analizzato nel capitolo 2, implementa le funzionalità essenziali per la consegna di pacchetti in un insieme di reti globalmente interconnesse. L'unità fondamentale di tale protocollo, il pacchetto IP, registra nell'intestazione gli indirizzi di mittente e destinatario e ulteriori informazioni di controllo.

Ciò che propone il gruppo di lavoro del progetto NDN [12] è un nuovo modello basato su una rete di distribuzione in cui viene data primaria importanza ai contenuti e alla loro identificazione nella rete (fig. 3.1). Lo strato *network*, quindi, non si occuperà più di consegnare i pacchetti a destinazione, ma permetterà ai dispositivi di richiedere dati identificati dai Nomi. Proprio su queste ultime risorse avviene il processo di generalizzazione rispetto a IP in quanto i Nomi possono identificare una qualsiasi informazione nella rete: un *host*, una porzione di video, un comando di accensione di una lampadina e così via.

In questo capitolo viene posta l'attenzione agli aspetti strutturali di NDN, tra cui il formato dei pacchetti, i Nomi, la presenza di una *cache* nell'infrastruttura di rete, l'instradamento e la sicurezza fornita nativamente.

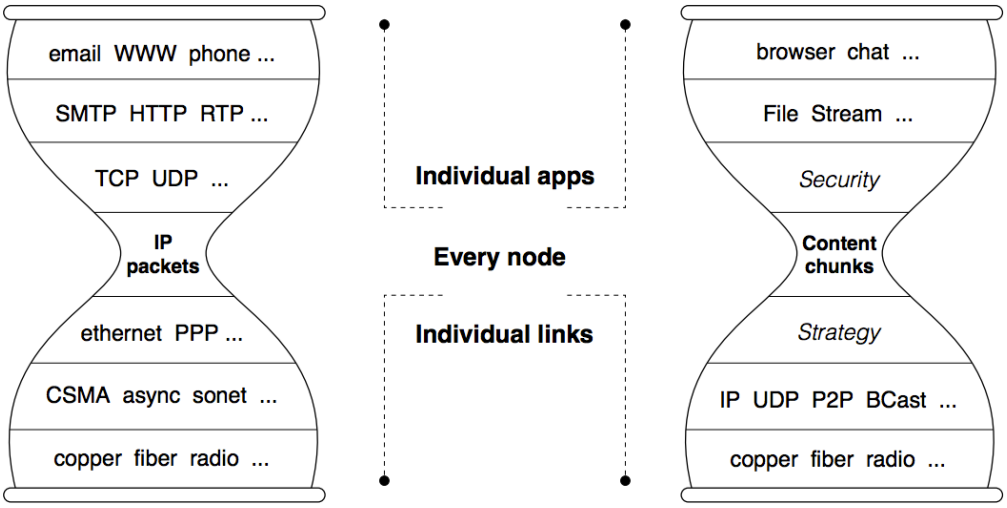


Figura 3.1: Confronto nella rappresentazione a clessidra tra l'architettura Internet attuale e NDN.[11]

3.1 Formato dei pacchetti

In una comunicazione NDN intervengono due attori principali: chi effettua la richiesta (*Consumer*) e chi serve le risorse agli utenti (*Producer*).

I *Consumer*, quindi, effettuano le richieste attraverso un pacchetto chiamato *Interest* che verrà inoltrato dai router verso il *Producer*. Quest'ultimo invierà le risposte in un pacchetto di nome *Data*.

Interest e Data, al contrario del protocollo IP, non fanno più riferimento agli indirizzi degli *host* a cui vengono effettuate le richieste, piuttosto specificano direttamente il Nome della risorsa.

Nelle successive sezioni verranno mostrate le modalità di instradamento dei pacchetti e la loro gestione da parte dei router.

Di seguito viene descritto il formato dei pacchetti secondo la specifica nella versione `0.3` rilasciata dai progettisti di NDN. In fig. 3.2 è presente una rappresentazione dei pacchetti *Interest* e *Data*.

3.1.1 Codifica Type-Length-Value

É opportuno osservare che i pacchetti in NDN non prevedono più, come accade in IP, un'intestazione che rispetti una struttura rigida: è stata infatti adottata la codifica TLV (*Type-Length-Value*) per una composizione flessibile delle opzioni dei pacchetti in grado di garantire il contenimento della dimensione degli stessi e un minore impatto prestazionale nella loro elaborazione.

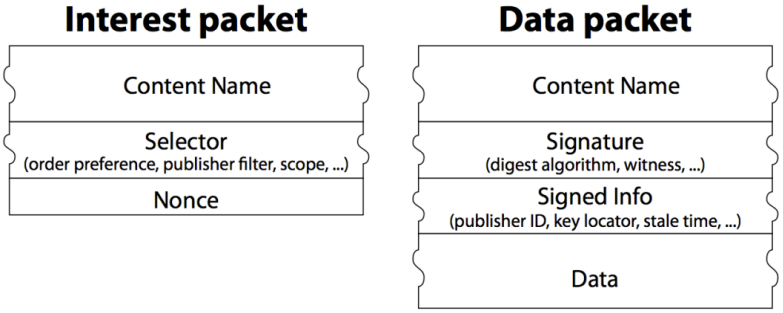


Figura 3.2: Struttura dei pacchetti NDN (precedente alla versione 0.3).[12]

TLV è una codifica composta di tre elementi fondamentali: Tipo (*Type*), Lunghezza (*Length*) e Valore (*Value*). *Type* consiste in un codice numerico che specifica il tipo di informazione contenuta nel campo del pacchetto, *Length* ne indica la lunghezza in byte mentre *Value* racchiude il vero e proprio contenuto informativo.

La codifica permette l'annidamento dei dati tra le stesse triplette di TLV, permettendo quindi una specifica a più livelli delle informazioni. In NDN il livello più esterno, indicato come TLV_0, è usato per distinguere il tipo di pacchetto, per un Interest il campo *Type* è posto a `5` mentre per Data viene usato il valore `6`.

3.1.2 Interest

Secondo la specifica TLV un pacchetto *Interest* è quindi formato dal tipo, dalla dimensione in byte dei dati e da una serie di campi innestati di cui il Nome è l'unico obbligatorio.[51]

Seguono quindi una serie di TLV opzionali utili all'instradamento, alla sicurezza o alle applicazioni:

- `CanBePrefix`: specifica che il Nome può riferirsi ad un prefisso di una risorsa e non al suo identificativo completo
- `MustBeFresh`: se presente, si richiede che il dato mantenuto in cache dai router, seppur ancora valido, venga comunque richiesto al Producer in quanto può aver generato nuovi dati
- `ForwardingHint`: specifica il percorso che l'*Interest* dovrebbe seguire
- `Nonce`: sequenza di caratteri casuale lunga 4 byte usata per identificare univocamente l'*Interest* con uno specifico Nome al fine di identificare cicli di instradamento dello stesso pacchetto; campo obbligatorio se l'*Interest* è instradato su collegamenti di rete

- `InterestLifetime`: tempo di vita dell'*Interest* specificato in millisecondi dalle applicazioni

- `HopLimit`: numero di salti (*hop*) che l'*Interest* può compiere durante l'instradamento. Ha dimensione 1 byte (massimo 255 salti), il suo valore viene decrementato di 1 ad ogni salto: se durante questa operazione `HopLimit` vale 0 allora l'*Interest* può essere instradato solamente alle interfacce locali (cache o applicativi)

- `ApplicationParameters`: parametri che il *Producer* userà per la generazione del Dato accompagnati dalla firma del pacchetto e dalle informazioni sull'algoritmo utilizzato (`InterestSignature`)

3.1.3 Data

I pacchetti *Data* consistono nelle risposte dei *Producer* ai relativi *Interest* inviati dai *Consumer* e si compongono quindi del contenuto informativo relativo al Nome espresso nella richiesta.

La componente *Value* della codifica TLV per un pacchetto *Data*, oltre al Nome, specifica inoltre [52]:

- `MetaInfo`: un campo opzionale in cui viene innestato un ulteriore livello di strutture TLV

  - `ContentType`: specifica il tipo di contenuto trasportato nel pacchetto *Data*. I principali contenuti supportati sono `BLOB` ossia l'informazione identificata dal Nome, `LINK` ossia una coppia di Nome e relativa priorità (chiamata delegato), `KEY` che indica una chiave pubblica e `NACK` che consiste in un NACK a livello applicativo
  - `FreshnessPeriod`: indicazione temporale di longevità del contenuto informativo. Allo scadere del tempo, l'informazione, pur essendo ancora valida, può essere stata aggiornata dal *Producer*. Il valore di `FreshnessPeriod` è legato all'indicazione `MustBeFresh` espressa nei pacchetti *Interest*
  - `FinalBlockId`: specifica qual è la parte finale di un Nome in una sequenza di frammenti

- `Content`: contenuto informativo espresso come sequenza di byte, opzionale

- `DataSignature`: campo obbligatorio che specifica l'algoritmo e la firma stessa usata per fornire autenticità

3.2 Indirizzamento: i Nomi

I principi fondamentali del protocollo prevedono che i Nomi siano l'unità centrale nella comunicazione tra *Consumer* e *Producer*.

I Nomi identificano le risorse ottenibili dalla rete e consistono in delle sequenze di caratteri divise in blocchi (detti componenti) dal separatore `/` (similmente agli URI [53]). Ogni blocco può far parte del prefisso di un Nome (che raggruppa logicamente più contenuti) oppure costituisce una porzione di identificativo della risorsa (*content identifier*). Lo spazio dei Nomi è gerarchico in quanto ogni blocco delimita un livello: ciò permette di distinguere il contesto di riferimento per le risorse (ad esempio aggregando le pubblicazioni di un autore sotto lo stesso Nome) e abilita l'uso di politiche di sicurezza mirate (come la possibilità di specificare in modo strutturato quali dati è in grado di firmare una chiave).

I router riconoscono i confini tra i componenti di un Nome, tuttavia ne ignorano il significato, ciò implica che ogni applicazione può adottare una propria struttura personalizzata indipendente dalla rete. Tale possibilità consente ai *Consumer* di effettuare richieste specificando il Nome incompleto di una risorsa (poiché non conosciuto a priori) ed essere informato dal *Producer* sulla convenzione utilizzata (principio di *In-Network Name Discovery*).

Ad esempio un *Consumer* può richiedere la prima versione pubblicata di un filmato ad un *Producer* con un *Interest* che avrà come campo `Name` il valore `/organization/videos/demo.mpg/1`. Il *Producer* risponderà, se possibile, con un pacchetto *Data* con Nome `/organization/videos/demo.mpg/1/1` evidenziando che i dati inviati corrispondono alla prima parte della versione `1` del video richiesto. In questo modo il *Consumer* avrà *scoperto* la convenzione usata per quel Nome e potrà costruire richieste coerenti per l'ottenimento di tutte le porzioni del video.

Come avviene per i pacchetti *Interest* e *Data*, anche i Nomi seguono la codifica *Type-Length-Value* [54]. La struttura di un Nome è caratterizzata dalla presenza di più livelli TLV contenuti nel campo `Value` (chiamato `NameComponent`):

- `GenericNameComponent`: rappresentazione testuale a lunghezza variabile e senza vincoli sul contenuto ospitato relativa al dato
- `ImplicitSha256DigestComponent`: impronta dell'algoritmo di *hashing* SHA-256 relativa all'intero pacchetto *Data*
- `ParametersSha256DigestComponent`: impronta dell'algoritmo di *hashing* SHA-256 relativa al TLV `ApplicationParameters` contenuto in un pacchetto *Interest*

- `OtherTypeComponent`: TLV relativo ad ulteriori componenti che è possibile aggiungere al Nome, come un numero di segmento, un *offset* in byte, la versione del contenuto, il numero di sequenza o il *timestamp*

In riferimento al TLV `OtherTypeComponent`, i ricercatori del progetto NDN hanno rilasciato in [55] una serie di linee guida sulle convenzioni che le applicazioni possono adottare per una migliore gestione nella distribuzione dei contenuti. Con il campo `SequenceNumNameComponent` è possibile organizzare Nomi che fanno riferimento ad una collezione di dati sequenziale opportunamente indicizzata con numeri incrementali non negativi. Un'applicazione di video *streaming* o di *real time conference* potrebbe beneficiare della suddivisione del flusso dati in più Nomi in cui ogni porzione identifica un segmento e viene marcata con un numero di sequenza (`SegmentNameComponent`) o con il suo *offset* in byte dall'inizio (`ByteOffsetNameComponent`). Può inoltre essere necessario rendere evidente la presenza di nuove versioni o della data di creazione del contenuto messo a disposizione dai *Producer*: a tal proposito viene raccomandato l'uso dei campi `VersionNameComponent` e `TimestampNameComponent`. Ciò può rendere efficiente la memorizzazione dei soli nuovi contenuti nelle memorie *cache* implementate nei router e descritte nella sezione 3.3.

3.3 Instradamento e inoltro

Affinché gli *Interest* di un *Consumer* vengano consegnati al rispettivo *Producer* al fine di ottenere una risposta relativa ad un dato Nome è necessario che i router applichino delle politiche di *forwarding* e rispettino, come per IP, dei protocolli di *routing*.

In NDN i router adottano delle strutture dati e delle strategie specifiche per gestire il processo di inoltro:

- *Pending Interest Table (PIT)*: memorizza le interfacce da cui sono stati ricevuti, per uno stesso Nome, degli *Interest* ancora insoddisfatti

- *Content Store (CS)*: archivio con funzionalità di *caching* per i pacchetti *Data* consegnati ai *Consumer*

- *Forwarding Information Base (FIB)*: registro delle interfacce tramite le quali è possibile raggiungere un *Producer* per uno specifico prefisso di Nome

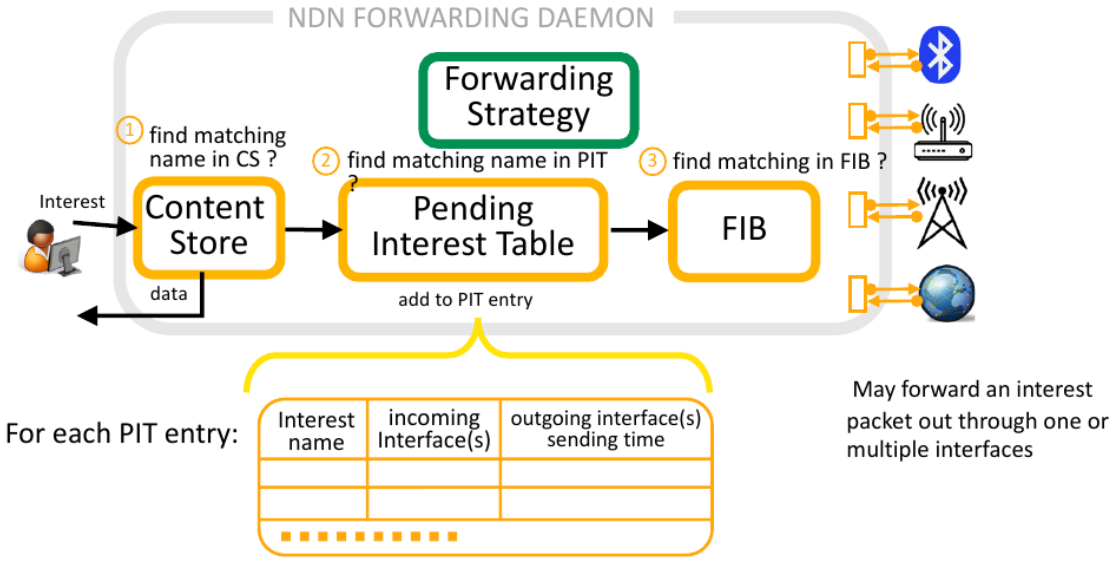


Figura 3.3: Passi e strutture dati presenti in un router per l'invio di un Interest in NDN.[13]

3.3.1 Invio e ricezione di pacchetti

L'operazione di *forwarding* degli *Interest* e la consegna dei *Data* coinvolge tutte e tre le strutture dati, in fig. 3.3 e 3.4 è possibile osservarne una rappresentazione schematizzata.

Alla ricezione di *Interest* i router inizialmente verificano che in *cache* (*Content Store*) sia presente un pacchetto *Data* utile a soddisfare la richiesta, in tal caso quest'ultimo viene direttamente inviato all'interfaccia di origine.

In caso contrario il router verifica che la richiesta sia presente nella *Pending Interest Table* con l'obiettivo di registrare unicamente l'interfaccia di provenienza nel caso in cui un *Interest* per lo stesso Nome sia già stato inoltrato.

Se la richiesta è "nuova" o se i precedenti *Interest* sono stati scartati (ad esempio per *timeout*) il router verifica che nella *Forwarding Information Base* siano presenti interfacce verso le quali è possibile inoltrare l'*Interest*: tale operazione viene effettuata mediante *longest prefix matching*, ovvero viene cercata la porzione di Nome più lunga che coincida con quanto memorizzato nella FIB al fine di inoltrare il pacchetto verso l'interfaccia più appropriata.

Nei router interviene un ulteriore componente chiamato *forwarding strategy* che influenza il processo di inoltro. Il suo compito è infatti quello di decretare se l'*Interest* verrà effettivamente inviato al prossimo nodo (che sia un altro router o il *Producer*) sulla base dello stato della rete, come la presenza di congestione o il sospetto di un attacco DoS. Una volta ricevuta indicazione dal componente FIB

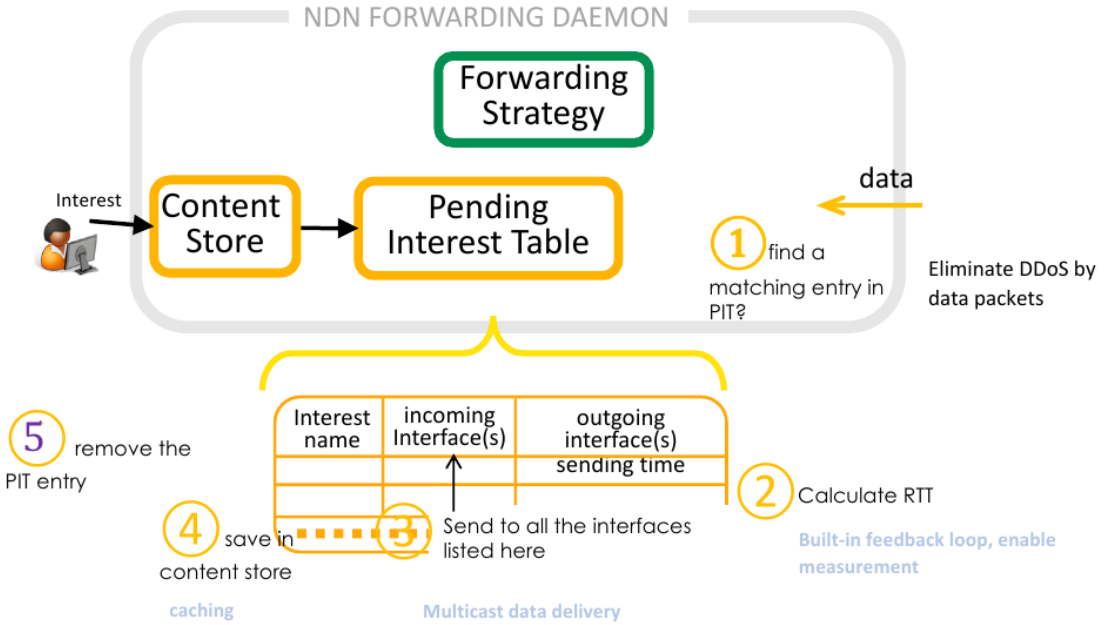


Figura 3.4: Passi e strutture dati presenti in un router alla ricezione di un Data in NDN.[13]

l'algoritmo individua un'opportuna strategia di inoltro sulla base di fattori come la presenza di *Interest* non soddisfatti nella PIT, la priorità delle richieste o il carico dei collegamenti che portano al *Producer*.

Ricevuto l'*Interest*, il *Producer* si occuperà dell'invio della risposta in un pacchetto *Data* relativamente al Nome espresso nella richiesta. Il percorso effettuato durante la consegna dell'*Interest* sarà lo stesso per *Data*, ma in direzione opposta, e ad ogni passaggio i router elimineranno la voce nella PIT, scarteranno il pacchetto se l'*Interest* è scaduto oppure lo inseriranno in CS per poi inoltrarlo verso le interfacce di cui si era tenuta traccia in precedenza, fino ad arrivare ai *Consumer* (fig. 3.4).[11]

L'adozione di tali meccanismi permette di effettuare delle osservazioni sul flusso del traffico interno ai router e nella rete. É evidente che viene minimizzato il numero di pacchetti inoltrati da un router all'altro: la presenza di CS e PIT permette di evitare che le richieste relative ad uno stesso Nome vengano instradate in quanto servite dalla *cache* o tutte in uno alla ricezione del *Data*.

3.3.2 Protocolli di routing

Per ciò che concerne i protocolli di *routing* è evidente che le soluzioni esistenti non siano compatibili con la nuova architettura proposta. Nonostante ciò gli algoritmi

che ne descrivono il comportamento hanno permesso ai ricercatori di ideare e realizzare dei protocolli con meccanismi simili. Inizialmente è stato infatti adattato [56] il protocollo OSPF realizzando una comunicazione tra router vicini che scambiano informazioni non più sui prefissi di indirizzi IP raggiungibili ma sui prefissi di Nomi che il router serve e che tiene memorizzati nella sua FIB.

Successivamente è stato sviluppato e adottato un nuovo protocollo chiamato NLSR (*Named-data Link State Routing protocol*) [57] basato sull'uso di *Interest* da parte dei router per richidere aggiornamenti sullo stato dei collegamenti.

I protocolli di *routing*, per la loro centralità nella gestione delle reti, costituiscono un ambito tutt'ora in esplorazione da parte della comunità di ricercatori che pone come obiettivo lo sviluppo di soluzioni che sfruttino nativamente le novità introdotte da NDN e che siano efficienti e sicure. [58, 59]

3.4 Security-by-design

Uno dei principi fondamentali su cui si basa NDN consiste nell'implementazione della sicurezza direttamente nei pacchetti, in particolare firmando i Dati (*Securing Data Directly*). Ciò permette da un lato di rimuovere la necessità di instaurare canali di comunicazione sicuri *end-to-end* come avviene su IP e dall'altro abilita la fruizione dei contenuti in modo sicuro e asincrono. La firma dei pacchetti *Data*, infatti, continua ad essere valida sia che essi siano in transito sia che vengano memorizzati nelle *cache* dei router.

Il processo di firma e verifica dei pacchetti deve essere necessariamente automatizzato per integrarsi con efficacia nel protocollo: i ricercatori si sono mossi in questo senso con la definizione del *NDN security framework* [60] ovvero un insieme di elementi e operazioni atti a fornire i meccanismi di sicurezza fondamentali per la comunicazione in rete.

Il *framework* è basato sulla crittografia a chiave pubblica [61] e prevede che ogni partecipante alla comunicazione (chiamato *entità*) sia dotato di un certificato (detto *identità*), ovvero un'associazione di chiavi al Nome.

I componenti di base del *framework* consistono in:

- chiavi: trattate come Nomi veri e propri, quindi ottenibili in pacchetti *Data* mediante degli *Interest*
- certificati: pacchetti *Data* che abbinano al Nome una chiave pubblica
- *trust policies*: in un contesto applicativo forniscono indicazioni sulle chiavi da utilizzare per firmare e verificare i dati

I tre elementi descritti permettono al *framework* di rispettare determinati obiettivi, tra cui:

- *security bootstrapping*: processo in cui un'entità viene informata sui *trust anchor*, sulle *trust policies* esistenti e riceve i certificati da utilizzare
- fornire autenticità: processo di verifica da parte dei *Consumer* delle firme sui Dati applicate dai *Producer*
- rendere confidenziali le comunicazioni mediante algoritmi crittografici
- garantire disponibilità dei dati e dei certificati: ciò è facilitato dalla presenza delle *cache* nei router

L'introduzione al *framework* adottato non costituisce tuttavia una trattazione completa degli aspetti di sicurezza che possono coinvolgere NDN e le relative implementazioni. Fornire meccanismi come quelli appena descritti, inoltre, non rende l'architettura esente da attacchi che possono minare la privacy degli utenti [62] o la rete stessa, come analizzato con gli attacchi DDoS nel capitolo 4.

Capitolo 4

Attacchi DDoS in NDN

Nel capitolo 1 è stata fornita una panoramica sulle minacce che possono coinvolgere le aziende o i singoli utenti in Internet nonché le loro implicazioni, come il furto di identità o la chiusura di intere catene produttive a fronte delle perdite economiche che ne derivano. É quindi di fondamentale importanza fornire adeguati ed efficaci meccanismi di protezione a supporto dei sistemi di comunicazione a partire dai livelli più bassi dell'architettura protocollare. Nel capitolo 3 è stato introdotto il *framework* di sicurezza che NDN implementa per fornire elementi basilari di protezione dei contenuti. Tuttavia è emerso che sono necessarie ulteriori misure di difesa nei confronti della rete e degli attori coinvolti nella comunicazione (*Consumer*, *Producer* e router). É fondamentale che, a fronte di un'accurata analisi e attività di ricerca, gli strumenti più adeguati per la mitigazione degli attacchi e la protezione degli utenti vengano integrati nativamente nell'architettura proposta e non più aggiunti in un secondo momento.

Come evidenziato nel capitolo 1, gli attacchi DDoS costituiscono ancora oggi un serio problema per la rete e per i fornitori di servizi in Internet. Le previsioni mostrano che il fenomeno è in crescita, in particolar modo a causa dei ridotti sforzi da impiegare per portare a termine un attacco di ampia scala che può richiedere notevoli risorse per essere mitigato.

A causa del radicale cambio di paradigma che NDN adotta rispetto ad IP è intuitivo supporre che anche gli attacchi DDoS mostrati nel capitolo 2 potrebbero mostrare delle differenze tra un protocollo e l'altro nel modo in cui vengono portati a termine. In tal senso in [63] vengono mostrati due nuovi attacchi che portano alla negazione del servizio in una rete NDN: IFA (*Interest Flooding Attack*) e *cache poisoning* (non approfondito in questo elaborato). Queste nuove minacce coinvolgono alcune strutture dati implementate nei router e descritte nel capitolo 3: PIT consiste in una tabella in cui vengono registrati gli *Interest* ancora insoddisfatti mentre in Content Store risiedono i pacchetti *Data* instradati verso i *Consumer* che li hanno richiesti.

Nell'articolo precedentemente citato vengono brevemente descritti i motivi per cui i principali attacchi DDoS mostrati nel capitolo 2 che coinvolgono IP non possono funzionare in NDN. L'attacco di *flooding* dei pacchetti non è efficace in NDN in quanto la moltitudine di richieste verrà soddisfatta una sola volta dal *Producer*: i rimanenti *Interest* potrebbero collassare nella PIT o comunque ottenere il relativo *Data* dalla *cache* dei router. Anche un attacco amplificato non è più possibile in quanto non essendovi in NDN una indicazione di indirizzo sorgente e destinazione, l'attaccante non sarà in grado di inviare le richieste dalle macchine compromesse ai riflettori e infine alla vittima. In ultimo, l'attacco di *black-holing* non può essere portato a compimento poiché i pacchetti che contengono le informazioni di controllo per gli aggiornamenti delle tabelle di routing sono firmati (così come, del resto, tutti i pacchetti *Data* in NDN).

Nel capitolo 5 viene proposta una valutazione sperimentale di quanto teorizzato in [63] circa l'inefficacia dell'attacco di *flooding* in NDN così come concepito in IP.

IFA consiste nell'invio di *Interest* che non possono essere soddisfatti dai *Producer* così da poter permanere il maggior tempo possibile nelle PIT dei router al fine di poter esaurire le risorse degli stessi e non consentire alle richieste legittime di poter essere soddisfatte. Le caratteristiche di IFA, le sue varianti e i sistemi di mitigazione proposti verranno analizzati nelle successive sezioni.

4.1 Interest Flooding Attack

Gli *Interest* che i *Consumer* inviano in una comunicazione NDN possono essere relativi a Nomi statici, generati dinamicamente o inesistenti.

Il primo caso è riconducibile a quanto riportato in precedenza in merito agli attacchi *flooding* in IP e pertanto, a causa della presenza delle *cache* nei router, non può essere una via praticabile per attacchi a reti NDN.

Inviare pacchetti con contenuto dinamico può avere effetti significativi in quanto i benefici delle *cache* vengono annullati, tutti gli *Interest* vengono inoltrati al *Producer* che dovrà impiegare risorse computazionali per produrre i dati e firmarli. Per un attacco di questo tipo è però necessario essere a conoscenza a priori dei contenuti dinamici che il *Producer* può servire nonché la *naming convention* adottata dall'applicazione.

L'attacco più efficiente che può essere effettuato consiste nell'inviare *Interest* che non possono essere soddisfatti in quanto relativi a Nomi inesistenti. Anche in questo caso vengono annullati i benefici del componente *Content Store* andando tuttavia ad impegnare le risorse dei router anziché quelle dei *Producer*.

In un IFA di questo tipo, infatti, i *Producer* possono liberamente ignorare le richieste scartando gli *Interest* poiché contenenti Nomi associati a risorse non esistenti. L'attaccante è però in grado di esaurire la memoria disponibile dei router,

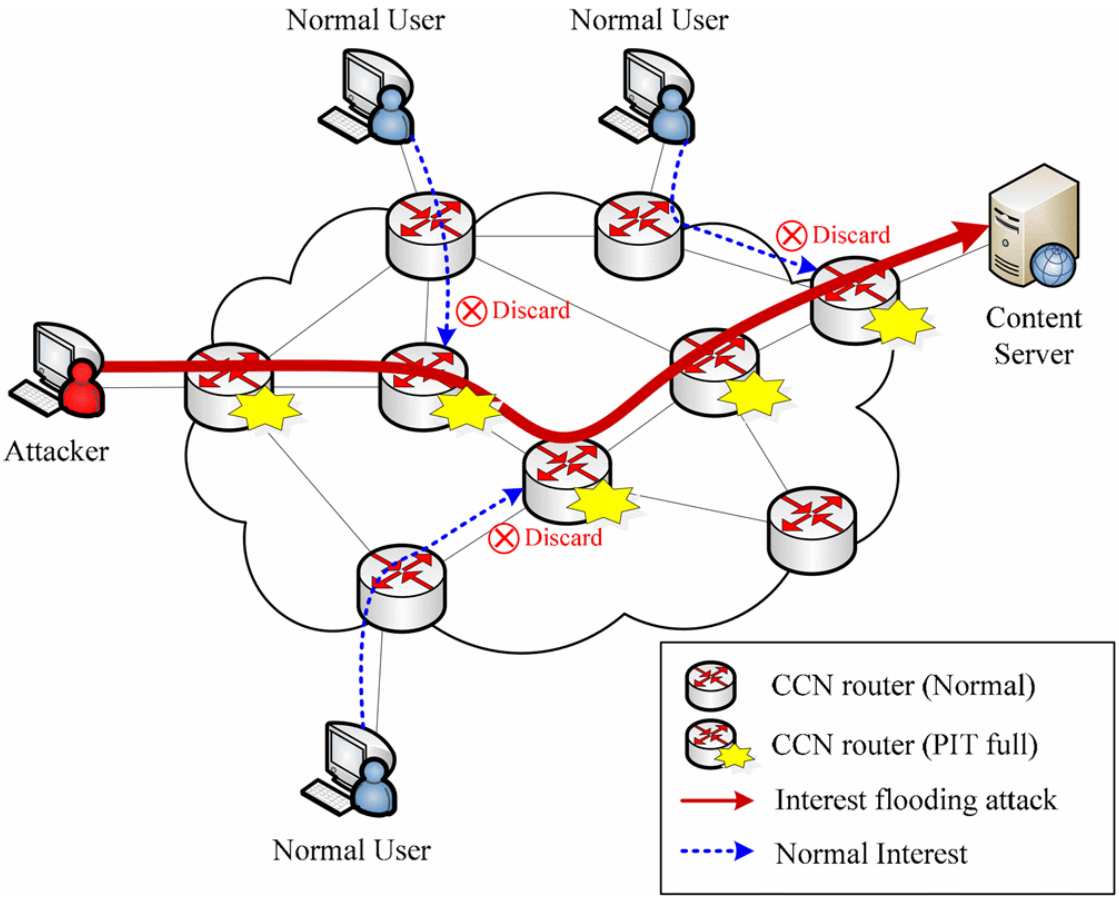


Figura 4.1: Rappresentazione di un Interest Flooding Attack.[14]

in particolare riempiendo la PIT con *Interest* la cui permanenza sarà unicamente vincolata al tempo di scadenza configurato nei router. In questo modo le richieste di *Consumer* legittimi non potranno essere instradate ai *Producer* fin tanto che i router non scarteranno i pacchetti dell'attaccante (fig. 4.1).

La generazione di *Interest* invalidi per un *Producer* può essere effettuata in modo triviale da parte di un attaccante. É infatti sufficiente usare un valore casuale come componente di un Nome servito dal *Producer*. Per fare un esempio sia `/prefix` un prefisso di Nome servito da un *Producer*. Una qualsiasi sequenza casuale di caratteri sarà valida per la creazione di un *Interest* malevolo con un Nome `/prefix/<random_sequence>`. Poiché i router non si interrogano sul contenuto delle richieste, tutti gli *Interest* con un prefisso valido (ovvero che abbiano una corrispondenza con un'interfaccia di uscita nella FIB) verranno instradati da un router all'altro verso il *Producer*. Inoltre, la componente casuale del Nome garantisce che gli *Interest* generati siano univoci: ciò implica che tali pacchetti non

possono essere collassati in un'unica voce nella PIT di un router, rendendo di fatto possibile l'attacco.

Sono inoltre possibili degli scenari che presentano tecniche più avanzate pur mantenendo la stessa direzione in termini di obiettivi che l'attaccante persegue. Sono state infatti proposte delle varianti di IFA in cui gli *Interest*, pur rimanendo invalidi per il *Producer*, contengono Nomi strettamente legati al contesto in cui lavora l'applicazione. In [64] viene definito uno scenario in cui l'attaccante costruisce *Interest* con Nomi falsi ma riconducibili a contenuti esistenti ed effettivamente serviti dal *Producer*. Nell'esempio viene proposta un'implementazione di Wikipedia in NDN i cui contenuti sono disponibili pubblicamente. L'attaccante può quindi servirsi di tale informazione per generare Nomi di pagine inesistenti sulla base di quelle servite da Wikipedia senza dover utilizzare componenti casuali. L'obiettivo di tale comportamento risiede nel rendere l'attacco più efficace in quanto poiché più complesso da individuare. Un meccanismo di riconoscimento di sequenze casuali può infatti essere implementato nei router per identificare gli *Interest* malevoli.

Ulteriori tecniche avanzate di IFA sono descritte nello stesso articolo. La prima variante è chiamata *blended IFA* e consiste nell'invio, da parte dell'attaccante, di *Interest* legittimi combinati ai malevoli. In *chameleonic IFA* invece l'attaccante cambia il prefisso del Nome coinvolto ad intervalli di tempo specifici.

In questo modo verranno influenzate le metriche registrate dai router per fare in modo che i valori soglia dei sistemi di identificazione vengano raggiunti tardivamente al fine di ridurre la probabilità che i pacchetti vengano scartati. In entrambi i casi l'obiettivo è quello di evitare che i sistemi di mitigazione entrino in azione fermando l'attacco.

4.2 Rilevamento dell'attacco

Affinché gli attacchi DDoS vengano correttamente mitigati è necessario che i router siano in grado di identificare i comportamenti anomali dei *Consumer* al fine di attivare quanto prima degli adeguati sistemi di difesa.

Il processo di inoltro degli *Interest* effettuato dai router permette agli stessi di registrare lo stato dei pacchetti mediante le strutture dati implementate. Tale operazione consente a sua volta di raccogliere una serie di dati che possono essere utilizzati ai fini di un attivo monitoraggio sulla salute della rete, che può influenzare la strategia di inoltro all'arrivo di nuovi pacchetti.

Le informazioni registrate, opportunamente interpretate, permettono di identificare un attacco DDoS in corso se i valori utilizzati superano determinate soglie.

I sistemi di mitigazione degli attacchi DDoS in NDN, a partire da quelli più semplici e basilari, vengono azionati sulla base di due metriche principali: *Interest Satisfaction Ratio* e la percentuale di utilizzo della PIT.

Nel primo caso, vengono rapportati il numero di *Interest* scartati per *timeout* e di quelli per cui invece è stato ricevuto un pacchetto *Data*.

La misura può essere raffinata tenendo conto dei valori appena descritti circoscrivendoli ad un'interfaccia o ad un particolare prefisso di Nome. In [65], infatti, viene inizialmente monitorato l'*Interest Satisfaction Ratio* per ogni interfaccia e solo se viene superata una soglia di allarme per numero di *Interest* malevoli si passa ad un'analisi più accurata per identificare esattamente qual è il prefisso coinvolto nell'attacco.

Un'operazione simile viene effettuata nel metodo di identificazione basato sullo stato della PIT. Poiché quest'ultima è la struttura dati maggiormente coinvolta in un attacco è più che ragionevole osservarne periodicamente l'occupazione. In questo caso il valore soglia viene fissato sulla base del rapporto tra il numero di *Interest* contenuti nella PIT ad un istante di tempo e il numero totale di *Interest* che può ospitare.

4.3 Mitigazione per IFA

La prima operazione che un router può effettuare non appena viene identificato un IFA è liberare le risorse occupate dai pacchetti malevoli scartandoli. Ciò tuttavia non limita in alcun modo l'attacco in quanto la stessa condizione può ripresentarsi in un istante di tempo immediatamente successivo oppure possono essere coinvolti altri router.

É quindi necessario che un router sotto attacco disponga di meccanismi di intervento immediati che possano mitigare l'azione malevola dell'attaccante e garantire piena operatività ai *Consumer* che effettuano richieste legittime.

I router possono adottare due comportamenti principali volti a contrastare un attacco IFA: in primo luogo, una volta identificata l'interfaccia da cui provengono gli *Interest* malevoli, ne limitano la banda a disposizione, inoltre possono ricorrere a sistemi collaborativi per rendere noto l'attacco ai router vicini.

I meccanismi di mitigazione presentati di seguito sono basati sulle statistiche raccolte dai router precedentemente mostrate, ovvero *Interest Satisfaction Ratio* (ISR) e percentuale di occupazione della PIT.

Satisfaction-Based Pushback [15] è unicamente basato sulla metrica ISR, il cui valore viene moltiplicato con un fattore di limite e inviato all'interfaccia da cui si stanno ricevendo *Interest* malevoli. Il limite calcolato dinamicamente da ogni router indicherà il numero di *Interest* che si è disposti a ricevere dalla relativa

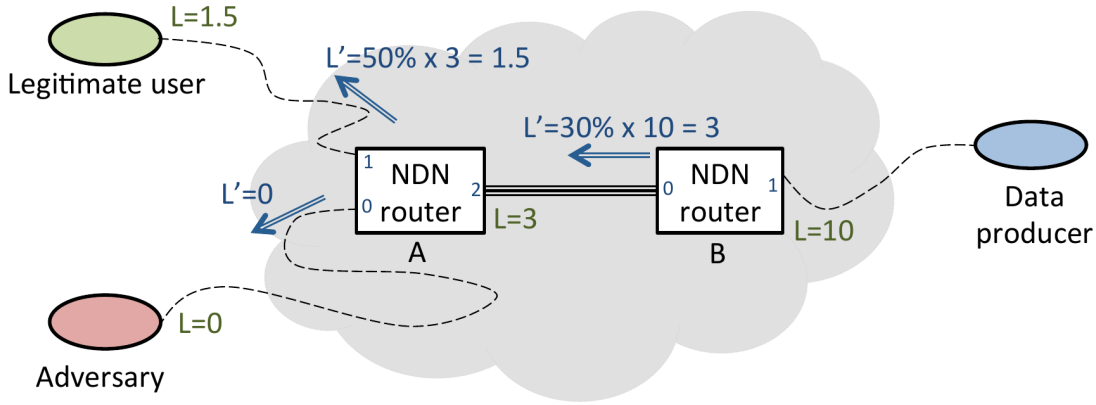


Figura 4.2: Funzionamento di Satisfaction-Based Pushback.[15]

interfaccia. La componente principale dell'algoritmo è mostrata in 1, mentre è possibile osservare in figura 4.2 uno schema rappresentativo di funzionamento.

Algorithm 1: Satisfaction-based pushback [15]

$\forall$ f $\in$ interfaces : $L'_f \leftarrow L_f$ ▷ *Per-incoming interface Interest limit*
▷ *Announcement from the neighbor*

function `InLimits`(*Interface in, Limit* L')
  $L_{in} \leftarrow L'$
function `AnnounceLimits()` ▷ *E.g., every second*

  foreach *outgoing interface out* **do**
    foreach *incoming interface in* **do**
      $L'_{in} = L_{out} \times (1 - U_{in}/F_{in})$
      AnnounceLimit(in, L'_{in})
    end
  end

Un metodo del tutto simile viene descritto in [66] in cui i router, ancora una volta mediante un algoritmo collaborativo, scambiano tra loro messaggi di allarme su un prefisso dedicato (`/pushback/alerts`) una volta superati i valori di soglia per le metriche ISR e utilizzo della PIT.

In [16] vengono introdotti due nuovi elementi a supporto del sistema collaborativo per la mitigazione di IFA: dei router di monitoraggio e un dispositivo di controllo centrale. I primi hanno il compito di raccogliere delle statistiche sullo stato della rete che invieranno al *controller*. Quest'ultimo, sulla base delle informazioni aggregate è in grado di determinare la presenza di un attacco in corso, identificarne l'origine e il prefisso coinvolto e in ultimo inviare indicazioni ai router

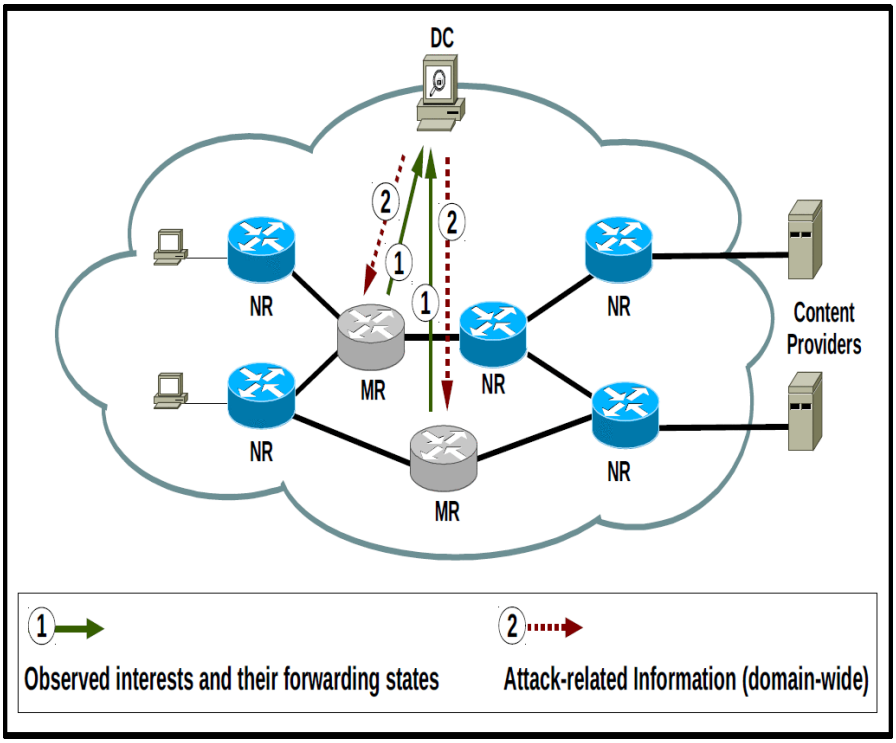


Figura 4.3: Schema di funzionamento del sistema collaborativo basato su router di monitoraggio (MR) e *controller* centrale (DC).[16]

sulle modalità operative di mitigazione (fig. 4.3).

Capitolo 5

Simulazione di Interest Flooding Attack

Nel capitolo 4 sono stati introdotti gli attacchi DDoS in NDN (meglio conosciuti come *Interest Flooding Attacks*) e i principali meccanismi di identificazione e mitigazione degli stessi. In particolare è stato evidenziato, sulla base di quanto scritto in [63], che un attacco di *flooding* in cui vengono richiesti Nomi esistenti e statici non è efficace in NDN in quanto non degrada le prestazioni dei router o dei *Producer*. A sostegno di questa ipotesi viene ricordata la presenza del componente *Pending Interest Table* dei router che si occupa di gestire le richieste da più *Consumer* per uno stesso Nome semplicemente instradandone una verso il *Producer* e registrando le interfacce da cui sono giunti i restanti *Interest*. Il progetto svolto dal candidato e descritto in questo capitolo rappresenta un tentativo di verifica sperimentale di quanto teorizzato in [63].

Il lavoro è stato svolto ideando un attacco DDoS in una rete NDN simulata adottando un processo di sviluppo che perseguisse i moderni standard *de facto* in merito alla riproducibilità delle sperimentazioni condotte mediante l'isolamento dell'ambiente. Una volta realizzati gli scenari è stato integrato un ulteriore processo per l'esecuzione automatizzata delle simulazioni, della raccolta degli artefatti e della loro elaborazione per una adeguata presentazione dei risultati.

5.1 Modello di attacco

Lo scenario descritto in questa sezione racchiude le caratteristiche generali dell'attacco ideato. Ogni elemento presentato, che al momento ne costituisce la parte concettuale, troverà un'implementazione effettiva nella simulazione dello scenario stesso.

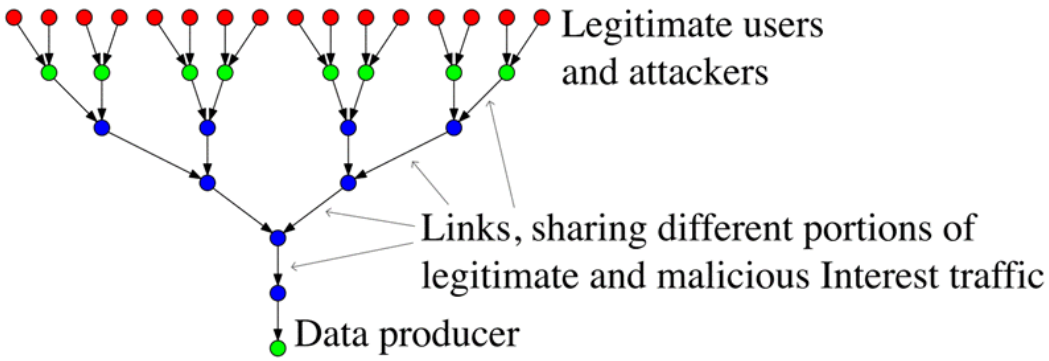


Figura 5.1: Topologia ad albero.[15]

Gli attori che interagiscono nella comunicazione sono quelli che è possibile trovare in una tipica rete NDN, ovvero *Consumer*, *Producer* e router. Come descritto in [15], è necessario considerare due tipi differenti di topologie: una che rappresenti il caso pessimo e una che permetta di effettuare valutazioni riconducendosi al mondo reale. Una topologia ad albero (fig. 5.1) ben rappresenta il caso pessimo, poiché il collegamento alla radice costituirà il collo di bottiglia verso il *Producer* e in quel caso è possibile osservare con precisione il degradamento delle prestazioni causato da un attacco DDoS. Volendo contestualizzare lo scenario in un caso reale è opportuno modellare la comunicazione in una rete che rispecchi un *Autonomous System* di grandi dimensioni (fig. 5.2).

Lo scenario prevede la presenza di un singolo *Producer* che serve contenuti relativi ad un solo Nome (`/prefix`). I *Consumer* sono locati nelle parti periferiche della rete e il loro numero dipende dalla topologia della rete, in ogni caso è fissata al 25% la quantità di *Consumer* malevoli. Questi ultimi inviano 1000 *Interest* al secondo per il Nome `/prefix` mentre le richieste legittime per lo stesso Nome vengono effettuate dal restante insieme di *Consumer* ad un tasso di 100 pacchetti al secondo. La comunicazione dura in totale 9 minuti, gli attaccanti inviano simultaneamente le richieste dal minuto 1 al minuto 6, mentre gli *Interest* legittimi vengono inviati per tutta la durata dello scenario.[64]

5.2 Ambiente di simulazione isolato

Di seguito viene presentato il simulatore utilizzato per l'implementazione dello scenario appena descritto, si elencano gli elementi basilari di una simulazione e viene mostrato il software usato per isolarne il contesto di esecuzione sfruttando le potenzialità dei *container*.

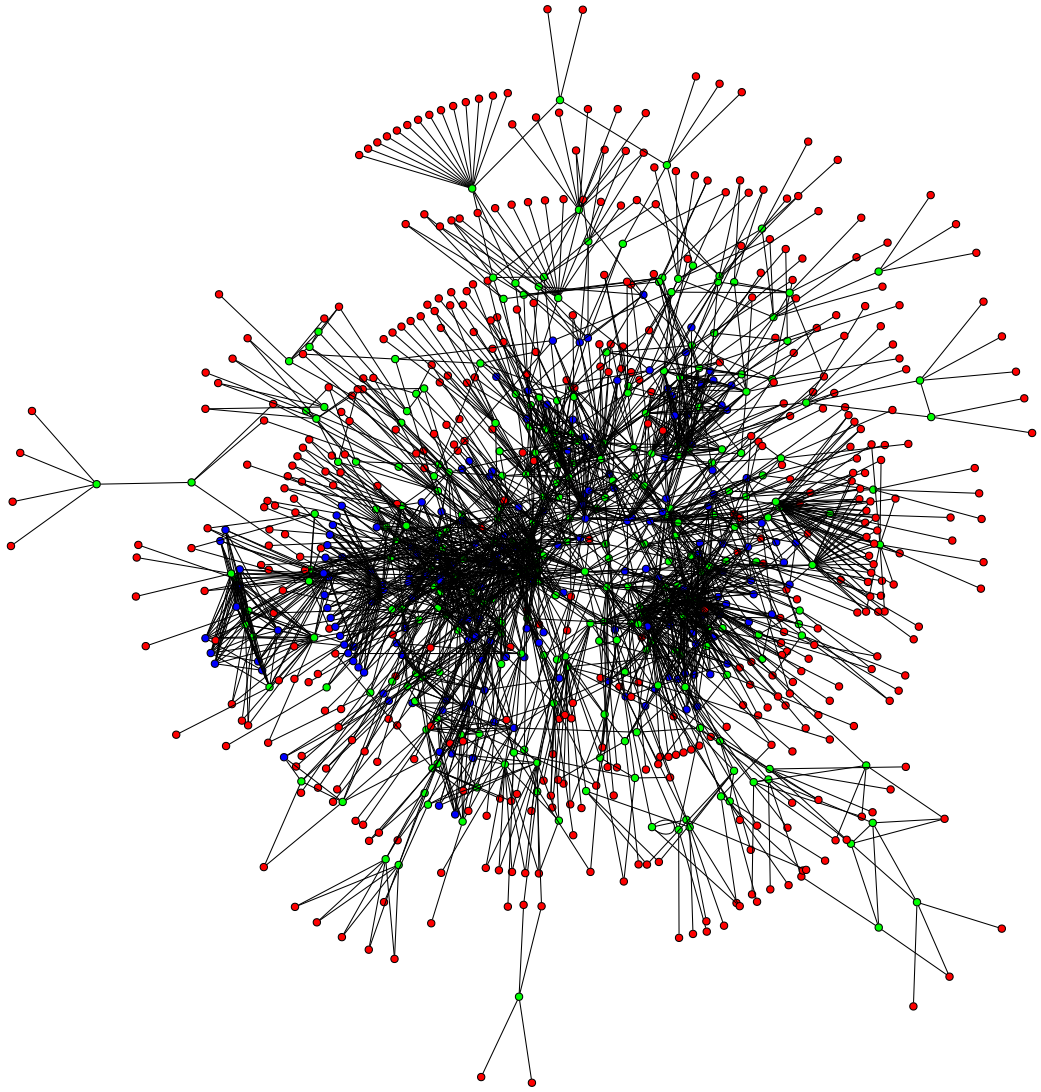


Figura 5.2: Topologia dell'Autonomous System 2914 (`NTT-COMMUNICATIONS-2914`) identificato dallo strumento Rocketfuel. [17]

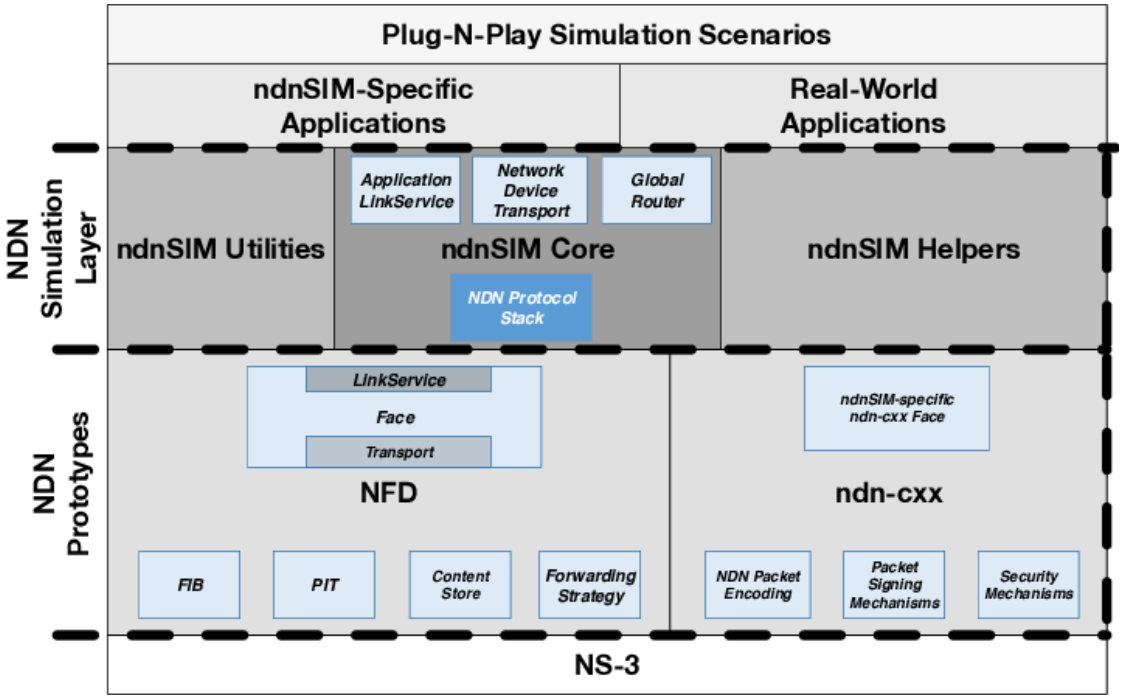


Figura 5.3: Struttura del simulatore ndnSIM.[18]

5.2.1 ndnSIM

Realizzare uno scenario che coinvolga centinaia o migliaia di dispositivi che comunicano tra loro ad intervalli di tempo prestabiliti, con collegamenti differenti e configurazioni personalizzate non è un compito banale in un contesto fisico. I simulatori di rete offrono una serie di vantaggi in questo senso, fornendo un approccio virtuale, più semplice e scalabile.

ndnSIM [18] è un simulatore *open source* per NDN che permette ai progettisti e ai ricercatori di valutare in un ambiente controllato le considerazioni effettuate sull'architettura. Il software è basato su ns-3, un'implementazione preesistente per la simulazione di reti informatiche i cui componenti fondamentali verranno descritti in seguito.

ndnSIM conta diversi livelli di astrazione (mostrati in fig. 5.3), ognuno dei quali è formato da moduli che implementano funzionalità specifiche dell'architettura. La sua struttura verrà dettagliatamente descritta nelle successive sezioni.

ns-3

ns-3 [67] è un simulatore ad eventi discreti nato nel 2006 come progetto *open source* per scopi didattici e di ricerca. Come descritto in precedenza, il compito di

un simulatore di rete è quello di semplificare la realizzazione di sperimentazioni che coinvolgono scenari difficilmente riproducibili in un contesto fisico. Tale obiettivo è reso possibile grazie a strumenti strutturati e complessi che racchiudono in moduli il funzionamento delle reti Internet.

I concetti principali su cui si basa ns-3 [68] permettono di modellare in modo generico un qualsiasi sistema di interconnessione tra reti in cui sono presenti delle unità computazionali chiamate *nodi*. Questi ultimi eseguono *applicazioni* che svolgono compiti per l'utente e sono dotati di *dispositivi di rete* che permettono di comunicare tra loro mediante dei *canali*. Nodi, dispositivi di rete e canali sono incapsulati in un'unico componente detto *topology helper* al fine di semplificare la creazione di entità di rete virtuali.

Ognuno dei concetti appena descritti è implementato in ns-3 mediante una classe e costituisce l'astrazione principale della programmazione ad oggetti per lo sviluppo di simulazioni.

ns-3 fornisce infine dei moduli per la creazione di simulazioni per reti di qualsiasi natura tecnologica, da quelle cablate alle reti wireless, supportando una moltitudine di protocolli e funzionalità utili alla simulazione, come il *logging*, una GUI o il tracciamento di tutti i pacchetti scambiati dai nodi.

NDN Forwarder Prototype

Come mostrato in fig. 5.3 ndnSIM è interamente basato su ns-3, tuttavia sono presenti degli ulteriori livelli di astrazione tra il simulatore appena descritto e gli scenari implementabili dagli utenti. Uno di questi è chiamato *NDN Forwarder Prototype* e consiste in due moduli fortemente legati tra loro.

Il primo, *NDN Forwarding Daemon*, contiene le implementazioni per le strutture dati e strategie usate dai router in NDN per realizzare l'inoltro dei pacchetti.

Ulteriori operazioni di NFD sono implementate nella libreria `ndn-cxx` che si occupa principalmente degli aspetti di codifica e di sicurezza dei pacchetti NDN.

In fig. 5.3 è inoltre possibile osservare che sia NFD che `ndn-cxx` comprendono un ulteriore componente chiamato *Face* che però assume ruoli differenti nei due diversi moduli. In NFD, *Face* consiste nell'astrazione principale usata dal *forwarder* per l'interfacciamento ai livelli inferiori e superiori dell'architettura (similmente alle *routine* del kernel per l'invio e la ricezione di pacchetti). `ndn-cxx` specifica invece *Face* come un'interfacciamento per la programmazione basilare di *Interest* e *Data* (equivalentemente alle *socket* BSD).

Principali componenti di ndnSIM

Il successivo livello di astrazione che è possibile osservare in fig. 5.3 è chiamato *NDN Simulation Layer* e comprende i tre moduli fondamentali di ndnSIM.

Il cuore del simulatore implementa: (i) il protocollo NDN (ii) l'interfacciamento con le astrazioni di ns-3 *NetDevice* e *Channel* mediante il sotto-componente *Trasport* di NFD (iii) un meccanismo di comunicazione tra le Applicazioni e il *forwarder* sfruttando il sotto-compomente *Transport* di NFD (iv) un configuratore globale per l'instradamento.

ndnSIM fornisce una serie di strumenti di utilità atti a semplificare lo sviluppo delle simulazioni, tra cui dei *packet tracer* in grado di catturare tutti i pacchetti sulle interfacce nei vari livelli dell'architettura (*link*, *network* e *application*) e un modulo per la lettura della topologia di rete da file.

Un ulteriore insieme di meccanismi consiste negli *Helpers* utili nella gestione della simulazione e nello svolgimento di compiti basilari come l'installazione dell'architettura NDN nei nodi, la specifica di strategie di inoltro o di politiche di *caching* nei router, modificare lo stato dei collegamenti e così via.

Il livello immediatamente superiore a quello appena descritto (*NDN Simulation layer*) consiste nelle Applicazioni di cui si vuole simulare il funzionamento e la comunicazione in rete.

Le Applicazioni presenti in questo livello possono essere di due tipi: specifiche di ndnSIM o derivanti da contesti reali.

Le prime vengono fornite direttamente come classi nel codice sorgente del simulatore: in fig. 5.4 ne viene mostrata una rappresentazione in UML. Come si evince in figura, `App` fa parte del `namespace ndn` a sua volta contenuto in `ns3`: ciò permette di comprendere a livello pratico quanto ndnSIM sia fortemente basato su ns-3 e sulle sue astrazioni. Le interfacce `Consumer` e `Producer` sono chiaramente riferite agli omonimi attori comunicanti in NDN, derivano da `App` e forniscono un'implementazione comune per le classi concrete.

5.2.2 Elementi di una simulazione

Come osservato, ndnSIM è ampiamente strutturato e ricco di moduli che si occupano di implementare l'architettura NDN, di interfacciarsi ad ns-3 e di rendere questa stessa struttura il più possibile trasparente ai ricercatori che intendono realizzare una simulazione.

L'implementazione di uno scenario fa infatti parte dell'ultimo livello di astrazione dell'architettura di ndnSIM mostrata in fig. 5.3. Una simulazione è costituita da tre elementi fondamentali: topologie, applicazioni e scenari. In questa sezione vengono esaminate le diverse implementazioni possibili degli elementi appena citati nonché il modo in cui sono relazionati nella simulazione stessa.

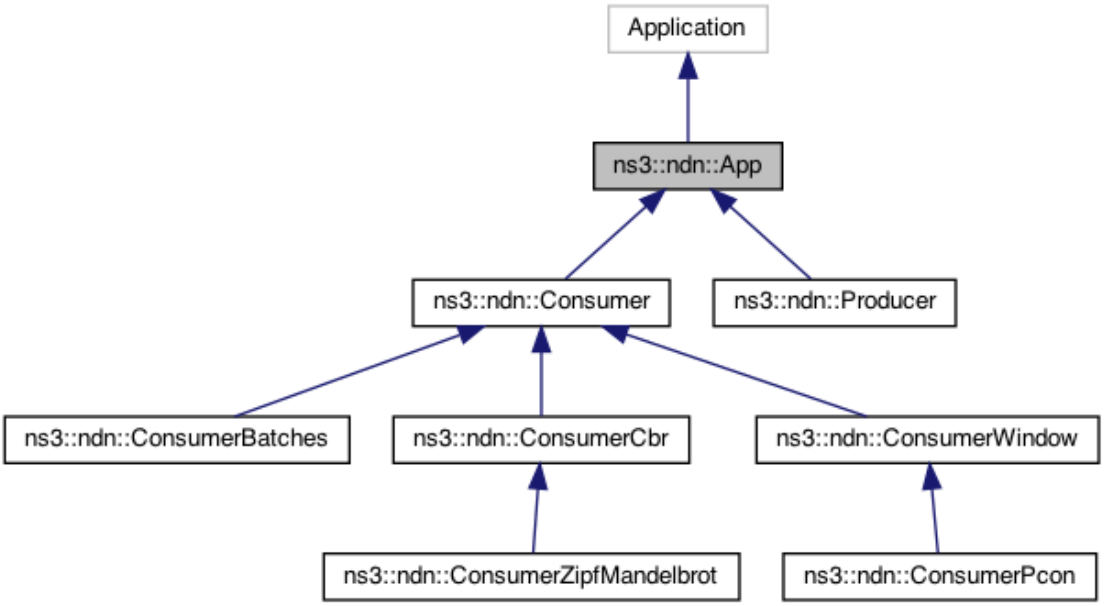


Figura 5.4: Diagramma UML delle classi relative alle applicazioni fornite con ndnSIM.[19]

Topologia della rete

Ogni scenario di comunicazione tra dispositivi di rete deve specificarne la disposizione dei nodi e le caratteristiche fisiche dei collegamenti: se via cavo o wireless, la velocità del mezzo trasmissivo, le latenze e così via.

Tali informazioni fanno parte della topologia della rete e in ndnSIM possono essere indicate direttamente nel codice sorgente dello scenario oppure in un generico file testuale con una sintassi opportuna e successivamente integrato nello scenario.

Nel primo caso si procede ad indicare con delle istruzioni nel linguaggio di programmazione scelto (C++ o Python) le informazioni sulla topologia: nell'esempio mostrato nel listato 5.1 vengono creati tre nodi e collegati tra loro con un canale a 1Mbps e latenza di 10 millisecondi.

Per semplificare il codice dello scenario è possibile specificare la topologia in un file testuale (listato 5.2) come mostrato nel seguente esempio. Il file deve avere una struttura e seguire una sintassi specifica affinché la topologia possa essere integrata correttamente nello scenario. La prima parte del file (sezione `router`) descrive i nodi e la loro disposizione spaziale. É inoltre possibile specificare un commento per ogni nodo e associarlo opzionalmente ad un gruppo (definito da un *system id*). Nella seconda parte del file (sezione `link`) vengono specificati i collegamenti indicando nodo sorgente e destinazione, banda a disposizione e latenza.

Listato 5.1: Definizione della topologia di rete all'interno dello scenario

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/ndnSIM-module.h"

namespace ns3 {

int
main(int argc, char* argv[])
{
  // setting default parameters for PointToPoint links
     and channels
  Config::SetDefault("ns3::PointToPointNetDevice::
     DataRate", StringValue("1Mbps"));
  Config::SetDefault("ns3::PointToPointChannel::Delay",
     StringValue("10ms"));
  Config::SetDefault("ns3::QueueBase::MaxSize",
     StringValue("20p"));

  // Creating nodes
  NodeContainer nodes;
  nodes.Create(3);

  // Connecting nodes using two links
  PointToPointHelper p2p;
  p2p.Install(nodes.Get(0), nodes.Get(1));
  p2p.Install(nodes.Get(1), nodes.Get(2));

  return 0;
}

} // namespace ns3
```

Listato 5.3: Lettura della topologia da file

```
AnnotatedTopologyReader topologyReader("", 25);
topologyReader.SetFileName("src/ndnSIM/examples/
   topologies/topo-6-node.txt");
topologyReader.Read();

// Getting containers for the consumer/producer
Ptr<Node> consumer1 = Names::Find<Node>("Src1");
```

Listato 5.2: Definizione della topologia di rete in un file testuale

```
router

# node   comment      yPos     xPos
Src1     NA           1        3
Rtr1     NA           2        5

link

# srcNode  dstNode   bandwidth   metric   delay   queue
Src1       Rtr1      10Mbps      1        10ms    20
```

Il nome del nodo è anche il suo identificativo che verrà richiamato nel codice sorgente. La scelta dei nomi è arbitraria, ma è buona pratica utilizzare il ruolo ricoperto nella comunicazione (ad esempio `Rtr` sta per router).

Il caricamento del file della topologia nello scenario viene mostrato nel listato 5.3. Si noti che nei listati che seguono non verranno più mostrati il caricamento delle librerie, la definizione del *namespace* di riferimento e la firma della funzione `main`.

Applicazioni

In ndnSIM è possibile sviluppare classi che implementano l'interfaccia `App` che a sua volta rappresenta l'astrazione dell'*Applicazione* di ns-3. Tali componenti definiscono il comportamento da adottare per l'invio di *Interest* e la ricezione di pacchetti *Data* (dal punto di vista del *Consumer*) e vice versa per i *Producer*.

Come mostrato nelle sezioni precedenti ndnSIM fornisce un supporto per applicazioni distribuite con il codice sorgente o definite in contesti reali, tuttavia gli sviluppatori possono crearne di nuove (listato 5.4).

Listato 5.4: Metodi che l'Applicazione deve implementare

```
class CustomApp : public ndn::App {
public:
  // register NS-3 type "CustomApp"
  static TypeId
  GetTypeId();

  virtual void
  StartApplication();

  virtual void
  StopApplication();

  virtual void
  OnInterest(std::shared_ptr<const ndn::Interest>
     interest);

  virtual void
  OnData(std::shared_ptr<const ndn::Data> contentObject)
     ;

private:
  void
  SendInterest();
};
```

Scenari

Infine, il codice sorgente dello scenario si occupa di caricare la topologia, assegnare le applicazioni ai nodi e definire le caratteristiche della comunicazione, come gli intervalli di tempo in cui *Consumer* e *Producer* vengono attivati, specificare le politiche di *caching*, di *routing* e le strategie di *forwarding*, nonché abilitare i *packet tracer* per la scrittura delle statistiche raccolte su file.

5.2.3 Docker per l'isolamento di ndnSIM

L'installazione di un *software* complesso come ndnSIM può risultare articolata e dispendiosa in termini di tempo per la quantità di operazioni preliminari da effettuare per ottenere un'istanza funzionante e pronta all'uso.

Sia ns-3 che ndnSIM, infatti, non vengono distribuiti come file binario precompilato, ma necessitano di alcuni passaggi più o meno onerosi (in base all'esperienza dell'utente) per la configurazione dell'ambiente di sviluppo. É innanzitutto necessario predisporre un dispositivo (fisico o virtuale) con un sistema operativo compatibile su cui installare le decine di librerie che rendono possibile il funzionamento del simulatore. In seguito è necessario effettuare il *download* del codice sorgente di ns-3, NFD, ndn-cxx e ndnSIM, configurare il *build system* Waf in base alle proprie esigenze e compilare il simulatore.

Un approccio di questo tipo presenta diversi svantaggi: in primo luogo se non si dispone in partenza di un sistema operativo compatibile con quello suggerito dagli sviluppatori, l'installazione del *software* può risultare ancora più complessa o talvolta impossibile a causa della mancanza di supporto dell'applicativo stesso o delle librerie che utilizza. Ricorrendo ad una macchina virtuale con un sistema operativo compatibile si incorre spesso in un dispendio di risorse inopportuno poiché, come mostrato in fig. 5.5, si riuscirebbero ad installare correttamente librerie e dipendenze per l'applicativo, tuttavia per disaccoppiare gli ambienti relativi alle singole applicazioni sarebbero necessarie più macchine virtuali.

I moderni sistemi di isolamento e distribuzione di applicativi sfruttano il concetto di *container*, ovvero un ambiente basato sull'esecuzione di singoli applicativi e relative dipendenze in un sistema operativo di dimensioni contenute e senza la necessità di virtualizzare interamente un computer, come effettuato dagli *hypervisor*.

Per fornire un ulteriore contributo alla comunità di ricercatori, il candidato ha implementato un'istanza del simulatore isolata distribuendola come *container* Docker. Il codice sorgente comprensivo di istruzioni per l'utilizzo è stato rilasciato in [69] e l'immagine compilata è stata rilasciata sulla piattaforma Docker Hub.

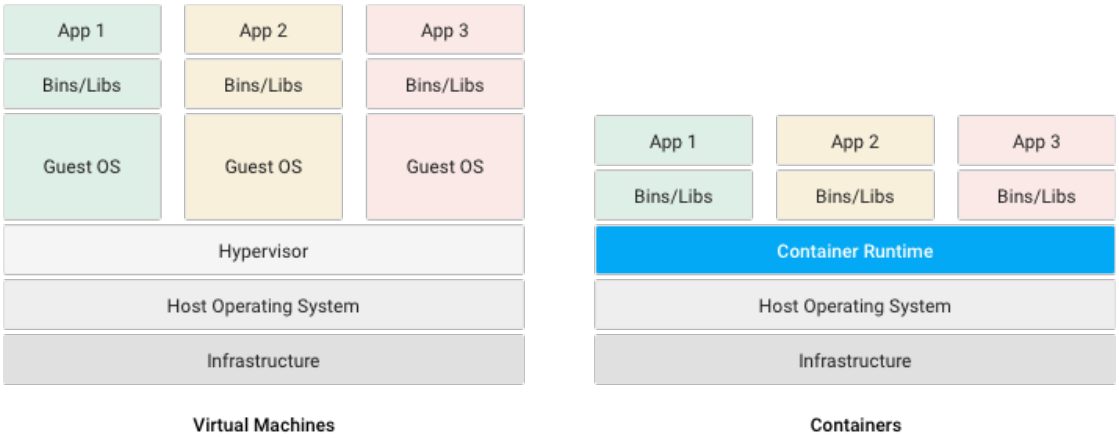


Figura 5.5: Differenze tra applicazioni installate su macchine virtuali (sinistra) e ambienti isolati in *container* (destra).[20]

5.3 Applicazione simulata

Le caratteristiche dell'attacco descritte in sezione 5.1 sono state implementate sulla base delle astrazioni fornite dal simulatore e citate in sezione 5.2.

É stata realizzata una classe chiamata `DumbConsumerCbr` che eredita dalla classe `App` del `namespace ns3::ndn` il comportamento generico di un'applicazione per l'invio di *Interest* e la ricezione di *Data* (nel caso di *Consumer*).

In particolare, `DumbConsumerCbr` invia *Interest* (listato 5.6) con uno stesso Nome configurato mediante l'API `AddAttribute` della classe `TypeId` (listato 5.5).

In ricezione viene semplicemente inviato alla console il pacchetto Data ricevuto e il relativo Nome mediante una funzione di *debug*.

5.4 Scenario

Nella sezione 5.2.2 sono stati elencati i principali elementi di cui si compone una simulazione in ndnSIM.

Tra questi è possibile trovare il codice sorgente dello scenario in cui vengono assemblati tutti i rimanenti elementi e le caratteristiche della simulazione.

Nel caso particolare dello scenario descritto in 5.1, l'implementazione comprende il caricamento del file della topologia specificato via CLI nonché la specifica del numero di *Consumer* malevoli e legittimi. Questi ultimi, in base alla quantità specificata, vengono scelti casualmente all'interno della topologia indicata (con opportune verifiche di correttezza).

In ultimo vengono installate (listato 5.7) le Applicazioni relative ai *Consumer* e al *Producer* nei rispettivi nodi, viene specificato l'intervallo temporale in

Listato 5.5: API per la configurazione del prefisso

```
TypeId
DumbConsumerCbr::GetTypeId(void)
{
  static TypeId tid =
    TypeId("ns3::ndn::DumbConsumerCbr")
      .SetGroupName("Ndn")
      .SetParent<ndn::App>()
      .AddConstructor<DumbConsumerCbr>()

      .AddAttribute ("Prefix",
                     "Name of the interest",
                     StringValue ("/dumb-interest"),
                     ndn::MakeNameAccessor (&
                        DumbConsumerCbr::m_interestName),
                     ndn::MakeNameChecker ())
```

cui devono essere attivi e viene chiamato il componente `L3Tracer` per effettuare il tracciamento (ogni secondo) di tutti i pacchetti che verranno inoltrati nelle interfacce durante la comunicazione.

L'ultimo componente, per quanto banale, è di fondamentale importanza in quanto colleziona i dati relativi alle connessioni effettuate tra i nodi e sui dati scambiati tra essi. Informazioni che verranno poi utilizzate in grafici generati mediante degli appositi *script* (in linguaggio R) per un'analisi qualitativa della simulazione effettuata.

5.5 Processo di automatismo

Seppure sia possibile scrivere il codice sorgente della simulazione all'interno delle cartelle di ndnSIM, le *best practices* di sviluppo suggeriscono di mantenere i relativi file in una cartella separata e gestita con lo strumento di DVCS (*Distributed Versioning Control System*) Git.

Gli sviluppatori forniscono un progetto da usare come *template* al livello di astrazione degli scenari "plug-n-play" in ndnSIM, come mostrato in fig. 5.3.

Tra gli strumenti distribuiti nello scenario di *template* è possibile trovare uno scheletro di implementazione di uno script Python per l'esecuzione parallela di simulazioni, il postprocessing dei dati registrati dai *packet tracer* e la loro elaborazione dagli script R per la generazione di grafici.

Listato 5.6: Implementazione parziale del metodo `sendInterest`

```
void
DumbConsumerCbr::SendInterest ()
{
  shared_ptr<ndn::Name> prefix = make_shared<ndn::Name>(
     m_interestName); // another way to create name

  // Create and configure ndn::InterestHeader
  shared_ptr<ndn::Interest> interest = make_shared<ndn::
     Interest> ();
  interest->setNonce(m_rand->GetValue(0, std::
     numeric_limits<uint32_t>::max()));
  interest->setName(*prefix);
  interest->setCanBePrefix(false);
  time::milliseconds interestLifeTime(m_interestLifeTime
     .GetMilliSeconds());
  interest->setInterestLifetime(interestLifeTime);

  NS_LOG_DEBUG("Sending Interest packet for " << *prefix
     );

  m_transmittedInterests(interest, this, m_face);
  m_appLink->onReceiveInterest(*interest);

  ScheduleNextPacket();
```

Listato 5.7: Installazione dell'applicazione malevola nei nodi preselezionati

```
ndn::AppHelper evilAppHelper("ns3::ndn::DumbConsumerCbr");
evilAppHelper.SetAttribute("Frequency", StringValue("1000"));
evilAppHelper.SetPrefix("/" + prefix);

for (NodeContainer::Iterator node = evilNodes.Begin(); node != evilNodes.End(); node++)
{
  ApplicationContainer evilApp;
  evilApp.Add (evilAppHelper.Install(*node));

  evilApp.Start(Seconds(60.0));
  evilApp.Stop(Seconds(360.0));
}
```

Per la facilità di utilizzo, il candidato ha adottato la metodologia appena descritta che prevede di eseguire parallelamente ogni esecuzione sui *core* della CPU, anziché adottare l'integrazione con MPI fornita dal simulatore in quanto necessita di un'ulteriore analisi sul partizionamento del dominio.

Lo script fornito è stato soggetto ad interventi di aggiornamento: è stata effettuata la conversione a Python 3 modificando gli *statement* incompatibili e sostituendo le chiamate alle librerie deprecate.

In particolare per sfruttare il parallelismo sono state adottate le funzionalità native di `concurrent.futures` in favore della libreria `workerpool` di Python 2, ormai non più mantenuta.

Le peculiarità di `concurrent.futures` consistono nell'utilizzo dell'interfacciamento comune di `multithreading` e `multiprocessing` per realizzare dei *pool* di `Executor` per l'esecuzione asincrona di attività. Queste entità, in base alla classe concreta utilizzata, consisteranno in *thread* o processi che si occuperanno dei *task* assegnati. L'interfaccia comune delle due classi consente loro di poter essere intercambiate con minimo sforzo.

Lo script adattato si occupa quindi di creare un *pool* di processi dello stesso numero di *core* della macchina. Ad ogni processo verrà quindi affidato il compito di lanciare una simulazione attraverso la libreria `subprocess`.

Terminata l'esecuzione delle simulazioni vengono allo stesso modo eseguiti gli script per la generazione dei grafici presenti in sezione 5.6.

5.6 Presentazione e analisi dei risultati

La simulazione è stata condotta secondo le caratteristiche descritte nelle sezioni precedenti, variando nello scenario la topologia utilizzata e il numero di attaccanti pur mantenendo la proporzione del 25% di *Consumer* malevoli sul numero totale.

Di seguito vengono proposti i risultati ottenuti nelle due topologie principali tra le undici sottoposte ad esame: la topologia ad albero e l'Autonomous System 2914 (il più grande in termini di numero di nodi) identificato dallo strumento Rocketfuel.

I grafici mostrati in questa sezione fanno riferimento alle metriche di numero di *Interest* soddisfatti e scaduti (rispettivamente gli indicatori `SatisfiedInterests` e `TimedOutInterests` di `L3Tracer`) per i *Consumer* nelle due topologie indicate.

In fig. 5.6 e 5.7 è possibile osservare che per i *Consumer* legittimi tutti i 100 *Interest* al secondo sono soddisfatti per l'intera durata della simulazione (nove minuti) in entrambe le topologie.

Il risultato è pressoché identico riferendosi ai *Consumer* malevoli (fig. 5.8 e 5.9): per tutta la durata dell'attacco (dal minuto 1 al 6) in entrambe le topologie tutti i 1000 *Interest* al secondo vengono soddisfatti.

L'indicatore relativo alla metrica `TimedOutInterests` assume quasi sempre il valore zero, indicando che non sono presenti degradazioni nelle prestazioni durante la comunicazione.

Sulla base di quanto mostrato è possibile quindi affermare che le ipotesi effettuate in [63] e a cui il corrente modello di attacco fa riferimento sono confermate. Un attacco di *packet flooding* basato sull'invio di *Interest* statici e riferiti allo stesso Nome non degrada le prestazioni della rete poiché i componenti dei router PIT e CS nativamente introdotti in NDN permettono di contrastare l'attacco.

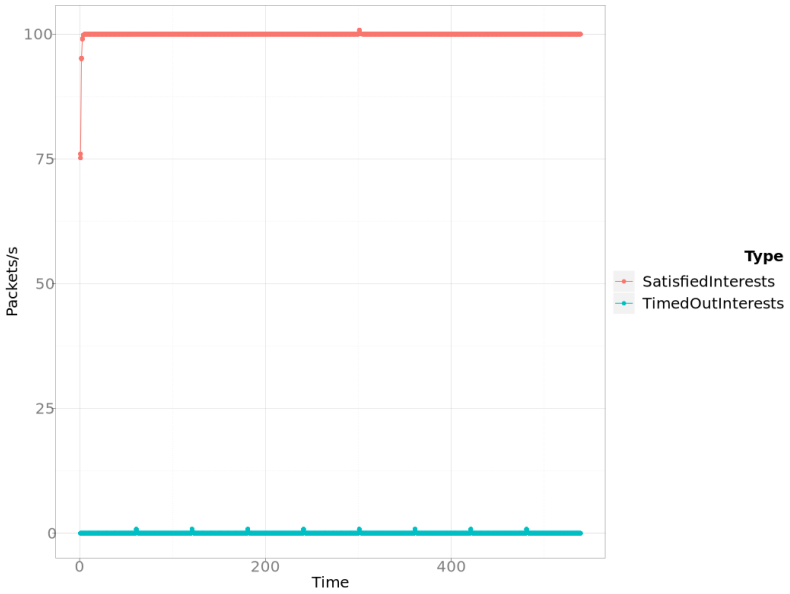


Figura 5.6: Rappresentazione degli *Interest* soddisfatti e scaduti nella topologia ad albero per i *Consumer* legittimi.

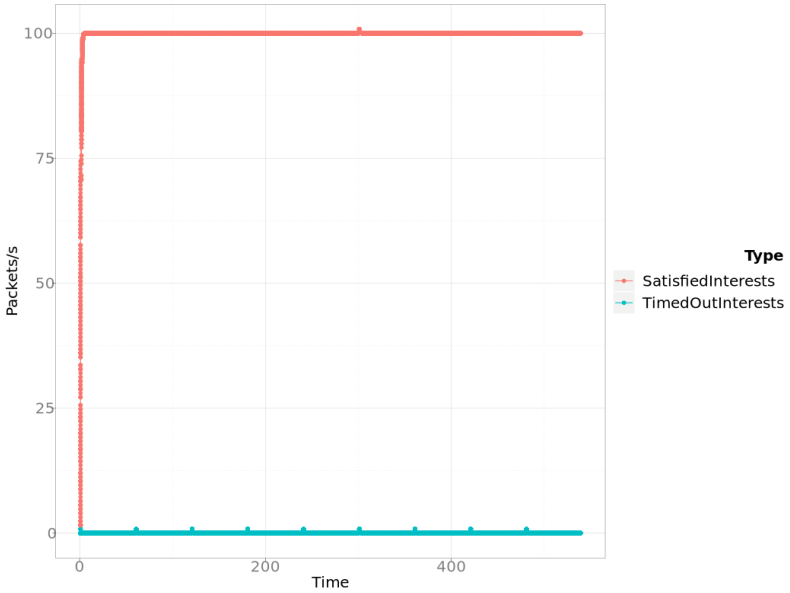


Figura 5.7: Rappresentazione degli *Interest* soddisfatti e scaduti nella topologia dell'AS 2914 per i *Consumer* legittimi.

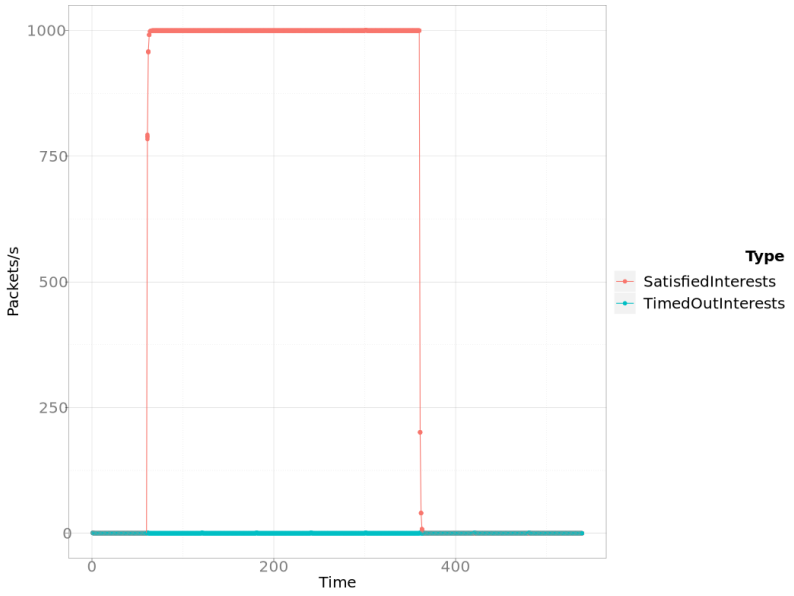


Figura 5.8: Rappresentazione degli *Interest* soddisfatti e scaduti nella topologia ad albero per i *Consumer* malevoli.

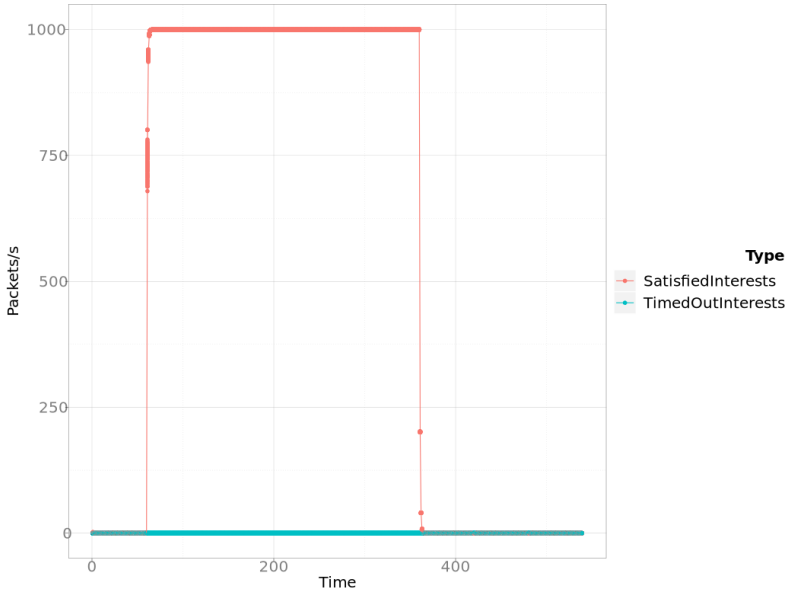


Figura 5.9: Rappresentazione degli *Interest* soddisfatti e scaduti nella topologia dell'AS 2914 per i *Consumer* malevoli.

Capitolo 6

Conclusioni

Il presente elaborato di tesi rappresenta un percorso non esclusivamente tecnico che tende a motivare un approfondimento nei confronti delle *Future Internet Architectures* e porta la sicurezza delle reti in primo piano per sensibilizzare sul valore che essa assume al giorno d'oggi.

Quanto viene proposto nei primi capitoli è necessario a comprendere le ragioni che risiedono dietro la nascita del progetto NDN e di quanto sia necessario studiare gli attacchi DDoS in relazione al forte impatto che hanno sulle aziende e sulle loro infrastrutture.

IP, infatti, nonostante possa vantare decine di anni di eccellente servizio, non può essere più ritenuto strettamente adeguato alle nuove sfide che il Web pone in termini di modalità di distribuzione dei contenuti.

É inoltre fondamentale che un progetto che mira a sostituire IP sia soggetto ad ampie fasi di analisi e verifiche di sicurezza, al fine di integrare appropriate misure di protezione contro gli attacchi durante la costruzione stessa del protocollo.

In questo elaborato, oltre a fornire un'introduzione sufficientemente dettagliata a NDN stesso, vengono descritti gli attacchi DDoS sotto una nuova veste: non potendo più essere effettuati come nel protocollo IP, gli *Interest Flooding Attacks* sfruttano il funzionamento intrinseco dei meccanismi di inoltro per compiere attacchi con Nomi inesistenti andando ad esaurire le risorse dei router.

É stato a tal proposito verificato sperimentalmente mediante una simulazione che gli scenari di attacco DDoS tipici di IP (effettuati con *packet flooding*) non hanno impatto in una rete NDN, confermando quindi quanto ipotizzato in [63].

Negli articoli esaminati e citati nel testo sono presenti proposte di identificazione e mitigazione per IFA che costituiscono un tema chiave nell'analisi degli attacchi DDoS in NDN. Ulteriori approfondimenti possono essere infatti correlati a tali meccanismi, in particolare è possibile verificare l'efficacia delle tecniche di mitigazione proposte in letteratura con degli attacchi avanzati come *blended IFA*

o *chameleonic IFA*. Inoltre si aprono nuove strade per la definizione di approcci più innovativi ed evoluti sia per le tecniche di difesa che per quelle di attacco.

Ulteriori sforzi possono essere volti a migliorare la metodologia di lavoro proposta e gli strumenti a corredo implementati: favorire l'utilizzo di MPI per parallelizzare il carico di lavoro della singola esecuzione potrebbe portare vantaggi in termini di scalabilità mentre l'immagine Docker implementata dal candidato può beneficiare di estensioni all'insieme di funzionalità attualmente fornito per poi essere proposta come strumento ufficiale al gruppo di lavoro di ndnSIM.

Bibliografia

[1] Hannah Ritchie Max Roser and Esteban Ortiz-Ospina. Internet. *Our World in Data*, 2015. https://ourworldindata.org/internet.

[2] Cisco. Annual Internet Report (2018–2023). Courtesy of Cisco Systems, Inc. Unauthorized use not permitted. Technical report, Cisco, March 2020.

[3] Fabrizio Margotta and Daniele Giacomini and Wikipedia contributors. Abbinamento tra il modello ISO-OSI e la semplicità dei protocolli TCP/IP. CC BY-SA 2.5 — Wikipedia, The Free Encyclopedia, 2012. [Online; accessed 07-October-2020].

[4] Wikipedia contributors. Encapsulation of application data descending through the layered IP architecture. GNU Free Documentation License, Version 1.2 — Wikipedia, The Free Encyclopedia, 2008. [Online; accessed 08-October-2020].

[5] Andrew S. Tanenbaum. The environment of the network layer protocols, 2011. Figure 5-1 from Computer Networks, 5 edition.

[6] Fabrizio Margotta and Wikipedia contributors. IP stack connections. CC BY-SA 3.0 — Wikipedia, The Free Encyclopedia, 2008. [Online; accessed 03-November-2020].

[7] MichelBakni and J. Postel and Wikipedia contributors. Data packet of IPv4. CC BY-SA 4.0 — Wikipedia, The Free Encyclopedia, 2019. [Online; accessed 10-October-2020].

[8] Andrew S. Tanenbaum. IP address formats, 2011. Figure 5-53 from Computer Networks, 5 edition.

[9] Fabrizio Margotta and Joao Gondim and Robson Albuquerque and Anderson Nascimento and Luis García Villalba and Tai-Hoon Kim. Amplified reflection attack representation. CC BY 4.0 — A Methodological Approach for Assessing Amplified Reflection Distributed Denial of Service on the Internet of Things, 2016.

[10] NSFOCUS. BGP Hijackings Come Back. ©COPYRIGHT 2020, NSFOCUS. ALL RIGHTS RESERVED.

[11] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[12] Lixia Zhang et al. Named Data Networking Tech Report 001. Technical report, NDN, October 2010.

[13] Lixia Zhang. NDN: A New Way to Communicate. NDN Tutorial @ MILCOM, November 2019.

[14] Seungoh Choi, Kwangsoo Kim, Seongmin Kim, and Byeong-hee Roh. Threat of DoS by interest flooding attack in content-centric networking. In *The International Conference on Information Networking 2013 (ICOIN)*, pages 315–319. IEEE, 2013.

[15] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun, and Lixia Zhang. Interest flooding attack and countermeasures in Named Data Networking. In *2013 IFIP Networking Conference*, pages 1–9. IEEE, 2013.

[16] Hani Salah, Julian Wulfheide, and Thorsten Strufe. Coordination supports security: A new defence mechanism against interest flooding in NDN. In *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pages 73–81. IEEE, 2015.

[17] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[18] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. On the Evolution of ndnSIM: an Open-Source Simulator for NDN Experimentation. *ACM SIGCOMM Computer Communication Review*, 47:19–33, 09 2017.

[19] ndnSIM development team. ns3::ndn::App Class Reference. `https://ndnsim.net/current/doxygen/classns3_1_1ndn_1_1App.html`.

[20] Google. Containers 101. Google and the Google logo are registered trademarks of Google LLC, used with permission.

[21] Andrew S. Tanenbaum. *Computer Networks*, chapter 1.5.1, pages 50–57. Prentice Hall PTR, 4 edition, 2003.

[22] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-Wide Web: the information universe. *Internet Research*, 1992.

[23] Andrew S. Tanenbaum. *Computer Networks*, chapter 1.1, pages 3–14. Prentice Hall PTR, 4 edition, 2003.

[24] Van Jacobson. A New Way to look at Networking. Google Tech Talk, August 2006.

[25] Van Jacobson et al. Networking named content. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies, Rome, Italy*, pages 1–12, December 2009.

[26] National Science Foundation. Future Internet Architecture Project. `http://www.nets-fia.net/`.

[27] ICNRG. Information-Centric Networking Research Group History. `https://datatracker.ietf.org/rg/icnrg/history/`.

[28] Damian Menscher. Exponential growth in DDoS attack volumes. Google Cloud Blog, 2020.

[29] Sam Kottler. February 28th DDoS Incident Report. GitHub Blog, 2018.

[30] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[31] Andrew S. Tanenbaum. *Computer Networks*, chapter 1.3, pages 26–30. Prentice Hall PTR, 4 edition, 2003.

[32] Andrew S. Tanenbaum. *Computer Networks*, chapter 1.4.2, pages 41–44. Prentice Hall PTR, 4 edition, 2003.

[33] Jon Postel. Transmission Control Protocol. STD 7, RFC Editor, September 1981. `http://www.rfc-editor.org/rfc/rfc793.txt`.

[34] J. Postel. User Datagram Protocol. STD 6, RFC Editor, August 1980. `http://www.rfc-editor.org/rfc/rfc768.txt`.

[35] Theodore John Socolofsky and Claudia Jeanne Kale. TCP/IP tutorial. RFC 1180, RFC Editor, January 1991. `http://www.rfc-editor.org/rfc/rfc1180.txt`.

[36] Jon Postel. Internet Protocol. STD 5, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc791.txt.

[37] Yakov Rekhter, Robert G. Moskowitz, Daniel Karrenberg, Geert Jan de Groot, and Eliot Lear. Address Allocation for Private Internets. BCP 5, RFC Editor, February 1996. http://www.rfc-editor.org/rfc/rfc1918.txt.

[38] Vince Fuller, Tony Li, Jessica (Jie Yun) Yu, and Kannan Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519, RFC Editor, September 1993. http://www.rfc-editor.org/rfc/rfc1519.txt.

[39] Andrew S. Tanenbaum. *Computer Networks*, chapter 5.2, pages 350–366. Prentice Hall PTR, 4 edition, 2003.

[40] John Moy. OSPF Version 2. STD 54, RFC Editor, April 1998. http://www.rfc-editor.org/rfc/rfc2328.txt.

[41] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, January 2006. http://www.rfc-editor.org/rfc/rfc4271.txt.

[42] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, RFC Editor, November 1998.

[43] B. Harris and R. Hunt. TCP/IP security threats and attack methods. *Computer Communications*, 22(10):885 – 897, 1999.

[44] Robbie Myers. Attacks on TCP/IP Protocols.

[45] CERT Coordination Center. CERT advisory CA 1996-21, TCP SYN flooding and IP spoofing attacks. September 1996. revised November 2000, 1996.

[46] M. Handley and E. Rescorla and. Internet Denial-of-Service Considerations. RFC 4732, RFC Editor, December 2006.

[47] Wikipedia contributors. Tcp normal. CC BY-SA 3.0 — Wikipedia, The Free Encyclopedia, 2012. [Online; accessed 15-October-2020].

[48] Wikipedia contributors. Syn flooding : several connections set by the attacker are half-open without proper acknowledgments, the server queue gets full and another client can not connect. CC BY-SA 2.5 — Wikipedia, The Free Encyclopedia, 2006. [Online; accessed 15-October-2020].

[49] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM SIGCOMM Computer Communication Review*, 31(3):38–47, 2001.

[50] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the Internet. *ACM SIGCOMM Computer Communication Review*, 37(4):265–276, 2007.

[51] NDN Team. NDN Packet Format Specification version 0.3: Interest Packet. `https://named-data.net/doc/NDN-packet-spec/current/interest.html`.

[52] NDN Team. NDN Packet Format Specification version 0.3: Data Packet. `https://named-data.net/doc/NDN-packet-spec/current/data.html`.

[53] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifier (uri): Generic syntax. STD 66, RFC Editor, January 2005. `http://www.rfc-editor.org/rfc/rfc3986.txt`.

[54] NDN Team. NDN Packet Format Specification version 0.3: Name. `https://named-data.net/doc/NDN-packet-spec/current/name.html`.

[55] NDN Project Team. NDN Technical Memo: Naming Conventions (Revision 2). *Technical Report NDN-0022*, 2019.

[56] Lan Wang, AKMM Hoque, Cheng Yi, Adam Alyyan, and Beichuan Zhang. OSPFN: An OSPF based routing protocol for named data networking. Technical report, Technical Report NDN-0003, 2012.

[57] Vince Lehman, AKM Mahmudul Hoque, Yingdi Yu, Lan Wang, Beichuan Zhang, and Lixia Zhang. A secure link state routing protocol for NDN. *Tech. Rep. NDN-0037*, 2016.

[58] Chavoosh Ghasemi, Hamed Yousefi, Kang G Shin, and Beichuan Zhang. MUCA: New routing for named data networking. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 289–297. IEEE, 2018.

[59] Yu Zhang, Zhongda Xia, Alexander Afanasyev, and Lixia Zhang. A note on routing scalability in named data networking. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2019.

[60] Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Spyridon Mastorakis, Yanbiao Li, Alexander Afanasyev, and Lixia Zhang. An overview of security support in Named Data Networking. *IEEE Communications Magazine*, 56(11):62–68, 2018.

[61] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[62] Gergely Acs, Mauro Conti, Paolo Gasti, Cesar Ghali, and Gene Tsudik. Cache privacy in named-data networking. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 41–51. IEEE, 2013.

[63] Paolo Gasti, Gene Tsudik, Ersin Uzun, and Lixia Zhang. DoS and DDoS in Named Data Networking. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7. IEEE, 2013.

[64] Salvatore Signorello, Samuel Marchal, Jérôme François, Olivier Festor, and Radu State. Advanced Interest Flooding Attacks in Named-Data Networking. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE, 2017.

[65] Jianqiang Tang, Zhongyue Zhang, Ying Liu, and Hongke Zhang. Identifying interest flooding in named data networking. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 306–310. IEEE, 2013.

[66] Alberto Compagno, Mauro Conti, Paolo Gasti, and Gene Tsudik. Poseidon: Mitigating interest flooding DDoS attacks in named data networking. In *38th annual IEEE conference on local computer networks*, pages 630–638. IEEE, 2013.

[67] ns-3 development team. ns-3 website. `https://www.nsnam.org/`.

[68] ns-3 development team. ns-3 Tutorial. `https://www.nsnam.org/docs/release/3.32/tutorial/ns-3-tutorial.pdf`.

[69] Fabrizio Margotta. ndnSIM Docker repository. `https://github.com/fam4r/docker-ndnsim`.

Ringraziamenti

Il futuro è in mano ai deboli che si sono fatti coraggio.
Mario Molinari

La conclusione di un percorso lungo e complesso porta a fare delle riflessioni su quanto affrontato e sulle persone meravigliose che mi sono state accanto. É a voi che dedico questo momento.

In primis devo tutto alla mia famiglia e a Giovanna per aver condiviso ogni istante, piacevole o meno, di questo viaggio.

Ai miei parenti, vicini e lontani, per il supporto costante e soprattutto a chi non c'è più, che tanto teneva a questo risultato.

Non posso di certo dimenticare tutti gli amici che mi hanno sempre dimostrato il loro affetto, la loro vicinanza e che hanno sempre creduto in me, specie quando io stesso avevo smesso di farlo.

Ringrazio anche il mio relatore, Giovanni Pau, per la disponibilità concessa nonostante i mille impegni, per la fiducia e per il rapporto autentico mostrato.

Una menzione speciale va al CeSeNA Security, qualcosa che è diventato più di un semplice gruppo di interesse.

Non basterebbe una pagina per elencare i nomi di tutti coloro che hanno contribuito a rendere questo percorso così speciale, ma sapete quanto siete importanti per me. É a tutti voi che va il mio ringraziamento più sincero e profondo, senza di voi non sarebbe stato lo stesso.