Università degli Studi di Torino

Dipartimento di informatica

Tesi di Laurea Triennale in Informatica

Analisi dell'efficacia delle CDN nel gestire i cambiamenti nel traffico degli utenti: Un approccio basato su simulazioni NS3.

Relatore:

Prof: Matteo Sereno

Candidato:

Andrea Fulcheri

ANNO ACCADEMICO 2022/2023

Per scoprire qualcosa, è meglio eseguire esperimenti accurati che impegnarsi in profonde discussioni filosofiche.
- Richard Phillips Feynman

Abstract

Questa tesi di laurea triennale in informatica esplora il cambiamento dinamico nel traffico utente delle reti ad alta velocità. L'evoluzione delle reti ad alta velocità è analizzata in dettaglio, focalizzandosi sulle trasformazioni nel tipo di traffico, tra cui lo streaming video, i giochi online, l'istruzione a distanza e l'Internet delle Cose (IoT). Si esaminano gli impatti economici e sociali di queste trasformazioni, evidenziando l'importanza delle Content Delivery Networks (CDN) nel panorama odierno. Questo studio si concentra anche sul concetto, l'implementazione e l'utilizzo delle CDN, esplorando le caratteristiche delle CDN e l'uso di Anycast in esse. Un focus particolare è posto sul ruolo cruciale delle simulazioni su prototipi di rete con cache, enfatizzando l'importanza dei protocolli UDP e QUIC per ottimizzare le prestazioni della rete. L'implementazione e la simulazione del meccanismo di cache a livello di rete sono esplorate attraverso NS3 Network Simulator, con un'analisi approfondita del protocollo QUIC e delle sue applicazioni nel contesto delle CDN. Inoltre, vengono presentati i dettagli sulla struttura di Docker e il suo utilizzo nel contesto specifico di questa ricerca. Il lavoro si conclude con un'analisi dettagliata dei moduli NS3 implementati, comprese le distribuzioni statistiche del traffico, l'implementazione della cache utilizzando strutture dati come Hashmap e Coda, e l'ottimizzazione tramite il pre-fetching. I risultati ottenuti sono analizzati attraverso telemetrie specifiche, fornendo un quadro delle prestazioni e delle implicazioni delle nuove tecnologie e strategie esplorate in questa tesi.

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.

Indice

Capitolo 1

L'Avvento delle Reti ad Alta Velocità: Cambiamenti nel Tipo di Traffico Utente

1.1 Introduzione

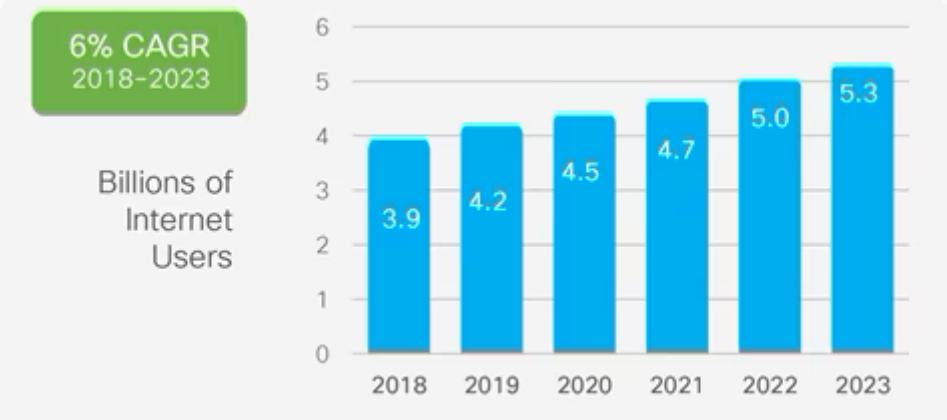


Figura 1.1: Proiezioni Cisco per il numero di utenti di Internet [Cis20]

L'avvento delle reti ad alta velocità ha rivoluzionato la società contemporanea, trasformando radicalmente la modalità di comunicazione, interazione e fruizione dei contenuti online. Questa evoluzione tecnologica ha portato a un notevole cambiamento nel tipo di traffico generato dagli utenti, con impatti significativi su diverse aree, tra cui l'economia digitale, la cultura, l'istruzione e l'innovazione. Secondo il Cisco Annual Internet Report, quasi due terzi della popolazione mondiale avrà accesso a Internet entro il 2023. Entro il 2023 ci saranno 5,3 miliardi di utenti totali di Internet (66% della popolazione globale), rispetto ai 3,9 miliardi (51% della popolazione globale) del 2018. [Cis20]

1.2 Evoluzione delle Reti ad Alta Velocità

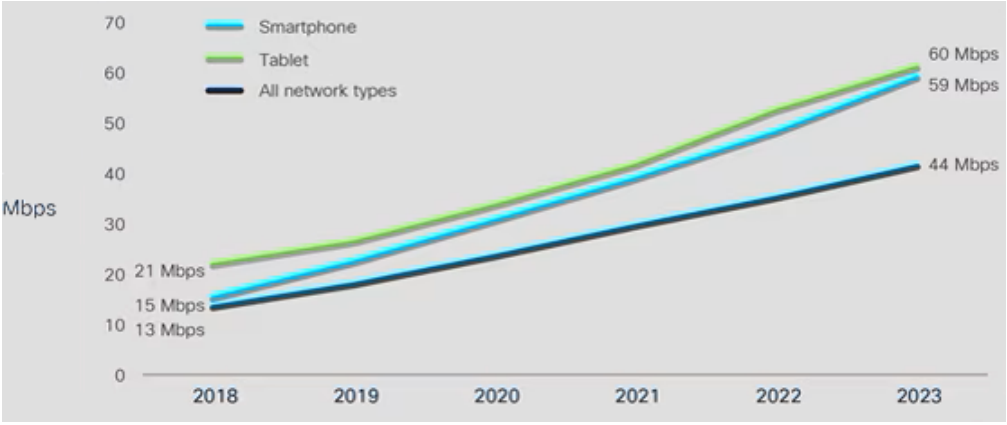


Figura 1.2: Proiezioni Cisco per la Velocità media globale della telefonia mobile per tipo di dispositivo [Cis20]

Negli ultimi decenni, la tecnologia delle reti è progredita rapidamente, passando da connessioni dial-up lente e limitate a reti ad alta velocità che offrono connettività con latenze ed affidabilità largamente migliori. La diffusione delle reti a banda larga, delle connessioni fibra ottica e delle reti mobili ad alta velocità ha reso possibile la trasmissione rapida di dati, audio e video in tempo reale. Questa evoluzione ha aperto la strada a nuove opportunità e possibilità per gli utenti online.

1.3 Cambiamenti nel Tipo di Traffico Web: Streaming Video e Contenuti Multimediali

L'incremento delle velocità di connessione ha avuto un impatto significativo sui modelli di traffico online. Prima dell'avvento delle reti ad alta velocità, il traffico Web era prevalentemente basato su testo e immagini statiche. Tuttavia, con la disponibilità di connessioni più veloci, si è verificato un aumento esponenziale nel consumo di contenuti multimediali ad alta definizione, streaming video, giochi online e applicazioni interattive.

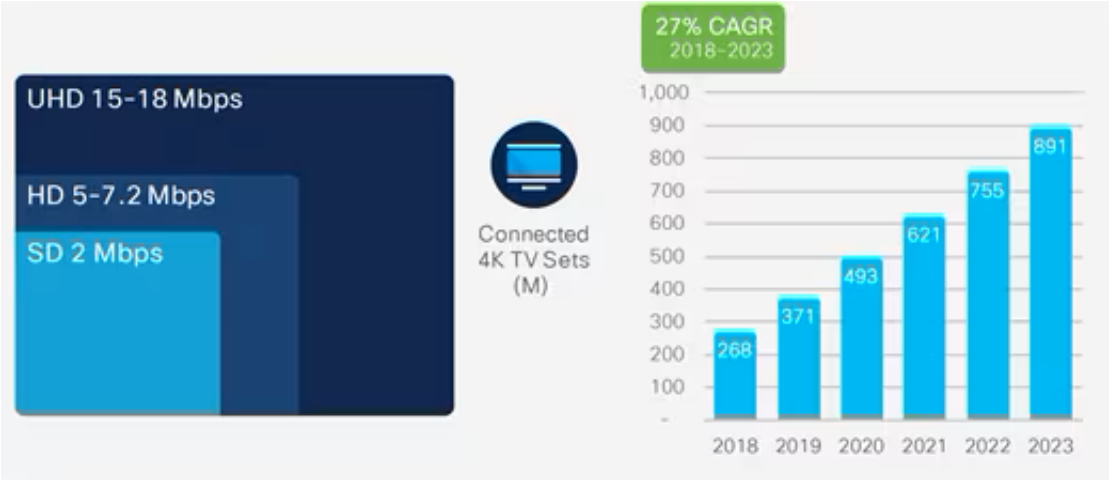


Figura 1.3: Proiezioni Cisco per l'aumento della definizione video: Entro il 2023, il 66% dei televisori a schermo piatto connessi avranno una risoluzione 4K. [Cis20]

Uno dei cambiamenti più evidenti nel traffico Internet è l'esplosione dello streaming video. Piattaforme come Netflix, YouTube e altre offrono una vasta gamma di contenuti video, da film e serie TV a tutorial e video musicali. Le reti ad alta velocità hanno reso possibile lo streaming di contenuti ad alta definizione senza interruzioni, portando a una maggiore adozione di questa modalità di consumo di contenuti.

1.3.1 Giochi Online e Realtà Virtuale

Le reti ad alta velocità hanno anche favorito lo sviluppo e la diffusione dei giochi online e della realtà virtuale. I giochi multiplayer online e le esperienze di realtà virtuale richiedono una connessione affidabile e veloce per garantire un'esperienza di gioco fluida e coinvolgente. Questo ha portato a un aumento nella domanda di larghezza di banda e a una crescita dell'industria dei giochi online.

1.3.2 Lavoro e Istruzione a Distanza

La pandemia da COVID-19 ha ulteriormente accentuato l'importanza delle reti ad alta velocità. Il lavoro da remoto e l'istruzione a distanza sono diventati la norma per molte persone in tutto il mondo. Videoconferenze, webinar e piattaforme di e-learning richiedono connessioni affidabili e veloci per consentire la partecipazione senza problemi.

1.3.3 Internet delle Cose (IoT)

La crescente adozione dell'Internet delle Cose ha introdotto un ulteriore cambiamento nel traffico Internet. Dispositivi come smart home, sensori industriali e dispositivi wearable generano un flusso costante di dati che richiede una connettività stabile e veloce. Questi dispositivi sono diventati parte integrante delle nostre vite quotidiane, contribuendo al traffico complessivo.

1.4 Impatti Economici e Sociali

Region	2018	2019	2020	2021	2022	2023	CAGR (2018–2023)
Global	45.9	52.9	61.2	77.4	97.8	110.4	20%
Asia Pacific	62.8	74.9	91.8	117.1	137.4	157.1	20%
Latin America	15.7	19.7	34.5	41.2	51.5	59.3	30%
North America	56.6	70.1	92.7	106.8	126.0	141.8	20%
Western Europe	45.6	53.2	72.3	87.4	105.6	123.0	22%
Central and Eastern Europe	35.0	37.2	57.0	65.5	77.8	87.7	20%
Middle East and Africa	9.7	11.7	25.0	29.0	34.9	41.2	33%

Figura 1.4: Proiezioni Cisco per La velocità della banda larga in base alle regioni [Cis20]

L'avvento delle reti ad alta velocità ha avuto profondi impatti economici e sociali. Da un lato, ha creato nuove opportunità commerciali e mercati emergenti legati all'infrastruttura delle reti e ai servizi online. D'altro canto, ha portato a una maggiore dipendenza dalla connettività e ha sollevato questioni sulla digital divide, poiché non tutte le regioni e le popolazioni hanno accesso alle stesse velocità di connessione. I progressi della tecnologia sono il principale motore della crescita economica, ma hanno anche portato a una maggiore incidenza di attacchi informatici. Le principali tendenze, come l'e-commerce, i pagamenti mobili, il cloud computing, i Big Data e l'analisi, l'IoT, l'AI, l'apprendimento automatico e i social media, aumentano il rischio informatico per utenti e aziende. Ad aggravare il problema, la natura delle minacce sta diventando sempre più varia. L'elenco comprende DDoS (Distributed Denial-of-Service), ransomware, APT (Advanced Persistent Threats), virus, worm, malware, spyware, botnet, spam, spoofing e phishing. [Cis20]

1.5 Conclusioni

Le reti ad alta velocità hanno rivoluzionato la nostra vita quotidiana, trasformando il modo in cui comunichiamo, apprendiamo, lavoriamo e ci divertiamo. L'evoluzione delle connessioni Internet ha determinato un cambiamento nel tipo di traffico

prodotto dagli utenti, spostandosi da contenuti testuali a contenuti multimediali interattivi, streaming video e applicazioni basate sulla connettività stabile e veloce. Questo cambiamento ha influenzato profondamente l'economia digitale e la società nel suo complesso, aprendo nuove opportunità e sfide nel panorama tecnologico in continua evoluzione.

Capitolo 2

Le Content Delivery Networks (CDN): Concetto, Implementazione e Utilizzo

Le Content Delivery Networks (CDN), rappresentano una componente chiave dell'infrastruttura Internet moderna, sviluppata per ottimizzare la distribuzione di contenuti digitali ai fruitori online. Le CDN si collocano all'intersezione tra tecnologie di rete avanzate e ottimizzazione dell'esperienza utente, consentendo una distribuzione efficiente dei contenuti su scala globale.

2.1 Definizione e Caratteristiche

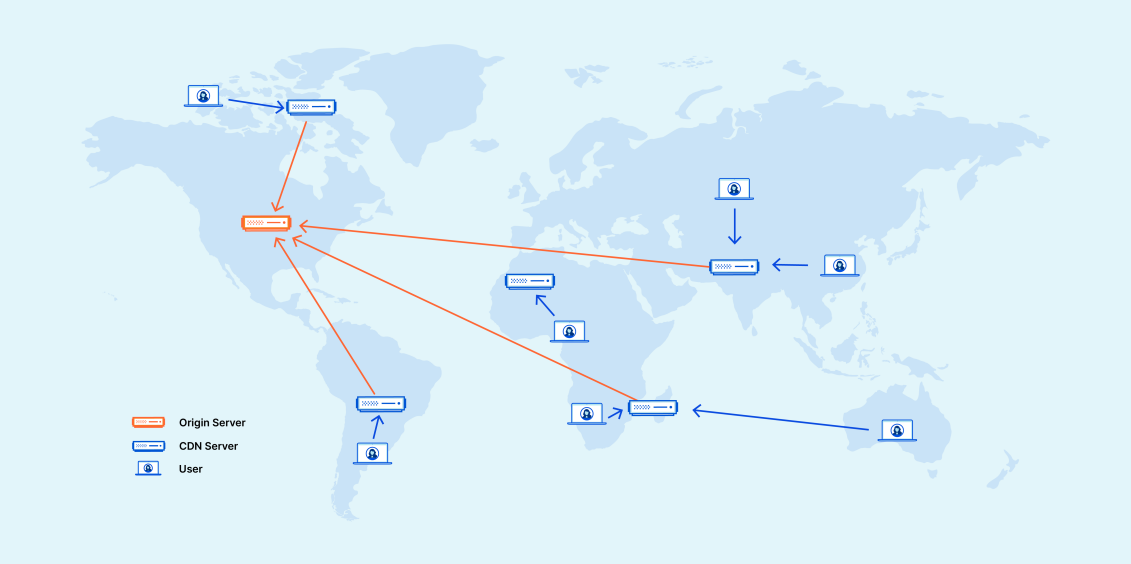


Figura 2.1: Infrastruttura CDN [Clo23]

Una CDN è una rete di server geograficamente distribuiti in punti strategici, chiamati "punti di presenza" (PoP), ognuno dei quali contiene una copia cache dei contenuti web. Questi contenuti possono includere immagini, video, documenti, pagine web e altri elementi multimediali. L'obiettivo principale di una CDN è quello di fornire

contenuti ai fruitori con la massima efficienza e velocità, riducendo il carico sui server di origine e migliorando la latenza per gli utenti finali.

2.2 Implementazione e Funzionamento

Le CDN vengono implementate da fornitori specializzati che dispiegano una rete globale di server. Quando un utente richiede un contenuto, la richiesta viene indirizzata automaticamente al server CDN più vicino geograficamente all'utente.

Questo server, chiamato "server edge" o "server di bordo", contiene una copia in cache del contenuto richiesto. Se il contenuto è già presente nella cache e non è stato modificato dall'ultima richiesta, il server edge può fornirlo direttamente all'utente, evitando di dover recuperare il contenuto dal server di origine.

Se il contenuto non è presente nella cache o è stato modificato, il server edge effettua una richiesta al server di origine per ottenere la versione più recente del contenuto. Dopo aver ottenuto il contenuto, il server edge lo memorizza nella cache per future richieste. Questo approccio riduce notevolmente la latenza e il carico sui server di origine, migliorando l'esperienza dell'utente.

2.2.1 Metodi di reindirizzamento alle CDN [WHR18]

Gli approcci esistenti per l'indirizzamento ai server edge includono il reindirizzamento HTTP, riscrittura degli URL, anycast e reindirizzamento dei server basato su DNS.

- Il reindirizzamento HTTP consente a un server Web di propagare il risultato della selezione del server all'utente finale tramite le intestazioni HTTP. In questo modo, l'utente finale può essere reindirizzato al server ottimale seguendo la risposta generata dal server Web. La debolezza del reindirizzamento HTTP risiede nella la sua dipendenza dal supporto del lato server al lato client. Inoltre, il reindirizzamento HTTP non è una soluzione leggera, poiché in ogni sessione HTTP viene introdotto un ulteriore ritardo di andata e ritorno e i costi di elaborazione dell'HTTP non sono trascurabili.

- Nella riscrittura degli URL, il server di origine riscrive i collegamenti URL delle pagine generate per indicare il miglior server surrogato. In seguito alle risposte riscritte, il cliente può essere reindirizzato in modo ottimale. Il costo principale della riscrittura degli URL è il ritardo nell'analisi degli URL. Peggio ancora, il costo è destinato ad aumentare perché l'URL riscritto non è memorizzabile per l'utente finale.Questa tecnica si rivolge principalmente alla distribuzione di contenuti con oggetti incorporati in risposta alle richieste dei client.

- Anycast è una tecnologia di livello rete per la selezione trasparente dei server e il reindirizzamento. In questo approccio, lo stesso indirizzo IP viene assegnato a più server a più server surrogati situati in modo distribuito. Quando il client invia richieste richieste all'indirizzo IP, le richieste vengono instradate al server surrogato più vicino definito dal server surrogato più vicino definito dal criterio di instradamento. Si noti che il reindirizzamento del server abilitato da anycast non è tecnicamente controllabile dai fornitori di contenuti pertanto, l'uso di anycast è controverso. Da un lato, la sua trasparenza sia per i fornitori di

contenuti che per gli utenti finali può essere un vantaggio, in quanto fornitori di contenuti sono liberati dall'overhead dell'instradamento delle richieste. D'altra parte i fornitori di contenuti possono perdere un po' di flessibilità nella selezione dei server, per esempio in uno scenario in cui anycast inoltra le richieste al server più vicino (ma sovraccarico), semplicemente rispettando una politica di instradamento della rete.

- Il reindirizzamento dei server basato su DNS si appoggia al reindirizzamento nelle risposte alle query DNS. Si tratta di una soluzione flessibile e leggera, pienamente compatibile con l'infrastruttura DNS esistente. Per questo motivo, ha guadagnato una notevole popolarità nelle importanti reti CDN come Akamai, Limelight Networks e Mirror Image. Essendo un substrato indispensabile dell'Internet di oggi, il DNS si comporta come una directory distribuita a livello globale. Il suo ruolo principale è quello di mappare i nomi di dominio agli indirizzi IP corrispondenti. In una tipica sessione DNS, il client invia prima una richiesta al server DNS ricorsivo designato. Un server DNS ricorsivo è gestito dagli ISP (Internet Service Provider) o fornito come servizio pubblico a tutti gli utenti del mondo (ad esempio, il servizio DNS di Google ospitato all'indirizzo 8.8.8.8). Il server DNS ricorsivo risolve la richiesta per conto del cliente. Passa iterativamente attraverso i server DNS autorevoli seguendo l'albero DNS fino a ottenere la risposta finale. Il server DNS ricorsivo non solo consegna la risposta al cliente, ma la memorizza anche nella cache per future interrogazioni. Un fornitore di CDN è probabile che gestisca i server autoritari come parte integrante del suo servizio di distribuzione dei contenuti.

2.3 Chi Installa le CDN

Ci sono 2 principali tipi di aziende che installano e gestiscono le CDN:

- **Aziende e Fornitori di Contenuti**: Le CDN sono installate principalmente da aziende, come fornitori di contenuti digitali, siti web di grandi dimensioni e servizi di streaming video. Queste entità cercano di migliorare l'esperienza degli utenti fornendo contenuti in modo più efficiente e affidabile. Inoltre, le CDN aiutano a gestire il carico di lavoro e a garantire la disponibilità dei contenuti anche durante picchi di traffico.

- **Provider di Servizi CDN**: Esistono anche provider di servizi CDN specializzati, come Akamai, Cloudflare e Amazon CloudFront. Questi provider mettono a disposizione delle aziende e dei fornitori di contenuti l'infrastruttura CDN, semplificando l'implementazione e la gestione delle CDN. Essi dispongono di una vasta rete di server globali e offrono servizi di caching, ottimizzazione delle prestazioni e sicurezza.

2.4 Conclusioni

I principali benefici delle CDN includono:

- **Riduzione della Latenza**: Riducono i tempi di caricamento dei contenuti migliorando l'esperienza dell'utente e diminuendo il rischio di abbandono del sito.

- **Riduzione del Carico**: Distribuiscono il traffico su più server, riducendo il carico sui server di origine e prevenendo eventuali congestioni di rete.
- **Affidabilità**: Forniscono ridondanza e tolleranza ai guasti, poiché i contenuti possono essere forniti da server diversi in caso di problemi.
- **Risparmio di Larghezza di Banda**: Riducono il consumo di larghezza di banda sulla rete di origine, poiché molte richieste possono essere soddisfatte localmente dai server edge.
- **Distribuzione Globale**: Consentono di distribuire contenuti in tutto il mondo, migliorando la latenza e l'accessibilità per gli utenti in diverse regioni geografiche.

Le Content Delivery Networks sono una componente fondamentale dell'ecosistema Internet moderno, migliorando l'efficienza e l'esperienza utente attraverso la distribuzione ottimizzata dei contenuti digitali. Grazie all'implementazione di server edge geograficamente distribuiti e alla cache dei contenuti, le CDN contribuiscono a ridurre la latenza, il carico sui server di origine e i tempi di caricamento dei contenuti, fornendo un'infrastruttura cruciale per il mondo digitale in continua evoluzione.

Capitolo 3

L'Importanza delle Simulazioni su Prototipi di Rete con Cache e il Ruolo del Protocollo UDP e QUIC

Le simulazioni su prototipi di rete che includono cache sono fondamentali per valutare e ottimizzare le prestazioni dei sistemi di distribuzione delle informazioni su Internet. Questi test sono cruciali per garantire che le reti siano in grado di gestire il crescente volume di dati e le richieste degli utenti in modo efficiente. Inoltre, il protocollo UDP, che in passato poteva sembrare obsoleto per alcune applicazioni, ha trovato nuova vita e significato con l'avvento di QUIC.

3.1 Ottimizzazione delle Prestazioni della Rete

Simulare prototipi di rete con cache consente di testare diverse configurazioni e strategie di distribuzione dei contenuti al fine di migliorare le prestazioni complessive. Questo è essenziale, soprattutto considerando che il contenuto memorizzato nella cache può variare notevolmente in base alle esigenze degli utenti e ai cambiamenti nel traffico di rete. Le simulazioni consentono di ottimizzare le politiche di caching e di valutare l'impatto sulle prestazioni globali della rete.

3.1.1 Valutazione delle Prestazioni del Protocollo UDP

Il protocollo UDP (User Datagram Protocol) è noto per la sua velocità e la mancanza di meccanismi di controllo di errore e di flusso. Tuttavia, questa mancanza di affidabilità lo ha reso inadatto per alcune applicazioni, come il trasferimento affidabile di dati sensibili. Tuttavia, è il protocollo a livello di trasporto su cui ancora si basa il protocollo DNS (fondamentale per il funzionamento di internet) e con l'introduzione di QUIC (Quick UDP Internet Connections) il panorama sta cambiando. [Jam17]

3.2 QUIC: Un protocollo di trasporto multiplexato e sicuro basato su UDP

Figura 3.1: Quic Logo

QUIC è un protocollo di trasporto sviluppato da Google che utilizza UDP come base. Introdotto nel RFC 9000 [J I21], è progettato per migliorare la velocità e l'affidabilità delle connessioni Internet. QUIC introduce meccanismi di controllo di errore e di flusso a livello di trasporto, rendendo UDP altamente adatto a una vasta gamma di applicazioni, tra cui il web, il video streaming e le comunicazioni in tempo reale.

3.2.1 Velocità e Sicurezza

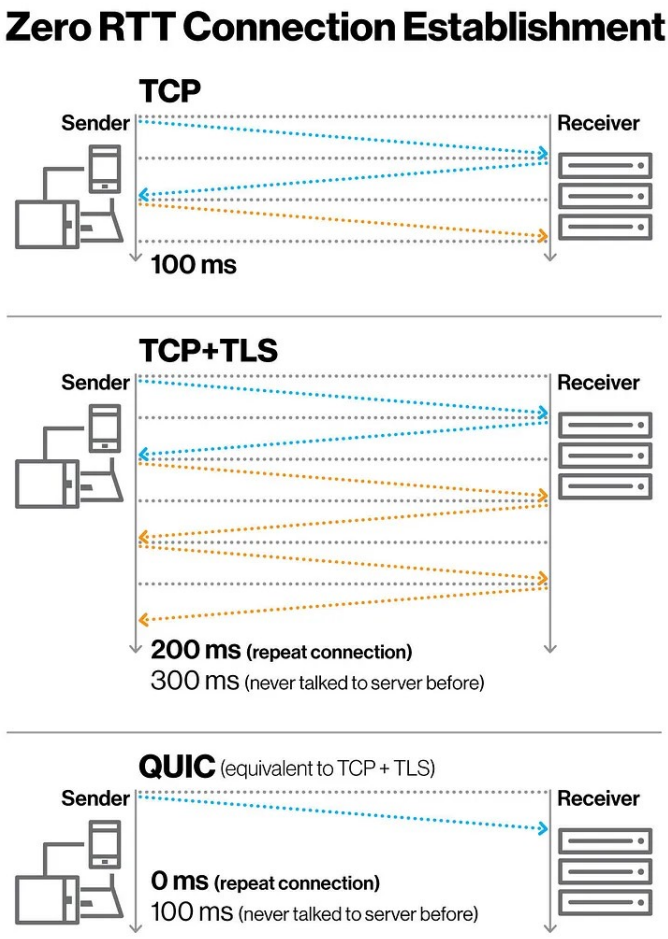


Figura 3.2: Tcp vs QUIC [Oro18]

QUIC offre una latenza inferiore rispetto a TCP (Transmission Control Protocol), il che lo rende ideale per le applicazioni che richiedono tempi di risposta minimi, come la visualizzazione di pagine web e lo streaming di contenuti. QUIC si affida su un protocollo di sicurezza integrato basato su TLS (Transport Layer Security), garantendo una connessione crittograficamente sicura, inoltre consente di inviare e ricevere dati su più "flussi" (multiplexing) all'interno di una singola connessione, questo significa che più richieste e risposte possono essere gestite simultaneamente, migliorando l'efficienza delle comunicazioni.

3.2.2 Benefici per le Applicazioni di Cache

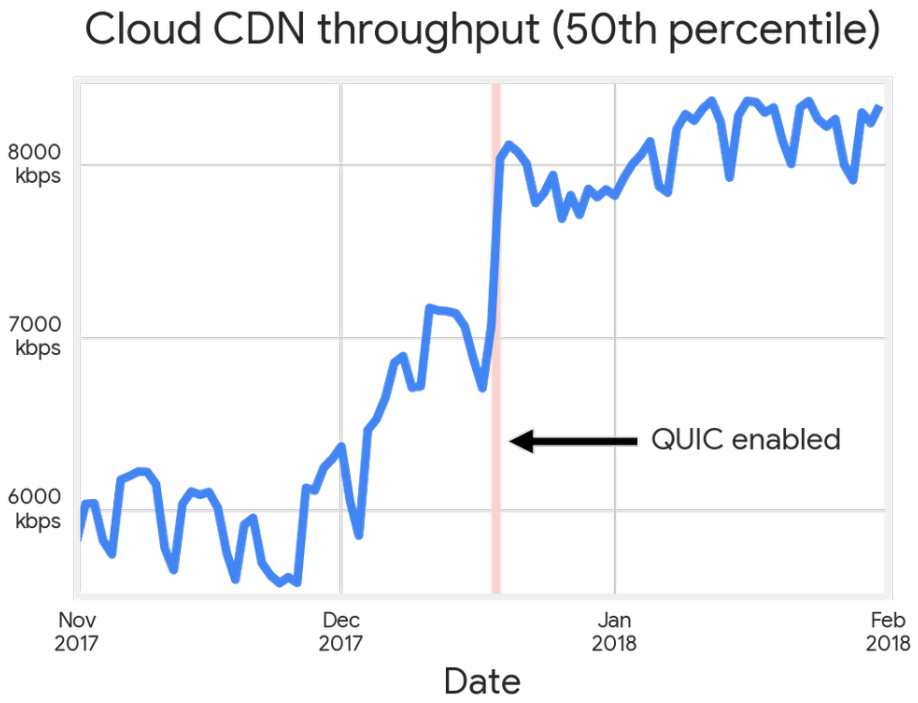


Figura 3.3: Benchmark di Cedexis sui Cloud CDN di Google [Ian18]

Le simulazioni su prototipi di rete che includono cache diventano ancora più rilevanti con l'uso di QUIC. L'ottimizzazione della distribuzione dei contenuti e delle politiche di caching diventa cruciale quando si utilizza QUIC per garantire un'esperienza utente ottimale.

In conclusione, le simulazioni su prototipi di rete con cache rimangono un elemento fondamentale per testare e ottimizzare le reti di distribuzione delle informazioni. Tuttavia, il ruolo del protocollo UDP sta vivendo una rinascita grazie a QUIC, che combina la velocità di UDP con la sicurezza e l'affidabilità richieste dalle applicazioni moderne. Di conseguenza, le simulazioni su reti con cache diventano ancor più pertinenti quando si considera l'utilizzo di QUIC, poiché consentono di massimizzare i benefici di questo nuovo protocollo in evoluzione.

Capitolo 4

Implementazione e simulazione del meccanismo di cache a livello rete: NS3 Network Simulator

Figura 4.1: NS3 logo [Ns323]

Citando la descrizione sul sito ufficiale "ns-3 is a discrete-event network simulator, targeted primarily for research and educational use. The goal of the ns-3 project is to develop a preferred, open simulation environment for networking research: it should be aligned with the simulation needs of modern networking research and should encourage community contribution, peer review, and validation of the software."[Ns323]

NS-3 è un software open source per la simulazione in tempo reale di prototipi di reti internet. L'architettura NS-3 è simile a quella del sistema operativo Linux, con interfacce interne (da rete a driver di dispositivo) e socket di interfacce applicative) che si adattano bene al modo in cui i computer sono costruiti oggi. NS-3 organizza le sue funzionalità nei cosiddetti moduli ed è programmabile utilizzando C++.

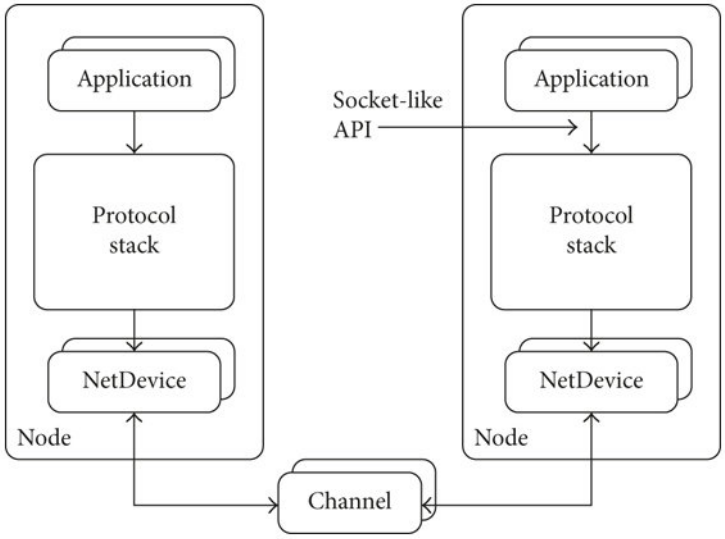


Figura 4.2: Struttura di NS3 [And18]

Questi sono i moduli più importanti:

- **Rete**: Questo modulo contiene i modelli di tutti i componenti base di una rete. Esso
- **Nodo**: Un nodo è l'endpoint della comunicazione. Un esempio di nodo è un computer in una LAN.
- **NetDevice**: Un NetDevice è il mezzo utilizzato da un nodo per interagire con la rete, come un adattatore Ethernet all'interno di un computer (il modulo di rete contiene anche un modello dell'indirizzo MAC).
- **Socket**: Un socket, proprio come nei sistemi operativi, è un'astrazione software che consente a un'applicazione di comunicare attraverso la rete.
- **Applicazione**: Un'applicazione è l'equivalente di un programma in esecuzione su un computer. Un'applicazione viene eseguita su un nodo specifico. Può utilizzare i socket per comunicare con altre applicazioni in esecuzione su altri nodi.
- **Internet**: Il modulo Internet contiene il modello per tutti i protocolli normalmente utilizzati Internet, come IPv4, IPv6, ICMP, TCP (con le sue varie versioni), UDP, ecc.
- **Moduli dispositivi**: Ci sono molti moduli che modellano vari tipi di dispositivi, come dispositivi Wi-Fi.

Capitolo 5

Uno strumento di sviluppo: Docker

Figura 5.1: Docker logo [Doc23a]

Docker è una piattaforma open-source che consente la creazione, la distribuzione e l'esecuzione di applicazioni all'interno di contenitori. I container Docker sono un modo per isolare e distribuire applicazioni e le relative dipendenze in modo efficiente e riproducibile.

Le piattaforme basate su container di Docker consentono carichi di lavoro altamente trasportabili. I container Docker possono essere eseguiti sul laptop locale di uno sviluppatore, su macchine fisiche o virtuali in un data center, su provider cloud o in una combinazione di ambienti. La portabilità e la leggerezza di Docker facilitano inoltre la gestione dinamica dei carichi di lavoro, aumentando o riducendo le applicazioni e i servizi in base alle esigenze aziendali, in tempo quasi reale.[Doc23b]

5.1 Struttura di Docker

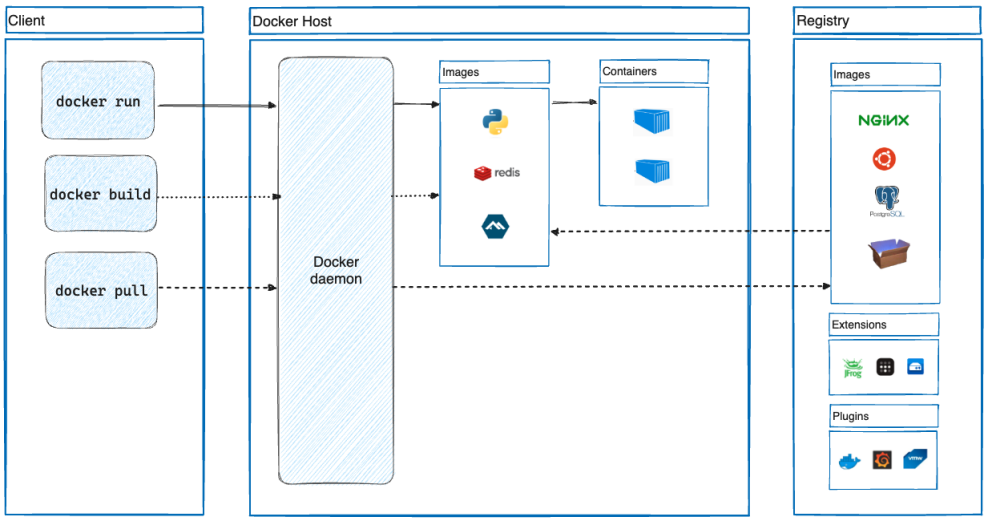


Figura 5.2: Docker architecture [Doc23b]

Ecco un riassunto della struttura di Docker:

- **Il Docker Engine** è il cuore di Docker. È un'applicazione leggera che gestisce i container. Include il server Docker, il daemon, l'API Docker e una serie di strumenti da riga di comando che consentono agli utenti di interagire con Docker. Il Docker Engine è responsabile della gestione dei container sul sistema host.
- **Un'immagine Docker** è un pacchetto di sola lettura che contiene tutti i file e le informazioni necessarie per creare un container. Le immagini sono la base per la creazione di container Docker. Ogni immagine è costituita da strati (layers), che rappresentano modifiche incrementali all'immagine di base. Questo sistema di strati consente di condividere e riutilizzare immagini in modo efficiente.
- **Un container Docker** è un'istanza in esecuzione di un'immagine Docker. È un ambiente isolato che include l'applicazione, le sue dipendenze e le risorse del sistema operativo necessarie per eseguire l'applicazione. I container sono leggeri, portabili e possono essere avviati, arrestati, spostati o eliminati facilmente. Ogni container è isolato dagli altri e dal sistema host, ma può comunque comunicare con il mondo esterno attraverso le porte e le reti virtuali.
- **Un Dockerfile** è un file di testo che contiene le istruzioni per la creazione di un'immagine Docker. Le istruzioni nel Dockerfile specificano cosa deve essere incluso nell'immagine, come configurare l'ambiente e come avviare l'applicazione. Con un Dockerfile, è possibile automatizzare il processo di creazione di immagini Docker, rendendolo riproducibile e gestibile.
- **Docker Compose** è uno strumento che semplifica la gestione di applicazioni multi-container. È utilizzato per definire e avviare un'applicazione composta da più container, specificando le dipendenze, le reti, le variabili d'ambiente e altre configurazioni. Questo rende facile orchestrare l'avvio di più container come un'applicazione unica.

- **Un Docker Registry** è un repository in cui è possibile archiviare e condividere immagini Docker. Il Docker Hub è uno dei registri pubblici più noti, ma è possibile creare registri privati per archiviare e distribuire immagini all'interno di un'organizzazione.

5.2 Il mio caso d'uso

Ho scelto di creare un'immagine Docker installandoci NS3 versione 3.38 per i seguenti motivi:

- NS3 va compilato da sorgente per poterlo utilizzare e non conoscendolo mi è successo alcune volte di comprometterne il funzionamento esplorandone le funzionalità, avendolo in un container mi bastava eliminarlo e farne partire uno nuovo.
- Dato che avevo necessità di poter lavorare da macchine con sistemi operativi diversi e anche con architetture diverse, creare un'immagine sul Docker Hub mi ha permesso di non perdere tempo nella configurazione del simulatore nel momento in cui io cambiassi computer.
- Viene consigliato l'utilizzo di linux come sistema operativo su cui eseguire NS3 ma non essendo un SO che uso stabilmente avrei dovuto compilare NS3 su Windows/MacOs incappando probabilmente in problemi di compatibilità, la possibilità di utilizzare il kernel linux come base per l'immagine Docker ha risolto questo problema diminuendo anche l'overhead sulla macchina host provocato da un'eventuale Macchina Virtuale Linux completa (es VirtualBox) nel caso non avessi scelto Docker.

5.3 Codice Docker

Ecco di deguito Il Docker-compose e il dockerfile da me utilizzati:

```
1  version: "3"
2
3  services:
4    client:
5      build:
6        context: "./"
7        dockerfile: "Dockerfile"
8      container_name: ns3-server
9      hostname: ns3-server
10     tty: true
11     cap_add:
12       - "ALL"
13     command: bash -c "/bin/bash"
```

Listing 5.1: Docker-compose

```
FROM ubuntu:20.04
RUN apt-get update
RUN apt-get upgrade -y

ARG DEBIAN_FRONTEND=noninteractive TZ=Europe/Rome
RUN apt-get install build-essential libsqlite3-dev libboost-all-dev
     libssl-dev git python3-setuptools castxml -y
ARG DEBIAN_FRONTEND=noninteractive TZ=Europe/Rome
RUN apt-get install gir1.2-goocanvas-2.0 gir1.2-gtk-3.0
    libgirepository1.0-dev python3-dev python3-gi python3-gi-cairo
    python3-pip python3-pygraphviz python3-pygccxml -y
ARG DEBIAN_FRONTEND=noninteractive TZ=Europe/Rome
RUN apt-get install wget g++ pkg-config sqlite3 qt5-default
    mercurial ipython3 openmpi-bin openmpi-common openmpi-doc
    libopenmpi-dev autoconf cvs bzr unrar gdb valgrind uncrustify
    doxygen graphviz imagemagick python3-sphinx dia tcpdump libxml2
    libxml2-dev cmake libc6-dev libclang-6.0-dev llvm-6.0-dev
    automake -y

WORKDIR /home

RUN wget -c https://www.nsnam.org/releases/ns-allinone-3.38.tar.bz2

RUN tar -xvjf ns-allinone-3.38.tar.bz2

WORKDIR /home/ns-allinone-3.38/ns-3.38

RUN ./ns3 configure --disable-werror --enable-examples --enable-
    tests

RUN ./ns3

RUN ./test.py

# start a new terminal on entry
CMD /bin/sh
```

Listing 5.2: Docker File

Capitolo 6

I Moduli NS3 che ho implementato: Traffic generator, CDN, CP

6.1 Application NS3

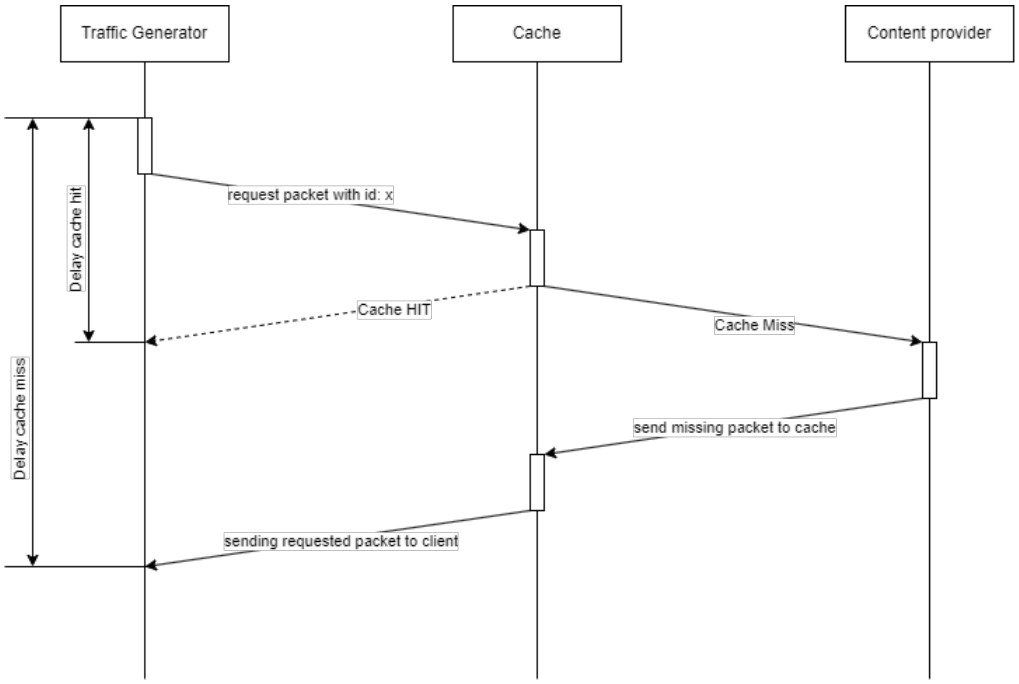


Figura 6.1: Schema di funzionamento

Per simulare il comportamento di client, CDN, CP sono state implementate 3 Application NS3 (si veda l'immagine 4.2) con il seguente comportamento:

- **Traffic generator**: come dice il nome ha lo scopo di generare traffico udp simulando dei client
- **CDN**: in questo caso d'uso la CDN viene modellata come una cache che risponde alle richieste del traffic generator e se l'id del pacchetto richiesto non è presente nella cache viene richiesto al CP e poi all'arrivo inoltrato al traffic generator.
- **CP**: il Content Provider viene simulato come un echo UDP

6.1.1 Distribuzione statistica del traffico

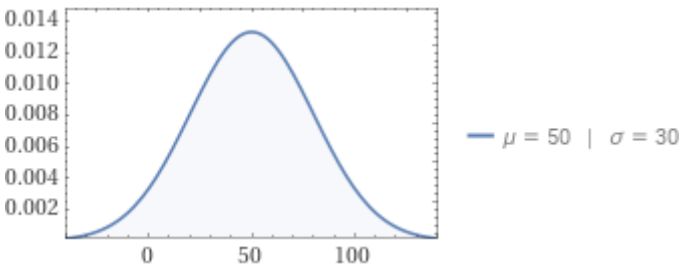


Figura 6.2: Esempio di distribuzione normale utilizzata (asse x id del pacchetto, asse y probabilità di invio)

Essendo in una simulazione di traffico su internet uno dei metodi più efficaci per simulare il traffico generato dagli utenti è la distribuzione normale perché modella accuratamente la variabilità naturale degli utenti, si allinea con la legge dei grandi numeri e la teoria del limite centrale, ed è facilmente manipolabile e implementabile. Utilizzando questa distribuzione, è possibile ottenere simulazioni realistiche del comportamento degli utenti su Internet, il che è essenziale per progettare e ottimizzare reti e servizi online.

6.1.2 Implementazione della cache: Hashmap e Coda

```
1 std::deque<uint32_t> m_cache;
2 std::multimap<uint32_t, Address> requestQueue;
```

Listing 6.1: Oggetti della libreria std utilizzati nella cache

Lista id contenuti presenti(m_cache)

```
1 void UdpCacheServer::pushInCache(const uint32_t& item) {
2     if (m_cache.size() == m_cacheSize) {
3         m_cache.pop_front();
4     }
5     m_cache.push_back(item);
6 }
```

Listing 6.2: La cache è implementata come un buffer di grandezza scelta durante la configurazione della simulazione(m_cacheSize) di variabili uint32_t (rappresentanti gli id dei pacchetti) con comportamento FIFO quando risulta piena.

Coda delle richieste(requestQueue)

Per gestire la possibilità che più client (o più traffic generator) utilizzino la cache, viene implementata una multimap con chiave uint32_t (id del pacchetto) e come risorsa uno o più Address per gestire anche l'eventualità che più utenti richiedano lo stesso id risorsa e per risparmiare una chiamata al content provider inserendo il nuovo Address nella multimap. Quando la risorsa sarà ricevuta dalla cache verrà inviata a tutti gli address inseriti nella multimap con chiave l'id del pacchetto arrivato.

```
1  void
2  UdpCacheServer::HandleReadClients(Ptr<Socket> socket)
3  {
4      /* log ns3 */
5
6      Ptr<Packet> packet;
7      Address from;
8      Address localAddress;
9      while ((packet = socket->RecvFrom(from)))
10     {
11         /*
12             funzioni interne ad ns3 di log e di traccia dei
       pacchetti
13         */
14
15         uint32_t value_from_pkt = getIdPacket(packet);
16         accesscount++;
17
18         /* log ns3 */
19
20         if (cacheContains(value_from_pkt))
21         {
22             sendPacketBackToClient(value_from_pkt, from);
23             hitcount++;
24
25             /* log ns3 */
26         }
27         else
28         {
29             /* log ns3 */
30             requestPacketToContentServer(value_from_pkt);
31             prefetchData(value_from_pkt);
32             requestQueue.insert(std::pair<uint32_t, Address>(
       value_from_pkt, from));
33         }
34
35     }
36 }
```

Listing 6.3: Funzione richiamata quando si scatena l'evento di ricezione di un pacchetto sul socket destinato al traffico in entrata dagli utenti (i commenti sostituiscono funzioni di logica NS3 superflui per comprendere il funzionamento) - udp-cache-server.cc

```
1  void
2  UdpCacheServer::HandleReadServer(Ptr<Socket> socketP2P)
3  {
4      /* log ns3 */
5
6      Ptr<Packet> packet;
7      Address from;
8      Address localAddress;
9
10     while ((packet = socketP2P->RecvFrom(from)))
11     {
12         /*
13             funzioni interne ad ns3 di log e di traccia dei
       pacchetti
14         */
15
16         uint32_t value_from_pkt = getIdPacket(packet);
17
18         /* log ns3 */
19
20         if (!cacheContains(value_from_pkt))
21         {
22             pushInCache(value_from_pkt);
23         }
24
25         std::multimap<uint32_t, ns3::Address>::iterator
       requestQueueIter = requestQueue.find(value_from_pkt);
26
27         while (requestQueueIter != requestQueue.end() &&
       requestQueueIter->first == value_from_pkt)
28         {
29             sendPacketBackToClient(value_from_pkt,
       requestQueueIter->second);
30             requestQueue.erase(requestQueueIter++);
31         }
32
33         /* log evento ns3*/
34     }
35 }
```

Listing 6.4: Funzione richiamata quando si scatena l'evento di ricezione di un pacchetto sul socket destinato al traffico in entrata dal content provider (i commenti sostituiscono funzioni di logica NS3 superflui per comprendere il funzionamento) - udp-cache-server.cc

6.1.3 Ottimizzazione della cache: Pre-fetching

```
1  void UdpCacheServer::prefetchData(uint32_t value) {
2
3      for (size_t i = 1; i < 3 && value-i > 0; i++)
4      {
5          if(!cacheContains(value-i)){
6              requestPacketToContentServer(value-i);
7          }
8      }
9      for (size_t i = 1; i < 3 && value+i < 101; i++)
10     {
11         if(!cacheContains(value+i)){
12             requestPacketToContentServer(value+i);
13         }
14     }
15 }
```

Listing 6.5: Metodo che esegue richieste adiacenti al pacchetto richiesto alla cache - udp-cache-server.cc

Per aumentare le prestazioni della CDN è stato implementato un meccanismo di pre-fetching basato sul modello probabilistico del traffico. Con il pre-fetching si tenta di prevedere il traffico futuro precaricando in cache i pacchetti con id adiacente a quello già richiesto. Nel nostro caso specifico quando un pacchetto viene richiesto e si verifica un cache-miss oltre a richiedere l'id del pacchetto vengono richiesti anche i 3 adiacenti superiori e inferiori.

6.2 Network

La rete che collega i tre nodi viene organizzata con due tipi di canali:

- **CSMA**: impiegata tra i nodi su cui vi sono le application e i loro router
- **P2P**: impiegata per simulare le connessioni point to point tra i router

6.3 Protocollo implementato

Per simulare il traffico utente verso il CP è stato implementato un rudimentale protocollo applicativo ispirato alle API REST ma semplificato e pensato su misura per questa simulazione.

Il messaggio codificato JSON ha tre campi:

- **sender**: tre possibili valori client / cache / server
- **type**: indicante il tipo di messaggio (richiesta oppure risposta)
- **id**: rappresentante l'id di un'ipotetica risorsa richiesta o fornita dal comunicante

```
{
    "sender": "client/cache/server",
    "type": "request/response",
    "id": integer
}
```

Listing 6.6: Prototype del contenuto dei pacchetti UDP

```
uint32_t
UdpCacheServer::getIdPacket(Ptr<Packet> packet)
{
    uint8_t* buffer = new uint8_t[packet->GetSize()];
    packet->CopyData(buffer, packet->GetSize());
    std::string payload = std::string(reinterpret_cast<char*>(
    buffer), packet->GetSize());
    std::regex six_digit_number_regex(R"(\b\d{1,6}\b)");

    std::smatch matches;
    if (std::regex_search(payload, matches,
    six_digit_number_regex))
    {
        std::string id = matches[0];
        return static_cast<uint32_t>(std::stoul(id));
    }
    else
    {
        // handle whenever the id is not found
        return 0;
    }
}
```

Listing 6.7: Metodo per ottenere l'id del pacchetto dal payload udp del pacchetto tramite regex - udp-cache-server.cc

6.4 Codice Moduli NS3 completi

Per evitare di appesantire questo documento il codice relativo ai moduli è reso pubblico nella repository GitHub al seguente link https://github.com/andreafulcheri/ns3-cache-simulations con le relative istruzioni per ripetere le simulazioni.

Capitolo 7

Telemetrie e risultati

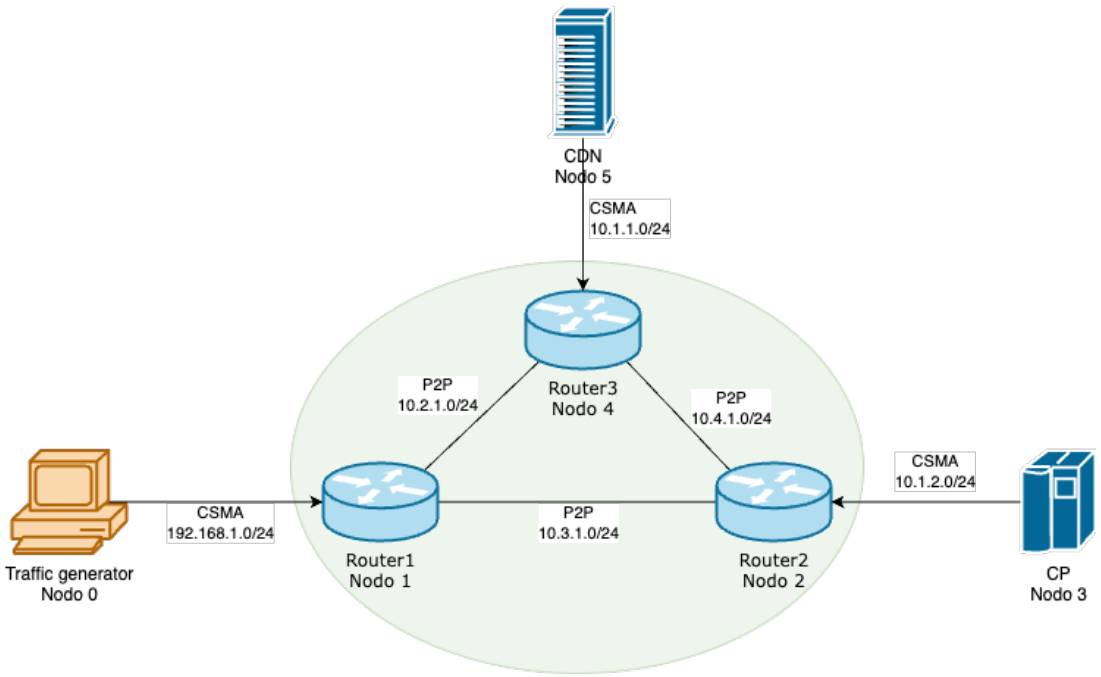


Figura 7.1: La topologia adottata nella simulazione

Allo scopo di simulare il più possibile lo scenario di utilizzo di una CDN è stato messo di proposito un collo di bottiglia sulla rete P2P tra il router relativo al traffic generator e quello del content provider per rendere realistico l'impiego di una CDN.

7.1 Telemetrie utilizzate

Per osservare il comportamento della cache e del flusso di pacchetti sono state utilizzate 3 telemetrie

- **File pcap**: generati automaticamente da ns3 e registrano tutte le attività di rete, inclusi pacchetti di dati, indirizzi IP, protocolli utilizzati e altre informazioni relative alla comunicazione di rete.

- **Conteggio cache hit/cache access**: utile per vedere l'efficacia della cache

- **Timestamp invio e ricezione pacchetti dal traffic generator**: utile per misurare il delay medio percepito dagli eventuali utenti

Nota: i file vengono creati in una directory chiamata "output" nello stesso path dell'eseguibile ns3.

7.2 Simulazione

7.2.1 Setup

L'obiettivo di questa simulazione è di generare un flusso di 100 pacchetti in totale alla velocità di 16 Kbps dal traffic generator, quindi va calcolato l'intervallo tra ogni pacchetto nel modo seguente:

$$\frac{Velocita' \, desiderata \, (Kbps)}{grandezza \, frame \, (b)} = frame \, al \, secondo$$

$$\frac{1}{frame \, al \, secondo} = intervallo \, tra \, ogni \, frame \, (s)$$

tramite i file pcap conosciamo la grandezza del frame essere 784 b, di conseguenza il tempo che intercorre tra ogni pacchetto deve essere 49 ms.

Varianza e media della distribuzione normale (limitata nell'intervallo di numeri naturali $[0; 100]$) del traffic generator sono impostate a:

$$\rho = 100 \, e \, \mu = 50$$

La cache è impostata ad una capienza di 20 id.

I canali della topologia sono così configurate (riferimento a 7.1):

- 3 canali CSMA che collegano i nodi su cui ci sono le 3 application NS3: DataRate 1Mbps, 5 ms delay , MTU 1400

- 2 canali P2P (R2-R3 e R3-R1): DataRate 1Gbps, 15 ms delay

- il canale P2P R1-R2 (collo di bottiglia): DataRate 1Gbps, 100 ms dalay

7.2.2 Risultati

Senza cache

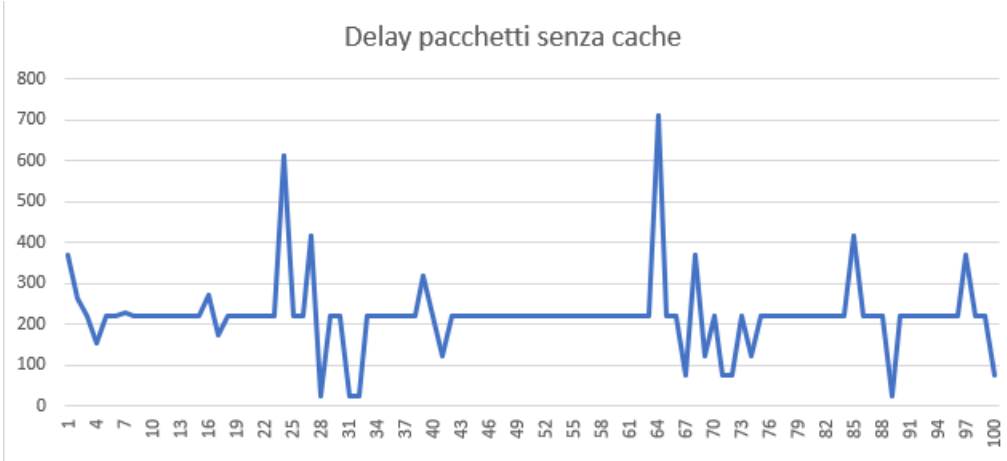


Figura 7.2: Grafico delay pacchetti 1

Delay medio: 224,23 ms

Con cache

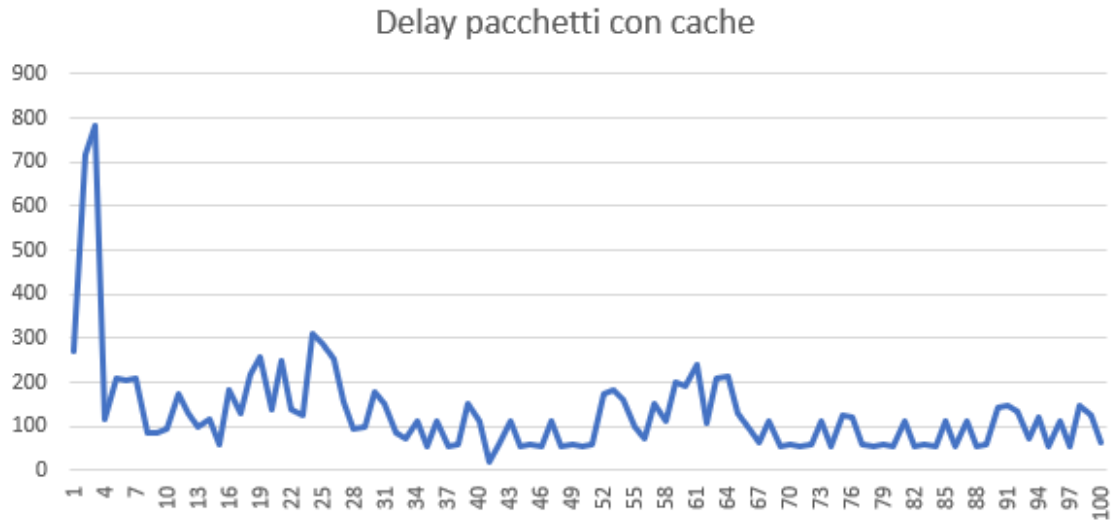
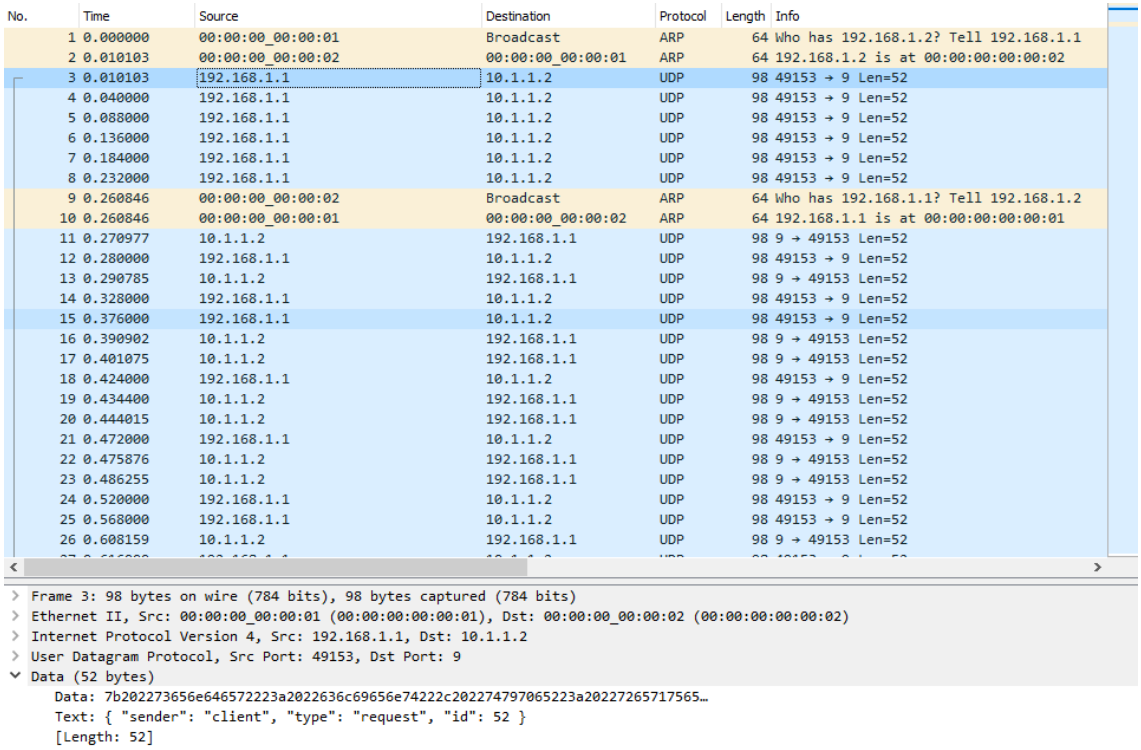


Figura 7.3: Grafico delay pacchetti 2

Cache hit ratio: 49%
Delay medio: 131,01 ms

Esempio di traffico



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:01	Broadcast	ARP	64	Who has 192.168.1.2? Tell 192.168.1.1
2	0.010103	00:00:00_00:00:02	00:00:00_00:00:01	ARP	64	192.168.1.2 is at 00:00:00:00:00:02
3	0.010103	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
4	0.040000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
5	0.088000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
6	0.136000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
7	0.184000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
8	0.232000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
9	0.260846	00:00:00_00:00:02	Broadcast	ARP	64	Who has 192.168.1.1? Tell 192.168.1.2
10	0.260846	00:00:00_00:00:01	00:00:00_00:00:02	ARP	64	192.168.1.1 is at 00:00:00:00:00:01
11	0.270977	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
12	0.280000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
13	0.290785	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
14	0.328000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
15	0.376000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
16	0.390902	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
17	0.401075	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
18	0.424000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
19	0.434400	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
20	0.444015	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
21	0.472000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
22	0.475876	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
23	0.486255	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52
24	0.520000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
25	0.568000	192.168.1.1	10.1.1.2	UDP	98	49153 → 9 Len=52
26	0.608159	10.1.1.2	192.168.1.1	UDP	98	9 → 49153 Len=52

Figura 7.4: Screenshot Wireshark di un pcap generato da NS3

Nota: questo è il traffico catturato nei file pcap sul canale CSMA tra Traffic Generator e Router1, infatti si può vedere che oltre al flusso UDP ci sono anche i pacchetti ARP discovery (Gialli).

7.2.3 Considerazioni

Quando si confrontano simulazioni con e senza l'utilizzo della cache di rete, è essenziale considerare diverse variabili. Innanzitutto, la presenza di una cache di rete può notevolmente migliorare le prestazioni quando le simulazioni coinvolgono l'accesso frequente a dati già memorizzati nella cache. Questo può ridurre drasticamente i tempi di latenza e migliorare il QoS percepito dagli utenti.

D'altra parte, nelle simulazioni in cui i dati richiesti sono altamente variabili e non possono essere facilmente previsti o memorizzati in anticipo nella cache di rete, l'utilizzo della cache potrebbe non portare a miglioramenti significativi delle prestazioni. In questi casi, l'overhead associato alla gestione della cache potrebbe addirittura rallentare il sistema anziché migliorarlo.

In conclusione, le considerazioni sull'effettivo miglioramento delle prestazioni tra le simulazioni senza cache e quelle con cache di rete dipendono dalla natura specifica delle simulazioni, dalla variabilità dei dati coinvolti e dai costi associati all'implementazione della cache.

Un'analisi dettagliata dei requisiti del sistema e del tipo di traffico è essenziale per determinare se l'uso della cache sia una scelta appropriata per ottimizzare le prestazioni.

Bibliografia

[Jam17] Keith W. Ross James F. Kurose. *Reti di calcolatori e internet. Un approccio top-down. Ediz. mylab. Con eText. Con aggiornamento online.* Pearson, 2017.

[And18] Jamal-Deen Abdulai Andy Bubune Amewuda Ferdinand Katsriku. «Implementation and Evaluation of WLAN 802.11ac for Residential Networks in NS-3.» In: *ResearchGate* (2018). URL: https://www.researchgate.net/figure/The-basic-ns-3-wireless-network-model_fig1_323521058.

[Ian18] Michael Behr Ian Swett. «Introducing QUIC support for HTTPS load balancing». In: *Google Cloud Blog* (2018). URL: https://cloud.google.com/blog/products/gcp/introducing-quic-support-https-load-balancing.

[Oro18] Frank Orozco. «How QUIC speeds up all web applications». In: *Edgecast* (2018). URL: https://edgecast.medium.com/how-quic-speeds-up-all-web-applications-62964aadb3d1.

[WHR18] Zheng Wang, Jun Huang e Scott Rose. «Evolution and challenges of DNS-based CDNs». In: *Digital Communications and Networks* 4.4 (2018), pp. 235–243. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2017.07.005. URL: https://www.sciencedirect.com/science/article/pii/S2352864817300731.

[Cis20] Cisco. «Cisco Annual Internet Report (2018–2023) White Paper». In: *Cisco Annual Internet Report* (2020). URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[J I21] M. Thomson J. Iyengar. «QUIC: A UDP-Based Multiplexed and Secure Transport». In: *Internet Engineering Task Force (IETF)* (2021). URL: https://www.rfc-editor.org/rfc/rfc9000.html.

[Clo23] CloudFlare. *Cos'è una CDN?* 2023. URL: https://www.cloudflare.com/it-it/learning/cdn/what-is-a-cdn/.

[Doc23a] Docker. «Docker». In: *Docker Official site* (2023). URL: https://www.docker.com/.

[Doc23b] Docker. «Docker Docs». In: *Docker Official site* (2023). URL: https://docs.docker.com/get-started/.

[Ns323] Ns3. «About NS3». In: *Nsnam site* (2023). URL: https://www.nsnam.org/about/.

Ringraziamenti

Vorrei dedicare qualche riga a coloro che hanno contribuito sostenendomi durante il percorso universitario.

Ringrazio i miei genitori che mi sono sempre stati accanto, con l'enorme pazienza che hanno dimostrato sopportandomi durante i periodi più pesanti.

Grazie al mio gruppo di amici di Torino, che è nato un po' in ritardo a causa del covid, per essere stati di supporto durante i periodi di studio e grazie per tutti i momenti di spensieratezza.

Infine grazie a tutti coloro che non ho citato per il loro contribuito a raggiungere questa tappa importante della mia vita.