Alma Mater Studiorum ·
Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso Laurea Magistrale in Ingegneria Informatica

Sviluppo di un ambiente virtualizzato per la simulazione di attacchi a reti time-sensitive

Relatore:
Chiar.mo Prof.
Marco Prandini

Correlatore:
Andrea Melis
Davide Berardi
Amir Al Sadi

Presentata da:
Andrea Giovine

Sessione III
Anno Accademico
2020/2021

Indice

Acronyms

5G 5th Generation. 7, 14

AS Time-Sensitive Application. 15

BMC Best Master Clock. 28

BMCA Best Master Clock Algorithm. 15

CI Continuous Integration. 51

detnet Deterministic Networking. 50

DoS Denial of Service. 21, 22, 41, 45, 48

FIFO First In First Out. 33

GPS Global Positioning System. 28

gPTP generic Precision Time Protocol. 15

IaC Infrastrucutre as Code. 6

IEEE Institute of Electrical and Electronics Engineers. 2, 14–16, 19–21, 23–25

IETF Internet Engineering Task Force. 50

IT Information Technology. 6

KVM Kernel-based Virtual Machine. 6, 26, 29, 34, 37

MIB Management Information Base. 24

NIC Network Interface Controller. 27

NTP Network Time Protocol. 27

OT Operational technology. 6

QCOW2 QEMU Copy On Write Version 2. 34

qdisc queuing disciplines. 32–34, 48, 49

QEMU Quick EMUlator. 6, 26, 29, 30, 34, 37

QoS Quality of Services. 14, 16, 21, 22, 25, 32

SNMP Simple Network Management Protocol. 23

SPB Shortest Path Bridging. 20

SRP Stream Reservation Protocol. 24

SSH Secure SHell. 26, 34

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege. 9, 10, 21

TC Traffic Control. 32, 35, 41

TSN Time-Sensitive Networking. 6, 7, 9, 14–16, 21, 22, 24–27, 33, 35, 37, 39, 41, 47, 50, 51

VM Virtual Machine. 7, 29, 31, 34–36, 38

Capitolo 1

Introduzione

Questo lavoro di tesi ha visto come obiettivo finale quello di realizzare una serie di attacchi, alcuni di questi totalmente originali, ai protocolli della famiglia Time-Sensitive Networking (TSN) attraverso lo sviluppo di un'infrastruttura virtualizzata. L'infrastruttura è stata costruita e progettata utilizzando macchine virtuali con Quick EMUlator (QEMU) come strato di virtualizzazione ed accelerate attraverso Kernel-based Virtual Machine (KVM). Il progetto è stato concepito come Infrastrucutre as Code (IaC), attraverso l'ausilio di Ansible e alcuni script shell utilizzati come collante per le varie parti del progetto.
La realizzazione di tale infrastruttura è risultata molto impegnativa in quanto sono state stressate alcune problematiche come ad esempio la virtualizzazione degli orologi all'interno delle macchine virtuali.
Il motivo per cui è utile e d'importanza strategica avere un'infrastruttura di test virtuale risiede nel fatto che testare queste tecnologie in ambiente fisico è molto costoso a causa del prezzo dei dispositivi TSN. Inoltre l'utilizzo di dispositivi fisici risulta essere molto più oneroso a livello di tempo rispetto ad una gestione software. La progettazione dell'infrastruttura virtualizzata quindi concorre alla realizzazione di un vero e proprio digital twin per le TSN che, come si analizzerà nel capitolo successivo, risulta un modello all'avanguardia per quanto concerne la creazione di un legame tra il mondo Operational technology (OT) e quello Information Technology (IT). La realizzazione del laboratorio virtualizzato di test TSN è stato concepito per essere IaC. Questa scelta di progettazione non soddisfa solo le moderne tecniche e pratiche per la realizzazione d'infrastrutture tecnologiche allo stato dell'arte. Infatti risulta anche importante poiché la tecnologia impiegata (Ansible) utilizza un linguaggio dichiarativo rendendo autoesplicativo il codice di configurazione e standardizza la configurazione dei vari nodi, in questo caso virtuali. L'utilizzo dell'approccio IaC introduce un

ulteriore vantaggio: qualora il progetto abbia una diffusione, colui che sarà incaricato degli ulteriori sviluppi dovrà solamente imparare la sintassi di Ansible, senza inventare nuove procedure non standard che rendono tediosa la parte di configurazione e deployment delle VM.
In ultima analisi, l'importanza delle TSN nel prossimo futuro risulta ad oggi chiaramente in ascesa. Questi protocolli sono riconosciti come il pilastro per le moderne reti emergenti. La ricerca applicata punta, ad esempio, all'integrazione di questi protocolli per le reti cablate con il 5G. Le applicazioni delle TSN sono legate al mondo dell'industria, specialmente nei dispositivi di sicurezza ma anche nel campo della visione artificiale, per realizzare scanner a bassissima latenza. Vista la possibilità di applicare queste tecnologie sia in ambito militare che avionico risultano indispensabili un'analisi di sicurezza e la realizzazione di un ambiente di test facilmente aggiornabile, al fine di collaudare le vulnerabilità scoperte e sviluppare contromisure adeguate.

1.1 Importanza di avere un'infrastruttura di test virtuale

Al giorno d'oggi la possibilità di avere un'infrastruttura di test virtualizzata è di fondamentale importanza dal punto di vista strategico. Lo stato dell'arte indica al giorno d'oggi la realizzazione di un digital twin come requisito fondamentale. Essendo ad oggi gli switch TSN molto costosi, diventa molto importante avere un ambiente di test e di raccolta delle informazioni digitalizzato. Nella tesi si è ricorso alle più moderne e più mature tecnologie di virtualizzazione, al fine di avere una vera e propria infrastruttura dematerializzata. Lo scopo perseguito è stato: non solo generare un laboratorio virtuale per gli attacchi a questi protocolli, ma anche un banco di prova in cui un'azienda possa testare la resilienza della propria infrastruttura TSN in pochi semplici passi. Così facendo risulta possibile avere un modo per stressare i vari parametri degli algoritmi di scheduling dei pacchetti, fino a comprendere i limiti della propria infrastruttura TSN.

1.1.1 Digital Twin

Il digital twin è la controparte virtuale o digitale di un dato prodotto o infrastruttura. Infatti è fondamentale sia durante lo sviluppo del prodotto sia dopo la sua produzione, ovvero durante la fase di rilascio al cliente, in quanto può essere utile per sondare lo stato del prodotto stesso e permette d'identificare

eventuali malfunzionamenti in modo preventivo. In un digital twin a modello ibrido è possibile raccogliere dati sia sull'impianto/prodotto appena rilasciato sia su quello digitale. Così è possibile avere una sufficiente quantità di dati per fare grandi analisi tramite tecniche di machine learning o qual si voglia tecnologia che ne richieda un massiccio accumulo.

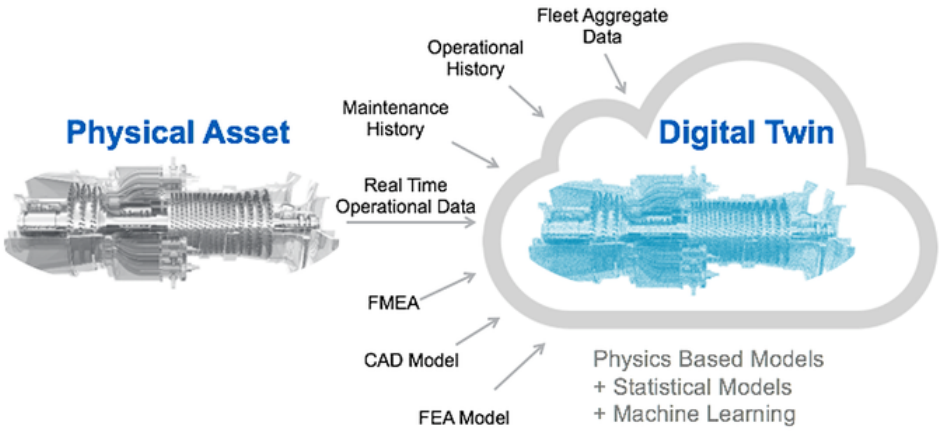

Figura 1.1: Esempio di quello che può fare un Digital Twin [7]

1.2 Modelli di minaccia

Nell'ambito della sicurezza informatica, una minaccia è un'azione potenzialmente negativa oppure un evento facilitato da una vulnerabilità che ha un impatto non desiderato sul sistema. Le minacce possono avere origini eterogenee: essere esterne alla propria organizzazione oppure in alcuni casi anche interne. Le conseguenze di queste minacce possono avere un impatto elevato come ad esempio alcuni attacchi sono in grado di disabilitare completamente i sistemi oppure possono causare perdite d'informazioni sensibili . Questo può portare a dei risultati nefasti sia da parte dei consumatori che potrebbero perdere fiducia nel fornitore dei servizi sia da parte degli investitori. Per prevenire che minacce di vario tipo possano essere usate per prendere il controllo su un sistema sono stati redatti dei metodi che esplorano i vari modelli di minacce esistenti, mettendo sul piatto contromisure ed un piano efficace nel caso l'organizzazione venga presa di mira. Per convenzione i modelli di minaccia tendono a creare:

- un modello astratto del sistema,

- profili di potenziali attaccanti, cercando anche di prevedere come ed in che modo possano attaccare,
- un catalogo di potenziali minacce che possono presentarsi

Nel corso degli anni sono stati sviluppati vari modelli per catalogare le minacce, che possono essere combinati tra loro perché alcuni hanno un focus maggiore sulle tematiche di privacy altri sulla gestione del rischio etc.. I più moderni esempi di modelli di gestione delle minacce sono [20]:

- STRIDE
- PASTA
- LINDDUN
- CVSS
- Attack Trees
- Persona non Grata
- Security Cards
- hTMM
- Quantitative Threat Modeling Method
- Trike
- VAST Modeling
- OCTAVE

Di conseguenza è importante evidenziare che non esiste un modello che prevale su un altro: bisogna scegliere quello che serve in base alle proprie necessità. Per il lavoro di tesi si è deciso di utilizzare il modello STRIDE per fare delle valutazioni di carattere generale sulla sicurezza della suite di protocolli TSN.

1.2.1 STRIDE

[9] Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege (STRIDE) è un modello per identificare le minacce di sicurezza informatica sviluppato da Praerit Garg e Loren Kohnfelder presso Microsoft. Il modello è risultato molto efficace per le minacce sui sistemi puramente informatici e quelli cyber-fisici. STRIDE è sicuramente anche il modello

più maturo a nostra disposizione [20]. STRIDE per fare le sue valutazioni valuta il progetto del sistema, in modo dettagliato, ovvero si va a valutare le possibili minacce al modello non alla sua implementazione. Lo scopo del primo step è quello di modellare il sistema posto in essere tramite la realizzazione di diagrammi di flusso, dove vengono identificati le entità del sistema, gli eventi e i legami con esso. Questa identificazione prevede una classificazione delle minacce in sei categorie. Le minacce identificate sono:

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

Minaccia	Proprietà Violata	Tipo di Minaccia
Identità contraffatta	Autenticazione	Pretendere di essere qualcuno o qualcosa che in realtà non si è.
Manomissione dei dati	Integrità	Modificare qualcosa.
Ripudio	Non-Ripudio	Pretendere che non si è stati a fare qualcosa o non essere responsabile, sia che sia in buonafede che altrimenti.
Rivelare Informazioni	Confidenzialità	Fornire informazioni a qualcuno che non è autorizzato a conoscerle.
Negare il servizio	Disponibilità	Esaurire le risorse di cui si ha bisogno per erogare un servizio.
Elevare i privilegi	Autorizzazione	Permettere a qualcuno di fare qualcosa che non è autorizzato a fare.

Tabella 1.1: Confronto tra minaccia proprietà violata

Spoofing

Un attacco di tipo spoofing si verifica quando il malintenzionato finge di essere un'altra persona, assumendo l'identità e le informazioni in tale identità per commettere una frode. Un esempio molto comune di questa minaccia è quando

un'email viene inviata da un indirizzo email falso, che sembra essere qualcun altro. In genere, queste email richiedono dati sensibili. Un destinatario vulnerabile o inconsapevole fornisce i dati richiesti, l'hacker è quindi in grado di assumere facilmente la nuova identità. Le identità contraffatte possono essere sia umane che software. Attraverso lo spoofing, l'attaccante può accedere al sistema attraverso un'identità vulnerabile per poi eseguire un attacco informatico molto più profondo ed esteso. Lo spoofing non si limita ad essere una vulnerabilità che sfrutta il fattore umano. Dal punto di vista tecnico, specialmente nel ambito delle reti, si può riassumere come la capacità tecnica di contraffare i pacchetti con il risultato di essere identificato come un altro attore della rete ottenendo così un vantaggio illegittimo.

Tampering

Un attacco di tipo Tampering ai dati vuole dire, in italiano, manometterli. Questo si verifica quando i dati o le informazioni vengono modificati senza autorizzazione. Una delle modalità in cui un malintenzionato potrebbe eseguire la manomissione è tramite la modifica o l'eliminazione di un file di configurazione oppure con l'inserimento di un file dannoso. Il monitoraggio delle modifiche, noto anche come monitoraggio dell'integrità dei file (FIM), è essenziale. Una componente essenziale da tenere in considerazione, come forma di difesa preventiva sarebbe quella di registrare tutto quello che avviene nel sistema in quanto in questo modo sarebbe possibile identificare se e quando la manomissione dei dati è andata a buon fine. Questo processo esamina in modo sistematico i files, istruito con delle indicazioni su come identificare un file "buono". La registrazione e l'archiviazione corrette sono fondamentali per supportare il monitoraggio dei file. Ovviamente la modifica dei soli files copre solo una porzione di quello che può essere un attacco di questo tipo, alternativamente si potrebbe presentare il caso in cui vengano alterate le informazioni al volo per esempio durante una comunicazione di dati via rete forgiando dei pacchetti ad arte.

Repudiation

Il ripudio come suggerisce il nome è un attacco in cui l'attore malevolo pretende di non avere commesso un'operazione, di solito illegale e non autorizzata, oppure riesce a disconoscerne la paternità. In questi attacchi, il sistema non ha la capacità di tracciare effettivamente l'attività dannosa per identificare un attaccante. Questo tipo di attacchi può essere usato per modificare le informazioni sulla creazione delle azioni eseguite da un utente malevolo al fine di registrare dati errati nei files preposti [19]. Gli attacchi di ripudio sono relativamente facili da

eseguire sui sistemi di posta elettronica, poiché pochissimi sistemi controllano la validità della posta in uscita. Come forma di difesa preventiva si potrebbero registrare tutte le attività di un soggetto e ovviamente renderle non alterabili, in questo modo sarebbe possibile risalire sempre ad un eventuale responsabilità dell'utente.

Information disclosure

La divulgazione di informazioni è anche nota come leak, perdita d'informazioni. Succede quando un'applicazione o un sito Web rivela involontariamente dati a utenti non autorizzati. Questo tipo di minaccia può influenzare il processo, il flusso di dati e l'archiviazione dei dati in un'applicazione. Alcuni esempi di divulgazione di informazioni includono l'accesso involontario ai file del codice sorgente tramite backup temporanei, l'esposizione non necessaria di informazioni sensibili come i numeri di carta di credito e la rivelazione di informazioni sul database nei messaggi di errore. Questi problemi sono comuni e possono derivare da contenuto interno condiviso pubblicamente, configurazioni dell'applicazione non sicure o risposte di errore errate nella progettazione dell'applicazione. Ad esempio: quando si cerca di autenticarsi presso un servizio web e si sbaglia la password il servizio potrebbe restituire un messaggio di errore dove viene esplicitamente detto che la password è sbagliata, questo comporta una potenziale informazione per un attaccante in quanto implicitamente ci viene suggerito che l'utente esiste ma abbiamo solo sbagliato la password, un comportamento migliore sarebbe sempre restituire come messaggio d'errore "utente o password errati" in modo tale da non fornire nessun tipo di informazione ad un utente male intenzionato.

Denial of service

Gli attacchi di tipo Denial of Service (DoS), negare il servizio, limitano a un utente autorizzato l'accesso alle risorse (sito web, applicazioni, server) a cui dovrebbe essere in grado di accedere. Esistono molte strade per rendere un servizio non disponibile ad un utente legittimo, ad esempio modificare pacchetti di rete, programmi, il flusso di esecuzione. Se un servizio riceve un gran numero di richieste potrebbe cessare di rispondere, oppure rispondere con un ritardo estremamente alto, e quindi risultare non disponibile anche per un utente legittimo. Analogamente un servizio può interrompersi se una vulnerabilità viene sfruttata facendo crashare l'applicazione oppure tramite il modo in cui il servizio gestisce le risorse che utilizza. [18] Questi attacchi introducono grandi ritardi nella ri-

sposta, perdite eccessive e interruzioni del servizio, con un impatto diretto sulla disponibilità.

Elevation of privilege

Tramite l'innalzamento dei privilegi un utente legittimo o meno può accedere a parti del sistema in cui non è autorizzato oppure eseguire programmi a lui non destinati. Così facendo viola la proprietà di autorizzazione di un sistema. Per realizzare questo genere di attacco di solito l'utente malintenzionato deve aver già accesso al sistema, magari sfruttando una falla oppure un errore di configurazione, quindi di solito avviene in una seconda fase di un attacco. Questo tipo di minaccia può essere catalogata in due modi:

- Scalata verticale
- Scalata orizzontale o movimento laterale

Nel primo caso l'attaccante vuole diventare l'utente più importante del sistema, l'amministratore.
Nel secondo caso invece vuole diventare un utente a pari livello di autorizzazione, ma con accesso ad aree differenti rispetto a questi ultimi, guadagna accesso a dette aree. La violazione di questa proprietà è molto pericolosa in quanto, una volta guadagnati i privilegi massimi, l'attaccante può manipolare tutto il sistema in modo totalmente legittimo. Ne segue che diventa molto difficile riuscire ad identificarlo oppure essere sicuri di averlo definitivamente eliminato dal sistema stesso, poiché una volta ottenuti i privilegi avrebbe potuto inserire degli altri account dormienti per persistere la minaccia oppure creare delle scorciatoie nascoste per rientrare in caso di individuazione (backdoor).

Capitolo 2

TSN

Time-Sensitive Networking (TSN) saranno un componente fondamentale delle nuove reti industriali, trovando applicazioni molto verticali nelle nuove reti emergenti come 5G. Il motivo per cui è individuabile una traiettoria di crescita nell'utilizzo di queste tecnologie è legato alla presenza di molti altri standard de facto che realizzano queste funzionalità ma tra loro sono quasi sempre tutti non interoperabili, TSN invece si pone come standard neutrale e non legato ad una azienda o produttore di dispositivi specifici. Utilizzi di questo standard, che ad oggi è considerato lo stato dell'arte nel ambito di trasmissioni a bassissima latenza (Ultra-low Latency Communication) e con garanzia di consegna temporale, possono trovarsi in ambito militare, avionico e nelle transazioni finanziare. L'utilizzo di questi protocolli in ambiente industriale è fortemente legato a quello che in letteratura vengono chiamati dispositivi di "safety", ovvero tutti quei dispositivi che vengono impiegati per la sicurezza degli impianti e delle persone che ci lavorano. Ad esempio quando un pulsante di arresto critico viene premuto a causa di un problema all'impianto, quest'ultimo deve arrestarsi istantaneamente, ma come sappiamo le reti tradizionali non prevedono questo genere di comportamento rendendo non predicibile il risultato dell'operazione. La famiglia di protocolli che compone TSN è composta da una parte di standard di base, come ad esempio IEEE 802.1Q-2018, che contiene una sezione nella quale si descrivono i meccanismi per la trasmissione TSN su applicazioni in tempo reale ed altri più ancillari. Lo standard TSN copre un ampio spettro di requisiti di trasmissione molto stringenti ed è un pilastro per la realizzazione di politiche che vanno nella direzione della qualità dei servizi, QoS.

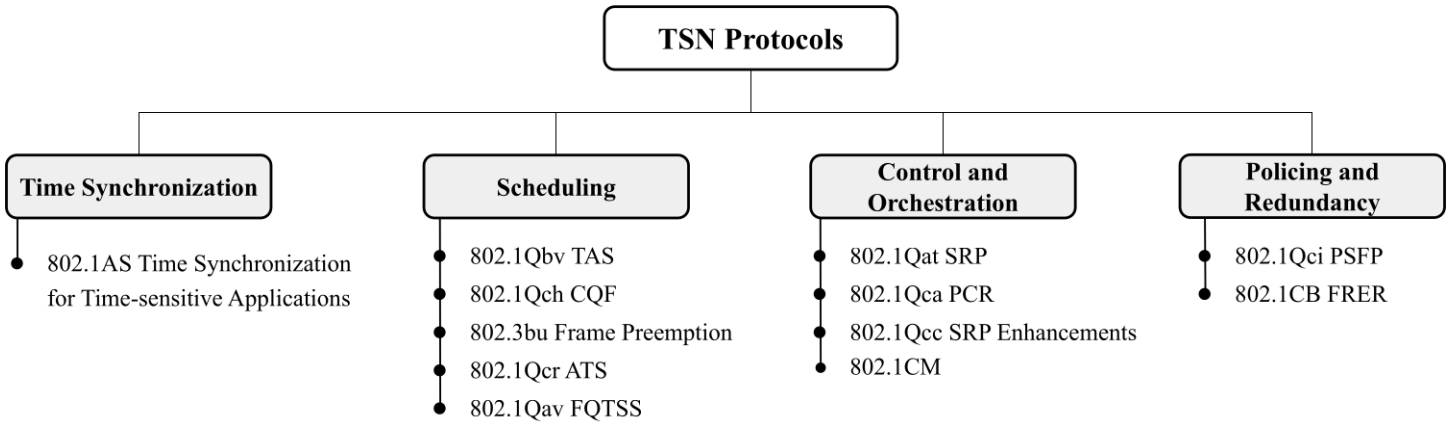

Figura 2.1: TSN Classification [8]

2.1 Time Synchronization

2.1.1 802.1AS - Timing and Synchronization

IEEE 802.1AS [12] è lo standard per applicazioni sensibili al tempo AS. Le comunicazioni con questa particolare novità diventano deterministiche attraverso uno standard uniforme per tutti i partecipanti della rete che richiedono la sincronizzazione di tutte le entità TSN. Lo standard IEEE 802.1AS [10] generic Precision Time Protocol (gPTP) si assicura che la sincronizzazione desiderata venga rispettata sfruttando generic Precision Time Protocol (gPTP). Il funzionamento di tale protocollo prevede una realizzazione gerarchica instaurata mediante un modello "leader/follower" tra tutti i partecipanti alle comunicazioni TSN. Il Best Master Clock Algorithm (BMCA) permette di scegliere un tempo di riferimento per tutta la rete TSN, questo appunto viene chiamato grandmaster oppure Clock Master. La scelta necessariamente ricade sull'entità con l'orologio più accurato tra tutti i partecipanti. Seguendolo gPTP scambia i messaggi tra il Clock Master e i "follower" al fine di calcolare il ritardo di propagazione tra i vari Clock Slaves. Il tempo in cui il frame viene allocato nel buffer di ogni Clock Slave è un requisito di trasmissione che prende in considerazione tutto il tempo in cui il frame passa dalle code d'ingresso a quelle d'uscita. Questo permette di avere una sincronizzazione in tempo reale di tutti i dispositivi della rete che diventano in questo modo consapevoli dello scorrere del tempo.

2.2 Scheduling

2.2.1 IEEE 802.1Qbv Enhancements to Traffic Scheduling: Time-Aware Shaper

Il protocollo IEEE 802.1Qbv è stato progettato per separare le comunicazioni che avvengono sulla rete Ethernet in finestre temporali o in frame di lunghezza fissata. Al fine di realizzare questo comportamento sono state proposte nel protocollo otto priorità. All'interno di queste finestre temporali è possibile configurare diversi intervalli di tempo che possono essere assegnati a una o più delle otto priorità Ethernet. Ad ogni priorità viene assegnata la sua finestra di trasmissione, questa previene la possibilità del frame di finire in starvation, ovvero l'impossibilità di essere schedulato, e si assicura che la trasmissione del traffico ad alta priorità abbia il canale di trasmissione occupato. In aggiunta, per prevenire l'indesiderato effetto di blocco oppure la sovrapposizione di trasmissione critiche che non possono essere rimandate, TAS introduce un intervallo di tempo chiamato "guard band", prima di una trasmissione critica per i vincoli di tempo. La possibilità di eseguire una "preemption" sui frame da trasmettere rende il sistema migliore nel garantire la QoS. Come contro altare, per garantire che il traffico ormai schedulato con una determinata priorità non possa subire interferenza da quello con una priorità più bassa, il protocollo richiede che tutti i partecipanti alla rete TSN debbono essere tempo sincronizzati.

2.2.2 IEEE 802.1Qav Forwarding and Queuing Enhancements for Time-Sensitive Streams

Il protocollo in questione è uno degli elementi fondamentali per la realizzazione di reti Ethernet in tempo reale, in quanto ci permette di definire varie tipologie di traffico da quelle best effert a quelle time-sensitive. Questa operazione di Trasmission Selection è svolta per ogni porta del dispositivo TSN. Ogni porta in questo modo è dotata di otto code (queue), una per ogni livello di priorità che si vuole assegnare. La strategia che viene applicata ad ogni porta per estrarre i vari frame da inoltrare e trasmettere è così determinata dall'algoritmo Trasmission Selection:

- è presente un frame da trasmettere su quella data coda
- ogni coda a priorità maggiore è vuota

Per distinguere quale frame inoltrare l'algoritmo si avvale di una tabella chiamata "Transmission Selection Algorithm Table" dove appunto sono specificate sia l'algoritmo possibile che il suo identificativo.

Algoritmo di Transmission Selection	Identificatore
Strict Priority	0
Credit-based shaper	1
Riservato per standardizzazione futura	2-255
Specifico del produttore	Un intero formato da 4 byte, dove i 3 byte più significativi contengono un valore OUI e il byte meno significativo contiene un valore intero compreso tra 0 e 255 assegnato dal proprietario dell'OUI.

Tabella 2.1: [11]

Come mostra la tabella 2.1 abbiamo a disposizione due tipi di algoritmi:

- Strict Priority
- Credit-based shaper

Algoritmo Strict Priority

In questo algoritmo la selezione del frame da trasmettere è fatta solamente sulla base della priorità. In questo caso appena la coda è pronta a trasmettere ed è presente un frame al suo interno questo viene trasmesso. Condizione necessaria è che non siano presenti altri frame in code a priorità maggiore.

Algoritmo Credit Based Shaper

Questo algoritmo risulta più conservativo del precedente e si va ad incastonare in quelli che in letteratura vengono chiamati "algoritmi di scheduling soft real time". Tramite questa strategia si evita di lasciare per un tempo indefinito nel buffer di trasmissione dei frame in quanto né arrivano sempre altri a priorità maggiore. Il modello di funzionamento è stato preso in prestito dal modello a token bucket. Il token bucket è un meccanismo di controllo di trasmissione che determina quando e quanto traffico dati (sotto forma di pacchetti) può essere trasmesso in base alla presenza o meno di token (gettoni) in un contenitore astratto, che detiene il traffico complessivo di rete da trasmettere, detto appunto bucket. Un flusso è autorizzato a trasmettere traffico solo quando sono presenti

token nel bucket. L'algoritmo presenta una serie di parametri interni associati ad ogni coda usati per realizzare tale comportamento:

transmit

assume il valore true per tutta la durata della trasmissione di un frame mentre assume il valore false quando la trasmissione del frame termina.

credit

Rappresenta, in bit, il credito di trasmissione disponibile alla coda. Se non ci sono frame in coda, transmit è false e credit è positivo, credit viene posto a 0.

transmitAllowed

Assume valore true se il credito è positivo o zero, false altrimenti

sendSlope

Sendslope è il tasso di crediti che viene esaurito quando è in corso una trasmissione. La formula che lo calcola è la seguente:
$sendSlope = (IdleSlope - portTransmitRate)$
Dove portTransmitRate rappresenta la velocità effettiva di trasmissione nel layer sottostante. Idleslope è il tasso di crediti che viene accumulato quando c'è almeno un pacchetto in attesa di trasmissione.

hicredit

Definisce l'ammontare massimo di crediti che possono essere accumulati. Questo valore dipende dalle caratteristiche del traffico che produce interferenza, il valore viene calcolato con la sequente espressione:
$hicredit = max\_interference\_size * (idleslope/portTransmitRate)$
Dove max_interference_size è la dimensione massima di ogni burst del traffico che può ritardare la trasmissione di un frame pronto alla trasmissione nella stessa classe di traffico. Superare la soglia di hicredit significherebbe introdurre un eccessivo ritardo.

locredit

Locredit è l'ammontare minimo di crediti che può essere raggiunto. Il calcolo di questo valore è una funzione del traffico che attraversa la qdisc.
$locredit = max\_frame\_size * (sendslope/portTransmitRate)$

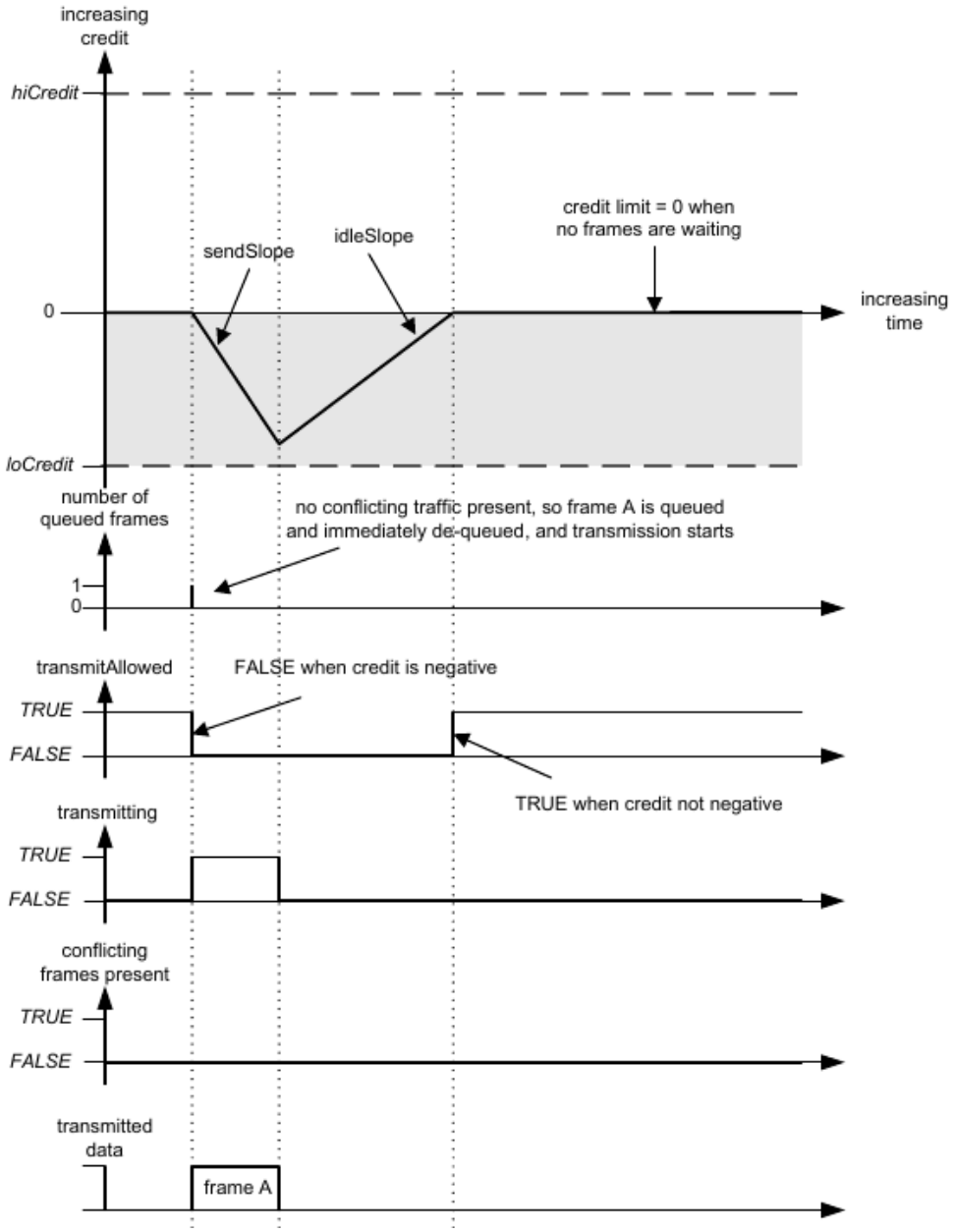

Figura 2.2: Credit-based shaper operation—no conflicting traffic [11]

2.2.3 IEEE 802.1Qch Cyclic Queuing and Forwarding

Il protocollo in questione permette la sincronizzazione e la trasmissione senza perdita dei frame, sia a causa della congestione sia a causa di ritardi nella latenza dei partecipanti. Di norma in base alla specifica topologia della rete

e degli switch, nel caso peggiore possono verificarsi dei ritardi di trasmissione. Queste reti però beneficiano della sincronizzazione che intercorre tra vari nodi. Per ovviare ai problemi prima citati questo protocollo quindi, introduce buffer doppi per le code; ciò permette ai "bridge" di sincronizzare le trasmissioni. Utilizzando dei buffer circolari il protocollo instaura un legame tra la latenza che dipende solo dal numero degli "hops" e il tempo ciclico, che è completamente indipendente dalla topologia.

2.2.4 IEEE 802.3br and 802.1Qbu Interspersing Express Traffic (IET) and Frame Preemption

L'unione dei due standard descritti in precedenza ha come scopo quello di diminuire gli inconvenienti dovuti all'introduzione del "guard band" implementato dal "TAS". Il problema dell'introduzione del "guard band" consiste nel potenziale tempo di trasmissione che viene perso, tempo in cui non si trasmette nulla. Per mitigare questo problema è stata introdotta la seguente strategia: la porta di uscita è separata in due interfacce: Preemptable MAC (pMAC) e express MAC (eMAC). I frame marcati come "espressi" posso scavalcare i frame pMAC facendogli subire preemption, quindi trasmettendoli in coda. Il risultato netto è la riduzione della "guard band" fino al tempo di trasmissione del più corto frame a priorità più bassa, così nel caso peggiore il frame meno prioritario è trasmesso prima che abbia inizio quello successivo ad alta priorità.

2.3 Control and Orchestration

2.3.1 IEEE 802.1Qca Path Control and Reservation

Questo standard è basato su Shortest Path Bridging (SPB) e specifica come collegare i vari bridge su sentieri (path) esplicitamente scelti. Possono determinarsi cammini multipli per frame trasmessi con strategia unicast oppure multicast, in dipendenza della topologia. Dopo aver messo insieme informazioni sulla topologia della rete dai vari nodi per trovare cammini ridondanti, il protocollo in questione riserva la larghezza di banda e le risorse per trasmissioni ridondanti e ottimali del traffico di rete.

2.3.2 IEEE 802.1Qat Stream Reservation Protocol (SRP) and IEEE 802.1Qcc Enhancements to SRP and Centralization Management

Garantire le risorse di rete lungo la strada individuata e accettare oppure rifiutare i flussi è compito del protocollo IEEE 802.1Qat SRP. Questo protocollo P2P distribuito permette di riservare le risorse di rete e di annunciare lo stream mantenendo la QoS. SRP riserva risorse e annuncia gli stream dal mittente (talker) al destinatario (listeners), garantendo la disponibilità di risorse di rete sufficienti lungo tutto il cammino, in questo caso flusso di trasmissione.

2.4 Policing and Redundancy

2.4.1 IEEE 802.1Qci Per-Stream Filtering and Policing

Quasi nessun protocollo della suite TSN si pone come obiettivo quello di effettuare una comunicazione sicura, ma questo invece migliora la robustezza della rete tramite l'individuazione di una strategia sul flusso dati in ingresso. In questo modo si possono definire delle regole che filtrano per ogni flusso dati andando a prevenire una saturazione delle risorse e quindi di conseguenza anche attacchi di tipo DoS.

2.4.2 IEEE 802.1CB Frame Replication and Elimination for Reliability

Altri miglioramenti alla affidabilità delle TSN possono essere imposti tramite questo protocollo. FRER permette di ridondare il traffico considerato critico, mandando frames duplicati su sentieri disgiunti. Prevede anche la possibilità di eliminare pacchetti duplicati se entrambi i frames arrivano alla destinazione, al fine di prevenire perdite dovute alla congestione della rete. La replica dei pacchetti può essere configurata assegnando delle classi di traffico, mentre l'informazioni del percorso vengono selezionate tramite l'identificazione dello stream TSN sfruttando una funzione di generazione della corretta sequenza. Per salvare le informazioni sulla ridondanza del frame nel tag apposito.

2.5 Sicurezza dei protocolli TSN

Per approcciare la sicurezza di questi protocolli si è deciso di adottare un modello per catalogare le minacce già discusso in precedenza STRIDE, la figura 2.3

fornisce un buono schema di analisi della problematica. Anche se il modello prevede l'analisi del non ripudio e della perdita d'informazioni, nel lavoro ci si è concentrati principalmente sulla parte DoS e indebolimento delle politiche QoS.

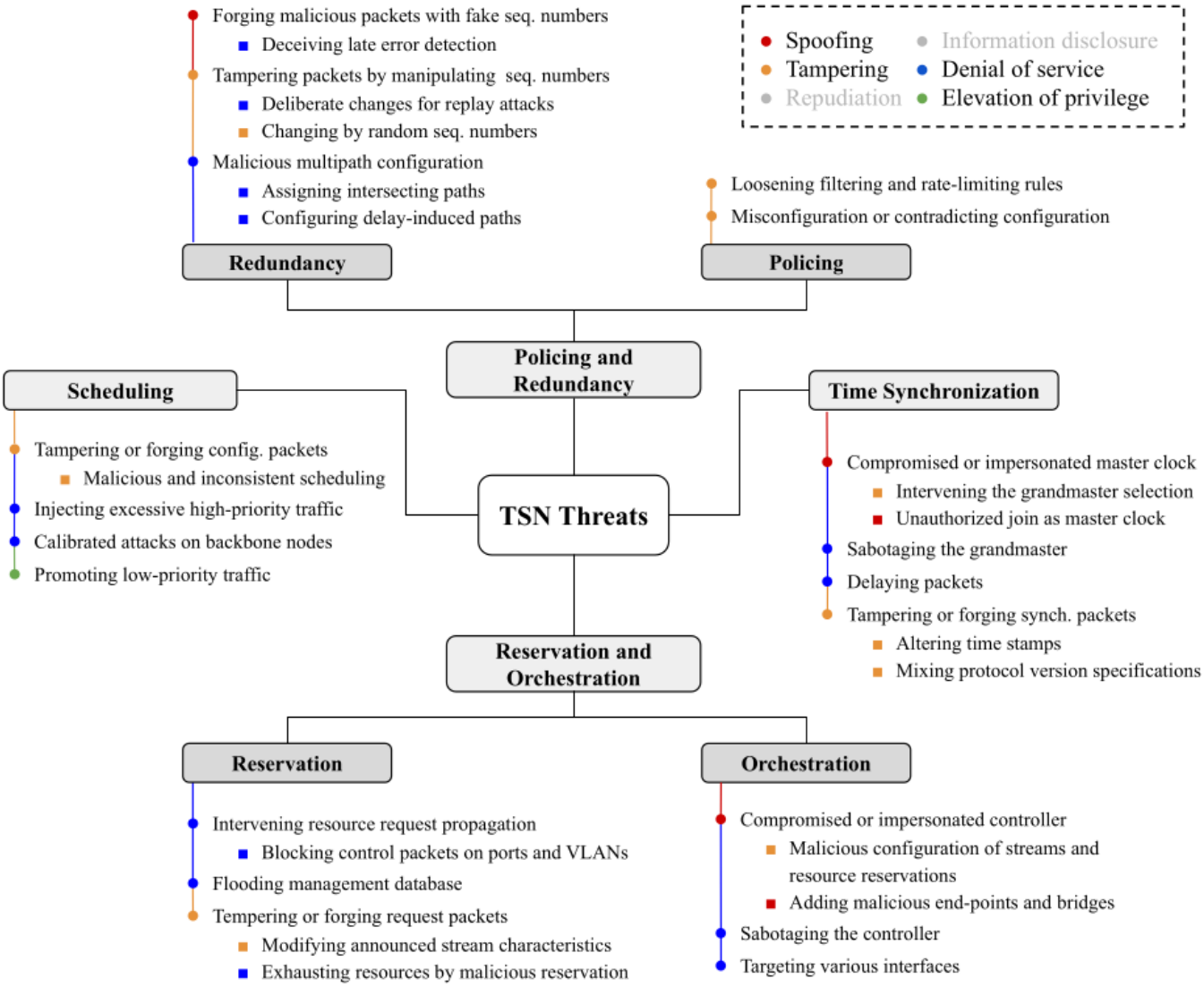

Figura 2.3: Minacce ai protocolli TSN con la corrispondente voce in STRIDE [8]

2.5.1 Minacce alla sincronizzazione

Visto che nelle TSN la sincronizzazione dei dispositivi è parte integrante della maggior parte dei protocolli possiamo annoverare anche il tempo come vettore

d'attacco. Se per esempio prendiamo il protocollo PTP, possiamo sicuramente compromettere l'orologio top-level, anche detto grandmaster, oppure possiamo impersonarlo ottenendo come risultato quello di diffondere nelle rete informazioni di sincronizzazioni inconsistenti o addirittura false. Per impersonare il grandmaster, ad esempio, si può andare ad agire durante il meccanismo d'elezione distribuito durante il passaggio in cui i vari nodi annunciano la loro priorità al fine di essere selezionati come grandmaster. Oltre a questo è possibile unirsi alla rete in modo non autorizzato direttamente come un orologio master, aggirando il meccanismo d'elezione e sovrascrivendo l'elezione del grandmaster. In alcune configurazioni, dove esiste un solo master, sabotarlo potrebbe causare una desincronizzazione delle rete fino a che un nuovo master viene eletto. Scegliere come bersaglio di un attacco il master clock dovrebbe essere complesso in quanto, vista la delicatezza del ruolo che ricopre, dovrebbe essere protetto da molte contromisure poste su altri livelli, non essendoci nessun meccanismo di difesa intrinseco nel protocollo. Forgiare o modificare pacchetti di sincronizzazione è un altro modo di intervenire nel processo di sincronizzazione. Il nuovo standard PTPv2, non compatibile con il precedente, prende in considerazione la questione della sicurezza introducendo meccanismi crittografici e di controllo della checksum. Nonostante queste contromisure, il problema persiste ancora nella forma di ritardare i pacchetti, introducendo una latenza nella rete.

2.5.2 Minacce allo scheduler dei pacchetti

Il protocollo IEEE 802.1Qbv TAS sfrutta lo scheduler per rendere le comunicazioni deterministiche. In questo caso le problematiche potrebbero sorgere durante la fase di configurazione delle politiche di instradamento del traffico, questa operazione può avvenire sia tramite un controllore esterno oppure locale. Qualora la configurazione avvenga in modo esterno come specificato nello standard IEEE 802.1Qcc sarebbe possibile manipolare o forgiare pacchetti del protocollo di gestione, come ad esempio SNMP. L'abuso da parte di un attaccante può avere come obiettivo una modifica della banda garantita ad un servizio, magari portandola a zero, ed avendo come conseguenza l'indisponibilità del servizio stesso. Quindi una mancanza di meccanismi di validazione a livello protocollare può portare ad avere uno scheduling dei pacchetti inconsistente, in questo è molto difficile da individuare. Il motivo di tale difficoltà risiede nel fatto che non esistono tecnologie mature per effettuare tale tipo di controllo [8].
Un altro attacco possibile a questo livello consiste nella possibilità di approfittare dei gate già configurati per iniettare traffico eccessivo nella rete ad alta priorità, creando una congestione critica che può portare alla perdita di pacchet-

ti importanti per la rete. Specialmente nella modalità in cui i frame subiscono preemption la possibilità di promuovere il traffico a priorità bassa a priorità alta. Ora qualora un attaccante si trovi di fronte ad una rete TSN potrebbe calibrare gli attacchi al fine di rompere quello che è lo scheduling nei nodi, sfruttando il fatto che i tempi di latenza sono calcolati con precisione sotto i microsecondi. Anche se quest'ultimo non è una vera problematica delle TSN è una problematica relativa a tutto ciò che è deterministico e predicibile.

2.5.3 Minacce all'orchestratore e al control plane

Le TSN usano meccanismi per riservare banda ed avere uno scheduling di tipo deterministico. Lo standard IEEE 802.1Qat, che realizza SRP è stato poi aggiornato con la realizzazione di Audio Video Bridging, alcune considerazioni sulla sicurezza si possono fare. Una configurazione malevola di un bridge può ad esempio bloccare i pacchetti di controllo inviandoli altrove oppure bloccandoli direttamente. Invece di intervenire sulla conversazione un'alternativa sarebbe modificare i pacchetti che annunciano il protocollo, questo può portare ad indurre in errore i vari nodi, magari generando un errore di calcolo delle latenze dello streaming con conseguenza lo sforamento della deadline. Oltre quindi a manipolare o forgiare pacchetti per lo stream reservation è possibile andare ad agire sul MIB, dove vengono salvati gli oggetti di cui si è fatto streaming, facendo flooding dei pacchetti.

2.5.4 Minacce alle policy e alla ridondanza

IEEE 802.1CB FRER non offre nessun meccanismo di sicurezza, nonostante le sue due principali funzionalità, replicazione dei pacchetti ed eliminazione dei pacchetti, sono vittime di minacce ben definite. Il meccanismo che regola l'eliminazione permette ai vari bridge di inoltrare il primo pacchetto in ordine di arrivo e scartare il secondo, quando viene ricevuto dallo stesso bridge. Un attaccante può forgiare pacchetti con un falso numero di sequenza al fine di far scartare il pacchetto originale. Oltre ad attacchi così mirati un processo malevolo potrebbe iniziare ad inviare pacchetti con numeri di sequenza casuali con conseguenze non predicibili a priori, come lo scarto o la consegna non in ordine dei pacchetti legittimi. Alternativamente sarebbe possibile cambiare il numero di sequenza di un pacchetto che è già replica di un'altro e poi inoltrarlo nuovamente. FRER non prevede la consegna in ordine dei pacchetti questo può portare un malintenzionato può configurare percorsi multipli per la ridondanza dello stream al fine di introdurre un ritardo nel percorso ridondato e rendere

possibile la consegna non in ordine dei pacchetti con conseguente degradazione della QoS.
In ultima analisi IEEE 802.1QCi PSFP offre anche la possibilità di filtrare i pacchetti, ottenendo come risultato quello di una scorretta configurazione ed interagendo con altri protocolli potrebbe produrre un comportamento in cui alcuni pacchetti vengono scartati in maniera del tutto non intenzionale.
PSFP risulta essere l'unica forma di protezione per le reti TSN.

Capitolo 3

Virtualized Adversarial Laboratory

Durante il lavoro di tesi si è realizzata un'infrastruttura virtualizzata per testare alcune tipologie d'attacco su alcuni protocolli della famiglia TSN. Nell'ottica di realizzare un ambiente di test tale da poter anche effettuare una messa appunto per un'infrastruttura fisica corrispondente.

3.1 Tecnologie esplorate per la realizzazione del progetto

3.1.1 Qemu-System

QEMU è un virtualizzatore ed emulatore open source per ogni tipo di sistema. Quando è usato come virtualizzatore riesce ad ottenere prestazioni molto vicine a quelle che si hanno in forma nativa, eseguendo il codice ospite direttamente nel host CPU. QEMU può usare il modulo KVM nel kernel Linux.

3.1.2 Ansible

Ansible è un motore di automazione IT open source che consente di automatizzare il provisioning, la gestione della configurazione, il deployment delle applicazioni, l'orchestrazione e molti altri processi IT.
Ansible utilizza il protocollo SSH per connettersi ad ogni nodo della rete e realizzare il deployment in modo sicuro. Si noti che Ansible è agentless ovvero nelle macchine target non ha bisogno di nessun software al di fuori di SSH.

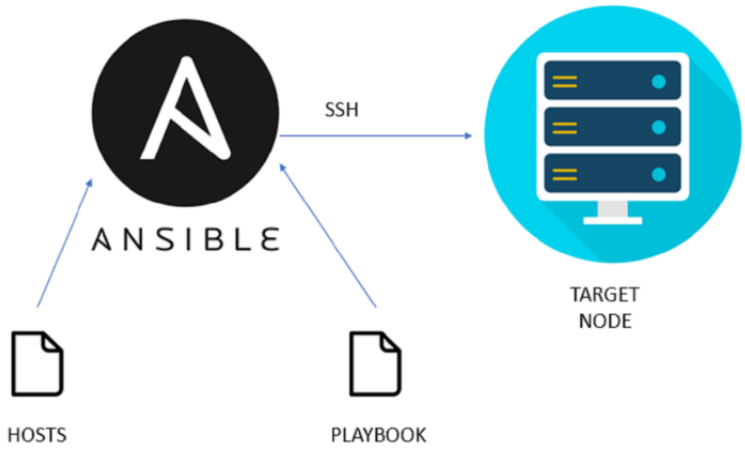

Figura 3.1: Ansible come automatizzare una infrastruttura GNU/LINUX [1]

3.1.3 PTP for Linux

La spina dorsale dei protocolli TSN è la possibilità di avere un certo grado di sincronizzazioni tra tutti i dispositivi della rete. Per potersi coordinare il protocollo alla base di tutto è il Precision Time Protocol, PTP.
PTP quindi è usato per sincronizzare gli orologi all'interno di una rete. Quando è usato sopra all'hardware che lo supporta, PTP è capace di essere accurato nell'ordine dei microsecondi ed anche sotto, il che è molto meglio di quello che si può raggiungere con NTP. Il supporto a PTP è fornito in tandem sia a livello kernel sia nello spazio utente. L'implementazione di PTP più usata in ambiente GNU/LINUX è linuxptp, un'implementazione dello standard PTPv2 in accordo con il documento IEEE 1588. Il pacchetto linuxptp include i programmi ptp4l e phc2sys per la sincronizzazione degli orologi. Il programma ptp4l implementa il meccanismo tra l'orologio gestito da PTP e quello normale di sistema. Sfruttare il timestamp hardware è quello che realizza il protocollo PTP per sincronizzare l'orologio master (master clock) e poi con il timestamp software sincronizzare l'orologio di sistema al master clock. Il programma phc2sys è necessario solamente per il timestamp hardware, per sincronizzare l'orologio di sistema all'hardware clock di PTP presente sulla scheda di rete NIC.

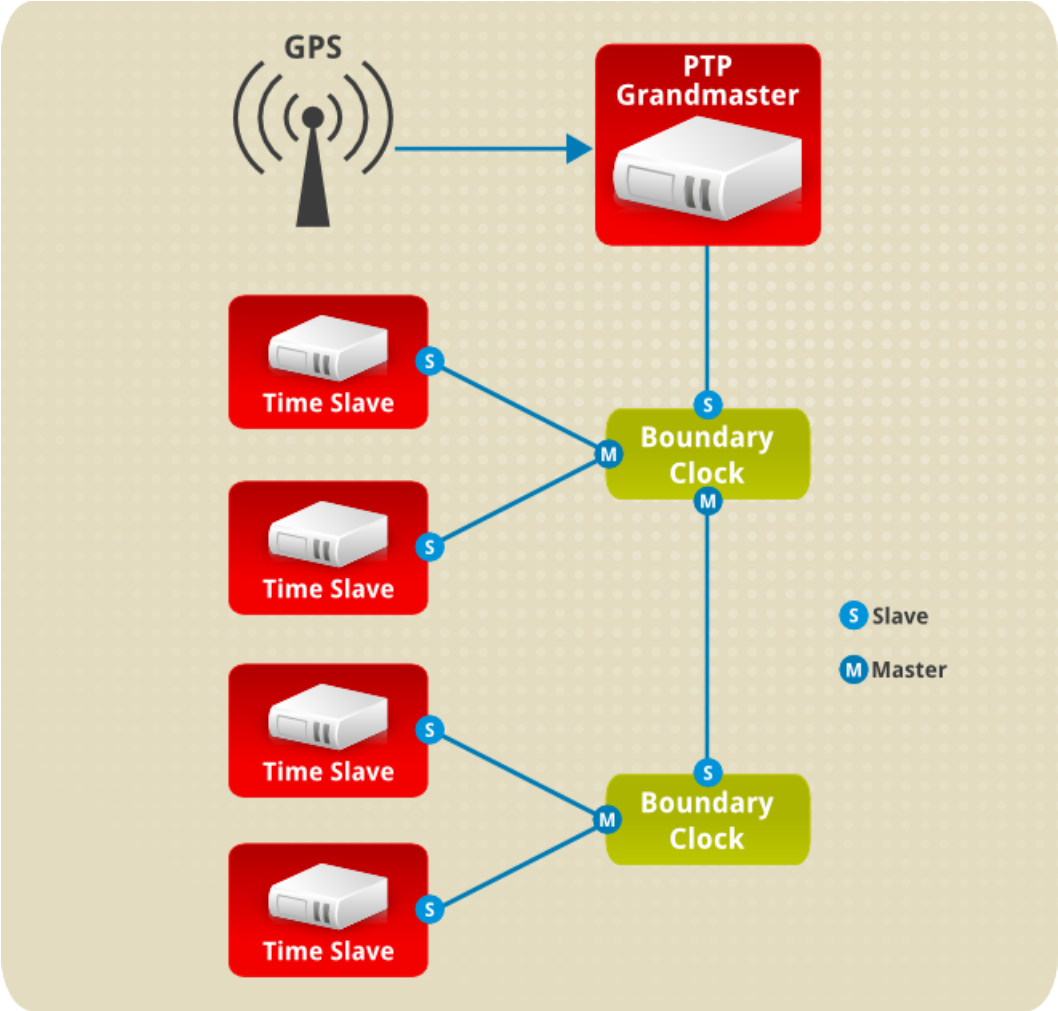

Figura 3.2: PTP grandmaster, boundary, and slave Clocks [4]

Gli orologi sincronizzati da PTP sono organizzati in modo gerarachico secondo un'architettura di tipo main- follower (Master/Slave). I follower sono sincronizzati ai rispettivi main come in figura 3.2. La gerarchia così creata è aggiornata automaticamente dal miglior orologio disponibile tra i main usando un algoritmo chiamato BMC presente ed in funzione in ogni orologio [3]. Quando un orologio ha una sola porta, può essere sia main che follower, questa tipologia di orologi è chiamata OC ordinary clock. Un orologio con più porte può essere sia main su una porta sia follower su un'altra, questa tipologia è chiamata BC Boundary Clock. L'orologio più importante è chiamato grandmaster, questo di norma è sincronizzato sfruttando Global Positioning System (GPS) come fonte, al fine di garantire ad un elevato grado di precisione del sistema.

3.1.4 Virtual Distributed Ethernet

VDE, acronimo per Virtual Distributed Ethernet, è composto da un insieme di programmi che nel complesso realizzano interfacce di rete Software-defined sia per dispositivi hardware sia per dispositivi virtuali. VDE è ad oggi sotto l'ombrello del progetto Virtual Square coordinato dal Prof. Renzo Davoli [22], l'idea dietro questo componente è quella di realizzare una piattaforma di comunicazione interoperabile tra le varie entità virtuali, ovvero VM, namespace, virtual switch e router. Le caratteristiche principali di VDE sono:

- comportamento consistente con le rete ethernet fisica
- abilita l'interconnessione tra VM, applicazione e strumenti di connettività virtuale
- non richiede privilegi di amministratore per funzionare

Per il lavoro di tesi si è usato VDE4plug [21]. Questo realizza un dispositivo di rete virtuale direttamente attaccabile a QEMU tramite la specifica dell'interfaccia.

3.2 Virtualizzazione dei Componenti Fisici

3.2.1 Virtio

Virtio è stato scelto come piattaforma principale per la virtualizzazione dell'IO in KVM, offrendo direttamente un modo efficiente, standard ed un meccanismo estendibile per gestire i dispositivi virtualizzati. Per realizzare questo comportamento sfrutta la possibilità del guest di condividere la memoria con l'host per gestire I/O.
La specifica di virtio è basata su due elementi: dispositivi e driver. In una implementazione tipica, l'hypervisor espone direttamente i dispositivi di virtio alla macchina guest attraverso modalità diverse di trasporto. Per scelta progettuale questi dispositivi danno l'illusione di essere hardware all'interno del guest, dentro la macchina virtuale. Per inviare un pacchetto, il driver manda al device un buffer che include alcuni metadati come ad esempio l'offloading del pacchetto. Il driver può anche dividere il buffer in diverse modalità, ad esempio può dividere i metadati dell'intestazione del pacchetto dal frame dello stesso. Il diagramma in figura 3.3 mostra il flusso di funzionamento dei dispositivi Virtio di rete, indicando sia a livello di configurazione, sia a livello di trasmissione dei pacchetti come viene sfruttato il virtio-net driver. Questo comunica con il

dispositivo corrispondente attraverso la PCI. Dopo aver riempito il pacchetto da inviare, il sistema innesca una notifica, invertendo il controllo a QEMU, in questo modo questo può inviare pacchetti attraverso il dispositivo TAP.

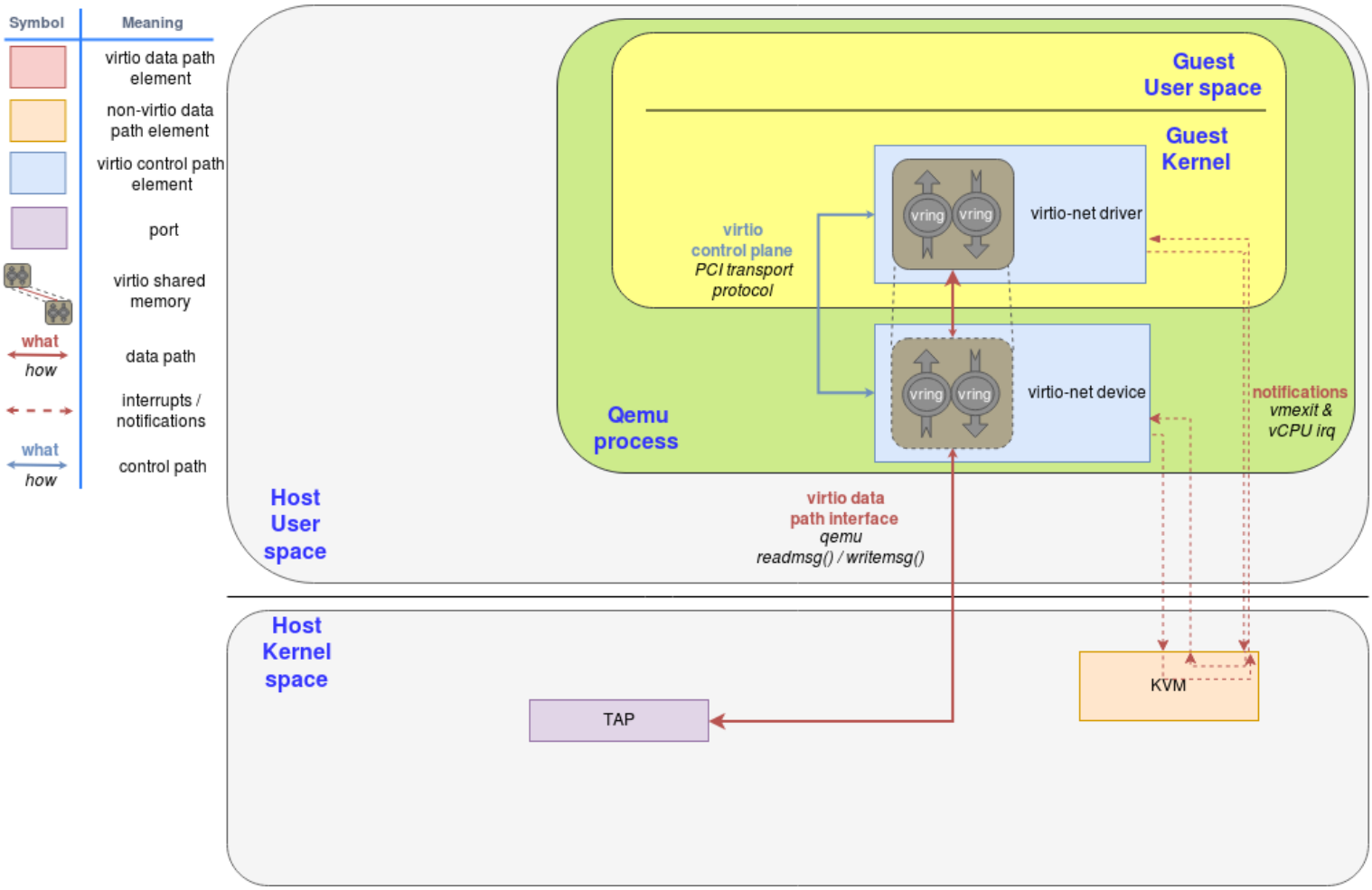

Figura 3.3: Networking con virtio: implementazione su qemu [16]

TUN TAP

Nelle reti di calcolatori, TUN e TAP sono dispositivi di rete virtuali a livello kernel. Essendo dispositivi di rete interamente software, differiscono dai normali dispositivi che sono controllati direttamente dalle schede di rete fisiche. Sebbene entrambi abbiano come scopo quello di creare un tunnel, TUN e TAP non possono essere utilizzati insieme perché trasmettono e ricevono pacchetti a livelli diversi dello stack di rete. TUN, che sta per TUNnel di rete, simula un dispositivo a livello di rete e opera nel livello 3, ovvero trasporta pacchetti IP.

TAP, cioè network TAP, simula un dispositivo a livello link e opera nel livello 2, ovvero trasporta frame Ethernet. TUN viene utilizzato con il routing. TAP può essere utilizzato per creare un bridge di rete nello spazio utente. I pacchetti inviati da un sistema operativo tramite un dispositivo TUN/TAP vengono consegnati a un programma nello spazio utente che si collega direttamente al dispositivo. Questo programma può quindi sfruttare questi dispositivi virtuali per mandare pacchetti e comunicare. In questo caso il dispositivo TUN/TAP consegna (o "inietta") questi pacchetti nello stack di rete del sistema operativo emulando così la loro ricezione da una sorgente esterna [17]. Per riassumere TUN/TAP forniscono un modo per ricevere e trasmettere pacchetti a programmi nello spazio utente. Possono essere visti come un semplice dispositivo Point-to-Point oppure Ethernet, che invece di ricevere o trasmettere pacchetti da un mezzo fisico, li riceve o trasmette da un programma nello spazio utente.

3.2.2 Multiqueue - KVM

Ad oggi i server corazzati di livello enterprise sono caratterizzati da molti processori e quindi ogni VM che risiede su questi server ha un elevato numero di vcpus, cpu virtuali. La scalabilità dello stack dei protocolli di rete nelle macchine virtuali guest è fortemente limitato dal uso di una sola coda in ingresso. Ad esempio se avessimo un caso in cui quattro vcpus fossero sempre al 100% di utilizzo per attività collegate all'uso della rete, avendo disponibile una solo coda d'ingresso, e una d'uscita avremmo una strozzatura nelle prestazioni della nostra applicazione in quanto l'assenza della multiqueue sarebbe il collo di bottiglia del sistema. Come conseguenza le performance di rete non scalano all'aumentare del numero di vcpus: questo porta direttamente ad avere delle VM che non possono trasmettere né ricevere pacchetti in parallelo in quanto avrebbero solo una coda TX e una RX. L'architettura per la gestione di più code è rappresentata in figura 3.4

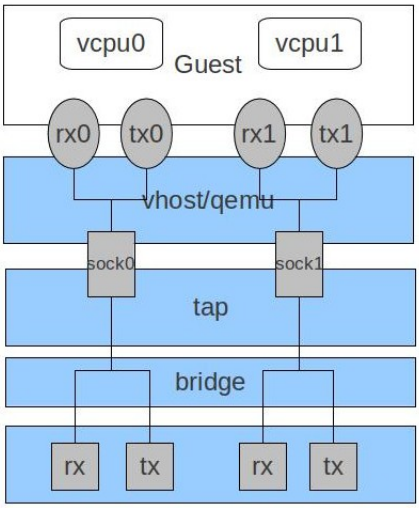

Figura 3.4: Trasmissione dei pacchetti con architettura virtio [14]

3.2.3 Linux Traffic Control

Linux Traffic Control (TC) è un programma di utilità che dà la possibilità a chi lo padroneggia la responsabilità di configurare lo scheduler dei pacchetti di rete direttamente del kernel. Tra i suoi molti utilizzi c'è ad esempio quello di limitare la banda ad un particolare servizio oppure di garantirla ad un altro impedendo che magari rimanga senza risorse. Attraverso l'uso di TC è possibile andare a manipolare direttamente il modulo "Linux Packet Scheduler" con il quale poi è possibile agire sulle queuing disciplines (qdisc). Per semplificare la visione sul componente possiamo pensare alle qdisc come uno scheduler, ovvero un insieme di politiche del traffico di rete per creare code e regole QoS per la ricezione e la trasmissione del traffico. Le code che rappresentano traffico in uscita dal dispositivo vengono chiamate "ingress qdisc", altrimenti quello in ingresso "egress qdisc". Le ultime appena descritte possono agire sul traffico con queste possibilità:

- manipolarlo
- schedularlo
- filtrarlo

Per le qdisc "egress" abbiamo a disposizione due tipi di politiche possibili:

classless qdisc A

Classless qdisc che non contengono altre qdisc, quindi non essendoci livelli di innestamento abbiamo una sola coda. Questa strategia serve solo a determinare come il pacchetto viene classificato, ritardato oppure scartato. La politica di dafault è FIFO.

classful qdisc A

Classfull qdisc a differenza della precedente può contenere altre qdisc, quindi possiamo avere più livelli di code. In questo caso, si possono applicare diverse politiche di filtraggio e determinare quale pacchetto trasmettere da quale qdisc
.

Per realizzare e supportare i protocolli TSN nello stack di rete nel kernel linux sono state sviluppate nuove qdisc.

- MQPRIO - The Multiqueue Priority Qdisc
- CBS - Credit Based Shaper (CBS) Qdisc
- ETF - Earliest TxTime First (ETF) Qdisc
- TAPRIO - Time Aware Priority Shaper

MQPRIO - The Multiqueue Priority Qdisc

MQPRIO è una qdisc che permette di mappare il flusso di traffico proveniente dal dispositivo fisico usando priorità configurabili. Una classe di traffico in questo contesto mappa in modo univoco le code esposte direttamente dall'hardware.

CBS - Credit Based Shaper (CBS) Qdisc

La "qdisc CBS" Credit Based Shaper implementa l'algoritmo che modella lo standard IEEE 802.1Q-2014 il quale applica alla banda un limite ben definito. Questa qdisc è stata implementata appositamente per essere usata al fine di modellare applicazioni TSN. Questa qdisc è progettata per essere installata sotto un'altra, come per le classfull, in questo modo si può mappare il flusso dei pacchetti alla classe del traffico.

ETF - Earliest TxTime First (ETF) Qdisc

Questa qdisc permette alle applicazioni di controllare quando un pacchetto deve essere estratto dalla coda dal controllore del traffico nel dispositivo di rete.

TAPRIO - Time Aware Priority Shaper

Questa qdisc implementa una versione semplificata della macchina a stati definita nello standard IEEE 802.1Q-2018, che permette la configurazione di una sequenza di "gate state", dove ognuno consente il traffico in uscita per un sottoinsieme (ptenzialmente anche vuoto) di classi di traffico. Ovvero con questa suite è possibile ad esempio dividere il traffico best effort da quello real time.

3.2.4 Creazione degli endpoint TSN virtualizzati

Il fulcro di tutta l'infrastruttura risiede nell'utilizzo di QEMU, come strato di virtualizzazione accelerato tramite KVM.
Per iniziare il progetto si è dovuto procedere a creare il disco, in formato QCOW2, nel quale si è installata una particolare distribuzione di Linux: Debian.
Debian è considerato la distribuzione che per natura è la più stabile e quindi la più usata nonché matura ad essere installata sui server di produzione.

```
$ qemu-img create -f qcow2 debian_base.qcow2 20G
```

Successivamente è stato installato il sistema operativo, senza ambiente grafico, sopra il disco appena creato.

```
$ qemu-system-x86_64
-cdrom debian-11.2.0-amd64-netinst.iso
-boot order=d
-drive file=debian_base.qcow2,format=qcow2
```

Una volta partita la VM si è iniziato a configurare il sistema; il primo passo è stato quello di installare le chiavi SSH. Ciò ha permesso, in un secondo stage, l'installazione dei software e la configurazione degli strumenti utilizzando Ansible.
Proseguendo nel design si è optato per una configurazione di rete di tipo "Private Virtual Bridge", in quanto l'obiettivo prefissato prevedeva la realizzazione una rete privata tra tutte le macchine virtuali. In questo modo tutte le VM non sarebbero state viste né da altre VM su un'altra rete né dalla rete reale.
Prima di procedere è stato necessario creare un bridge in cui in un secondo momento si sono attaccati i vari "TAP", uno per ogni macchina virtuale.

```
# Crea il bridge
ip link add br0 type bridge
# attiva il bridge
ip link set br0 up
```

Per dividere il traffico su più code, proprio come prevedono alcuni protocolli TSN, si è dovuto assegnare ad ogni VM più di una coda d'ingresso, realizzando quindi un sistema multiqueue, successivamente è stato necessario modificare lo script fornito nella documentazione di "Red Hat" [15] con l'introduzione del multiqueue:

```
# Crea il dispositivo tap chiamato come primo parametro
ip tuntap add $1 mode tap user $(whoami) multi_queue
```

Questo ha permesso ad ogni "TAP" di avere il concetto di multiqueue. In questo modo si è potuto utilizzare il TC di Linux e Iptables per realizzare comunicazioni su più code. Mettendo insieme i vari pezzi si è proceduto alla stesura di uno script "shell" per orchestrare le varie VM. Qui in seguito riportata la parte più importante con la quale vengono create:

```
# $2 rappresenta il numero di code che vogliamo avere
(( vectors="$2"*2+2 ))
qemu-system-x86_64
    -daemonize
    -enable-kvm
    -machine q35
    -device intel-iommu
    -cpu host
    -m 1024
    clone_#_debian_disk.qcow2
    -device virtio-net,
    netdev=net0,
    mac=$(printf 'DE:AD:BE:EF:%02X:%02X\n'
        $((RANDOM%256))
        $((RANDOM%256))),
    mq=on,vectors="$vectors"
    -netdev tap,
    id=net0,
    queues="$2",
    vhost=on,
    script=$(dirname $0)/qemu-ifup.sh
    -nic user,hostfwd=tcp::602#-:22
```

Lo script, una volta orchestrate le macchine virtuali, lancia la configurazione di quest'ultime attraverso Ansible. In questo modo è possibile creare tutta l'infrastruttura in pochi passi, in modo automatico ed idempotente. Per far

comunicare tra loro le varie macchine virtuali si è deciso di configurarle con un ip statico della rete “192.168.1.0/24” dopo di che Il primo passo per ogni VM è stato quello di andare effettivamente ad abilitare le multiqueue attraverso il programma “ethtool”:

```
ethtool -L interfaccia combined num_queue
```

Nel nostro caso per l’attività proposta nelle sezioni successive, si è deciso di lavorare con 3 code. La verifica che tutta la configurazione sia andata per il meglio si è ottenuta andando a controllare l’effettiva realizzazione delle code poste nel file system:

```
ls /sys/class/net/enp3s0/queues/
```

Capitolo 4

Attacchi TSN

4.1 PTP Clock Poisoning

Il primo attacco effettivamente realizzato ha avuto come obiettivo la desincronizzazione degli orologi, compromettendo il protocollo PTP. Come abbiamo già descritto PTP si trova alla base di molti meccanismi delle TSN. In questo caso si è dovuto creare 3 macchine virtuali: una che facesse da Main e due che facessero da Follower per gli orologi. Il problema della deriva degli orologi nella macchine virtuali purtroppo è stato oggetto d'indagine in quanto è considerato uno degli aspetti più critici legati alla virtualizzazione [23]. Avendo utilizzato un orologio paravirtualizzato, ovvero che viene gestito direttamente dall'hardware tramite l'implementazione di hypercall, quindi senza emulazione e con un overhead molto basso. Sfruttando tale meccanismo si è osservato che la problematica era poco presente; a differenza di altri sistemi di virtualizzazione come VirtualBox nel quale invece la desincronia era molto evidente. In QEMU KVM è stato possibile sfruttare un componente paravirtualizzato chiamato "kvm_clock" [23]. L'attacco in questo caso si basa sul fatto che i frame di tipo TLV (Type Lenght Value), sono completamente esposti. Il fatto è l'assenza in modo intrinseco al protocollo di nessun tipo di autenticazione. Il TLV è un tipo di frame che viene usato per configurare i vari settaggi di PTP. In pratica questo attacco prevede di andare a manipolare questi pacchetti in modo tale da riconfigurare o spegnere completamente la sincronizzazione degli orologi. Il risultato ottenuto rende possibile quindi andare a modificare le porte oppure spostare gli orologi nella gerarchia preconfigurata.
In questa parte del lavoro per creare l'attacco si è deciso per optare per VDE4Plug come gestore delle schede di rete virtuali in quanto la sua modalità di funziona-

mento a P2P a socket Unix sembrava essere quelle che introducesse una latenza di comunicazione minore.
Durante il lavoro sono emersi delle problematiche per attaccare in modo diretto l'ultima versione di VDE. Siccome il sistema operativo usato per i test è una particolare distribuzione basata su Arch, il programma qemu-system con cui è stato possibile lanciare le VM è stato compilato con una versione datata di VDE, ovvero la due. Per risolvere tale idiosincrasia tra le versione, si è provveduto, una volta compilato VDE4plug, a creare dei link simbolici per ingannare qemu-system e farlo funzionare con la versione aggiornata della libreria. Siccome anche dopo questo espediente ancora non funzionava si è usato strace per tracciare tutte le systemcall usate da qemu-system per rendersi conto appunto delle versioni difformi della libreria in oggetto, inoltre si è usato la variabile di ambiente "LD_LIBRARY_PATH=" per agganciare la nuova libreria installata a qemu-system.
Il funzionamento dell'attacco è il seguente:
Si è definito una topologia usando un file di configurazione, in questo caso una topologia con due entità legittime attaccate direttamente e poi un attaccante.
Si è deciso un nodo dove inserire un orologio PTP, in questo caso sfruttando PTP4L, si è inserito il demone PTPd nei due nodi che sono poi diventati OC, Ordinary Clock.
L'attaccante poi inizia a manipolare il frame TLV.
Gli altri nodi iniziano ad avere il loro orologio non più conforme a quello che ci si aspetterebbe.

```
[global]
clockAccuracy 0x1
priority1 0
priority2 0
```

Con questo file di configurazione di PTP si è riusciti ad avvelenare l'orologio in modo tale da poi durante la fase di elezione si viene scelti come Master.

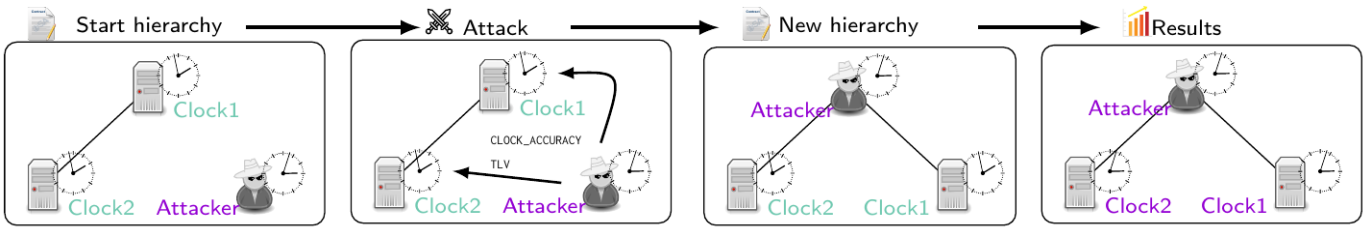

Figura 4.1: PTP Spoofing [2]

4.2 TSN Credit Based Shaper Denial of service

L'attacco che si è realizzato ha avuto lo scopo di dimostrare come la mancanza di meccanismi di autenticazione reciproca tra i vari dispositivi TSN possa far si che un malintenzionato possa introdursi nella rete e generare traffico ad alta priorità rallentando il traffico critico.

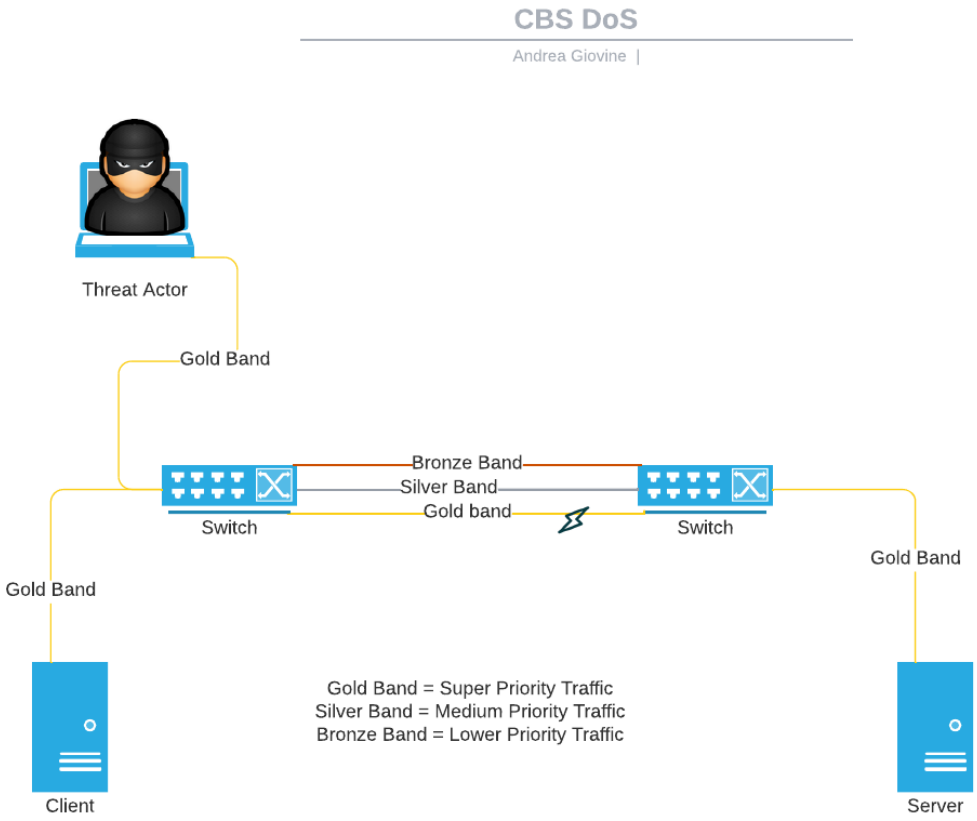

Figura 4.2: Modello architetturale attacco DoS CBS

La figura 4.2 vede lo svolgimento dell'attacco, dove tutti i nodi sono stati realizzati in modo virtualizzato e gli switch sono le macchine virtuale create in precedenza.
Per realizzare questo attacco dobbiamo introdurre altri strumenti che sono stati usati per realizzare la simulazione dell'attacco:

- Iptables
- h3ping

- iperf3

4.2.1 Iptables

Iptables è lo strumento principe nei sistemi Linux per configurare il firewall, il suo funzionamento si base sul fatto che esistono delle tabelle di default INPUT, DROP, FORWARD da cui poter intervenire filtrando il traffico. Nel lavoro di tesi è stato usato questo strumento in quanto oltre a filtrare i pacchetti può applicare ad essi un mark tramite la tabella di MANGLE, introdotta apposta per questa caratteristica. La tabella di MANGLE in generale è usata per modificare i pacchetti.

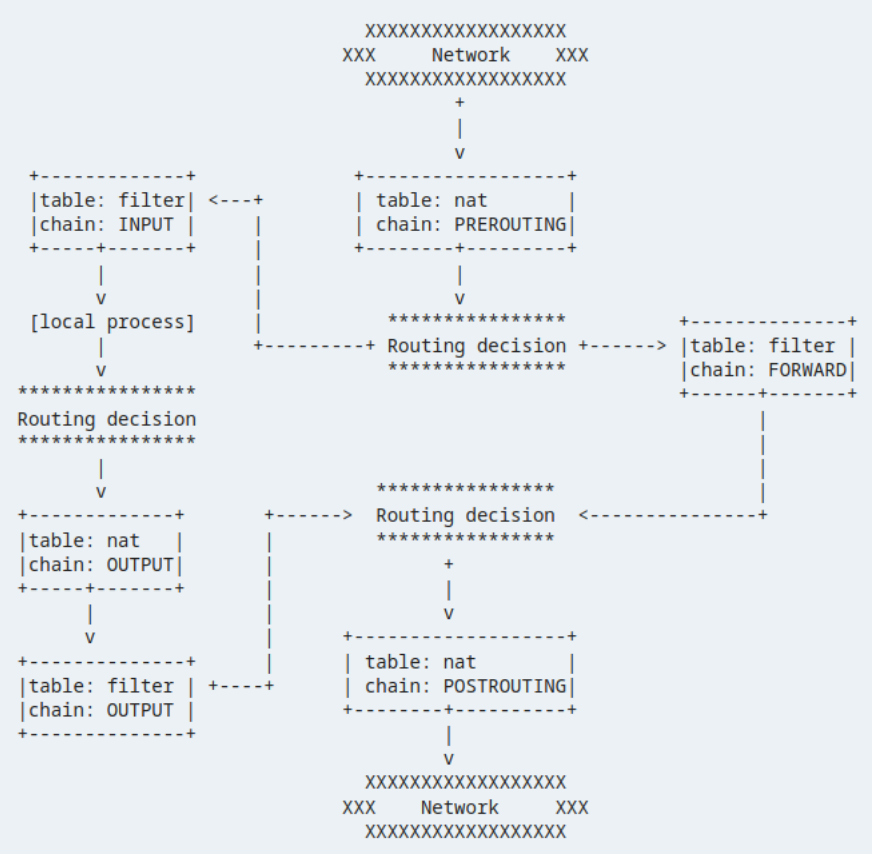

Figura 4.3: Flusso del traffico in Iptables [13]

4.2.2 Iperf3

Iperf3 é uno strumento open source per la misura attiva della banda massima di una rete IP. Supporta una pletora di parametri relativi al tempo, protocolli e

modifiche al buffer. Per ogni esperimento crea un report dove risultano misurati portata, bitrate, perdita di bit e altre funzionalità. Ad oggi viene da molti considerato lo standard de facto per questo genere di misurazioni. L'architettura dello strumento è di tipo Cliente Servitore.

4.2.3 hping3

Hping3 è uno strumento open source con il quale è possibile forgiare pacchetti di tipo ICMP/UDP/TCP arbitrariamente modificati. Hping3 è molto usato per testare le corrette configurazione dei firewall e testare le performance di rete. Con Hping3 è possibile quindi creare un elevato volume di traffico, funzionalità che useremo poi per simulare attacchi di tipo DoS.

La preparazione di quest'attacco ha avuto come fase preliminare la configurazione delle varie qdisc. Come abbiamo detto nel capitolo dedicato alle TSN per realizzare lo scheduling dei pacchetti con questa strategia il primo passo è stato creare la qdisc con i privilegi d'amministratore.
Specifichiamo che CBS opera in base ad ogni coda. Per esporre le code di trasmissione a livello hardware, bisogna utilizzare MQPRIO qdisc. MQPRIO fa molto di più che esporre le code di trasmissione hardware, definisce anche come le priorità della rete su Linux vengono mappate nelle classi di traffico e come le classi di traffico vengono mappate nelle code hardware.

```
tc qdisc add dev enp0s3 \
    parent root handle 6666 mqprio \
    num_tc 3 \
    map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 \
    queues 1@0 1@1 2@2 \
    hw 0
```

In questo modo si è creato una qdisc di tipo mqprio che ha tre classi di traffico diverse, da 0 a 2, il TC di Linux ne supporta fino a 16. Così però si è mappato la priorità 3 nella classe di traffico 0, la priorità 2 nella classe di traffico 1, e tutto il resto mappato nella classe di traffico 2. Il valore map indica la priorità alla classe di traffico mappata, le priorità vanno da 0 a 15 per ogni specifica classe di traffico. Il parametro queues fornisce quante e a che offset trovare le varie classi di traffico. Ora per proseguire bisogna creare effettivamente le qdisc con la strategia CBS.
Ricordando i vari paramentri specificati nel capitolo delle TSN

- idleSlope: l'ammontare di crediti che sono accumulati quando non si trasmette in una coda

- sendSlope: l'ammontare di crediti che sono spesi quando una coda sta trasmettendo
- hicredit: l'ammontare massimo di crediti di una coda che gli è permesso avere
- locredit: il minimo di crediti che è permesso avere ad una coda

Per la simulazione dell'attacco si è deciso di simulare due AVB streaming con le seguenti proprietà:

- Stream A: SR class A, AVTP Compressed Video Format, H.264 profile High, 1920x1080, 30fps.
- Stream B: SR class B, AVTP Audio Format, PCM 16-bit sample, 48 kHz, stereo, 12 frames perAVTPDU.

Per calcolare i vari parametri di supporto allo streaming si è deciso di utilizzare uno script Python fornito da Intel [5] in quanto tale calcolo può essere triviale. Il risultato di tale calcolo porta ad avere:

- 1st priority queue: idleslope 98688 sendslope -901312 hicredit 153 locredit -1389
- 2nd priority queue: idleslope 3648 sendslope -996352 hicredit 12 locredit -113

Quindi per configurare la prima qdisc:

```
tc qdisc replace dev enp0s3 parent 6666:1 \
    cbs idleslope 98688 \
    sendslope -901312 \
    hicredit 153 \
    locredit -1389 \
    offload 0
```

Per configurare la seconda qdisc:

```
tc qdisc replace dev enp0s3 parent 6666:2 \
    cbs idleslope 3648 \
    sendslope -996352 \
    hicredit 12 \
    locredit -113 \
    offload 0
```

Arrivati a questo punto bisogna instradare il traffico dentro le code appena create, per fare questo ci viene in soccorso Iptables, con il quale è stato possibile impostare le varie classi di traffico basandosi sulle porte:

```
iptables -t mangle -A POSTROUTING -p udp --sport 7777 \
    -j CLASSIFY --set-class 6666:
iptables -t mangle -A POSTROUTING -p udp --dport 7777 \
    -j CLASSIFY --set-class 6666:2
iptables -t mangle -A POSTROUTING -p udp --dport 6666 \
    -j CLASSIFY --set-class 6666:3
iptables -t mangle -A POSTROUTING -p udp --sport 6666 \
    -j CLASSIFY --set-class 6666:3
```

Il campo che identifica la priorità può essere settato usando l'argomento '-j CLASSIFY' con argomento '-set-class' in modo tale da definire il valore della priorità.
Dopo aver configurato tutto, si è proceduto a rifarlo tramite Ansible. Il passo successivo è stato quello di testare la configurazione corretta della rete tramite Iperf, si è deciso di eleggere quindi un nodo come server.

```
iperf3 -s -p 7777
```

L'altro nodo delle rete invece è stato eletto per convenzione client

```
# IP del server nel nostro ambiente di test
# 192.168.1.34
iperf3 -c 192.168.1.34 --udp -p 7777 -b 1G
```

In questo modo il client trasferisce o cerca di trasferire 1Gb/s al server per le statistiche usando il protocollo UDP ovviamente sulla coda a priorità più elevata. Ovviamente non riuscirà a trasferire quanto richiesto in quanto abbiamo predisposto dei limiti più stringenti per la coda che siede dietro la porta 7777

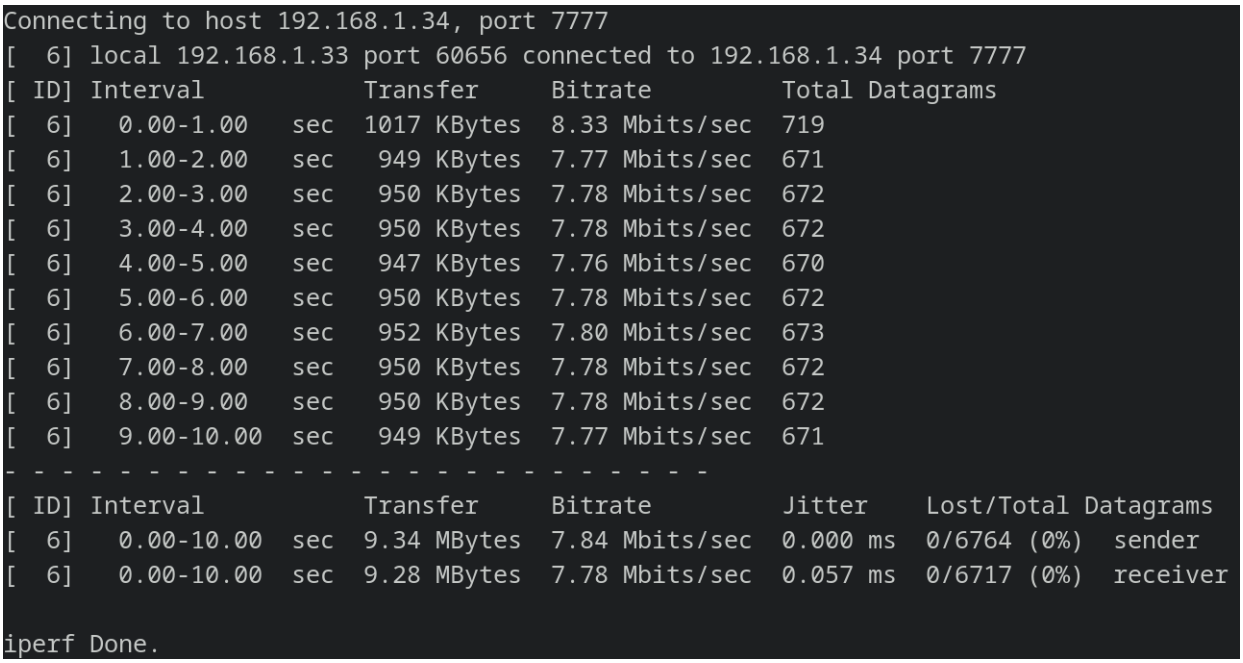

Figura 4.4: Risultato del trasferimento dal client al server sotto le qdisc implementate

```
-----------------------------------------------------------
Server listening on 7777
-----------------------------------------------------------
Accepted connection from 192.168.1.33, port 48164
[  6] local 192.168.1.34 port 7777 connected to 192.168.1.33 port 60656
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  6]   0.00-1.00   sec   950 KBytes  7.78 Mbits/sec  0.044 ms  0/672 (0%)
[  6]   1.00-2.00   sec   949 KBytes  7.77 Mbits/sec  0.038 ms  0/671 (0%)
[  6]   2.00-3.00   sec   950 KBytes  7.79 Mbits/sec  0.061 ms  0/672 (0%)
[  6]   3.00-4.00   sec   950 KBytes  7.78 Mbits/sec  0.069 ms  0/672 (0%)
[  6]   4.00-5.00   sec   947 KBytes  7.76 Mbits/sec  0.048 ms  0/670 (0%)
[  6]   5.00-6.00   sec   949 KBytes  7.77 Mbits/sec  0.049 ms  0/671 (0%)
[  6]   6.00-7.00   sec   953 KBytes  7.81 Mbits/sec  0.067 ms  0/674 (0%)
[  6]   7.00-8.00   sec   950 KBytes  7.78 Mbits/sec  0.034 ms  0/672 (0%)
[  6]   8.00-9.00   sec   950 KBytes  7.79 Mbits/sec  0.063 ms  0/672 (0%)
[  6]   9.00-10.00  sec   949 KBytes  7.77 Mbits/sec  0.057 ms  0/671 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  6]   0.00-10.00  sec  9.28 MBytes  7.78 Mbits/sec  0.057 ms  0/6717 (0%)  receiver
-----------------------------------------------------------
Server listening on 7777
-----------------------------------------------------------
```

Figura 4.5: Risultato del trasferimento dal client al server sotto le qdisc implementate

Ora che si hanno i risultati sperimentali dell'infrastruttura ben configurata andiamo a sondare il tutto quando inizia una attacco di tipo DoS. Per configurare l'attacco si è reso indispensabile usare hping3 per generare il volume di traffico necessario a stressare il sistema.

```
# prima sulla coda a maggiore priorita'
hping3 --udp --flood 192.168.1.34 --destport 7777 -d 512
# poi sulla coda a minore priorita'
hping3 --udp --flood 192.168.1.34 --destport 6666 -d 512
```

```
Connecting to host 192.168.1.34, port 7777
[  6] local 192.168.1.33 port 49522 connected to 192.168.1.34 port 7777
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  6]   0.00-1.00   sec  1017 KBytes  8.33 Mbits/sec  719
[  6]   1.00-2.00   sec   836 KBytes  6.85 Mbits/sec  591
[  6]   2.00-3.00   sec   474 KBytes  3.88 Mbits/sec  335
[  6]   3.00-4.00   sec   525 KBytes  4.30 Mbits/sec  371
[  6]   4.00-5.00   sec   478 KBytes  3.92 Mbits/sec  338
[  6]   5.00-6.00   sec   519 KBytes  4.25 Mbits/sec  367
[  6]   6.00-7.00   sec   485 KBytes  3.97 Mbits/sec  343
[  6]   7.00-8.00   sec   512 KBytes  4.19 Mbits/sec  362
[  6]   8.00-9.00   sec   492 KBytes  4.03 Mbits/sec  348
[  6]   9.00-10.00  sec   505 KBytes  4.14 Mbits/sec  357
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  6]   0.00-10.00  sec  5.70 MBytes  4.79 Mbits/sec  0.000 ms  0/4131 (0%)  sender
[  6]   0.00-10.00  sec  5.64 MBytes  4.73 Mbits/sec  0.068 ms  0/4084 (0%)  receiver

iperf Done.
```

Figura 4.6: Risultato del trasferimento dal client al server sotto le qdisc implementate con presenza Dos

```
-----------------------------------------------------------
Server listening on 7777
-----------------------------------------------------------
Accepted connection from 192.168.1.33, port 48176
[  6] local 192.168.1.34 port 7777 connected to 192.168.1.33 port 49522
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  6]   0.00-1.00   sec   950 KBytes  7.78 Mbits/sec  0.084 ms  0/672 (0%)
[  6]   1.00-2.00   sec   836 KBytes  6.85 Mbits/sec  0.539 ms  0/591 (0%)
[  6]   2.00-3.00   sec   474 KBytes  3.88 Mbits/sec  0.100 ms  0/335 (0%)
[  6]   3.00-4.00   sec   525 KBytes  4.30 Mbits/sec  0.051 ms  0/371 (0%)
[  6]   4.00-5.00   sec   478 KBytes  3.92 Mbits/sec  0.074 ms  0/338 (0%)
[  6]   5.00-6.00   sec   519 KBytes  4.25 Mbits/sec  0.070 ms  0/367 (0%)
[  6]   6.00-7.00   sec   485 KBytes  3.97 Mbits/sec  0.165 ms  0/343 (0%)
[  6]   7.00-8.00   sec   512 KBytes  4.19 Mbits/sec  0.066 ms  0/362 (0%)
[  6]   8.00-9.00   sec   491 KBytes  4.02 Mbits/sec  0.074 ms  0/347 (0%)
[  6]   9.00-10.00  sec   506 KBytes  4.15 Mbits/sec  0.068 ms  0/358 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  6]   0.00-10.00  sec  5.64 MBytes  4.73 Mbits/sec  0.068 ms  0/4084 (0%)  receiver
-----------------------------------------------------------
Server listening on 7777
-----------------------------------------------------------
```

Figura 4.7: Risultato del trasferimento dal client al server sotto le qdisc implementate con presenza DoS

Come si vede dalle figure 4.7 e 4.6 un attaccante che riesce a guadagnare l'accesso alla rete TSN riesce a introdurre un ritardo di trasmissione non indifferente sulla coda a priorità maggiore in quanto inizia a consumare tutte tutti i credit riservati allo stream A. Questo è estremamente facile da realizzare in quanto non essendo presenti nessun meccanismo di autenticazione della fonte che genera i pacchetti.

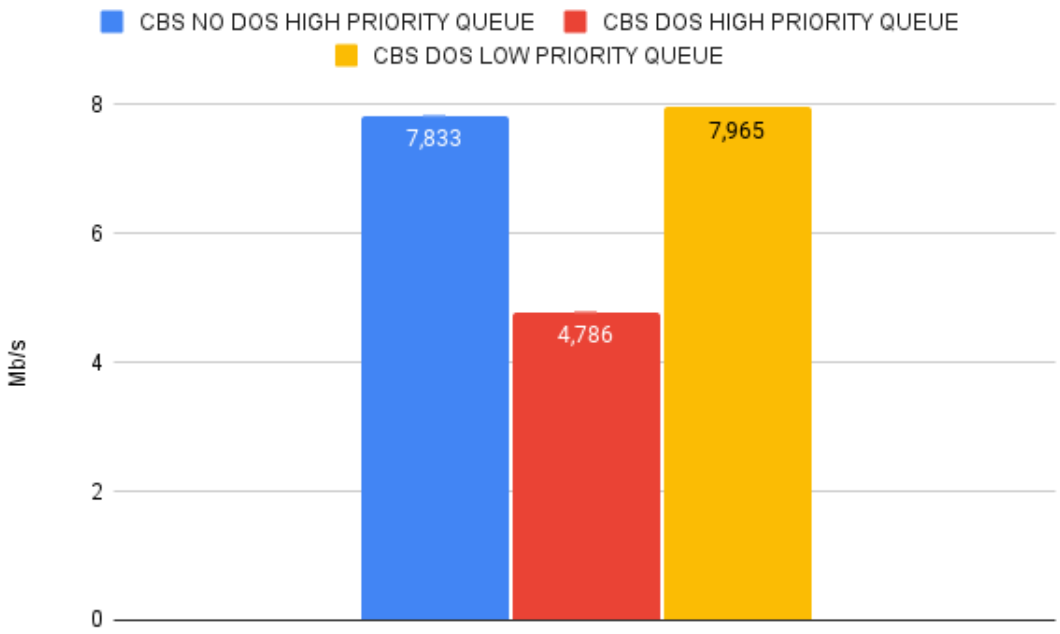

Figura 4.8: Variazione del bitrate durante l'attacco

Nella figura 4.8 è rappresentata la degradazione delle prestazioni dovute all'invio da parte di un malintenzionato di pacchetti sulla coda a massima priorità. I vari istogrammi sono frutto della media aritmetica degli esperimenti effettuati con Iperf3.
La colonna etichettata come "CBS NO DOS PRIORITY QUEUE" rappresenta il traffico sondato durante il normale funzionamento del sistema senza nessuna interferenza, quella etichettata come "CBS DOS HIGH PRIORITY QUEUE" invece rappresenta il funzionamento del sistema sotto stress da parte dell'attaccante, la colonna "CBS DOS LOW PRIORITY QUEUE" è stata usata come cartina di tornasole in quanto rappresenta il sistema quando la coda a bassa priorità è inondata di pacchetti malevoli.
Ad ogni colonna è associata il proprio intervallo di confidenza:

- 0,1749317327
- 1,524767378
- 0,1288625452

Per quanto non sia presente nessun meccanismo di autenticazione per le varie qdisc esponendole ad attacchi sia DoS, come appena dimostrato, sia ad attacchi

di spoofing e tampering. Come si evince dalle prove sperimentali la segmentazione di rete effettuata tramite l'uso di qdisc ha reso l'infrastruttura più resiliente all'attacco. Questo ci suggerisce di spostare le difese di rete, in quanto non presenti a livello protocollare, a livello applicativo o fisico. Riuscire a difendere il traffico sensibile, ad alta priorità, sarebbe una contromisura efficace, in quanto anche se un'attaccante compromettesse il traffico best effort, la parte ad alta priorità rimarrebbe perfettamente funzionante.

Capitolo 5

Conclusioni e Sviluppi Futuri

Essendo parte di un progetto più ambizioso nell'ambito della costruzione di un vero e proprio digital twin, il progetto di tesi ha già realizzato la parte di switch TSN che utilizzano il protocollo CBS, Credit Based Shaper. Come descritto nel capitolo precedente si è costruita l'infrastruttura virtualizzata di test sfruttando le tecnologie poste in analisi nell'introduzione. I risultati ottenuti negli esperimenti confermano le ipotesi studiate e riportate sullo stato dell'arte. L'esperienza accumulata durante lo svolgimento della tesi ha evidenziato alcune criticità dei protocolli TSN, una su tutte è che questa famiglia di protocolli non ha intrinsecamente proprietà di sicurezza. Il motivo principale di questa scelta progettuale risiede nella complessità di gestire lo scheduling con vincoli temporali introducendo un overhead, come potrebbe essere un livello crittografico. Ad esempio se dovessimo introdurre l'autenticazione del mittente dei pacchetti dovremmo anche introdurre meccanismi crittografici, meccanismi che consumano risorse non solo di computazione ma anche temporali; rendendo inefficace lo scheduler realtime. Probabilmente la soluzione migliore sarebbe inserire la sicurezza altrove, ad esempio spostandola a livello applicativo. Un'altra proposta che si pone nella direzione delle TSN è quella del Internet Engineering Task Force (IETF) Deterministic Networking (detnet). Questo documento di specifica, in competizione con TSN, prevede a livello di standard meccanismi di sicurezza "by desing" ma tuttora non risulta completata la standardizzazione della totalità delle proposte [6].

Nel futuro sarebbe possibile testare e verificare altri attacchi sfruttando il testbed TSN in modo tale da avere una pletora di funzionalità aggiuntive. Ulteriore

sviluppo potrebbe essere quello di analizzare le varie configurazioni di rete TSN non ancora implementate nell'infrastruttura. Il risultato poi potrebbe essere usato per avere disponibili sia attacchi che configurazioni, in un modello virtuoso per gli sviluppatori di applicazioni TSN, integrando l'infrastruttura nella Continuous Integration (CI) di sviluppo. Una proposta aggiuntiva è quella di realizzare un vero e proprio digital twin ibrido in modo tale da avere anche la controparte fisica dell'infrastruttura.

Bibliografia

[1] *Ansible configuration.* URL: https://www.lffl.org/wp-content/uploads/2021/03/ansible-managed-host-768x471.png.

[2] Davide Berardi. "Security Enhancements and Flaws of Emerging Communication Technologies". In: Alma Mater Studiorum – Università degli studi di Bologna. 2022.

[3] *Chapter 23. Configuring PTP Using ptp4l Red Hat Enterprise Linux.* URL: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/ch-configuring_ptp_using_ptp4l.

[4] *Configuring PTP Using ptp4l.* URL: %60https://access.redhat.com/webassets/avalon/d/Red_Hat_Enterprise_Linux-6-Deployment_Guide-en-US/images/90d1d6fec9026c26b1ade9b2494c79b7/ptp_grandmaster_boundary_and_slaves.png'.

[5] *Configuring TSN Qdiscs — TSN Documentation Project for Linux* 0.1 documentation.* URL: https://tsn.readthedocs.io/_downloads/cfb635037882ccb0bf5365acfd10f552/calc-cbs-params.py.

[6] *Deterministic Networking (DetNet) Security Considerations.* URL: https://datatracker.ietf.org/doc/rfc9055/.

[7] *Digital Twin capabilities.* URL: https://www.entsoe.eu/Technopedia/images/uploads/picture-digital-twin.png.

[8] Doğanalp Ergenç et al. "On the Security of IEEE 802.1 Time-Sensitive Networking". In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2021, pp. 1–6. DOI: 10.1109/ICCWorkshops50388.2021.9473542.

[9] Praerit Garg e Loren Kohnfelder. *The threats to our products.* URL: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx.

[10] "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks". In: *IEEE Std 802.1AS-2011* (2011), pp. 1–292. DOI: 10.1109/IEEESTD.2011.5741898.

[11] "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams". In: *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)* (2010), pp. C1–72. DOI: 10.1109/IEEESTD.2009.5375704.

[12] "IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications". In: *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)* (2020), pp. 1–421. DOI: 10.1109/IEEESTD.2020.9121845.

[13] *iptables - ArchWiki*. URL: https://wiki.archlinux.org/title/iptables.

[14] *Multiqueue - KVM*. URL: https://www.linux-kvm.org/images/e/e3/Ver1.jpg.

[15] *Networking - KVM*. URL: https://www.linux-kvm.org/page/Networking.

[16] *Networking with virtio: qemu implementation*. URL: https://www.redhat.com/cms/managed-files/2019-09-12-virtio-networking-fig1.png.

[17] *Networking, TUN TAP*. URL: https://www.kernel.org/doc/Documentation/networking/tuntap.txt.

[18] OWASP. *Denial of Service Software Attack — OWASP Foundation*. URL: https://owasp.org/www-community/attacks/Denial_of_Service.

[19] OWASP. *Repudiation Attack Software Attack — OWASP Foundation*. URL: https://owasp.org/www-community/attacks/Repudiation_Attack.

[20] Nataliya Shevchenko et al. *Threat modeling: a summary of available methods*. Rapp. tecn. Carnegie Mellon University Software Engineering Institute Pittsburgh United ..., 2018.

[21] *Virtual Distributed Ethernet*. URL: https://github.com/rd235/vdeplug4.

[22] *VirtualSquare*. URL: https://wiki.virtualsquare.org/.

[23] *VM Guest clock settings — Virtualization Guide*. URL: https://doc.opensuse.org/documentation/leap/virtualization/html/book-virtualization/sec-kvm-managing-clock.html.