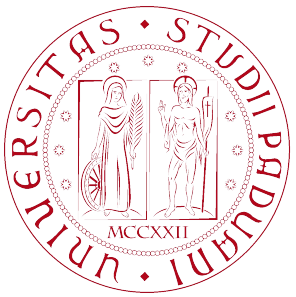


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Tesi di laurea triennale

ANALISI DI TRACCE USB PROVENIENTI DA VISORI OCULUS QUEST

Relatore:
Prof. Zanella Andrea

Laureando:
Fincato Saverio

Correlatore:
Dott. Chiariotti Federico
Dott. Testolina Paolo

ANNO ACCADEMICO: 2021/2022

Data di laurea: 21 Settembre 2022

Abstract

Questo elaborato, nella sua parte iniziale, presenta un breve riepilogo della storia e delle principali tecnologie utilizzate nella realtà virtuale dandone una definizione. Successivamente verrà esposto il funzionamento del protocollo Universal Serial Bus (USB) il quale viene utilizzato dal visore Oculus Quest 2 per la trasmissione dati.
Nella seconda parte verrà presentata un'analisi del traffico USB proveniente dal visore Oculus Quest 2. Questi dati sono stati registrati utilizzando il visore presente nel laboratorio dell'Università di Padova. I dati, poi, sono stati raccolti in tabelle e grafici in modo tale da poter rappresentare meglio i risultati ottenuti e visualizzare più chiaramente l'andamento e le caratteristiche della trasmissione dati delle applicazioni prese in esame.
L'analisi vuole caratterizzare la comunicazione tra visore e computer, concentrandosi su aspetti che riguardano caratteristiche quantitative e temporali del traffico dati. Per svolgere queste analisi sono stati utilizzati i seguenti software: Wireshark, USBPcap, Oculus Monitor.
Obbiettivo di questo studio è quindi fornire informazioni e analisi utili per le future ricerche riguardanti questa tematica.

Indice

Introduzione

Quando si parla di Realtà Virtuale (VR), molto spesso si pensa a film di fantascienza. Tuttavia, la verità è che oggi questa tecnologia si integra completamente nella nostra vita quotidiana. Dalla sua nascita, nella seconda metà del ventesimo secolo, ad oggi ha subito una notevole crescita grazie allo sviluppo tecnologico che ha permesso di ridurre le dimensioni dei dispositivi e abbassare i costi.
Infatti, negli ultimi tre anni questo mercato ha subito un vero e proprio boom, spinto soprattutto dalla pandemia causata dal Covid-19 che ha obbligato la rivalutazione dei mezzi d'intrattenimento e della comunicazione.
Il fattore che maggiormente ha limitato l'espansione di questo mercato nel passato, oltre alle limitate tecnologie relative agli schermi e al rendering 3D, è stata la necessità di utilizzare dispositivi in grado di elaborare e trasmettere grandi quantità di dati per garantire sufficienti prestazioni. Pertanto, studiando il traffico dati tra il visore e il computer sarà possibile migliorare gli algoritmi di compressione o predizione, i quali permettono di aumentare le prestazioni dei dispositivi e di ridurre i costi.
In questo elaborato andremo ad analizzare la trasmissione tra Oculus Quest 2 e un computer. Questa trasmissione, che avviene tramite collegamento USB, è stata registrata durante le nostre ricerche nel laboratorio dell'Università di Padova e in questo elaborato ne verranno analizzati alcuni aspetti. Per registrare la trasmissione sono stati utilizzati i software Wireshark, con l'ausilio di USBPcap, e Oculus Monitor, che verranno descritti nel capitolo relativo al setup utilizzato. È stato registrato il traffico dati per quattro diverse applicazioni, per un totale di 1h di utilizzo.
Nel primo capitolo dell'elaborato andremo a ripercorrere brevemente la storia della tecnologia VR.
Nel secondo capitolo si presenterà il protocollo USB, utilizzato per la connessione dati tra visore e PC.
Nel terzo e quarto capitolo verranno presentati il setup e l'analisi quantitativa e temporale di questo traffico dati. Più precisamente andremo ad analizzare: la velocità di trasferimento dati (cioè la quantità di dati per unità di tempo), la distribuzione temporale del trasferimento, il tempo inter pacchetto e la lunghezza dei pacchetti. Inoltre, utilizzando il software Oculus Monitor, che ci permette di registrare i dati dei sensori del visore, confronteremo il movimento della testa con la quantità di dati trasmessi.

Capitolo 1

Tecnologia VR

1.1 Definizione

La realtà estesa (eXtended Reality - XR) è un termine generico che include tutte le tecnologie immersive che ampliano il nostro mondo reale e lo combinano a elementi virtuali [1]. Essa rappresenta tutte le opportunità derivanti da questo campo, come la Realtà virtuale (Virtual Reality - VR) e la Realtà Aumentata (Augmented Reality - AR).
La Virtual Reality (VR) è la tecnologia più diffusa e conosciuta appartenente a questo mondo. Attraverso l'utilizzo di dispositivi, quali visori indossabili (Head-mounted display - HMD), cuffie e guanti aptici o controller, permette di proiettare l'utente in qualsiasi luogo consentendogli di vivere avventure ed esperienze in prima persona simulando qualsiasi ambientazione.
La Augmented Reality (AR) ha come obbiettivo la sovrapposizione di contenuti generati dal computer sul mondo reale. I principali esempi sono visori muniti di un display trasparente (smartglass) capaci di proiettare sul proprio campo visivo elementi digitali, oppure l'utilizzo di telecamere che permettono di proiettare oggetti digitali nella realtà.

1.2 Storia

Figura 1.1: Sensorama [2]

La nascita di questa tecnologia è attribuita alla realizzazione, nel 1957, del dispositivo "Sensorama" di Morton Heilig. Questo dispositivo non interattivo permetteva la visione 3D e creava stimoli sonori, olfattivi (utilizzando dei profumi) e tattili grazie a movimenti della poltrona. Sensorama proiettava il video di una moto e, grazie a questi stimoli, ricreava all'utente un'esperienza immersiva e realistica. Il progetto fu poi abbandonato per i costi troppo alti e la mancanza di fondi. Nel 1968 l'informatico Ivan Sutherland creò quello che viene considerato il primo vero visore per la realtà aumentata, un dispositivo che proiettava immagini 3D sovrapposte al mondo reale. Queste immagini, diversamente da Sensorama, erano create da un computer e per questo motivo Sutherland è conosciuto come "padre della computer grafica". Questo visore fu soprannominato "Spada di Damocle" [3] perché era così pesante da dover essere sostenuto da un braccio meccanico attaccato al soffitto.

Figura 1.2: VIEW NASA [4]

Successivamente, grazie al rapido sviluppo dei sistemi informatici, furono avviati vari progetti che avevano lo scopo di sviluppare questa nuova tecnologia e applicarla a diversi settori.
Il più famoso fra questi fu il progetto Virtual Interface Environment Workstation (VIEW) [4], iniziato nel 1985 dalla NASA, che riuscì in pochi anni a creare un visore VR in grado di trasmettere computer grafica 3D. Questo visore, di dimensioni ridotte e con dei particolari guanti che permettevano di tenere traccia dei movimenti della mano dell'utente, fu utilizzato dalla NASA per l'addestramento degli astronauti. In questi anni tuttavia la tecnologia VR trovò poco spazio nel mondo video-ludico. Alcuni progetti come il VR di SEGA [5], sviluppato nel 1992, o il Virtual Boy di Nintendo [6], non trovarono mai spazio nel mercato e in alcuni casi non passarono il periodo di test a causa dei feedback negativi.
La più grande spinta per la realtà virtuale è arrivata sotto forma di Oculus Rift [7].
Questo progetto nacque del 2012 da Palmer Luckey, 21 enne californiano, in un momento storico in cui la VR sembrava una tecnologia fallimentare. Luckey riuscì a conquistare l'interesse dei giocatori e a ottenere finanziamenti per un totale di 2.4 milioni fondando Oculus. In un paio di anni Oculus fu acquistata da Facebook (ora chiamata Meta e attualmente proprietaria) per due miliardi di dollari. Nel 2016 venne ufficialmente commercializzato il primo dispositivo che fece sold-out in pochi minuti. Attualmente Oculus è considerata leader nel settore VR e dal lancio di Oculus Quest 2 nel 2020, Meta ha registrato più di 15 mln di dispositivi venduti, confermando che questo dispositivo è indubbiamente un'icona culturale per questa tecnologia.

Figura 1.3: Oculus rift [7]

Capitolo 2

Protocollo USB

In questo capitolo si vuole descrivere il protocollo USB, il quale caratterizzerà l'analisi del traffico presentata successivamente. Si parlerà principalmente dello standard USB 3.0, definito nello IEC 62680-3-1:2017, che viene utilizzato nel nostro caso per collegare il dispositivo Oculus al PC. Tutta la documentazione utilizzata per questo capitolo è presente nel sito ufficiale [8]

2.1 Versioni

Il protocollo USB nasce nel 1994 quando le principali aziende informatiche, come IBM, Microsoft e NEC, si unirono per creare un collegamento per i dispositivi esterni che doveva sostituire i vecchi connettori, unificandoli in un'unica porta e aumentandone la velocità. La specifica USB 1.0 è stata introdotta nel gennaio 1996 e successivamente divenne popolare grazie al suo utilizzo nei computer di Microsoft e Apple. La forma del connettore USB è stata pensata per semplificare il suo inserimento e la sua rimozione. Le porte USB sono dotate di sistema "Plug and Play" e supportano, se il sistema operativo lo consente, la rimozione "a caldo" delle periferiche e il reinserimento senza dover necessariamente riavviare il computer. Di seguito viene riportata la tabella delle principali versioni rilasciate nel corso degli anni con la velocità teorica di trasferimento e quella reale. Quest'ultima differisce in base al sistema operativo o all'hardware utilizzato e nella tabella è inserita la media tra essi.

Nome	Versione	Velocità teorica	Velocità reale	Data di pubblicazione
Low-Speed	USB 1.0	1.5 Mb (187.5 KB/s)	1 Mbps (125 KB/s)	Gennaio 1996
Full-Speed	USB 1.1	12 Mbps (1.5 MB/s)	7 Mbps (875 KB/s)	Agosto 1998
Hi-Speed	USB 2.0	480 Mbps (60 MB/s)	280 Mbps (35 MB/s)	Aprile 2000
SuperSpeed	USB 3.0	5.0 Gbps (625 MB/s)	3.2 Gbps (400 MB/s)	Settembre 2008
SuperSpeed 10Gbps	USB 3.1	10 Gbps (1.25 GB/s)	7.2 Gbps (900 MB/s)	Luglio 2013
SuperSpeed 20Gbps	USB 3.2	20 Gbps (2.5 GB/s)	N/A	Settembre 2017

Tabella 2.1: Versioni USB [9]

La seguente tabella mostra i risultati delle diverse velocità di trasferimento dati per le varie versioni nell'utilizzo pratico:

Dimensione dati	Tempo impiegato		
	USB 1.0	USB 2.0	USB 3.0
Immagine/MP3 (4MB)	5.3 s	0.1 s	0.01 s
Flash Drive (1 GB)	22 min	33 s	3.3 s
HD-Movie (16 GB)	9.3 h	13.9 min	70 s

Tabella 2.2: Velocità di trasferimento Dati

Vantaggi dell'USB

- **Facilità d'uso** – USB è stato progettato per essere un'interfaccia semplificata. La semplicità dell'interfaccia risiede nei seguenti aspetti:
  - **Interfaccia singola per più dispositivi**: la natura versatile dell'USB elimina la complessità dei diversi tipi di connettori e requisiti hardware per ciascuna periferica.
  - **Configurazione automatica**: il sistema operativo del dispositivo host deve installare il driver del dispositivo USB solo una volta. Successivamente, ogni volta che il dispositivo periferico viene collegato, il driver viene caricato automaticamente per configurare il dispositivo collegato. Solitamente, il driver del dispositivo USB viene installato automaticamente la prima volta che il dispositivo viene connesso all'host.

  - **Facilità di espansione**: generalmente i computer possiedono 3 o 4 porte USB. Nel caso in cui siano necessarie più porte USB, è possibile utilizzare hub USB per aggiungere porte esterne.
  - **Dimensioni compatte**: i connettori USB sono di piccole dimensioni rispetto alle porte RS232 o simili.
  - **Non necessita di alimentazione esterna**: l'interfaccia USB è stata sviluppata sin dal primo giorno per fungere da alimentatore CC. Qualsiasi dispositivo host tramite la sua porta USB può fornire 5V CC fornendo da 500 mA (USB 1.0 e 2.0) a 900 mA (USB 3.0) alla periferica.

- **Velocità** – USB fornisce varie modalità di velocità che lo rendono più efficiente e veloce rispetto alle porte RS232 o simili.
- **Affidabilità** - il protocollo USB può rilevare gli errori durante il trasferimento dei dati e notificare al trasmettitore di ritrasmettere i dati. Il driver USB generico e il software del driver specifico garantiscono una comunicazione dei dati priva di errori.
- **Basso costo** - grazie alla sua natura versatile e all'elevata domanda, è diventato poco costoso produrre dispositivi supportati da USB.
- **Basso consumo energetico** - i dispositivi USB funzionano generalmente a +5V e consumano corrente nell'ordine di grandezza di un ampere. Inoltre USB possiede una modalità di sospensione, in cui la periferica consuma 500 microampere, o meno, per USB 2.0 e 2.5 milliampere o meno per USB 3.0 .

Limitazioni

Oltre a tanti vantaggi, ci sono però delle limitazioni che rendono l'USB inefficace per alcune attività:

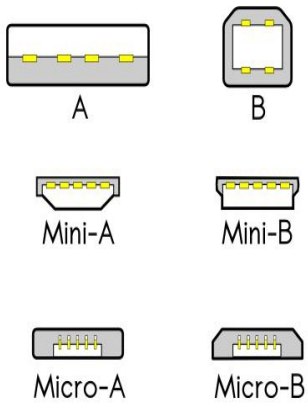
- **Velocità** – con l'introduzione di USB 3.0, è possibile raggiungere 5 Gb/sec. Questa, tuttavia, risulta essere ancora inferiore ad altre interfacce, come ad esempio Gigabit Ethernet o FireWire.
- **Comunicazione "Peer to peer"** – secondo lo standard USB, la comunicazione avviene solo tra l'host e la periferica. Due host non possono comunicare direttamente tra loro, e lo stesso vale per le periferiche. In altre interfacce, come FireWire, questa comunicazione da periferica a periferica è supportata. Per superare questa limitazione, USB ha introdotto il concetto di On The Go (OTG) . Il dispositivo OTG normalmente funziona come periferica, ma può anche funzionare come host, anche se con alcune capacità limitate.

- **Distanza** – secondo gli standard USB, il cavo di collegamento può essere lungo fino a 5 metri, oltre i quali è necessario utilizzare un hub USB per espandere la connettività.
- **Trasmissione** – USB non fornisce la funzione di trasmissione; solo i singoli messaggi possono essere comunicati tra host e periferica.

2.2 Piedinatura

Di seguito viene presentato uno schema che illustra la disposizione dei pin di contatto e la loro funzione nei connettori USB.

USB 1.0 - 2.0

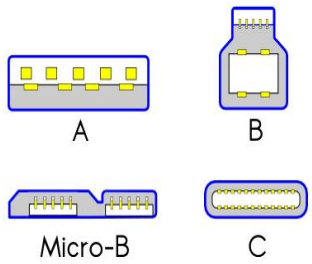


Piedinatura USB 1.0 e 2.0 [9]

Sono i primi connettori creati, supportano lo standard USB 2.0, e tutt'ora rimangono i più utilizzati nei dispositivi elettronici. Il connettore A è utilizzato per connettere la maggior parte delle periferiche odierne e permette la connessione di hub USB. Micro-B fu progettato per dispositivi di piccole dimensioni e nel 2011 divenne il principale connettore dei telefoni cellulare.

Latenza: La latenza dell'USB 1.0 è di circa 1 ms, invece la latenza dell'USB 2.0 scende a circa 125 µs.

USB 3.0



Piedinatura USB 3.0 [9]

Questi nuovi connettori 3.0 mantengono la retrocompatibilità con USB 2.0 e 1.0, permettendo l'inserzione in una parte dei suoi piedini. Le nuove porte e i connettori USB 3.0 di tipo A si riconoscono subito in quanto hanno il supporto interno dei contatti di colore blu. Il connettore tipo C, presentato nel 2014, prevede due gruppi di 12 pin ciascuno (12 sopra, 12 sotto) per permettere la reversibilità.

Latenza:
La latenza per l'USB 3.0, solo per i pin SSTX e SSRX è di circa 950 ns.

2.3 Architettura e funzionamento

Il protocollo USB, per permettere una comunicazione affidabile e il mantenimento della compatibilità tra le varie versioni, ha fornito un'interfaccia standard e un formato per i pacchetti dati. Questo protocollo permette di collegare un'interfaccia principale chiamata "Root Host", solitamente computer, laptop, smartphone, ecc., al dispositivo periferico o ad un'altra interfaccia chiamata "HUB", che permette il collegamento di più dispositivi al bus.
Il cavo di collegamento è costituito da 4 fili schermati di cui due per l'alimentazione (+5V e massa) e i restanti sono segnali dati differenziali a doppio intrecciato che utilizzano lo schema NRZI (Non Return to Zero invert) per trasmettere i dati.

2.3.1 Endpoints

Gli endpoint identificano il ricevitore o la fonte di informazioni in un canale di comunicazione. Essi sono blocchi di memoria situati in un chip controller che contiene un buffer per la trasmissione e uno per la ricezione.
Due endpoint possono avere lo stesso numero identificativo ma devono avere direzioni diverse. Quando il dispositivo si collega è accessibile solo l'endpoint predefinito 0. Quest'ultimo riceve le richieste di controllo e di stato dall'host durante il processo di enumerazione. Gli altri endpoint vengono dichiarati dopo la configurazione del dispositivo.

2.3.2 Formato dati

Per inviare dati tramite USB vengono utilizzati 4 tipi di pacchetti: Token, Data, Handshake e Start of Frame. Ogni pacchetto è formato da diversi campi come: SYNC, PID, Address, Data, Endpoint, CRC e EOP di seguito esposti. I pacchetti vengono uniti in "frame" per creare un messaggio USB [10]. I dati USB sono trasmessi nel formato Least Significant Bit (LSB).
Di seguito analizzeremo questi campi per capire di che informazioni necessita questo protocollo.

Pacchetti USB

USB Token

Il pacchetto token USB viene utilizzato per accedere all'indirizzo e all'endpoint corretti e viene inviato solamente dall'Host. È costituito dai campi SYNC, PID, un campo PID a 8 bit, un indirizzo a 7 bit, un endpoint a 4 bit e un CRC a 5 bit.

FIELD	SYNC	PID	ADDRESS	ENDPOINT	CRC5	EOP
BITS	8/32	8	7	4	5	3

Tabella 2.3: pacchetto TOKEN

Tipi di pacchetti Token inviati:

- In – Notifica al dispositivo USB che l'host desidera leggere le informazioni.
- Out: Notifica al dispositivo USB che l'host desidera inviare le informazioni.
- Setup – Questo pacchetto viene utilizzato per avviare il trasferimento di controllo.

Con l'USB 2.0 sono stati aggiunti altri due pacchetti:

- Ping – Prima di inviare i pacchetti OUT/DATA, questo token chiede al dispositivo USB se è pronto a ricevere la coppia di pacchetti OUT/DATA.
- Split - questo token viene utilizzato per comunicare con un dispositivo a bassa/massima velocità su un bus ad alta velocità.

Handshake

I pacchetti di handshake vengono utilizzati per segnalare lo stato della comunicazione, ad esempio pass/fail e vengono inviati principalmente in risposta a pacchetti di dati.

FIELD	SYNC	PID	EOP
BITS	8/32	8	3

Tabella 2.4: pacchetto Handshake

Tipi di pacchetti handshake inviati:

- ACK – Conferma per il pacchetto ricevuto
- NAK – indica che i pacchetti non possono essere ricevuti o inviati temporaneamente. Utilizzato anche per indicare che non ci sono dati da inviare
- STALL – indica che il dispositivo è in stato di errore e necessita dell'intervento dell'host.

Con USB 2.0 sono stati aggiunti altri due pacchetti:

- NYET – indica che la transazione Split non è ancora completa.
- ERR – indica che la transazione Split non è riuscita.

Start of Frame

Per completare un messaggio i pacchetti sono raggruppati in frame. Per identificare il frame è necessario indicare il suo inizio (Start of Frame Packets - SOP).

FIELD	SYNC	PID	Frame Number	CRC5	EOP
BITS	8/32	8	11	5	3

Tabella 2.5: Start of Frame

Il SOP viene inviato ogni 1 ms su un sistema USB per facilitare la sincronizzazione. Su un sistema USB Hi-Speed un SOP viene trasferito ogni 125 us.

Dati

Il pacchetto di dati può essere di lunghezza variabile, a seconda dei dati. Tuttavia, il campo dati deve necessariamente essere un numero intero di byte.

FIELD	SYNC	PID	DATA	CRC16	EOP
BITS	8/32	8	0-N	16	3

Tabella 2.6: pacchetto Dati

I tipi di pacchetti inizialmente disponibili per i dati sono Data0 e Data1.
Per i dispositivi a bassa velocità, il campo dati massimo è di 8 byte.
Per i dispositivi ad alta velocità fino a USB 2.0, il campo dati massimo è di 1023 byte.
Dopo USB 2.0, ne sono stati aggiunti altri due tipi, Data2 e MData. Questi vengono utilizzati solo nel trasferimento ad alta velocità di tipo isocrono a larghezza di banda elevata, quando è necessario trasferire più di 1024 byte a 8192 KB/s.

Campi USB

Sync

Ogni pacchetto USB inizia con il campo SYNC. Questo è utilizzato per sincronizzare il trasmettitore e il ricevitore in modo che i dati possano essere trasferiti in modo accurato.

PID

Il campo PID serve per identificare il tipo di pacchetto che viene inviato ed è composto di un identificatore a 4 bit e altri 4 bit che sono il complemento dell'identificatore per rilevare gli errori.

PID0	PID1	PID2	PID3	/PID0	/PID1	/PID2	/PID3

Tabella 2.7: campo PID

Nella tabella seguente vengono rappresentati i tipi di pacchetto inviati per le tipologie di frame:

Tipo di Pacchetto	Valore del PID	Identificatore di pacchetto
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake	0010	ACK
	1010	NAK
	1110	STALL
	0110	NYET
Special	1100	Preamble
	1100	ERR
	1000	Split
	0100	Ping

Tabella 2.8: valori di PID

Address

Viene utilizzato insieme al campo endpoint per identificare il dispositivo e indirizzare la comunicazione. L'indirizzo 0 è un caso speciale riservato all'accesso subito dopo l'accensione e il ripristino.

ADDR0	ADDR1	ADDR2	ADDR3	ADDR4	ADDR5	ADDR6

Tabella 2.9: campo Address

Endpoint

Consente una maggiore flessibilità nell'indirizzamento. Gli endpoint sono generalmente divisi per i dati d'ingresso o uscita. La lunghezza di 4 bit consente di creare 16 endpoint per ogni dispositivo.

EP0	EP1	EP2	EP3

Tabella 2.10: campo Endpoint

Dati

Il campo dati non ha una lunghezza fissa, essa è compresa tra 0 e 8192 bit e deve essere necessariamente sempre un numero intero di byte.

CRC

Questo campo viene utilizzato per il controllo della ridondanza ciclica (Cyclic Redundancy Check), ovvero ha lo scopo di rilevamento degli errori.
Ci sono due campi CRC:

Il CRC5 è lungo 5 bit e viene utilizzato con il pacchetto token e SOP.

CRC 0	CRC 1	CRC 2	CRC 3	CRC 4

Tabella 2.11: campo CRC5

Il CRC16 è lungo 16 bit e viene utilizzato per il pacchetto Dati.

CRC 1	CRC 2	CRC 3	CRC 4	CRC 5	CRC 6	CRC 7	CRC 8	CRC 9	...	CRC 14	CRC 15	CRC 16

Tabella 2.12: campo CRC16

EOP

Ogni pacchetto viene terminato con un campo End of Packet (EOP). Questo è costituito da un singolo zero finito (SE0).

2.3.3 Transazioni

Una transazione riuscita consiste in tre fasi che si verificano in sequenza. Queste sono: la fase dei token, la fase dei dati e la fase di handshake.

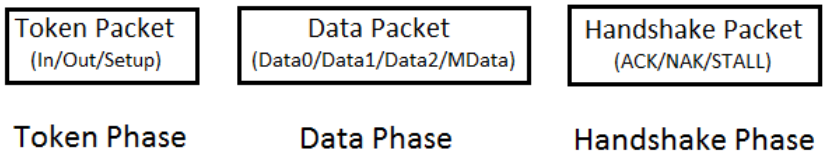


Figura 2.1: Schema a blocchi della transazione USB [11]

Queste fasi garantiscono il trasferimento sicuro dei dati.
Nella tabella seguente vengono riportati i tre tipi esistenti di transazione.

Transaction Type	Data Source	Transfers which uses this Transaction Type	Packet Contents
IN	Device(Peripheral)	Control, Bulk, Interrupt, Isochronous	Data or Status
OUT	Host	Control, Bulk, Interrupt, Isochronous	Data or Status
SETUP	Host	Control	Request

Tabella 2.13: Tabella che elenca i tipi di transazioni USB [11]

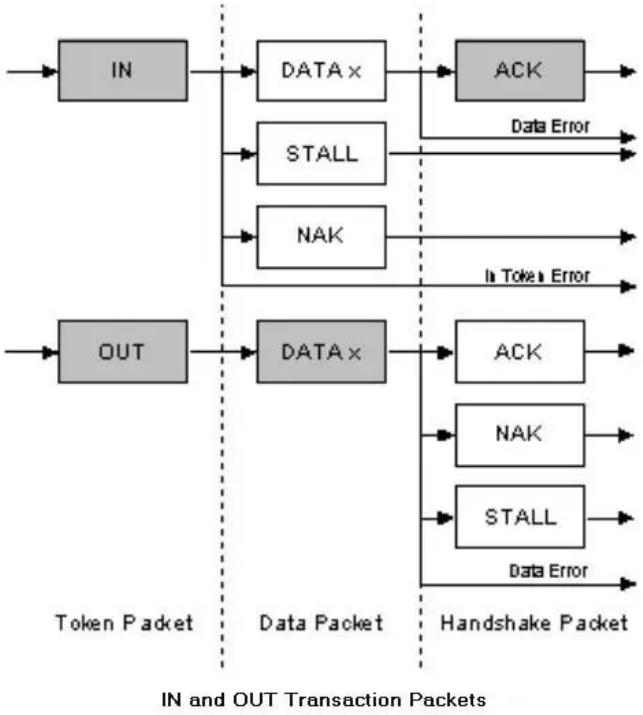


Figura 2.2: Pacchetti nella transazione IN e OUT [11]

2.3.4 Tipi di trasferimenti

Nella terza colonna della tabella 2.13 sono riportati i tipi di trasferimenti che utilizzano quel tipo di Transazione. USB attualmente supporta quattro modalità di trasferimento e ognuno di essi è progettato per risolvere scopi diversi. Queste sono: trasferimenti di controllo, trasferimenti in blocco, trasferimenti isocroni, trasferimenti Interrupt.

	Control	Bulk	Isochronous	Interrupt
Correzione errori	Yes	Yes	Yes	No
programmazione garantita	No	No	Yes	Yes

Tabella 2.14: Tabella che confronta i tipi di trasferimenti USB [11]

Transfer Type		Number of Transactions	Packets
Control	Setup Stage	One Transaction (Setup)	Token
			Data
			Handshake
	Data Stage	Zero or More Transaction (IN or OUT)	Token
			Data
			Handshake
	Status Stage	One Transaction (Opposite direction from Data stage)	Token
			Data
			Handshake
Bulk		One or More Transaction (IN or OUT)	Token
			Data
			Handshake
Interrupt		One or More Transaction (IN or OUT)	Token
			Data
			Handshake
Isochronous		One or More Transaction (IN or OUT)	Token
			Data
			Handshake

Tabella 2.15: Tabella che elenca le transazioni di diversi trasferimenti USB [11]

1. Trasferimenti di controllo

I trasferimenti di controllo vengono utilizzati per trasportare informazioni relative alla configurazione del dispositivo periferico. L'host viene a conoscenza della periferica tramite questo trasferimento. L'endpoint di controllo predefinito è sempre zero ed è esso che risponde alle richieste dell'host, come la descrizione del dispositivo. Ci sono tre fasi nel trasferimento di controllo e ciascuna fase è composta da una o più transazioni.
Queste fasi sono:

- **Fase di configurazione** – Il trasferimento di controllo inizia sempre con questa fase. L'host invia la query (richiesta) alla periferica USB.
- **Fase dati** – In questa fase vengono effettuate diverse transazioni IN o OUT. Il Pacchetto Dati contiene le informazioni relative alla richiesta effettuata nella fase precedente (configurazione).
- **Fase di stato** - questa fase utilizza la transazione IN o OUT. Questa fase avviene sempre per fornire lo stato/risultato della richiesta effettuata dall'host.

I trasferimenti di controllo sono supportati da tutte le velocità e versioni. La dimensione massima del payload (carico utile) per il pacchetto nella fase dati è diversa per ciascuna modalità di velocità: per i dispositivi a bassa velocità è di 8 byte, per i dispositivi a piena velocità può essere 8, 16, 32 o 64 byte, e per i dispositivi ad alta velocità è 64 byte. Questi byte non includono il bit PID e CRC.

2. Trasferimenti in blocco (Bulk)

I trasferimenti in blocco vengono utilizzati per trasferire grandi quantità di dati in sequenza e non hanno una larghezza di banda garantita.
I trasferimenti all'ingrosso sono unidirezionali. Questi sono supportati solo dai dispositivi Full Speed e High Speed. La dimensione massima del payload per il pacchetto di dati è diversa per ciascuna modalità di velocità: per i dispositivi a piena velocità è 8, 16, 32 o 64 byte, mentre per i dispositivi ad alta velocità arriva a 512 byte. Questi byte non includono i bit PID e CRC.

3. Trasferimenti isocroni (Isochronous)

Il trasferimento isocrono viene utilizzato quando è richiesta una consegna di dati ad una velocità costante, anche se alcuni dati vengono persi o danneggiati. La larghezza di banda è garantita per il trasferimento isocrono, ma non vi è alcuna garanzia per la consegna senza errori dei dati. In genere viene utilizzato per trasmettere informazioni sensibili al fattore tempo, come audio o video.
Questo trasferimento è unidirezionale, utilizza transazioni IN o OUT ed è supportato solo dai

dispositivi Full e High Speed. Una caratteristica speciale del trasferimento Isocrono è l'assenza dei pacchetti di handshake nella transazione.

4. Trasferimenti Interrupt

Il trasferimento Interrupt utilizza il "polling" (sondaggio) per verificare se è presente un dispositivo che desidera trasferire i dati. Se il dispositivo non ha dati da inviare risponde con un pacchetto handshake NAK. Questo sondaggio avviene regolarmente. Il trasferimento Interrupt viene utilizzato anche per trasmettere dati a un dispositivo su base programmata e il dispositivo risponde con NAK se non è pronto ad accettare dati.
In questo tipo di trasferimento, le transazioni IN e OUT vengono eseguite regolarmente su base programmata. La dimensione massima del payload per il trasferimento interrupt è diversa per ciascuna modalità di velocità: per i dispositivi a bassa velocità arriva a 8 byte, per i dispositivi a piena velocità arriva a 64 byte, per i dispositivi ad alta velocità arriva a 1024 byte. Questi byte non includono i bit PID e CRC.

2.3.5 Stati dei dispositivi

Un dispositivo USB può avere diversi stati possibili:

- **Collegato** – questo stato si verifica quando il dispositivo è collegato all'Host.
- **Alimentato** – dopo che il dispositivo è stato collegato, l'Host fornisce l'alimentazione al dispositivo se non dispone di un proprio alimentatore. Il dispositivo non dovrebbe assorbire più di 100 mA in questo stato.
- **Default** – questo stato si verifica quando il dispositivo viene ripristinato e non gli è stato assegnato un nuovo indirizzo. In questo stato il dispositivo utilizza l'indirizzo endpoint 0.
- **Indirizzato** – il dispositivo USB entra in questo stato dopo aver ottenuto un indirizzo che verrà utilizzato per comunicazioni future.
- **Configurato** – quando l'Host ottiene le informazioni richieste dal dispositivo, carica il driver appropriato per esso e successivamente configura il dispositivo selezionando una configurazione. Il dispositivo è ora pronto per eseguire le operazioni per cui è stato progettato.
- **Sospeso** – il dispositivo USB entra in questo stato quando il bus rimane inattivo per più di 3 ms. In questo stato il dispositivo non deve assorbire più di 500 uA di corrente.

2.3.6 Classe dei dispositivi

Ogni periferica USB ha una classe di dispositivo che definisce la funzionalità e lo scopo di quella periferica. L'host carica il driver adatto in base alla classe del dispositivo. Le classi di dispositivi più comuni sono:

- **Human Interface Device (HID)** – questa classe è generalmente utilizzata dai dispositivi che utilizzano Interrupt Transfer per la comunicazione dei dati. Esempi sono tastiera, mouse o joystick.
- **Audio** – questa classe è implementata da dispositivi audio come microfono, altoparlante, scheda audio esterna, ecc. Il dispositivo utilizza il trasferimento isocrono per la comunicazione dei dati.
- **Hub USB** – viene utilizzato dagli hub USB per espandere il numero di porte.
- **Immagine** – utilizzata da un dispositivo come una webcam o uno scanner.
- **Printer** – utilizzata ad esempio da stampanti laser, stampanti a getto d'inchiostro o macchine CNC.
- **Wireless** – utilizzato dall'adattatore esterno per la comunicazione wireless[Bluetooth 802.11].

2.3.7 Codifica

L'USB utilizza il meccanismo di codifica NRZI (Non Return to Zero Inversion) per codificare i dati sul bus. Nella codifica NRZI, un '1' è rappresentato da nessun cambiamento di livello, mentre uno '0' è rappresentato da un cambiamento di livello. Insieme alla codifica NRZI, il bit stiffing e il campo SYNC vengono utilizzati per la sincronizzazione tra host e dispositivo.

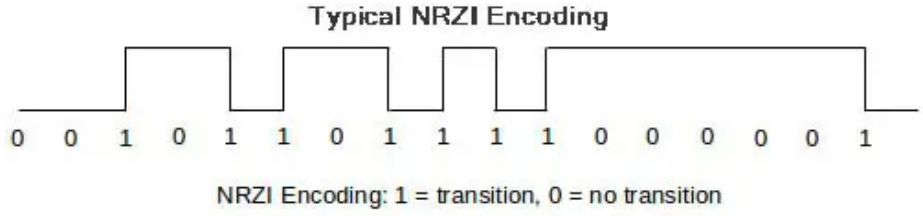


Figura 2.3: Diagramma del segnale con schema di codifica NRZI [11]

2.4 Sviluppo futuro

Attualmente l'USB si muove verso due direzioni: una prevede di rimuovere cavi e comunicare tramite onde radio, il Wireless USB, e un'altra cerca di aumentare la velocità di trasferimento dati, l'USB 4.0.

Figura 2.4: wireless USB

Wireless USB (WUSB) Questo standard utilizza la comunicazione tramite onde radio. Queste onde sono dotate di elevata ampiezza di banda a corto raggio, che combinano la velocità dei dispositivi USB 2.0 con la praticità della tecnologia wireless. Utilizzando le onde radio, l'USB subisce i limiti fisici che esse posseggono, riducendo la banda in base alla distanza. Inoltre, la conversione del segnale cablato in segnale a radiofrequenze e viceversa, introduce del ritardo rallentando ulteriormente la comunicazione. Attualmente è largamente utilizzato nei dispositivi come mouse, tastiere e cuffie, anche se non è consigliato nei casi in cui si necessita di precisione e velocità, come nel gaming. Infatti tutt'ora questo standard non è consigliato per la comunicazione dei visori a realtà aumentata, i quali necessitano di elevata larghezza di banda mantenendo una bassa latenza.

USB 4.0 Questo standard è stato presentato nel 2019 e porta alcune novità rispetto agli standard precedenti. Una di queste è l'utilizzo dell'interfaccia Thunderbolt, che al momento è l'interfaccia più veloce nel mercato, permettendo una velocità di trasferimento fino a 40 Gbps in entrambe le direzioni. Per utilizzare questo standard risulta necessario usare appositi cavi, i quali contengono diversi canali dati e video, che permettono di sommare la loro banda utilizzabile.
Il connettore USB-C, il quale già supporta un'ampia varietà di protocolli che consentono di collegare interfacce come HDMI e DisplayPort, con l'USB 4.0 subirà ulteriori miglioramenti su questo versante. Il consorzio VESA ha infatti recentemente annunciato il DisplayPort Alt Mode 2.0, una nuova interfaccia utilizzabile in questi connettori, i quali permetteranno di trasferire segnali video con risoluzione fino a 16K e frequenza di aggiornamento pari a 60 Hz, utilizzando proprio una connessione USB 4.0. Questo standard sicuramente porterà benefici nel campo VR.

Capitolo 3

Setup e acquisizione tracce

3.1 Dispositivi

Computer

Il computer utilizzato in questa sperimentazione è un PC desktop HP - Omen Obelisk, dotato di un processore Intel i7-9700k, 64 GB di RAM, una scheda grafica NVIDIA GTX 2080Ti e Windows 10. Nel computer vengono eseguite le applicazioni e il rendering grafico, che poi vengono trasmessi via cavo Meta Link [12] al visore.

Visore Oculus (Meta) Quest 2

Il dispositivo noto con il nome di Oculus Quest 2, che diventò Meta Quest 2 nel 2021, è un visore di Realtà Virtuale di casa Facebook. È dotato di un display single-panel RGB-stripe LCD a commutazione rapida, diviso in due lenti regolabili, con una risoluzione di 1832x1920 per occhio e una frequenza di aggiornamento che può essere impostata a 60, 72, 90 e 120 Hz. Il tracking è fatto con 6 gradi di libertà (3 di posizione e 3 di rotazione), riuscendo a catturare i movimenti della testa e del corpo senza la necessità di sensori esterni. Il visore è dotato di due controller, il cui uso è opzionale ma consigliato nelle applicazioni in cui si necessita dell'utilizzo delle mani, poiché il tracciamento fatto dal solo visore risulta poco preciso. Il dispositivo è dotato di audio posizionale 3D integrato direttamente nel visore, ma è possibile collegare delle cuffie tramite la porta audio da 3,5mm. Il visore è di tipo all-in-one, ovvero permette l'utilizzo del visore e dei controller senza l'ausilio di un PC. In questa modalità viene utilizzata solamente la potenza di calcolo del visore, il quale possiede un processore Qualcomm Snapdragon XR2 e 6 GB ram.
Nella nostra sperimentazione abbiamo utilizzato il visore collegato al PC, così da sfruttare la sua potenza di calcolo e permetterci di registrare i dati che vengono scambiati. I due dispositivi sono collegati tramite il cavo Meta Link [12], che sfrutta l'USB type-C versione 3.0 per la comunicazione, con velocità di trasferimento di almeno 2 Gb/s.

3.2 Setup e Software

Come già visto, nel nostro studio andremo ad analizzare i dati che vengono scambiati tra il computer e il visore Oculus. Per registrare questi dati sono stati utilizzati tre software: Wireshark, USBPcap e Oculus Monitor.

Fase di registrazione Inizialmente abbiamo collegato il visore al PC e abbiamo attivato la connessione tramite Oculus Link. Successivamente abbiamo aperto e impostato il software Oculus Monitor, abbiamo avviato Wireshark per leggere i dati nell'interfaccia USB utilizzata dal collegamento e abbiamo aperto l'applicazione che intendevamo registrare tramite la piattaforma di giochi Steam. Dopo che l'applicazione si è avviata abbiamo iniziato la registrazione sia su Oculus Monitor che su Wireshark e abbiamo iniziato ad utilizzare il visore. Al termine della cattura abbiamo fermato la registrazione in entrambi i software e abbiamo salvato i dati.

Limiti del setup I principali limiti che questo setup ci pone sono: la modalità di registrazione dei dati USB, la composizione dei dati e la dimensione dei essi.
Il primo limite che abbiamo riscontrato è la **modalità di registrazione dei dati** utilizzata dal software USBPcap. Infatti, USBPcap accorpa varie transazioni in un unico frame e questo causa la perdita di alcune informazioni relative ai singoli pacchetti USB.
Un altro limite riguarda la **composizione dei dati** poiché non è stato possibile decodificare il contenuto del payload dei pacchetti catturati. Questo a causa del fatto che Oculus non rivela la formattazione di essi e anche dopo vari tentativi, non è stato possibile decifrarli. Riuscire a decifrarli permetterebbe di caratterizzare il traffico dati in relazione al tipo di contenuto inviato rendendo l'analisi molto più interessante.
L'ultimo limite che abbiamo riscontrato riguarda la **dimensione dei dati**, infatti, la registrazione fatta ha prodotto due differenti tipi di dati. I dati prodotti da Oculus Monitor, salvati in file CSV, sono di piccole dimensioni e generalmente questo software produce circa 1 MB di dati al minuto. Invece Wireshark e USBPcap, salvando tutto ciò che transita per la porta USB, restituiscono una grossa quantità di dati. Infatti, questi software durante le nostre registrazioni hanno prodotto poco meno di 1 GB al minuto. Questo impedisce la registrazione di tracce per lungo tempo poiché per 20 minuti di registrazione vengono registrati circa 20 GB di dati. Risulta quindi complesso fare questo tipo di registrazioni se non si dispone di un sufficiente spazio di memoria e una sufficiente potenza di calcolo, anche per le successive analisi.

3.2.1 Wireshark

Il software Wireshark è il packet sniffer più utilizzato al mondo [13]. E' in grado di leggere e registrare i dati che passano in molte delle interfacce presenti nel computer come ethernet, wifi, bluetooth, ecc. Nella nostra ricerca lo abbiamo utilizzato, nella versione 3.6.6, per la registrazione dei dati che transitavano nella porta USB-C della scheda madre con l'aiuto del software USBPcap. Wireshark, inoltre, permette di utilizzare dei filtri, da applicare ai dati, per ottenere solo le informazioni di cui si necessita.

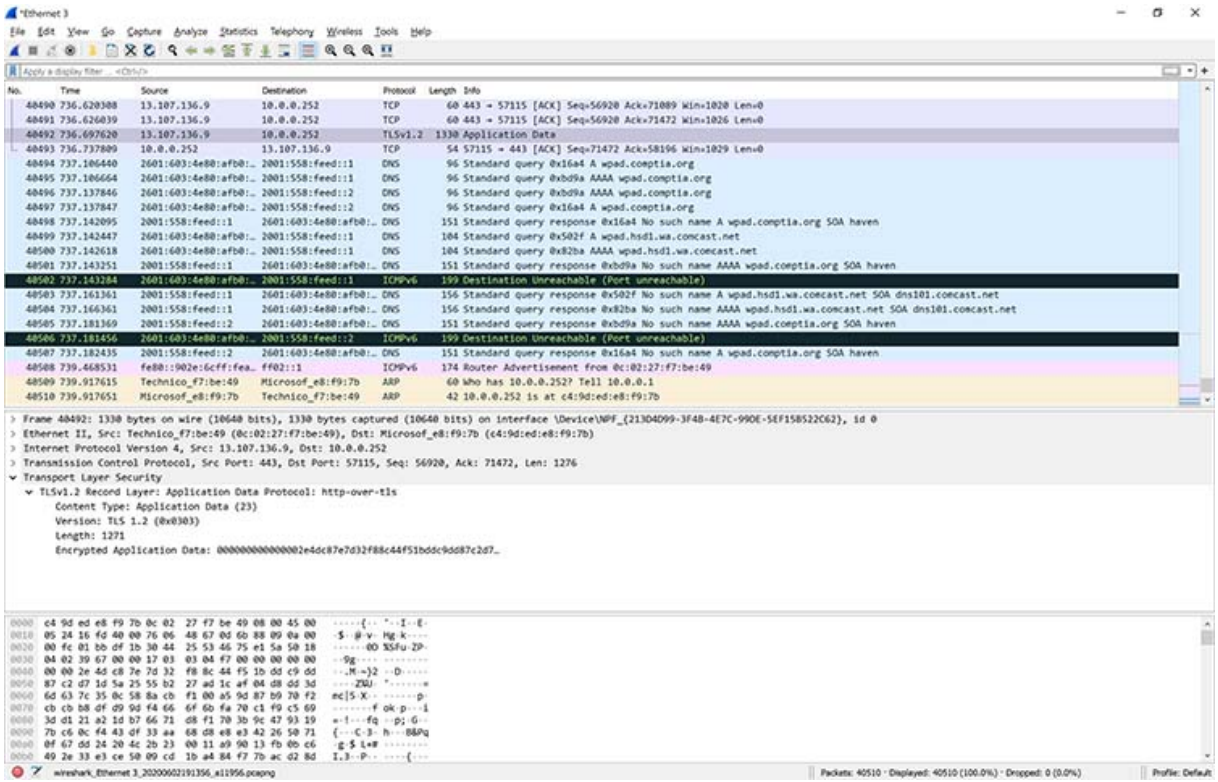


Figura 3.1: Esempio di schermata di Wireshark [13]

3.2.2 USBPcap

Questo software permette di catturare le tracce USB che altrimenti sarebbero invisibili a Wireshark [14]. Il software registra i dati inserendo un header ad ogni pacchetto. USBPcap cattura i dati degli USB Request Block (URB) che vengono poi visualizzati come frame da Wireshark. Questo software però non salva i dati esattamente nello stesso formato delle specifiche USB, quindi i frame visualizzati su Wireshark risulteranno in alcuni aspetti diversi da esse. Elementi che non si vedono in USBPcap ma che sono presenti nel formato USB sono:

- Stati del bus (Sospeso, Acceso, Spento, Ripristino, Handshake).
- ID pacchetto (PID).

- Tempo utilizzato per trasferire il pacchetto via cavo.
- Velocità di trasferimento.
- Enumerazione USB completa.

Nelle tracce registrate la maggior parte dei trasferimenti saranno di tipo Bulk. Queste sono costituite da tre pacchetti: IN o OUT DATA e ACK. Ogni trasferimento Bulk risulta essere un frame USBPcap, questo contenente i dati combinati di tutte le transazioni appartenenti a quel determinato trasferimento.

3.2.3 Oculus Monitor

Oculus Monitor è un software che ci permette di catturare le informazioni in tempo reale provenienti dal visore [15]. Permette di accedere ai valori dei sensori come posizione, accelerazione, rotazione, ecc., sia del visore che dei controller. Oltre alla visualizzazione in tempo reale permette di registrare tutti questi valori e di salvarli in file di testo in formato CSV. Questo software ci permette di registrare i dati di movimento e associarli ai dati registrati tramite USB.

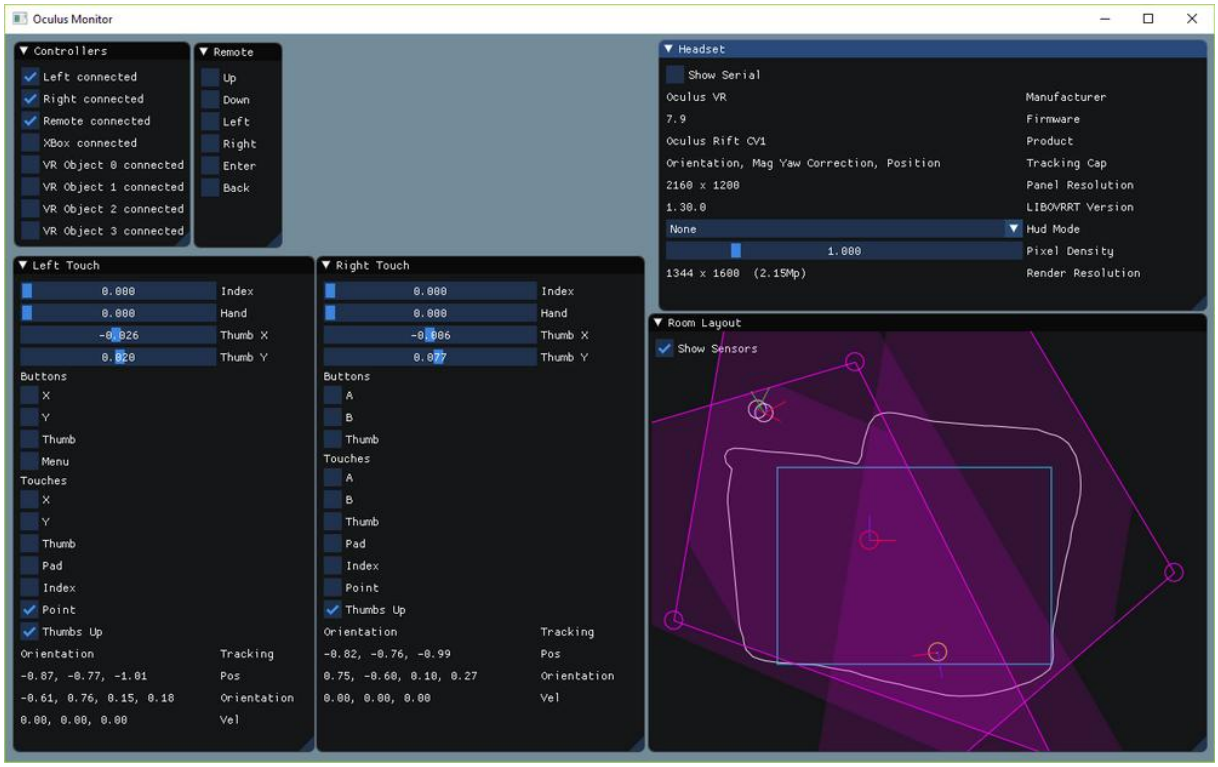


Figura 3.2: Esempio di schermata di Oculus Monitor [16]

3.3 Applicazioni

3.3.1 Beat Saber

La prima applicazione utilizzata è Beat Saber, sviluppata e pubblicata da Beat Games. E' un gioco a sfondo musicale in cui si devono tagliare dei cubi con delle spade che vengono mosse tramite i controller. Questo gioco utilizza tutti i gradi di libertà del visore ma i controller vengono utilizzati solo come puntatori e non vengono registrati i movimenti delle dita. Sono state registrate varie sessioni di 3/4 minuti ciascuno per un totale di 20 minuti. È stato utilizzato con la risoluzione del visore a 90 Hz.

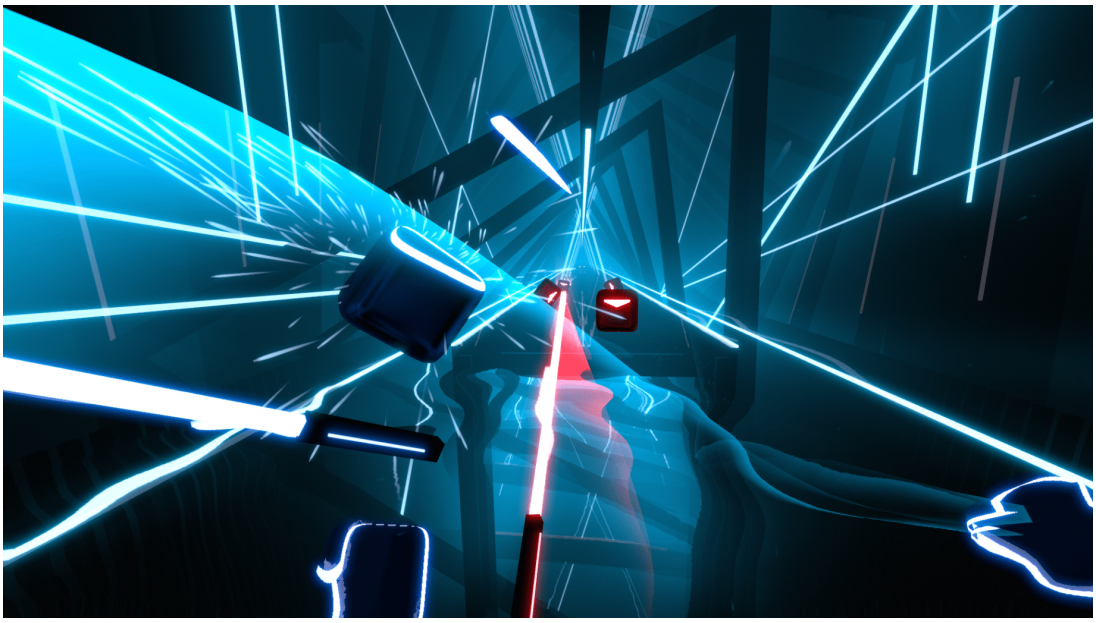


Figura 3.3: Beat Saber gameplay [17]

3.3.2 Half-Life: Alyx

È un videogioco di genere "sparatutto" in prima persona creato e pubblicato da Valve Corporation esclusivamente per la realtà virtuale. Questo gioco è stato scelto principalmente perché si voleva testare un'ambientazione che rimanesse stabile a meno di movimenti volontari della testa. In questo gioco vengono registrati i movimenti delle dita dai controller che servono per afferrare gli oggetti nel gioco. Di questa applicazione è stato registrato un gameplay di 20 m più qualche test che verrà presentato in seguito. È stato utilizzato con la risoluzione del visore a 90 Hz.

Figura 3.4: Alyx gameplay [18]

3.3.3 Medal of Honor: Above and Beyond

È un videogioco di genere “sparatutto” in prima persona prodotto da Respawn Entertainment e distribuito da EA. In questo gioco l’ambientazione è principalmente stabile ed è il personaggio a muoversi come su Alyx, ma la grafica risulta meno qualitativa. Anche in questo gioco vengono registrati i movimenti delle dita dai controller. Di questa applicazione sono stati registrati due gameplay per un totale di 15 m ed è stato utilizzato con la risoluzione del visore a 90 Hz.

Figura 3.5: Medal of Honor gameplay [19]

3.3.4 Microsoft Flight Simulator

È un simulatore di volo di casa Microsoft. In questo gioco l'ambientazione non è stabile, soprattutto durante il volo, e i controller vengono utilizzati solo come puntatori e non vengono registrati i movimenti delle dita. Di questa applicazione è stato registrato un gameplay di circa 7 minuti. E' stato utilizzato con la risoluzione del visore a 90 Hz.

Figura 3.6: Flight Simulator gameplay [20]

Capitolo 4

Analisi delle tracce

In questa sezione verranno analizzati i dati che sono stati registrati con i software sopra elencati. Inizialmente vediamo come i dati vengono visualizzati su Wireshark dopo la registrazione usando USBPcap:

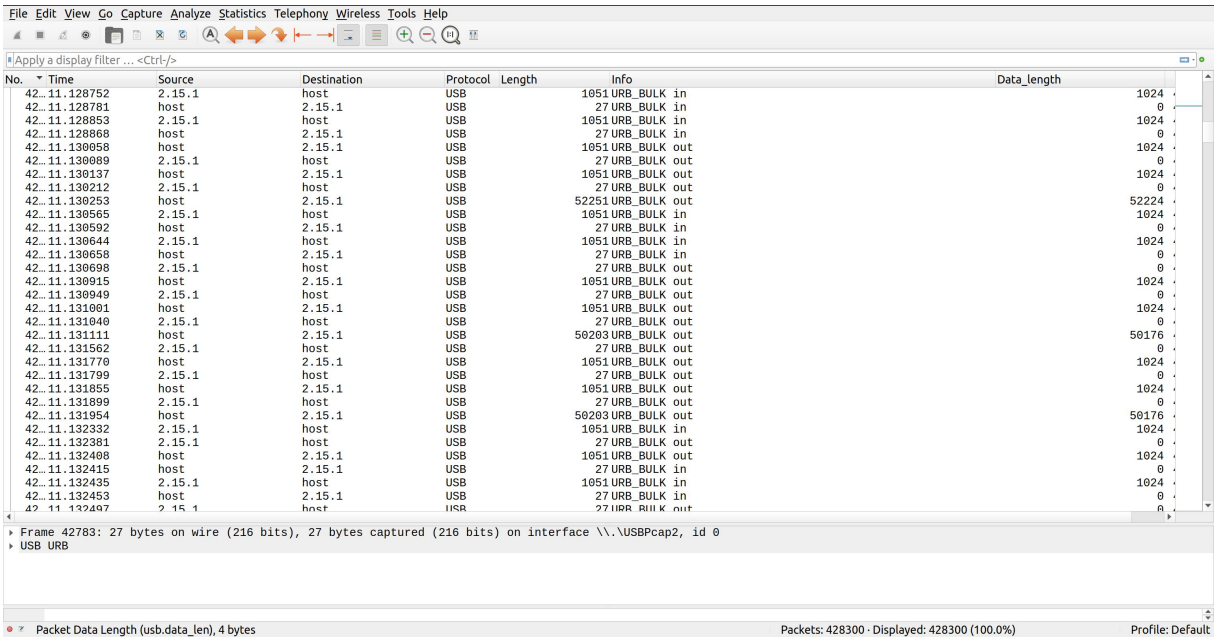


Figura 4.1: Visualizzazione dati su Wireshark

La figura 4.1 riporta una schermata che illustra come i dati vengono visualizzati su Wireshark. In questa schermata è possibile vedere subito le principali informazioni dei pacchetti che ci servono per la classificazione. In ordine queste informazioni sono:

- **No** – è un numero progressivo che identifica la transazione.

- **Time** – sono i secondi trasmessi dall'inizio della registrazione, cioè il tempo relativo, ma come vedremo USBPcap salva anche il tempo assoluto sotto forma di Unix Epoch Time.

- **Source e Destination** – sono la sorgente o la destinazione della transazione. Host rappresenta l'interfaccia del PC dove avviene la registrazione, mentre "2.15.1" è, in ordine, l'indirizzo

USB nel bus, indirizzo del dispositivo ed endpoint associati alla periferica che comunica, nel nostro caso il visore.

- **Protocol e info** – sono il protocollo utilizzato per la comunicazione e, in info, il tipo di trasferimento utilizzato. Nel nostro caso per quasi la totalità dei pacchetti si utilizza il trasferimento USB BULK, descritto nella capitolo dedicato all'USB, che è utilizzato per la comunicazione di grosse quantità di dati. Notiamo che "USB_BULK in" e "USB_BULK out" si riferiscono alla direzione del trasferimento, "in" da dispositivo a PC e "out" da PC a dispositivo.

- **Length e Data_length** – sono la lunghezza in byte del frame e la lunghezza in byte del campo data (payload) contenuto nel frame.

Queste non sono tutte e sole le informazioni che abbiamo dei dati, ma nella nostra ricerca andremo a svolgere solamente un analisi statistica della trasmissione, poiché il payload è codificato con il formato proprietario di Oculus, non disponibile pubblicamente, e non è stato possibile risalire alla tipologia e al contenuto dei dati trasmessi.
Nello specifico, nelle seguenti sezioni andremo a dividere il traffico nelle due direzioni "out" in cui i dati sono generati dal PC e inviati al visore e "in" dove è il visore a generare i dati e inviarli al PC. Questa divisione ci permette di associare meglio gli aspetti che andremo ad analizzare alla direzione della trasmissione.
Più precisamente andremo ad analizzare :

- Data rate, ovvero la quantità di dati generati e inviati dalla sorgente in un determinato lasso di tempo, differenziandolo per applicazione.

- Distribuzione temporale.

- Tempo inter pacchetto.

- Lunghezza media dei pacchetti.

4.1 OUT Direction

In questa sezione andremo ad analizzare il traffico con direzione "out". Per ottenere il traffico di questa sezione è sufficiente controllare che la direzione del campo USB Info risulti "out", inoltre per le analisi come data rate e tempo inter pacchetto sono stati eliminati dal traffico tutti i pacchetti di risposta di ricezione avvenuta (ACK) [21] poiché non contengono dati e quindi non sono rilevanti per queste analisi. In altre analisi invece questi pacchetti andranno conteggiati.

4.1.1 Data rate

In questa sezione parleremo della quantità di dati che ogni applicazione genera durante il suo utilizzo. Per ogni applicazione verrà riportato graficamente:

- l'andamento del data rate, in questo caso considerato come velocità di trasferimento dati, quindi il numero di dati inviati per unità di tempo (nel nostro caso MegaByte per secondo).
- la regressione polinomiale di ordine 10 dell'andamento del data rate.
- la media totale.

E' stato deciso di raggruppare i byte in finestre di 1 secondo per rendere l'analisi più leggibile, poiché se graficate le dimensioni di ogni pacchetto singolarmente si presentano con eccessive oscillazioni, che ne rendono impossibile la visualizzazione. Per questo motivo abbiamo inserito la regressione polinomiale che rende più chiaro l'andamento del grafico. La media totale, invece, ci aiuta a confrontare le applicazioni tra loro, classificandole in base alla mole di dati che genera. Questo dato è significativo in quanto tutte le catture sono state fatte su un intervallo di tempo abbastanza ampio, dai 6 ai 20 minuti, che dovrebbe essere sufficientemente grande per rappresentare l'utilizzo reale medio dell'applicazione.

Beat Saber

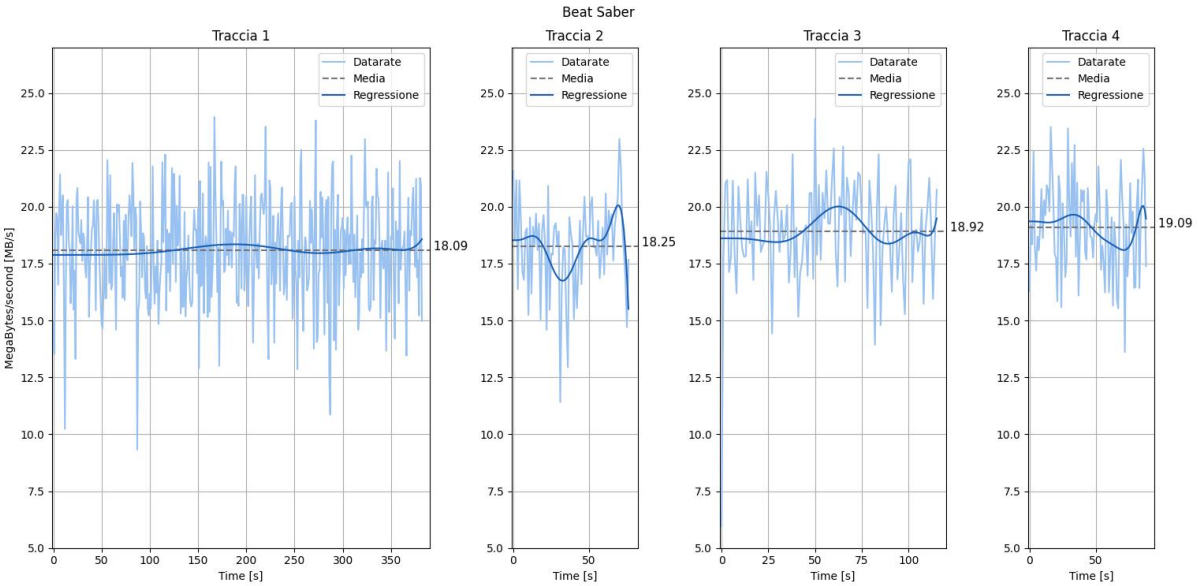


Figura 4.2: Data rate di Beat Saber

In questo grafico sono presenti i dati delle quattro catture fatte con diverse mappe di gioco. Possiamo notare che il data rate risulta molto simile per tutte le catture.
Nel grafico delle catture 2 e 4 si può notare un calo dell'andamento in alcuni punti, questo è

stato causato dalla “morte” durante il gameplay che ha interrotto il gioco e riportato alla schermata di avvio. Questa schermata essendo prettamente statica ha sicuramente comportato un abbassamento dei dati relativi all’ambientazione grafica e quindi un conseguente abbassamento del traffico scambiato.
Invece durante la cattura 3 è stata utilizzata una mappa di gioco che era poco dinamica tranne per un momento centrale in cui bisognava muoversi molto, infatti dal grafico è possibile notare un innalzamento dell’andamento causato sicuramente da questo movimento.

Medal of Honor: Above and Beyond

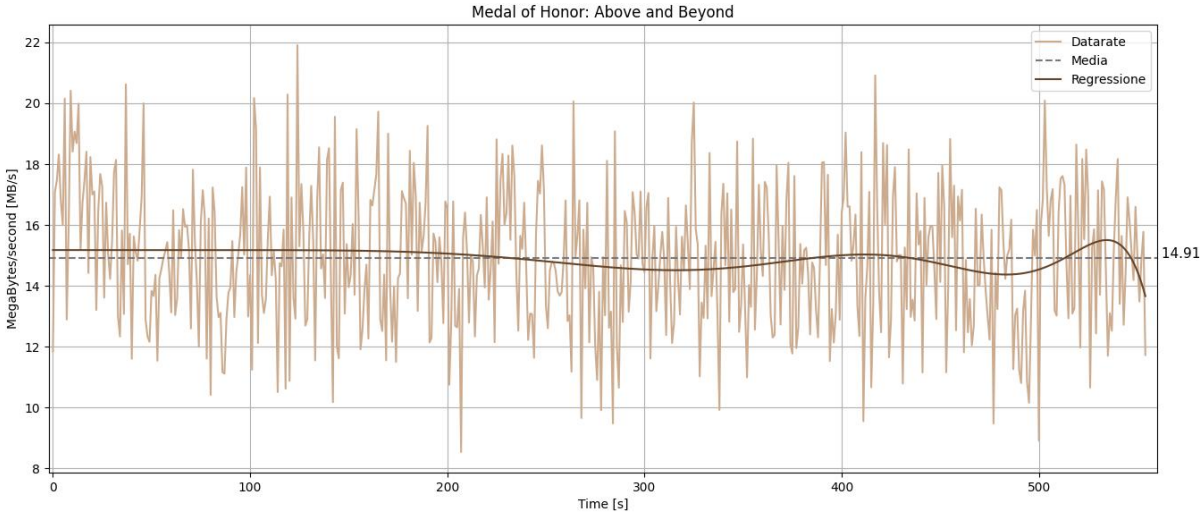


Figura 4.3: Data rate di Medal of Honor

Dal grafico si può notare che l’andamento risulta abbastanza regolare per tutta la durata della cattura, tranne alcuni punti in cui cala poiché, ricordando che l’applicazione è un gioco di categoria “sparatutto”, in alcuni momenti il giocatore si è fermato all’interno di strutture, quindi l’ambientazione si è stabilizzata, probabilmente, facendo diminuire il traffico dati.

Flight Simulator

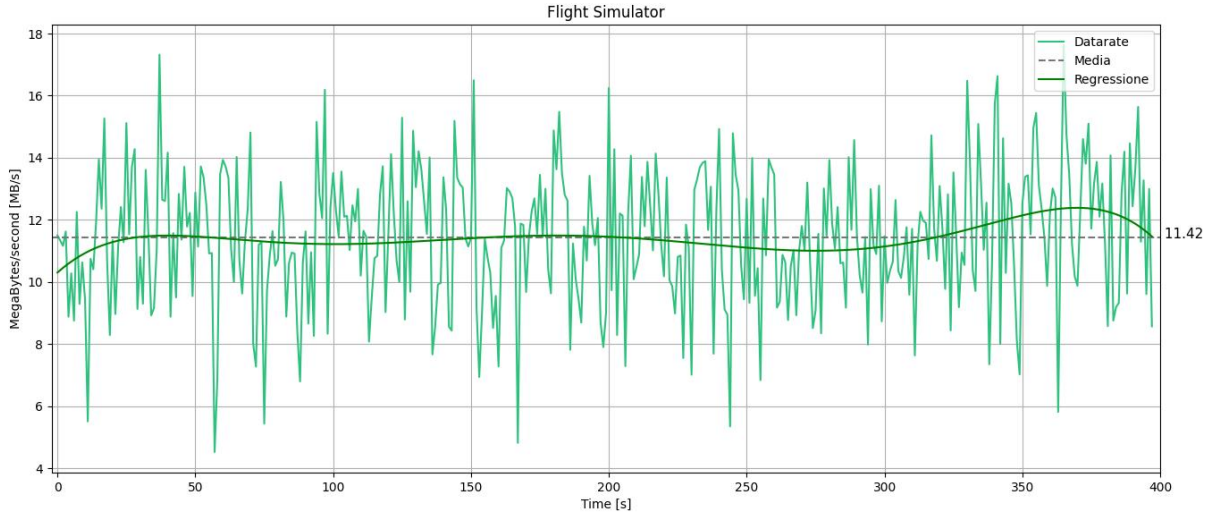


Figura 4.4: Traccia di Flight Simulator

Anche in questo grafico possiamo notare che il traffico risulta abbastanza costante e la regressione segue molto l'andamento medio. Gli unici punti in cui si nota un andamento diverso sono l'inizio, causato della presenza di un piccolo tutorial che mostra la strumentazione e non è visibile l'esterno del veivolo, questa fase sicuramente richiede pochi dati ambientali e dal grafico si vede che inizialmente ci sono meno dati inviati. Successivamente si inizia ad usare il veivolo ma essendo fermi nella pista l'ambiente attorno è pressoché stabile e dal grafico si nota che l'andamento tende a stabilizzarsi. Dopo alcuni minuti, nella parte finale della cattura, il veivolo era in volo e in questa parte del gioco è sicuramente richiesto un gran numero di dati ambientali, infatti si nota che verso la conclusione della registrazione è presente una crescita del numero di dati trasmessi.

Half-Life: Alyx

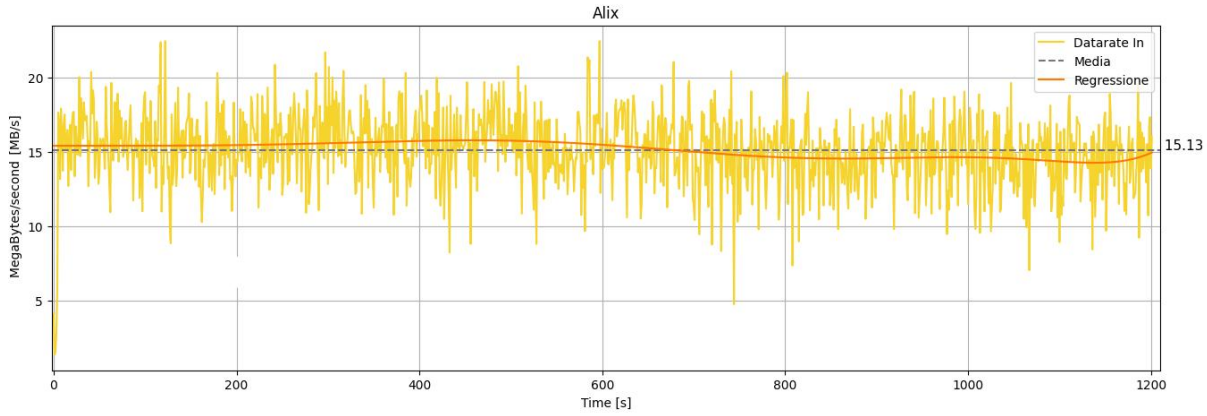


Figura 4.5: Traccia di Alix

Anche in questo grafico possiamo notare che il data rate risulta abbastanza costante e la sua regressione segue molto l'andamento medio. Questo potrebbe essere dovuto all'ambientazione che rimane relativamente stabile per tutta la durata del gioco.

Confronto

Applicazione	**Data rate medio [MB/s]**
Beat Saber	18.14
Alix	15.13
Medal of Honor	14.50
Flight Simulator	11.48

Tabella 4.1: Confronto tra le medie dei data

Nella tabella 4.1 sono stati rappresentati i valori medi del data rate delle quattro applicazioni. Da questo grafico possiamo notare che tra le loro medie c'è una differenza che può raggiungere anche il 40%. Questa differenza potrebbe essere data dal tipo di gioco utilizzato, infatti va sottolineato che il gioco con la media più alta, cioè Beat Saber, risulta quello con l'ambientazione più dinamica, mentre il gioco con la media più bassa, Fligt Simulator, ha l'ambientazione più stabile e richiede meno movimento della testa.

4.1.2 Distribuzione temporale

In questa sezione parleremo del modo in cui dati vengono inviati nell'arco temporale, in particolare analizzeremo la frequenza con cui il PC invia i dati al dispositivo. Per caratterizzare meglio la distribuzione temporale dei pacchetti abbiamo analizzato diversi intervalli di tempo: 1s, 3s, 10s e 30s.
Per ogni applicazione andremo a selezionare solo i pacchetti che hanno direzione in uscita, cioè da PC a visore, controllando che il campo usb.irp_info.direction valga 0, senza tener conto dei messaggi di ACK. Per ogni intervallo di tempo che vogliamo analizzare prenderemo questi pacchetti e andremo a registrare l'istante in cui vengono inviati. Successivamente andremo a raggruppare questi istanti in una finestra di 100ms e andremo a rappresentarli graficamente. Questo ci mostrerà la frequenza di invio dei dati generati dall'applicazione.

Beat Saber

Questa applicazione è stata utilizzata con una frequenza d'aggiornamento (refresh rate) di 90Hz, ciò significa che il visore dovrebbe aggiornare l'immagine visualizzata nel display 90 volte al secondo, cioè 9 volte ogni 100 ms. Queste immagini, quando il visore viene utilizzato con l'ausilio del PC sono generate da esso e inviate tramite il link USB. Per questo motivo nei grafici si dovrebbero notare, tra i pacchetti con peso maggiore in cui si suppone siano presenti i dati relativi alle immagini, questi nove gruppi di trasmissione.

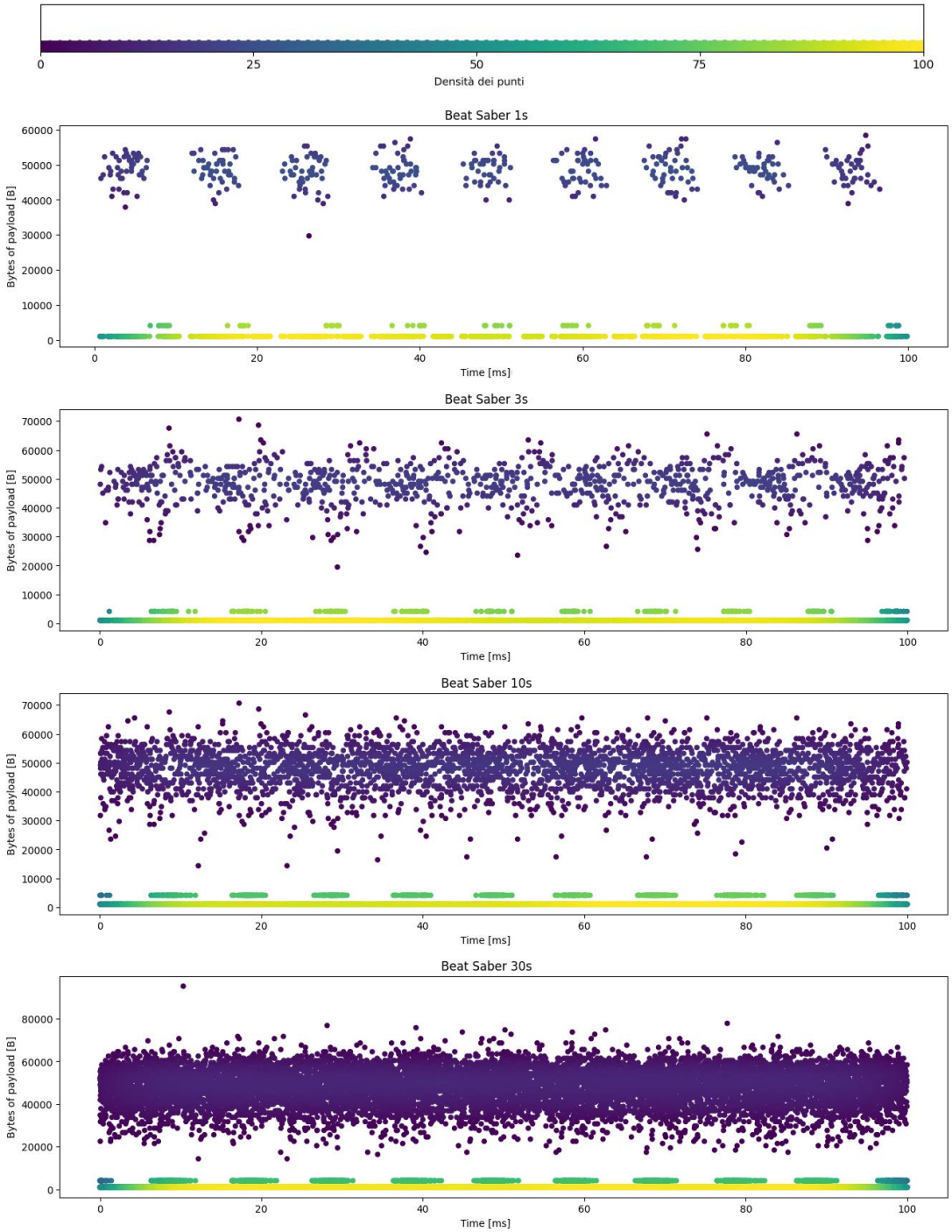


Figura 4.6: Traccia di Beat Saber originale

Dai grafici in figura 4.6 notiamo che in intervalli piccoli, da 1 a 3 secondi, questi gruppi sono visibili, invece per intervalli maggiori, da 10 a 30 secondi, i gruppi non sono più distinguibili. Analizzando la traccia abbiamo notato che ad ogni secondo questi gruppi risultano essere traslati in ritardo di 2 ms. Nei grafici sottostanti viene riportata la stessa traccia in cui si è andato a sottrarre 2 ms per ogni secondo al tempo di invio, che rende possibile osservare tutti e nove gli istanti di invio anche per intervalli di tempo più grandi.

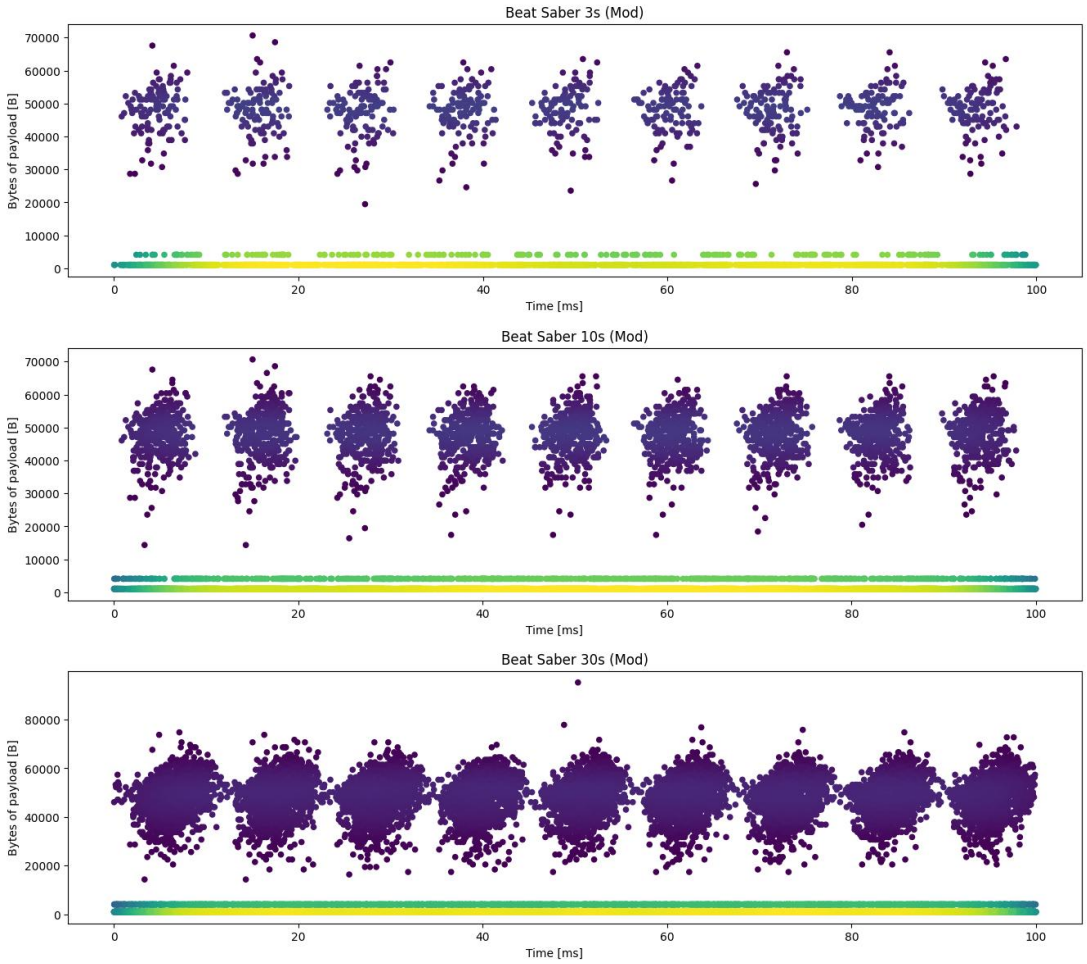


Figura 4.7: Traccia di Beat Saber modificata

Inoltre da questi grafici in figura 4.7 possiamo notare che i pacchetti con dati di grandi dimensione, risultano essere concentrati lungo l'asse Y in un intervallo che va dai 20 ai 70 kB. Nella sezione dedicata alla lunghezza dei pacchetti approfondiremo quest'aspetto.

Flight Simulator

Anche questa applicazione è stata utilizzata con 90 Hz di refresh rate. Quindi anche per questa applicazione ci aspettiamo di osservare i gruppi visti precedentemente. Di seguito viene riportato il grafico con i dati reali.

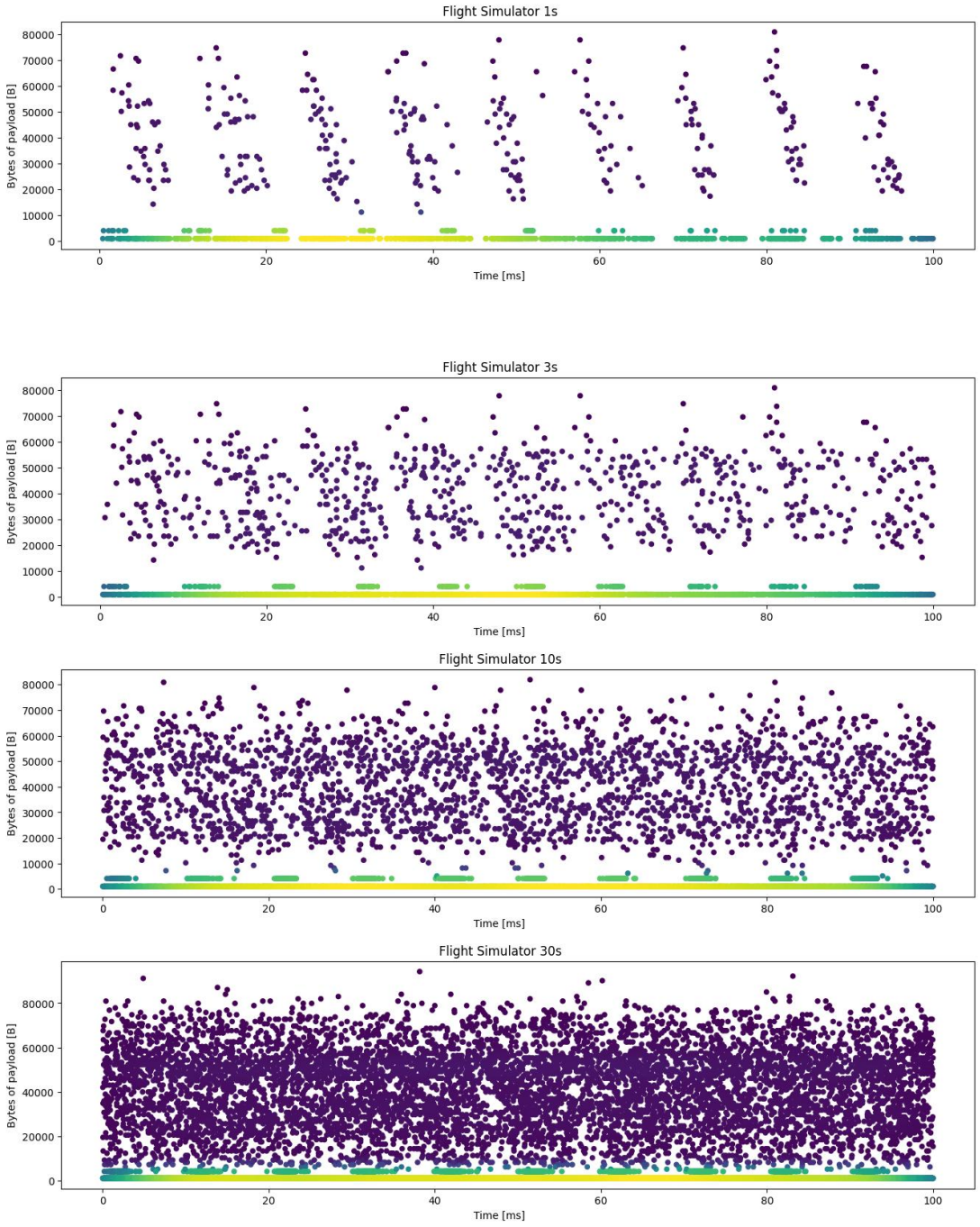


Figura 4.8: Traccia di Flight Simulator originale

Anche anche per questa applicazione abbiamo lo stesso risultato visto per Beat Saber. Perciò abbiamo utilizzato il metodo sopra citato per allineare tutti gli intervalli e di seguito verrà mostrato il grafico con i valori traslati.

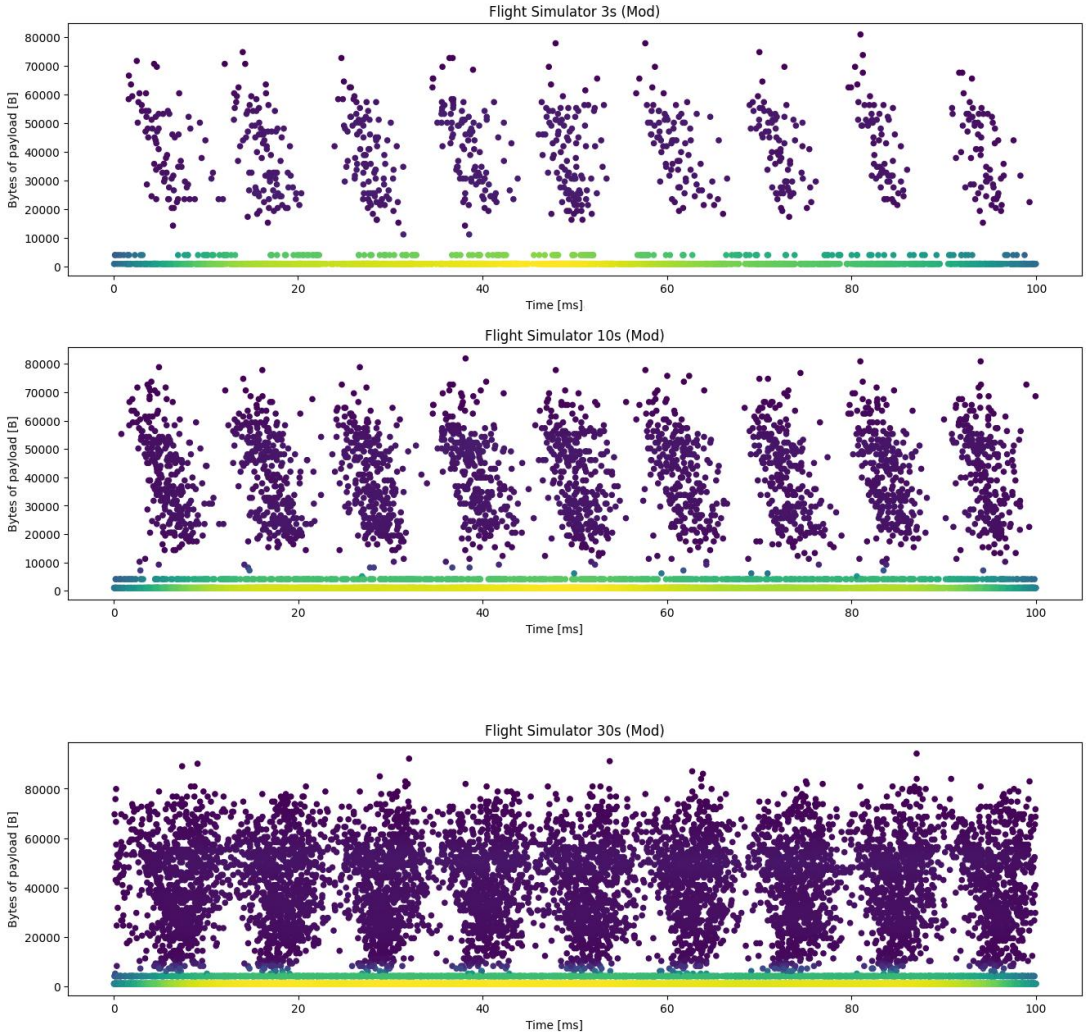


Figura 4.9: Traccia di Flight Simulator modificata

Notiamo che contrariamente ai dati risultanti da Beat Saber, nel grafico di Flight simulator è possibile notare che i dati di dimensioni maggiori sono più sparsi lungo l'asse Y che rappresenta il numero di byte, ciò significa che la dimensione dei dati non è sempre simile, ma invia dati di dimensioni variabile. Nella sezione riguardante la lunghezza dei pacchetti andremo ad approfondire questa analisi.

Medal of Honor

L'applicazione è stata utilizzata come per le altre, con 90 Hz di refresh rate, quindi anche per questa ci aspettiamo gli stessi risultati.

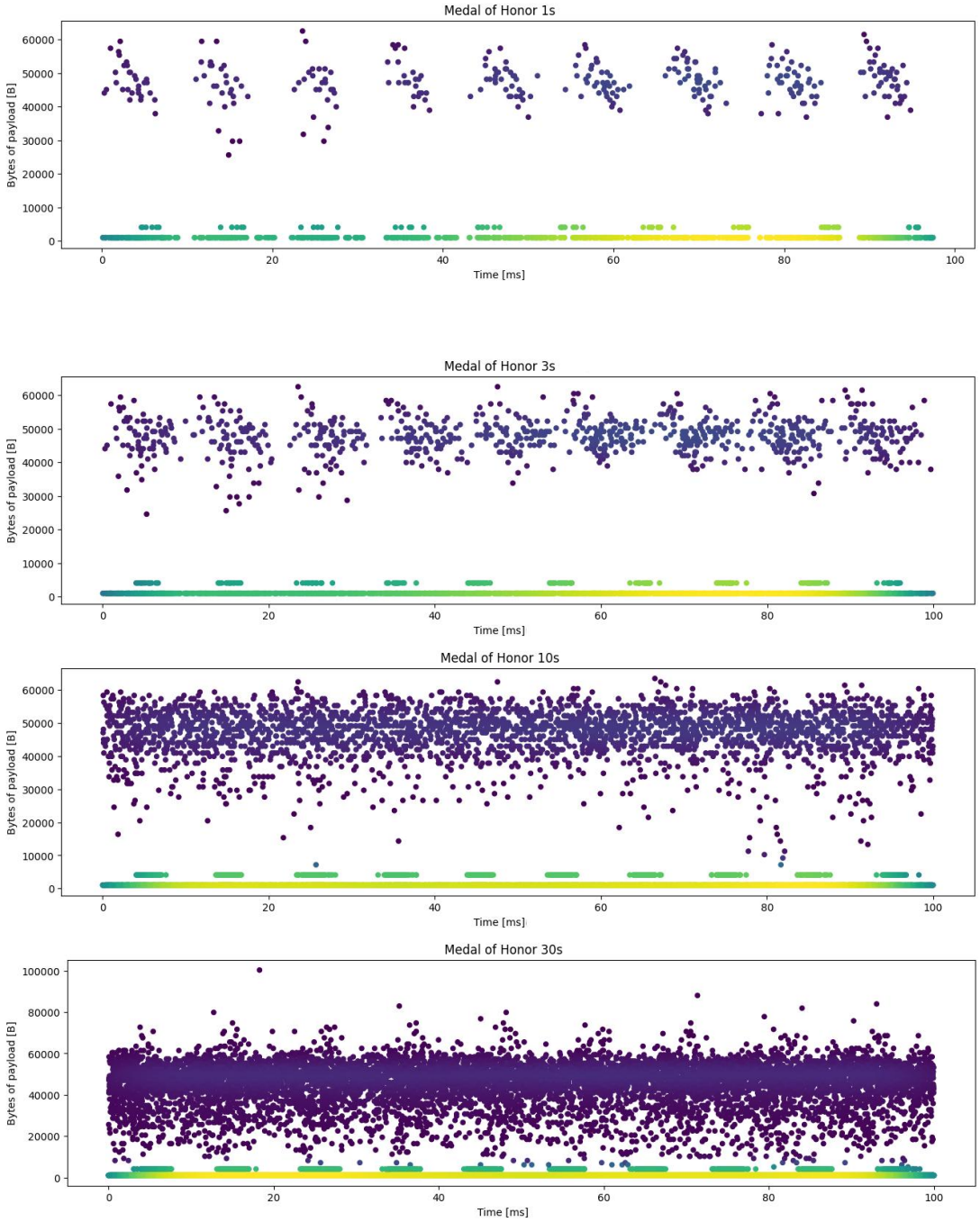


Figura 4.10: Traccia di Medal of Honor originale

Anche anche per questa traccia abbiamo utilizzato il metodo sopra citato per allineare tutti gli intervalli.

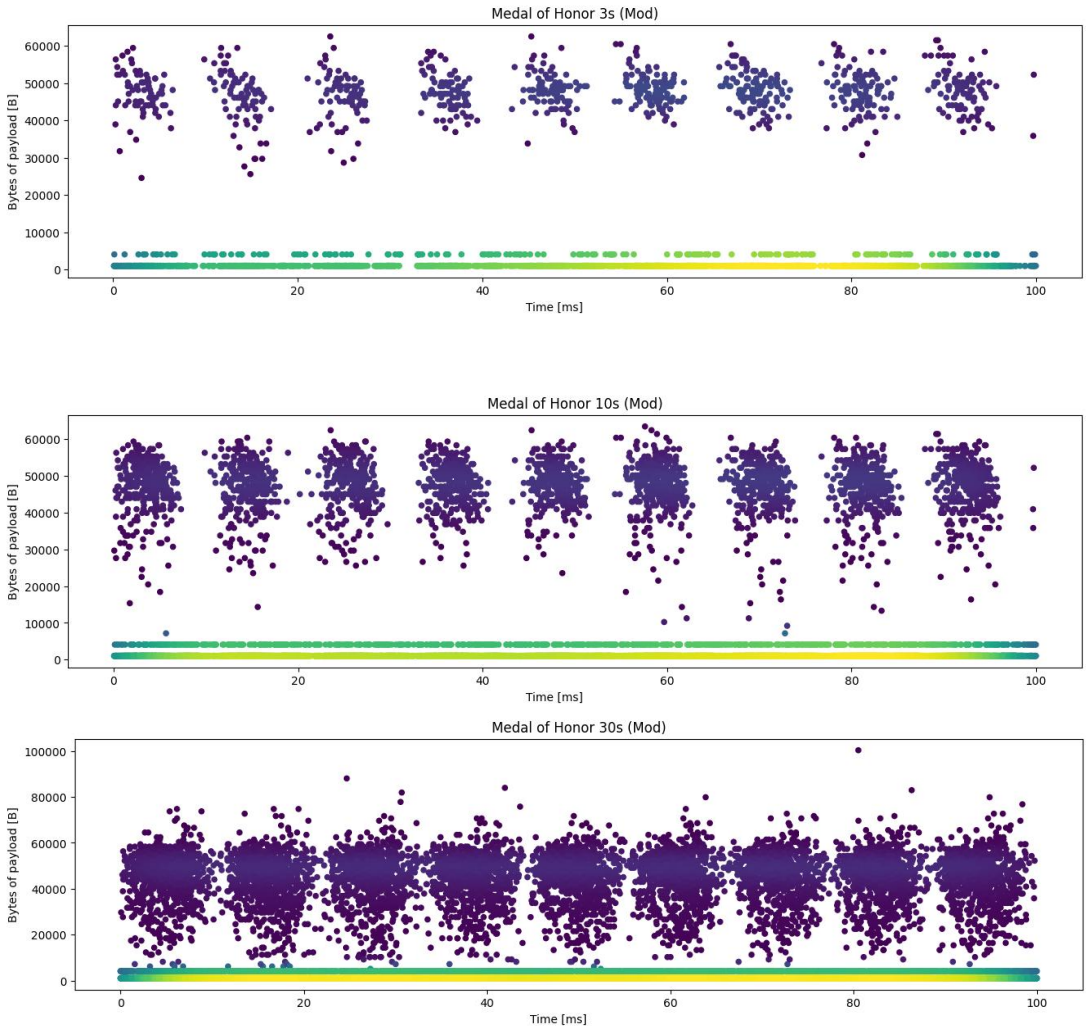


Figura 4.11: Traccia di Medal of Honor modificata

Come per le altre applicazioni notiamo che la maggior parte dei pacchetti inviati sono di piccola dimensione e senza una frequenza stabilita, invece i pacchetti che contengono grandi dati risultano seguire il frame rate utilizzato per mantenere il refresh rate di 90 Hz.
Inoltre, i pacchetti di grandi dimensioni risultano poco sparsi come per i pacchetti provenienti da Beat Saber.

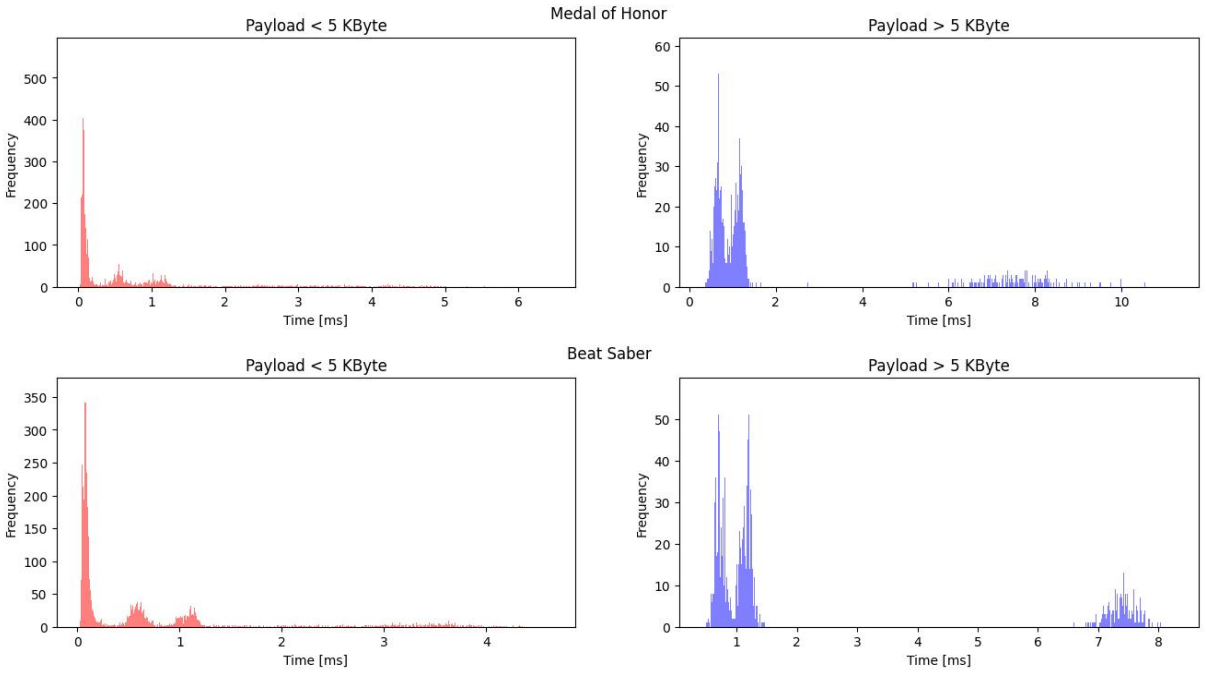
Considerazioni generali

Dai grafici notiamo che in tutte le applicazioni è presente un ritardo di circa 2 ms tra intervalli di 1 secondo, il quale disallinea gli istanti di invio dei pacchetti. Il fatto che sia presente in tutte la catture ci indica che non è un fattore dipendente dall'applicazione utilizzata. Notiamo però che questo ritardo potrebbe derivare da diversi fattori come la registrazione delle tracce USB utilizzando il software esterno USBPcap, oppure per scelte implementative del software di Oculus.

Molto probabilmente la causa di questo ritardo risiede nella necessità del PC di inviare altri tipi di dati, differenti da immagini, continuativamente senza limiti temporali, rallentando il buffer e di conseguenza impedendo al PC l'invio dei frame con perfetta frequenza. Questo comporta la diminuzione del frame rate di poco e quindi esso risulta in ritardo di qualche millesimo di secondo. Questo ritardo comunque non risulta compromettere la fluidità dal dispositivo, infatti un ritardo di 2 ms è il risultato di un frame rate abbassato a 89,82 Hz cioè solo 0.2% più lento del previsto.
Oltre a questo gruppo di pacchetti notiamo che sono presenti molti pacchetti di dimensione inferiore che non mostrano alcun tipo di frequenza. Questi pacchetti dovrebbero contenere tutti quei dati di dimensione minore, come l'Audio, o che non necessitano di costanza, ma dipendono dall'utente come dati di controllo, dati di stato e output Tattile [22].

4.1.3 Tempo inter pacchetto

Di seguito vengono riportati gli istogrammi relativi al tempo che intercorre tra i frame generati dal PC, quindi i pacchetti con direzione "Bulk out" e sorgente "host", differenziando tra frame di dimensioni superiore a 5 kB (i punti scuri nei grafici della distribuzione temporale) con quelli di dimensione inferiore. Prendiamo solo i pacchetti con sorgente "host" poiché i pacchetti inviati dal visore durante il trasferimento "Bulk" con direzione "out" sono solamente messaggi di ACK con payload nullo.
Per ogni applicazione abbiamo preso un intervallo di 30 s che permette di visualizzare i dati correttamente ed essere abbastanza ampio per rappresentare una statistica veritiera.



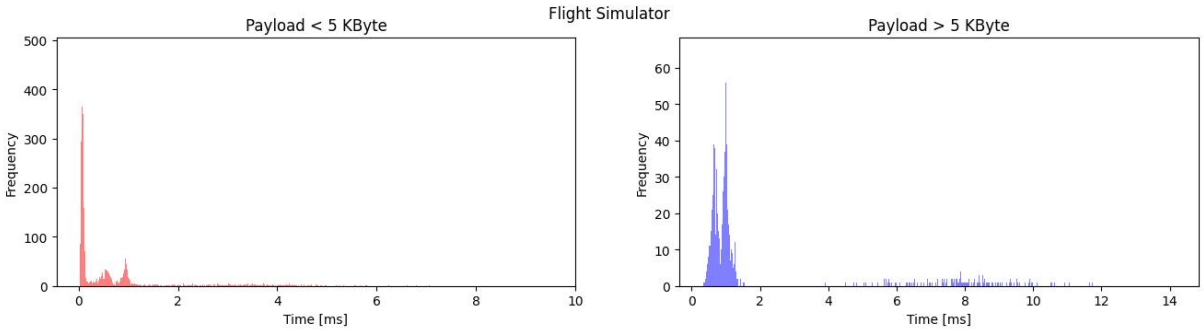


Figura 4.12: Tempo inter pacchetto per i dati generati dalle applicazioni.

Dagli istogrammi in figura 4.12 si nota che i pacchetti che contengono meno di 5 kB per la maggior parte vengono inviati con un intervallo minore di 250 μs e quasi la totalità di essi non viene inviato con intervalli che superano gli 1.2 ms.
Per i pacchetti che contengono più di 5 kB possiamo osservare che la maggior parte viene inviata in un intervallo di 1 ms e una parte minore in un intervallo che va dai 7 ai 10 ms. Questo a prima vista risulta in contrapposizione con le analisi fatte nella sezione precedente, da cui erroneamente ci aspettiamo che siano inviati 9 pacchetti in 100 ms con un intervallo tra loro di 11.1 ms. Nel paragrafo seguente vedremo che questi ultimi risultati in realtà sono coerenti con le ipotesi fatte precedentemente.

Pacchetti con payload maggiore di 5 kB

Nel grafico sottostante vengono rappresentati gli istanti temporali in cui vengono inviati i pacchetti di grandi dimensioni prendendo una piccola finestra di 50 ms. In questo caso abbiamo raffigurato i dati generati da Beat Saber ma potremmo avere lo stesso risultato anche con le altre applicazioni e nel paragrafo sottostante verranno presentati.

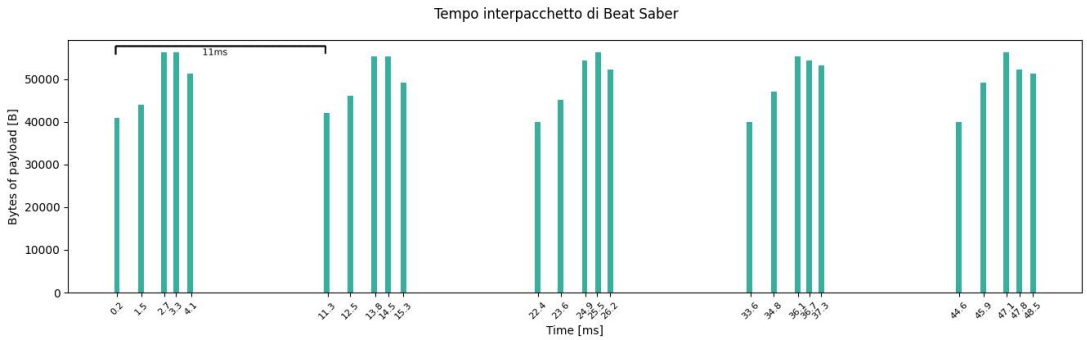


Figura 4.13: Zoom su una finestra di invio dati di Beat Saber

Dal grafico in figura 4.13 è possibile notare che effettivamente con intervalli di 11 ms viene inviato un piccolo numero di pacchetti di grandi dimensioni. Questo infatti è coerente con la statistica vista precedentemente in relazione all'utilizzo di un refresh rate di 90 Hz. La presenza di un gruppo di più pacchetti sicuramente dovuta alla dimensione elevata dei dati multimediali da inviare, che costringe la suddivisione di essi in più pacchetti. Questo gruppo è generalmente

composto da non più di 5 pacchetti, inviati in un intervallo massimo di 4 ms. Quindi nei grafici in figura 4.12 il tempo inter pacchetto per questi pacchetti di grandi dimensioni risulta per la maggior parte 1 ms, e in minor parte 7 o 8 ms. Questo perché, come appena visto, solitamente vengono inviati in sequenza 5 pacchetti con un tempo di 1 ms tra loro e dopo di che si dovrà attendete il tempo restate per raggiungere gli 11 ms e inviare il prossimo gruppo, per questo il primo pacchetto del nuovo gruppo registrerà un tempo superiore.

Tutti i pacchetti

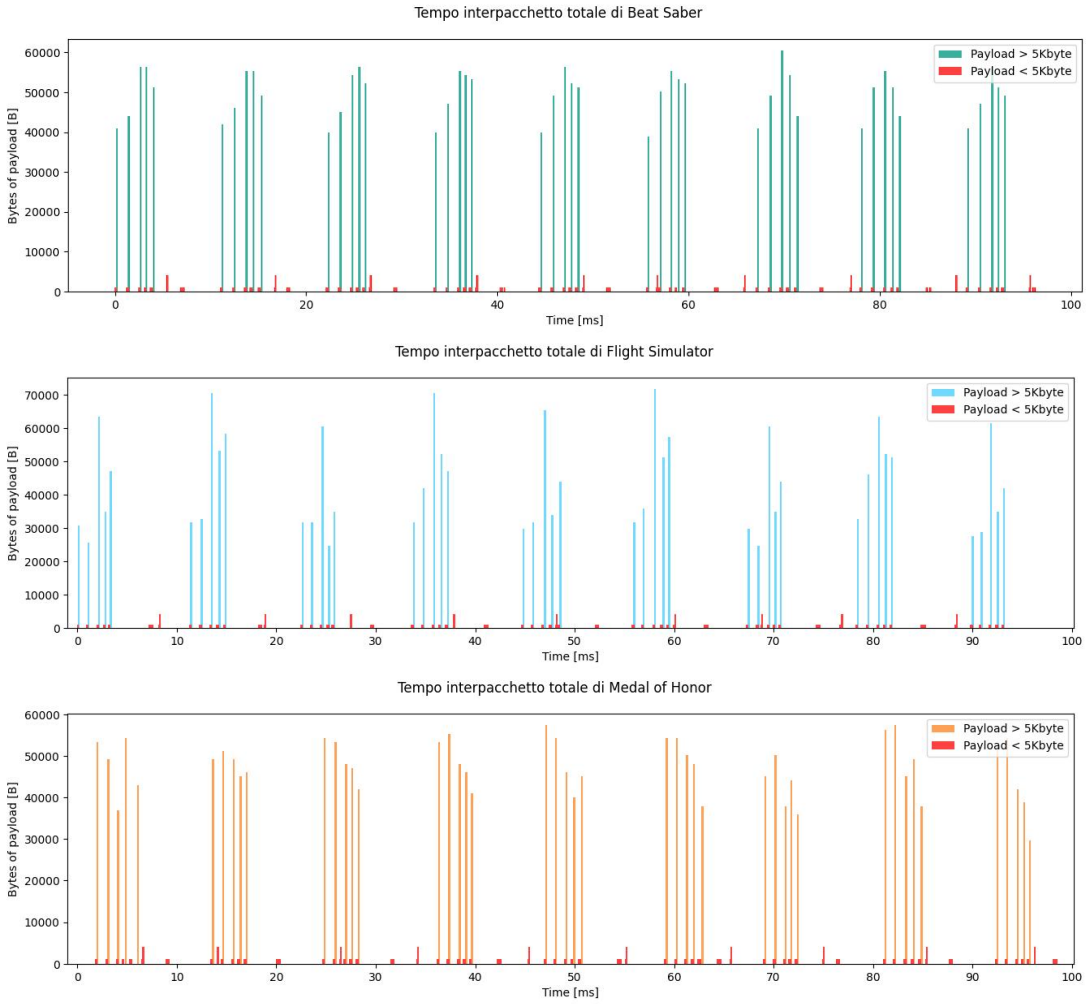


Figura 4.14: Tempo inter pacchetto su tutti i dati inviati dalle applicazioni

Nel grafico in figura 4.14 sono rappresentati i tempi inter pacchetto, di tutte le applicazioni, nella la finestra di tempo presa in precedenza e aumentata a 100 ms per renderla più esplicativa, in cui compaiono anche i pacchetti di dimensione inferiore a 5 kB. Dal grafico notiamo che per questi pacchetti di dimensione minore non è possibile identificare un pattern di invio e non potendo determinare il contenuto dei dati non è stato possibile fare ulteriori analisi.

4.1.4 Lunghezza pacchetti

Per ogni applicazione prenderemo un intervallo di 2 minuti e andremo a graficare, con un istogramma, la frequenza della dimensione del payload dei pacchetti da esse generati. Per permettere una migliore visione dei dati abbiamo diviso il grafico in 3 sezioni.

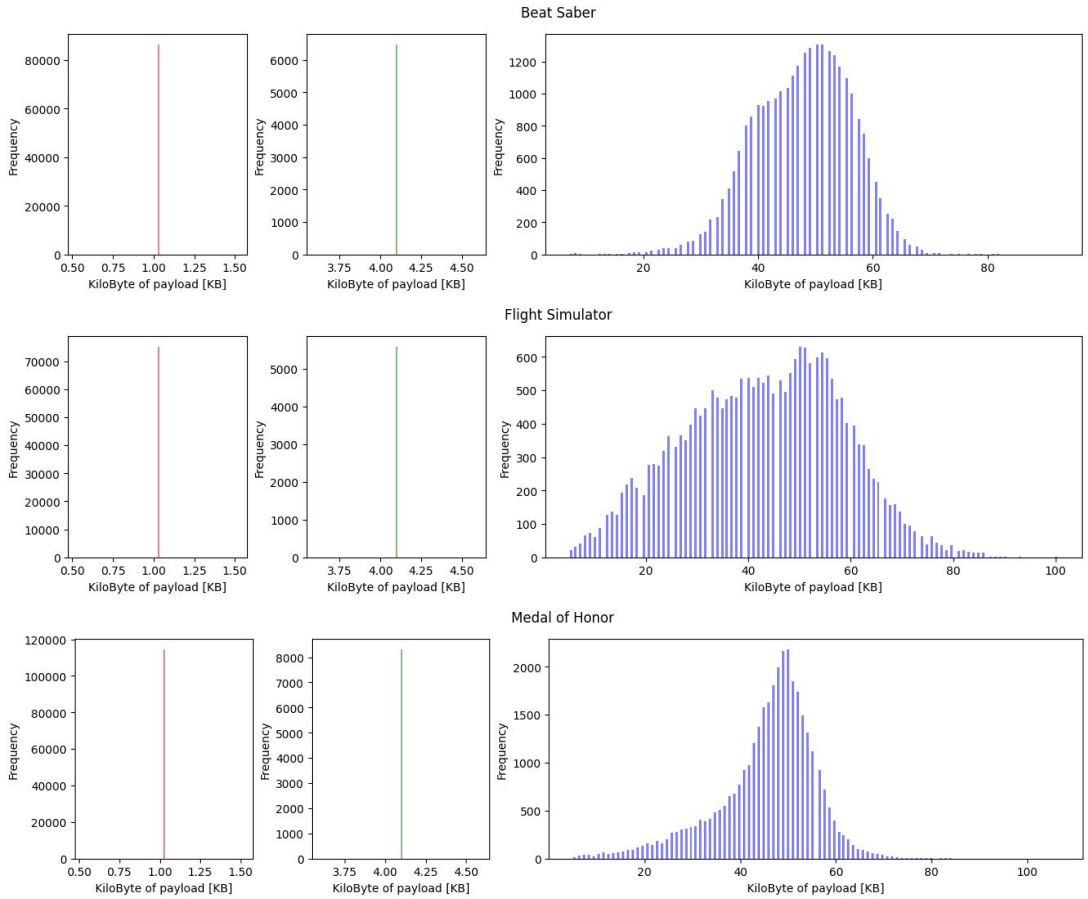


Figura 4.15: Lunghezza pacchetti “out” delle applicazioni

Dai grafici in figura 4.15 possiamo notare che il pacchetto con la frequenza di invio maggiore (nel grafico di sinistra in rosso) ha dimensione del payload di 1024 B. Possiamo notare inoltre che questo pacchetto ha una frequenza simile per le applicazioni Flight Simulator e Beat Saber, invece differisce in Medal of Honor in cui è poco più frequente. Differenze più evidenti si possono notare per quanto riguarda i pacchetti con payload di grandi dimensioni, infatti, possiamo notare che in Flight Simulator questi pacchetti hanno dati di dimensione molto variabile, superando anche gli 80 kB. Per le applicazioni Beat Saber e Medal of Honor, invece, questi pacchetti tendono ad avere un payload poco variabile, con una dimensione che si discosta di poco dai 50 kB.

Tabella 4.2: Flight Simulator

Byte di Payload	Numero di pacchetti		Quantità di dati [MB]	
1024	75202	***72.16 %***	77.0	***6.95 %***
4096	5583	***5.35 %***	22.8	***2.06 %***
Other	23420	***22.47 %***	1006.59	***90.97 %***
Totale:	104205		1106.47 (1.1 GB)	

Tabella 4.3: Medal of Honor

Byte di Payload	Numero di pacchetti		Quantità di dati [MB]	
1024	114457	***72.20 %***	117.2	***6.58 %***
4096	8308	***5.24 %***	34.0	***1.91 %***
Other	35758	***22.55 %***	1629.6	***91.5 %***
Totale:	158523		1780.8 (1.7 GB)	

Tabella 4.4: Beat Saber

Byte di Payload	Numero di pacchetti		Quantità di dati [MB]	
1024	86402	***71.66 %***	88.47	***6.15 %***
4096	6472	***5.37 %***	26.5	***1.85 %***
Other	27702	***22.97 %***	1324	***92.0 %***
Totale:	120576		1439 (1.4 GB)	

Nelle tabelle 4.2, 4.3, 4.4 abbiamo confrontato i tre gruppi di pacchetti divisi per dimensione di payload, mettendo in relazione la frequenza con cui questi pacchetti vengono inviati con la quantità di dati totale che trasmettono. Per ogni applicazione abbiamo preso 2 minuti di traccia e abbiamo riportato nella tabella il numero totale di pacchetti inviati e il totale di dati generati, poi, per ogni gruppo abbiamo riportato il numero di pacchetti e la quantità di dati con la percentuale in relazione al totale.

Da queste tabelle si può notare che i valori delle percentuali risultanti sono pressoché identici per tutte le applicazioni. Infatti, notiamo che a fronte di un numero di pacchetti maggiore, 55% in più in Medal of Honor rispetto a Flight Simulator, le percentuali rimangono comunque invariate.

Notiamo quindi che i pacchetti con payload di 1024 B sono i più frequenti, circa 72% del totale,

ma forniscono al massimo il 7% dei dati totali. I pacchetti che possiedono un payload di 4096 B risultano, in tutte le tracce, poco frequenti e con pochi dati trasmessi, meno del 6% dei pacchetti e il 2% dei dati trasmessi. Invece, i pacchetti rimanenti, con payload maggiore di 4096 B, sono meno frequenti, circa il 23%, ma trasportano fino al 92% dei dati.

4.2 IN Direction

In questa sezione andremo ad analizzare il traffico con direzione "IN". Per ottenere il traffico di questa sezione è sufficiente controllare che la direzione del campo USB_Info risulti "IN", inoltre per le analisi come data rate e tempo inter pacchetto sono stati eliminati dal traffico, come per la sezione OUT, i pacchetti di risposta di ricezione avvenuta (ACK) [15].

4.2.1 Distribuzione temporale

In questa sezione parleremo del modo in cui i dati vengono inviati nell'arco temporale, in particolare osserveremo la frequenza con cui il visore invia i dati al PC. Per ogni applicazione prenderemo una finestra di 1 s e andremo a selezionare solo i pacchetti che hanno direzione in ingresso, controllando che il campo usb.irp_info.direction valga '1', senza tener conto dei messaggi di ACK. Di questi pacchetti andremo a registrare l'istante in cui vengono inviati e successivamente andremo a graficare questi istanti in una finestra di 100 ms.

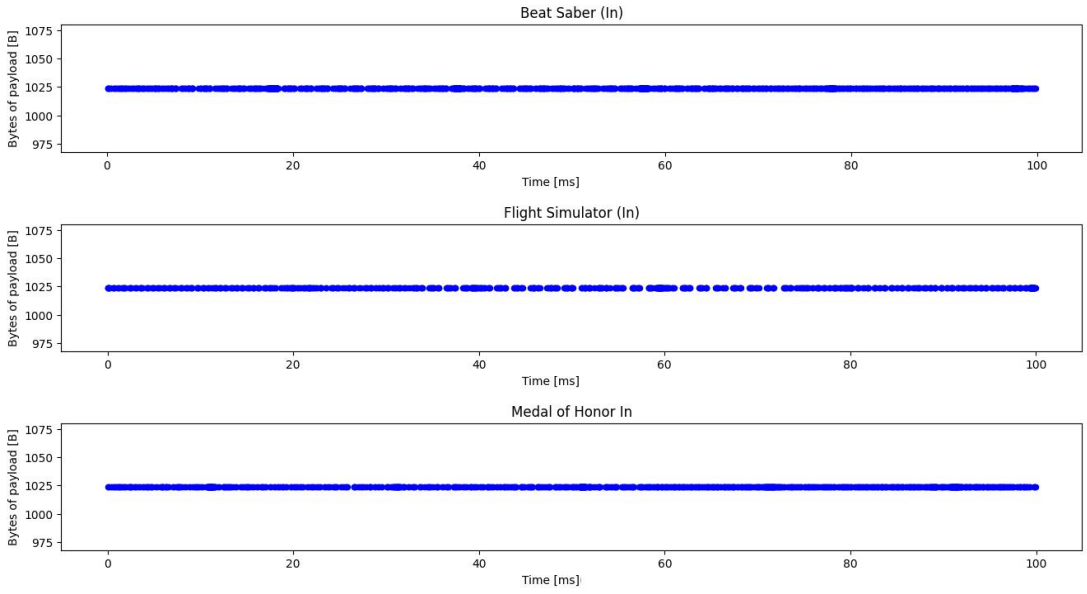


Figura 4.16: Grafici del data rate in ingresso

Dai grafici in figura 4.16 possiamo notare che la frequenza d'invio di questi pacchetti nei 100 ms non è costante. Questo potrebbe significare che questi pacchetti non hanno una frequenza

d'invio prestabilita, al contrario di alcuni pacchetti in uscita, ma vengano inviati all'occorrenza. Inoltre, possiamo notare che hanno una dimensione del payload sempre uguale. Approfondiremo questo aspetto in seguito.

4.2.2 Tempo inter pacchetto

Di seguito vengono riportati gli istogrammi relativi al tempo che intercorre tra i frame con direzione in ingresso. Per ogni applicazione abbiamo preso un intervallo di 30 secondi.

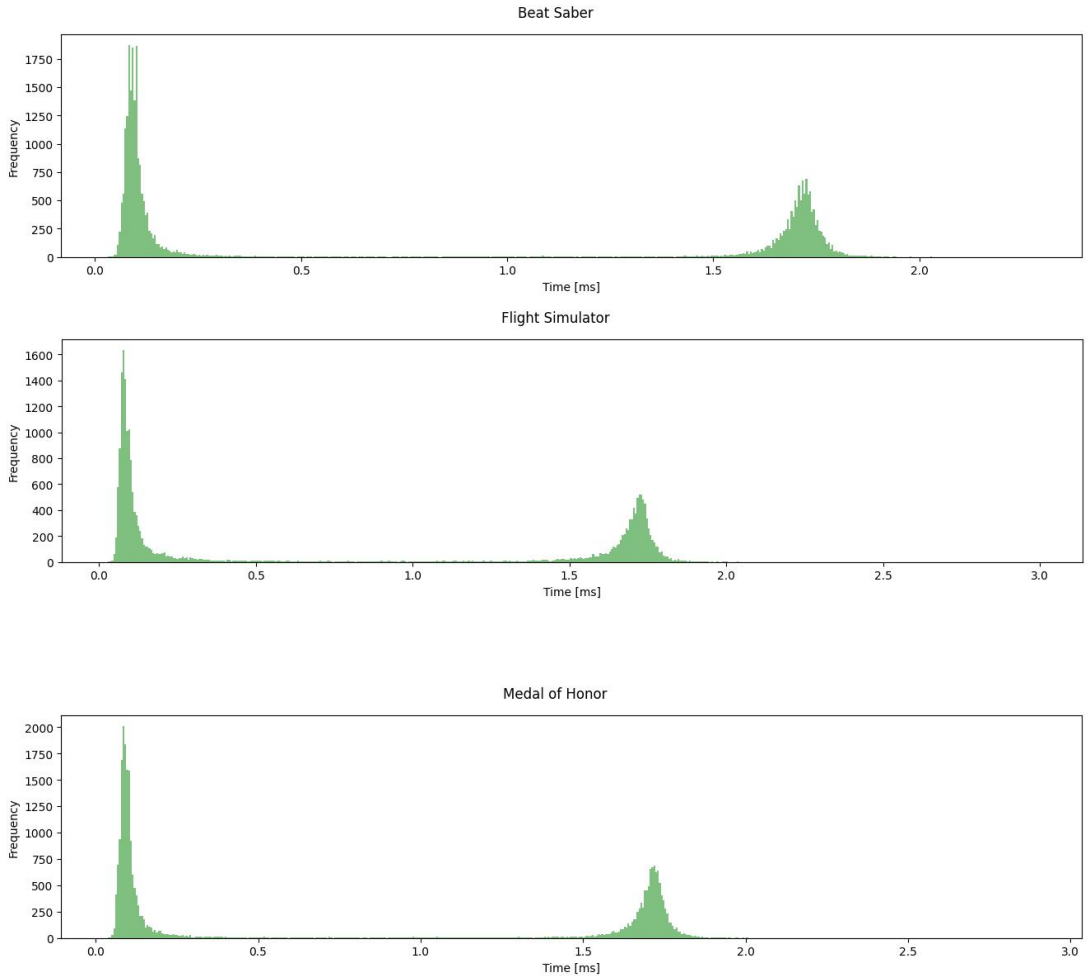


Figura 4.17: Grafici del tempo inter pacchetto

Dagli istogrammi in figura 4.17 si nota che il tempo inter pacchetto ha un valore che si aggira principalmente intorno a 0.1 ms e in minor parte intorno a 1.7 ms, per tutte le applicazioni. Inoltre, anche se dai grafici è difficile da intravedere, ci sono molti pacchetti che hanno un tempo variabile nell'intervallo tra 0.1 ms e 2 ms.

4.2.3 Lunghezza pacchetti

Per ogni applicazione andremo a graficare con un istogramma la frequenza della dimensione del payload dei pacchetti generati dal visore.

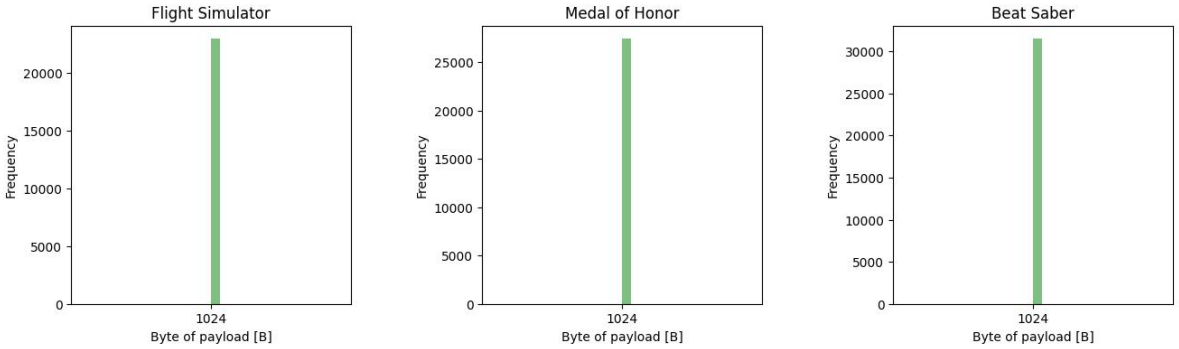


Figura 4.18: Istogrammi della lunghezza del payload

Da questi grafici notiamo che i dati inviati dal visore, per ogni applicazione, hanno una dimensione sempre pari a 1024 B. Questo tuttavia potrebbe non rappresentare sempre la giusta quantità di dati inviata, in quanto, questo è solamente il valore della grandezza del payload dichiarata nell'intestazione dei pacchetti. Controllando all'interno di questi notiamo che alcuni di essi presentano solo una parte dei 1024 B con un valore effettivo e i restanti risultano essere byte nulli, come se si dovesse riempire la parte rimanente.

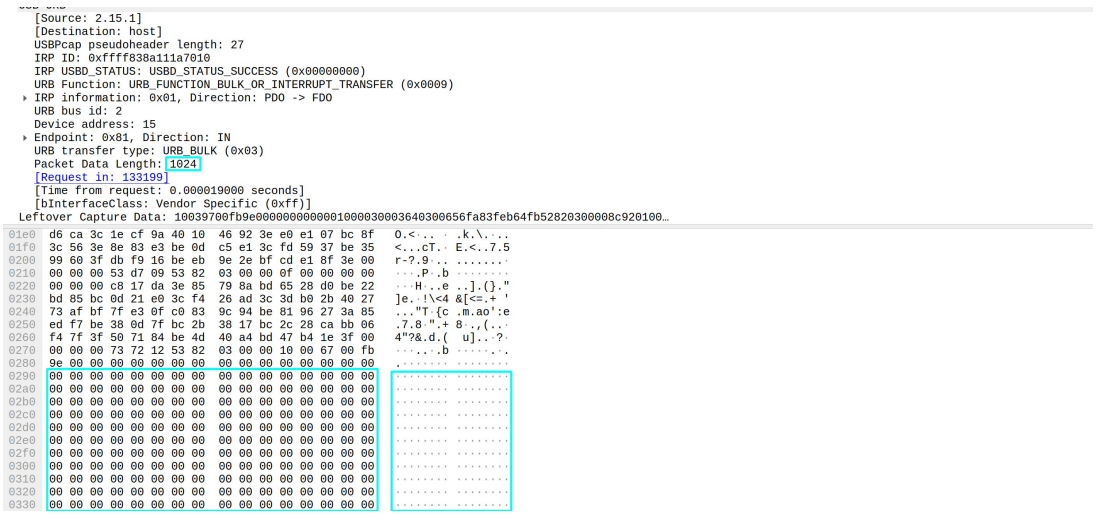


Figura 4.19: Schermata Wireshark

Nella figura 4.19 è raffigurato un esempio di quanto detto prima. Si può notare infatti che nel payload visualizzato alcuni byte hanno un valore definito, invece altri hanno valore nullo. Questo probabilmente potrebbe essere dato da uno dei due seguenti fattori: il primo derivante dal processo di registrazione dati con USBPcap, il secondo riguarda l'utilizzo del trasferimento USB, che potrebbe richiedere una dimensione minima del payload.

4.3 Oculus Monitor

In questa sezione andremo ad analizzare il data rate delle applicazioni mettendolo in relazione al movimento della testa, registrato tramite Oculus Monitor. Per ogni applicazione andremo a mostrare il movimento della testa, graficando i valori di orientazione dei 4 assi gestiti dal visore (X, Y, Z, W) e poi il data rate dei dati inviati dal PC. Considerando che Oculus Monitor campiona i valori degli assi 60 volte al secondo, mentre USBPcap registra circa 400 frame al secondo, per visualizzare chiaramente i dati, abbiamo rappresentato la media dell'orientamento 3 volte al secondo e allo stesso modo il data rate è stato raggruppato in 3 intervalli al secondo.

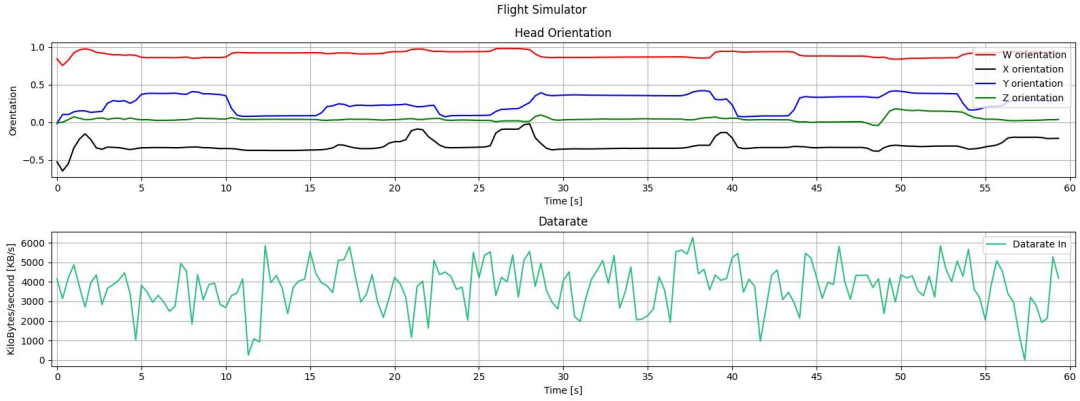


Figura 4.20: Movimento della testa e data rate di Flight Simulator

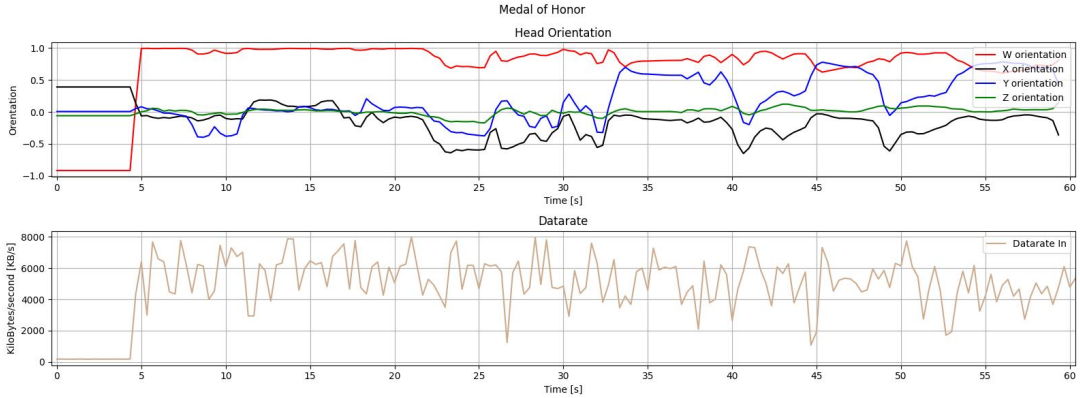


Figura 4.21: Movimento della testa e data rate di Medal of Honor

Dai grafici in figura 4.20 e 4.21 possiamo notare che il movimento della testa e l'andamento del data rate non sembrano relazionati. Tuttavia in figura 4.21 si può osservare che nei punti in cui l'orientazione della testa si stabilizza, ad esempio al secondo 35 o 46, il data rate tende a calare per qualche secondo. Per poter ottenere questi dati abbiamo riscontrato alcuni problemi, infatti, in Oculus Monitor non viene segnata l'ora locale, neanche all'inizio della registrazione, ma solo il tempo relativo dall'inizio della registrazione e questo crea dei problemi nel sincronizzare le due registrazioni.

Capitolo 5

Conclusioni

Gli studi effettuati hanno consentito di ottenere alcune informazioni utili riguardo la trasmissione dati che avviene tra il computer e il visore Oculus Quest 2.
Dalle analisi riguardanti i dati trasmessi dal PC abbiamo riscontrato, che, anche avendo utilizzato tutte le applicazioni con la stessa qualità grafica e la stessa frequenza di aggiornamento dello schermo (90 Hz), la differenza tra il data rate di diverse applicazioni raggiunge il 40%. Le cause di questa differenza molto probabilmente sono: il tipo di ambientazione presente in esse e la necessità di muovere maggiormente la testa durante l'utilizzo. Un'ambientazione dinamica, anche se con meno dettagli come nel caso di Beat Saber, richiede sicuramente un maggior numero di dati da inviare rispetto ad una statica. Allo stesso modo, anche considerando un'ambientazione statica, maggiori movimenti dalla testa risultano aumentare i dati trasmessi, come si riscontra analizzando le tracce delle applicazioni Flight Simulator e Medal of Honor (o Alix). Il primo, nel quale mediamente si muove poco la testa, risulta avere un data rate medio inferiore al secondo, che, essendo uno "sparatutto", richiede un maggiore movimento della testa.
Un altro aspetto che abbiamo osservato nella comunicazione analizzata riguarda il lieve ritardo riscontrato nell'invio dei pacchetti di grandi dimensioni, i quali riteniamo contengano i dati grafici. Questo ritardo ammonta a circa 2 ms per ogni secondo di trasmissione, il quale è il risultato di un invio ritardato di 0.022 ms dei gruppi di pacchetti, descritti nel capitolo 4. Esso comporta una minima diminuzione del frame rate, che dal valore ideale di 90 Hz risulta abbassato a 89,82 Hz, cioè 0.2% in meno. Questo evento è stato osservato in tutte le applicazioni analizzate, quindi, potrebbe essere il risultato di alcuni meccanismi di trasmissione utilizzati da Oculus ed essere indipendente dal tipo di applicazione usata.
Un altro aspetto che è stato riscontrato durante le analisi in tutte le applicazioni, riguarda la lunghezza dei pacchetti e la quantità di dati che essi trasmettono. Abbiamo visto che i pacchetti con payload piccolo, inferiore a 4100 B, risultano i più inviati, circa il 75% dei pacchetti, ma trasportano solo il 10% dei dati totali. Invece, i pacchetti con il payload più grande di 4100 B sono meno inviati, circa 25% dei pacchetti totali, ma trasportano il 90% di tutti i dati trasmessi.
Durante le analisi riguardanti il traffico in entrata, cioè i dati generati dal visore, non è stato possibile ottenere tutte le informazioni cercate. Questo a causa di alcune discrepanze riscontrate nei dati, infatti, in questi pacchetti sono contenuti sempre 1024 B di payload ma, analizzato

pacchetto per pacchetto, abbiamo notato che alcuni di questi byte hanno realmente un valore e altri hanno valore nullo.
A causa dei limiti del setup e delle poche informazioni pubbliche possedute riguardo il visore Oculus, non è stato possibile approfondire le analisi. In futuro andando a migliorare il setup, utilizzando altri software per la registrazione USB o andando a migliorare il software USBPcap che noi abbiamo utilizzato, e riuscendo a ottenere più informazioni riguardanti la formattazione dei dati utilizzata da Oculus, si potranno approfondire ulteriormente le analisi e si potrà dividere il traffico in base al tipo di dato inviato.
Inoltre, trovando un modo di sincronizzare perfettamente la registrazione eseguita con Oculus Monitor e la registrazione dei dati USB, si potranno ottenere maggiori informazioni riguardo la relazione che intercorre fra essi.
Infine, in futuro sarebbe interessante ripetere le analisi svolte andando a variare anche gli aspetti grafici delle applicazioni, come refresh rate o la risoluzione grafica, oppure confrontando le prestazioni che si ottengono usando l'ausilio del PC o il solo Visore senza ausilio di esso.

Ringraziamenti

Volevo ringraziare il professor Zanella Andrea che mi ha permesso di partecipare a questo progetto e di svolgere questa tesi. Inoltre volevo ringraziare i dottorandi: Matteo, Alessandro, Federico e Paolo che mi hanno seguito sia durante l'acquisizione dati sia durante la stesura della tesi.

Bibliografia

[1] Ionos. ≪Realtà estesa: cos'è la XR?≫ (2020), indirizzo: https://www.ionos.it/digitalguide/online-marketing/vendere-online/realta-estesa-xr/ (visitato il 22/08/2022).

[2] C. Isi, *Baffling history of virtual reality headsets: From sensorama to sega*. mysmartprice.com, 2019.

[3] Wikipedia. ≪The Sword of Damocles (virtual reality).≫ (), indirizzo: https://en.wikipedia.org/wiki/The_Sword_of_Damocles_(virtual_reality) (visitato il 25/08/2022).

[4] Nasa. ≪The Virtual Interface Environment Workstation (VIEW), 1990.≫ (), indirizzo: https://www.nasa.gov/ames/spinoff/new_continent_of_ideas/ (visitato il 25/08/2022).

[5] S. M. Drive. ≪SEGA VR (prototipo).≫ (), indirizzo: https://www.segamegadrive.it/sega-vr-prototipo/ (visitato il 25/08/2022).

[6] Wikipedia. ≪Virtual Boy.≫ (), indirizzo: https://it.wikipedia.org/wiki/Virtual_Boy (visitato il 25/08/2022).

[7] L. Grossman, *facebook-oculus-vr-inside-story*. time.com, 2014.

[8] U. I. Forum, *fUSB Specification*. www.usb.org, 2021.

[9] wikipedia. ≪USB.≫ (), indirizzo: https://it.wikipedia.org/wiki/USB (visitato il 22/08/2022).

[10] F. T. D. I. Ltd, *USB Data Packet Structure*. ftdichip.com, 2009.

[11] A. Singh, *Types of USB Packets and USB Transfers*. www.engineersgarage.com.

[12] M. Inc. ≪Meta link cable.≫ (2021), indirizzo: https://store.facebook.com/it/quest/accessories/quest-2/link-cable/ (visitato il 09/08/2022).

[13] CompTIA. ≪Wireshark.≫ (), indirizzo: https://www.comptia.org/content/articles/what-is-wireshark-and-how-to-use-it (visitato il 10/08/2022).

[14] T. Moń. ≪USBPcap - USB Packet capture for Windows.≫ (), indirizzo: https://desowin.org/usbpcap/ (visitato il 10/08/2022).

[15] kojack. ≪Oculus Monitor.≫ (2020), indirizzo: https://github.com/rajetic/OculusMonitor (visitato il 11/08/2022).

[16] ——, «Oculus Community - Oculus Monitor.» (2018), indirizzo: https://forums.oculusvr.com/t5/Games-and-Apps/Oculus-Monitor-v0-2-2-27-Mar-20/td-p/708659 (visitato il 11/08/2022).

[17] B. Games. «Steam - Beat Saber.» (2019), indirizzo: https://store.steampowered.com/app/620980/Beat_Saber/ (visitato il 11/08/2022).

[18] V. Corporation. «Half-life, Alyx.» (2020), indirizzo: https://www.half-life.com/it/alyx/ (visitato il 16/08/2022).

[19] Meta. «Medal of Honor.» (2020), indirizzo: https://www.oculus.com/medal-of-honor/ (visitato il 16/08/2022).

[20] Microsoft. «Flight Simulator.» (2020), indirizzo: https://www.flightsimulator.com/ (visitato il 16/08/2022).

[21] Wikipedia. «Acknowledgement.» (), indirizzo: https://en.wikipedia.org/wiki/Acknowledgement_(data_networks) (visitato il 23/08/2022).

[22] Oculus. «The Architecture, Pipeline and AADT Explained.» (2019), indirizzo: https://developer.oculus.com/blog/how-does-oculus-link-work-the-architecture-pipeline-and-aadt-explained/ (visitato il 20/08/2022).

[23] Wireshark. (), indirizzo: https://www.wireshark.org/.