UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Triennale in Ingegneria dell'Informazione

TESI DI LAUREA TRIENNALE

MODELLO DI TRAFFICO E ANALISI SPERIMENTALE DI TRACCE USB DI VISORI VR

Relatore:
Prof. Andrea Zanella

Laureando:
Leonardo Bellin

Anno Accademico 2021/2022

Data di Laurea 19/07/2022

Indice

Abstract

La realtà virtuale sta subendo un'enorme crescita in tutti i settori. Per questo motivo è importante avere dei solidi modelli di traffico generali, che siano successivamente d'appoggio a sviluppi futuri.
In questo elaborato verrà esposto il funzionamento del protocollo USB (Universal Serial Bus) e, in particolare, il suo funzionamento nel trasferimento di dati, in modo da poter comprendere al meglio l'analisi di traffico successiva. Verranno inizialmente confrontate tutte le sue versioni esistenti e, successivamente, verranno definite le specifiche del protocollo USB 3.0 che è il più diffuso, per poi passare all'attuale stato dell'arte con le tecnologie USB 3.2 SuperSpeed, Wireless USB ed infine accenni al futuro USB 4.0.
Nella seconda parte verrà presentata un'analisi di traffico derivata da tracce sperimentali acquisite da un visore VR, collegato tramite USB al computer. L'analisi ha riportato una certa costanza nella quantità di dati scambiati, con delle lievi variazioni per ogni specifica applicazione, evidenziando un possibile schema da seguire per gli sviluppi futuri. In questa analisi si cerca di dare delle informazioni di base sugli scambi di dati che sono necessari al visore, creando un modello di traffico generale che servirà come fondamento per lo studio e lo sviluppo futuro di possibili algoritmi di compressione predittivi e protocolli di trasmissione specifici per questa applicazione.

Capitolo 1

Introduzione

La realtà virtuale è un campo che è stato in crescita fin dal suo debutto, come dimostrato in [1], e che negli ultimi anni ha subito un boom anche grazie allo sviluppo e alla diffusione che tutti i mezzi di interazione e comunicazione da remoto hanno subito a causa della pandemia. Il settore VR è in continua crescita anche per quanto riguarda le applicazioni di questa tecnologia. La diffusione ha permeato tutti i principali settori: l'educazione [2], l'utilizzo militare [3], l'automobilistico [4], il turistico [5], lo sportivo [6], il social [7], il medico [8] [9] e infine, quello dell'intrattenimento [10].
Il settore videoludico in particolare è stato quello che ha subito la maggior crescita, ne è testimone la campagna di crowd-funding di Palmer Lucky su Kickstarter che nel 2012 riuscì a raccogliere quasi 2 milioni e mezzo di dollari [11]. L'azienda è stata da allora comprata da Meta, solamente 2 anni dopo per ben 2 miliardi di dollari.
I visori VR sono sempre stati abbastanza costosi, e questo ha ridotto notevolmente la loro diffusione. Ad aumentare questo effetto si aggiunge il fatto che giocare a un gioco VR richieda da notevoli capacità di calcolo hardware, e questo è sempre stato un fattore limitante, ma ultimamente la tecnologia nel settore ha fatto passi da gigante ed ha permesso che il prezzo dei visori calasse notevolmente.
Con la diffusione dei visori, sempre più aziende decidono di sviluppare le loro applicazioni in realtà virtuale, ma il fattore limitante rimane però la circoscritta capacità di calcolo dovuta alla necessaria limitatezza delle dimensioni del visore, che al suo interno dispone di tutto l'hardware che necessita. Alternativamente è possibile utilizzare le risorse di un computer esterno, ma in questo caso tutti i dati dovrebbero essere trasmessi al visore per essere mostrati. Quest'ultima soluzione è la più realizzabile e commercializzabile, specialmente per quanto riguarda il prezzo. È allora chiaro perché ci sia la necessità di creare un modello di traffico durante l'utilizzo di un visore, che in questo studio sarà il *Meta Quest 2*, uno tra i modelli di ultima generazione attualmente più diffusi.

Il progetto intende seguire e arricchire il lavoro svolto in [12, 13, 14], dove al posto di utilizzare uno smarthphone inserito in un adattatore da testa, questa volta si intende usare un visore VR. L'analisi che viene riportata di seguito rappresenta il primo passo verso un modello di traffico preciso e dettagliato, e perciò sarà più generale e andrà meno in profondità di quelle sopracitate.
Gli studi che attualmente esistono nella letteratura si concentrano principalmente sull'utilizzo del cloud gaming, che permette di risolvere i problemi hardware locali, come la limitata capacità di calcolo o il surriscaldamento, introducendo però il problema della latenza in quanto i dati dovrebbero essere trasmessi tramite internet. Riguardo questo ambito è interessante [15], in quanto utilizza Il Meta Quest 2, lo stesso visore che verrà utilizzato per lo

studio in questo progetto. Sempre riguardo al cloud gaming, è interessante [16], in quanto lo studio si focalizza sulla possibilità di utilizzare reti 5G a bassa latenza per ovviare al principale problema del servizio.
Altri studi esistenti si concentrano su analisi specifiche, ad esempio in [17], si presenta un esame riguardo all'accuratezza del tracciamento delle mani da parte di Meta Quest 2, in [18], si studia la differenza tra la distanza percepita e quella reale, rispetto a diversi visori, tra cui il Meta Quest 2. In [19] si esplora la possibilità di utilizzare un simulatore a realtà virtuale per la formazione di esperti chirurghi dentali. In [20] si dimostra che è possibile ottenere dati ottimi per la valutazione degli equilibri a partire da visori VR, invece che da più costosi dispositivi appositi.
Sebbene la letteratura sia piena di questi studi specifici, sembrerebbe essere scarna di modelli generici. Questo progetto ha lo scopo di creare un modello di traffico di base, che sia poi utilizzabile per la creazione di algoritmi di compressione predittivi, applicazioni o studi specifici.

La tecnologia VR, sebbene presenti benefici in numerosi campi, è ancora limitata da alcuni fattori fisici, quali ad esempio le limitate prestazioni dovute alle dimensioni e peso del visore, oppure la trasmissione di dati via cavo o wireless nel caso si voglia utilizzare l'hardware più potente di un computer.
La seguente analisi ha, quindi, anche lo scopo di facilitare e stimolare lo sviluppo di algoritmi e protocolli specifici per questa piattaforma in modo da ottimizzarne l'utilizzo e, in particolare, velocizzarne la trasmissione dei dati. Alternativamente, lo stesso risultato si potrebbe ottenere ottimizzando i protocolli di trasmissione USB attualmente in uso, che però presentano ancora il problema fisico del dover legare il visore tramite cavo.
Con questa particolare applicazione in mente e come scopo principale di questo progetto, nel capitolo 2 vengono presentate le specifiche del protocollo di trasmissione dati USB: a partire dal funzionamento di base, fino ad arrivare agli standard più veloci e recenti 3.0, 3.1, 3.2. Successivamente si parlerà di qualche spunto di miglioramento specifico per il nostro utilizzo preso dalla letteratura, come protocolli USB wireless e USB 4.0.

Capitolo 2

Protocollo USB

Il seguente capitolo ha lo scopo di descrivere il funzionamento di base del protocollo di trasmissione USB, per quanto riguarda le versioni 2.0 e superiori. Queste sono definite negli standard IEC 62680-2-1:2015 e IEC 62680-3-1:2017, mentre per questa sezione si fanno riferimento soprattutto ai rapporti implementativi [21, 22, 23, 24, 25, 26, 27, 28].

2.1 Storia e Versioni

Lo standard USB nasce nella metà degli anni 90, quando alcune tra le principali società informatiche decisero di sviluppare uno standard per rendere più semplice il collegamento di dispositivi esterni ai PC sostituendo la moltitudine di connettori sul retro dei PC, affrontando i problemi di usabilità delle interfacce esistenti e semplificando la configurazione software di tutti i dispositivi collegati a USB, oltre a consentire una maggiore velocità dati per dispositivi esterni.

Dagli anni 90 ad oggi sono state rilasciate diverse versioni del protocollo, che ne migliorano notevolmente la velocità di trasmissione dati.
Di seguito viene riportata la tabella delle principali versioni rilasciate nel corso degli anni, insieme alle velocità teoriche di trasferimento, e quelle reali. Si noti che la differenza tra queste due velocità è dovuta da molteplici fattori: alcuni tra i più influenti sono gli overhead e il controllo di flusso, che inevitabilmente occupano banda senza realmente trasmettere dati utili all'utente.

Nome	Versione	Velocità teorica	Velocità reale	Data di pubblicazione
Low-Speed	USB 1.0	1,5 Mb/s (187,5 KB/s)	1 Mb/s (125 KB/s)	Gennaio 1996
Full-Speed	USB 1.1	12 Mb/s (1,5 MB/s)	7 Mb/s (875 KB/s)	Agosto 1998
Hi-Speed	USB 2.0 [21]	480 Mb/s (60 MB/s)	280 Mb/s (35 MB/s)	Aprile 2000
SuperSpeed USB	USB 3.0 [22]	5 Gb/s (625 MB/s)	3,2 Gb/s (400 MB/s)	Settembre 2008
SuperSpeed USB 10Gbps	USB 3.1 [24]	10 Gb/s (1,25 GB/s)	7,2 Gb/s (900 MB/s)	Luglio 2013
SuperSpeed USB 20Gbps	USB 3.2 [24]	20 Gb/s (2,5 GB/s)	N/A	Settembre 2017

Tabella 2.1: Lista delle attuali versioni esistenti del protocollo USB e caratteristiche di trasferimento

È notevole il fatto che l'USB di tipo A e B può collegare periferiche che richiedono trasferimento dati, ma non può trasportare segnali video, tale limite è stato comunque superato con il nuovo formato USB Type-C, per questo, per i componenti multimediali, ormai lo standard USB è il metodo di collegamento più utilizzato.

Altro punto interessante da analizzare è la latenza delle diverse versioni [29]:

- la latenza dell'USB 1.0 è di circa 1ms;
- la latenza dell'USB 2.0 è di circa 125 µs;
- la latenza dell'USB 3.0 è di circa 950 ns.

2.2 Architettura

L'USB è molto facile da usare e fornisce un mezzo affidabile ed efficace per il trasferimento dei dati. Che sia USB 1, USB 2, USB 3 o anche USB 4, i dati richiedono un metodo standardizzato di trasferimento sull'interfaccia USB insieme a un formato standard per i pacchetti di dati USB.

Per ottenere questo, si è definito il protocollo di trasferimento dati USB, il formato dei pacchetti dati, ecc., che permette ai dati di essere formattati e trasportati in un modo definito che fornisce una comunicazione affidabile. Il bus funziona anche in un modo particolare, vedendo i vari dispositivi attaccati ad esso indipendentemente dall'effettiva disposizione fisica del sistema.
Anche se sono stati fatti alcuni cambiamenti tra i diversi aggiornamenti dello standard USB nel passaggio da USB 1 a USB 2, USB 3 e USB 4, il funzionamento di base, il protocollo, le modalità di segnalazione e trasferimento dati e il formato dei pacchetti di dati sono fondamentalmente gli stessi.

Per capire come si trasferiscono i dati e come il protocollo opera all'interno dell'ambiente USB, è necessario prima capire come USB è fisicamente impostato e come appare al sistema o host.
Per la maggior parte dei sistemi che usano USB, l'host sarà una qualche forma di computer: desktop, laptop, tablet, ecc, anche se non è sempre necessario che sia così.

2.2.1 Elementi architetturali

In termini di definizioni delle entità in una rete USB, ci sono tre elementi principali:

- Host: L'host è il computer o l'oggetto che funge da elemento principale o controllore del sistema USB. L'host ha un hub contenuto al suo interno e questo è chiamato Root Hub.
- Hub: L'hub è un dispositivo che effettivamente espande il numero di porte disponibili; avrà una connessione a monte e diverse a valle. È possibile collegare un hub ad un altro per espandere ulteriormente la capacità e la connettività.
- Dispositivo: Queste sono le periferiche o gli oggetti a cui è collegato il collegamento USB. Mouse, tastiere, memorie flash, ecc.

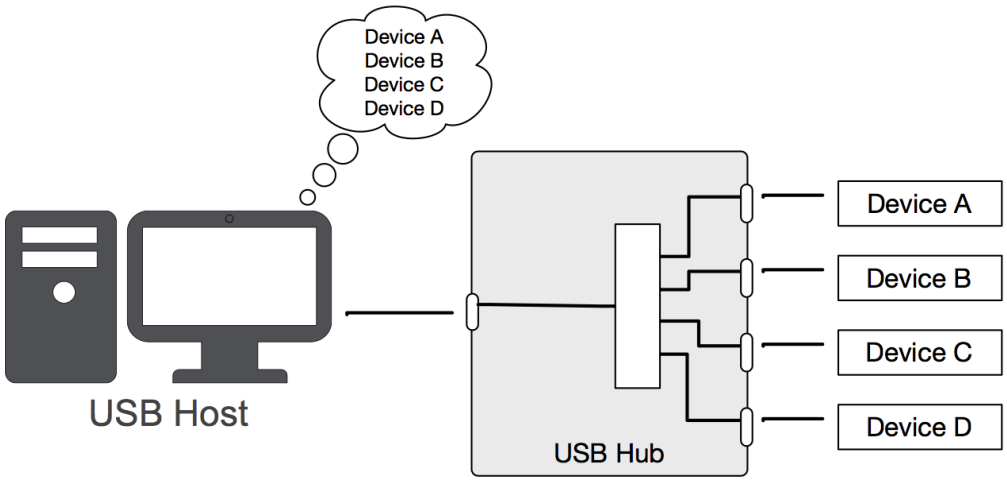


Figura 2.1: Diagramma dell'architettura USB [30]

2.2.2 Flusso dati

In termini di flusso di dati, i segnali viaggiano lungo il bus in modo che sia accessibile da tutti i dispositivi, ma solo il destinatario li accetta.
Questa funzionalità implica che sia necessario, per il corretto funzionamento USB, che i dati siano accettati solo dal dispositivo richiesto.
Per ottenere questo, quando un dispositivo è collegato al bus gli viene assegnato un numero univoco o un indirizzo dall'host durante il periodo di tempo in cui è collegato. Oltre all'indirizzo, il dispositivo contiene anche degli endpoint. Questi sono le fonti e le destinazioni effettive per le comunicazioni tra l'host e il dispositivo.
Gli endpoint possono operare solo in una direzione, cioè in ingresso o in uscita, ma non in entrambe, e i dispositivi possono averne fino a 16 per ogni direzione, anche se raramente vengono usati tutti. Di questi, uno ciascuno per l'ingresso e l'uscita deve essere riservato

come "Endpoint Zero" per quella direzione.
Gli endpoint zero sono usati per una moltitudine di attività tra cui l'auto-rilevamento e la configurazione del dispositivo sul bus, inoltre, secondo quanto detto in [21], sono gli unici accessibili fino a quando il dispositivo non è correttamente collegato sul bus.

2.2.2.1 Data Pipe

La comunicazione all'interno dell'USB si basa sul concetto di utilizzo di data pipe. Questi possono essere considerati come canali logici all'interno del flusso di dati sul bus.
In realtà, un data pipe USB è una connessione dall' host controllore a un'entità logica all'interno di un dispositivo, cioè l'endpoint. Poiché i pipe corrispondono agli endpoint, i termini sono talvolta usati in modo intercambiabile.
L'host utilizza quindi i data pipe per garantire che i dati da e verso un dispositivo siano correttamente indirizzati e che la fonte sia nota. Il data pipe usa una combinazione di indirizzo, endpoint e la direzione di esso per essere definito univocamente.

Per comunicare con gli endpoint zero è necessaria una forma speciale di data pipe perché deve essere usata per stabilire la comunicazione iniziale. Questa si chiama Default Control Pipe e può essere usata quando viene fatta la connessione fisica iniziale.

Ci sono due tipi di pipe USB:

- Message Pipe : Questo tipo è un canale USB bidirezionale ed è usato per i dati di controllo. I message pipe sono tipicamente usati per comandi brevi e semplici al dispositivo, e per risposte di stato da esso. Possono essere usati dal Default Control Pipe.

- Stream Pipe: Questa forma di canale USB è unidirezionale ed è collegata a un endpoint che trasferisce i dati usando un trasferimento isocrono, a interrupt o bulk. Queste tre modalità verranno discusse in seguito.

2.3 Livello Fisico

Per USB 1 e 2 viene impiegato un sistema a quattro fili. I cavi portano: alimentazione, terra e poi c'è una doppino intrecciato per il trasferimento dei dati differenziali.
Le linee sono designate Data+ (D+) e Data- (D-) per USB 1 e USB 2, e perciò permettono il trasferimento in modalità half-duplex [21]. Per USB 3, sono state introdotte alcune nuove linee. Per ogni porta ci sono TX1+ TX1- e TX2+ TX2- per coprire i dati trasmessi e poi RX1+ RX1- e RX2+ RX2- per i dati ricevuti. Questo permette la trasmissione in modalità dual simplex [24], ovvero la comunicazione è divisa in due canali distinti e in senso opposto, ciascuno in modalità simplex, cioè unidirezionale.
L'uso di doppini intrecciati e della segnalazione differenziale riduce gli effetti delle interferenze esterne. Riduce anche l'effetto di qualsiasi loop di ronzio che potrebbe causare problemi visto che nessun cavo è legato alla terra, ma il segnale deriva direttamente dalla differenza tra le due linee, il che riduce significativamente il disturbo.

2.3.1 Trasmissione

Per semplicità di argomentazione, riportiamo la tabella degli stati di transizione che può avere una linea USB: in particolare tutte le varie combinazioni logiche che possono avere le linee riservate ai dati (D+, D-), quindi 4 combinazioni totali. Per la nostra discussione ci concentriamo sullo standard Full Speed, poiché la differenza con i protocolli successivi è insita principalmente nella differenza del numero di linee, e non nel loro funzionamento generale.

Segnale	Stato della linea	Descrizione	D+	D-
J	Stato di inattività	Presente nelle transizioni di linea di trasmissione	alto	basso
K	Opposto allo stato J	Presente nelle transizioni di linea di trasmissione	basso	alto
SE0	Single-Ended Zero	Indica la fine di un pacchetto o la disconnessione di un dispositivo	basso	basso
SE1	Single-Ended One	Questo stato non dovrebbe mai accadere, è visto come un errore	alto	alto

Tabella 2.2: Lista dei possibili stati di transizione della linea di trasmissione

La trasmissione di dati si ottiene alternando gli stati J e K nella linea di trasmissione, dove:

- il bit 0 è trasmesso alternando la linea dallo stato J allo stato K o viceversa, cioè si cambia lo stato corrente.

- il bit 1 è trasmesso lasciando la linea inalterata, cioè si mantiene lo stato corrente.

Come detto in precedenza, la tabella riporta i dati per lo standard USB 2.0. Lo stesso sarebbe valido per gli standard successivi, dove al posto di una sola linea di trasmissione (D+ e D-), sono presenti due linee (TX1 e RX1; TX2 e RX2), dove per ciascuna di queste valgono considerazioni analoghe alla singola linea del protocollo Full Speed.

2.3.2 Stati della linea

Di seguito vengono riportati gli stati in cui una linea si può trovare, associati al segnale fisico che li rappresenta.

Stato	Descrizione	segnale fisico
Staccato	Nessun dispositivo rilevato	SE0 per almeno 2 μs
Connesso	Inizia la fase di enumerazione e imposta lo stato a idle	una delle due linee viene portata a livello logico alto
Idle	Host e dispositivo connessi, in attesa di informazioni. Porta la linea in stato di transizione J.	inizializzato dallo stato connesso
Sincronizzazione	Inizio della trasmissione	segnale di sync
Fine di Pacchetto (EOP)	Fine della trasmissione	SE0, SE0, J
Reset	Reset del dispositivo	SE0 per almeno 2.5 ms
Sospensione	Stato di sospensione per ridurre il consumo energetico. Si esce da questo stato dopo un segnale di ripresa o reset	J per almeno 3 ms
Ripresa	L'host richiede che il dispositivo si riprenda dallo stato di sospensione. Può essere il dispositivo a inizializzare la sequenza	K per almeno 20 ms, poi EOP. Se inizializzata dal dispositivo, K per almeno 1 ms, poi segnale di Ripresa da parte dell'Host.

Tabella 2.3: Lista dei possibili stati della linea di trasmissione

2.3.3 Segnale di sincronizzazione

Un pacchetto USB inizia con una sequenza di sincronizzazione a 8 bit, 00000001. Quindi, dopo lo stato inattivo iniziale J, le linee alternano KJKJKJKK. L'ultimo bit uguale a 1 (ripetizione dello stato K) segna la fine del segnale di sincronizzazione e l'inizio del frame USB. Per applicazioni ad alta larghezza di banda, il pacchetto inizia con una sequenza di sincronizzazione da 32 bit, in cui i primi 30 rappresentano la ripetizione della coppia KJ e gli ultimi due la coppia KK.

2.3.4 Fine di pacchetto

L'EOP (End Of Packet) è indicato dalle linee di trasmissione come due volte lo stato SE0, seguito dallo stato J, riportando le linee di trasmissione nello stato di inattività.

2.3.5 Reset del bus

Il segnale di reset del bus USB è formato da un uso prolungato (circa 10-20 ms) dello stato SE0.
Dopo questa procedura, se il dispositivo è ad alta velocità, ne viene effettuata una ulteriore detta chirp (da parte del dispositivo stesso). Questo specifica all'host che la connessione è disponibile a larghezza di banda elevata.

2.3.6 Velocità di trasferimento

Ogni versione di USB è caratterizzata da una sempre maggiore velocità di trasferimento, come abbiamo visto nella tabella 3.1. Questa velocità è distinta in quella teorica e in quella

reale, questo perché la prima è la larghezza di banda totale del bus, mentre quella reale è quella effettivamente utilizzabile, una volta tolti i dati scambiati per il controllo e per le intestazioni dei pacchetti, che secondo quanto analizzato in [31], occupano dal 20% al 40% del flusso totale di informazioni.

2.3.7 Latenza

Come già detto in precedenza, lo standard SuperSpeed ha una latenza inferiore al μs, e questo è ciò che ci si può aspettare da tutte le versioni successive. Questo può aiutare a capire l'incremento nelle velocità delle diverse versioni, anche se verrà poi analizzato più avanti nella discussione, quando si parlerà di tempo inter pacchetto.

2.4 Livello Protocollare

Il livello protocollare definisce tutte le convenzioni scelte per la comunicazione tra host e dispositivo. In particolare vedremo come avvengono le trasmissioni tramite pacchetti e le diverse tipologie di questi.

2.4.1 Tipologie di scambio

Come accennato in precedenza, ci sono quattro tipi fondamentali di scambio di dati che possono essere effettuate in USB:

- Controllo: Questo tipo di scambio dati all'interno del protocollo USB è usato dall'host per inviare comandi o parametri di una richiesta. Le lunghezze dei pacchetti sono definite all'interno del protocollo come 8 byte per la bassa velocità (USB 1.0), 8-64 byte per la piena (USB 1.1), e 64 byte per i dispositivi ad alta velocità (USB 2.0 o superiore).
  Questa tipologia è l'unica che è obbligatoria.

- Interruzione: Il protocollo USB definisce un messaggio di interruzione (interrupt). Questo è spesso usato dai dispositivi che inviano piccole quantità di dati, ad esempio mouse o tastiere. Si tratta di un messaggio di polling (sondaggio) da parte dell'host che deve richiedere dati specifici del dispositivo remoto, e quindi non si tratta di un'interruzione vera e propria perché inizializzata dall'host e non dal dispositivo.

- Bulk: Questo messaggio del protocollo USB è usato da dispositivi come le stampanti per le quali sono richieste quantità di dati molto più grandi. In questa forma di trasferimento dati, blocchi di dati di lunghezza variabile sono inviati o richiesti dall'host. La lunghezza massima è di 64 byte per i dispositivi a piena velocità, 512 byte per quelli ad alta velocità e 1024 byte per i protocolli successivi. L'integrità dei dati è verificata usando il controllo di ridondanza ciclica (CRC) e poi viene inviata una conferma. Questo meccanismo di trasferimento dati USB non è utilizzato dalle periferiche in cui il tempo è critico perché utilizza la larghezza di banda non utilizzata dagli altri meccanismi, e quindi può risultare molto lenta.
  Questo è il metodo principalmente utilizzato dai visori VR, perché è quello che permette il trasferimento della maggior quantità di dati. Il limite sulla larghezza di banda non è presente in questo caso perché si usa un canale dedicato.

- Isocrono: Questa forma di trasferimento dati è usata per trasmettere dati in tempo reale ed è usata per applicazioni come canali audio dal vivo, streaming video, ecc. Non usa il controllo dati, poiché non c'è tempo per reinviare i pacchetti di dati con errori: i dati persi possono essere sistemati meglio dei ritardi sostenuti dal reinvio degli stessi. Ciò significa che per applicazioni in cui il tempo è fondamentale, reinviare i pacchetti comporterebbe un disturbo maggiore che non quello dovuto alla correzione degli errori a posteriori, in cui, per esempio, i dati mancanti di un pacchetto possono essere parzialmente ricostruite a partire da quelle dei precedenti. Le dimensioni dei pacchetti possono essere fino a 1024 byte.

2.4.2 Differenze tra USB 2.0 e 3.0+

Come delineato in precedenza, questa trattazione si sofferma principalmente sul protocollo USB 2.0, specificando poi di volta in volta le differenze, se presenti, con i protocolli successivi. Questa scelta è stata fatta perché le principali differenze insitano nel livello fisico, in cui il diverso numero di pin per connettore, che in USB 3.0 e superiore è il type-C, permette trasmissioni a larghezze di banda elevate. Nei livelli superiori, invece, i due protocolli sono molto simili tra loro, con solamente qualche differenza che verrà sottolineata in seguito ove necessario.
È inoltre importante ricordare che sebbene USB 3.0 e superiori siano nuove versioni, queste non sostituiscono gli standard 2.0 Full speed, in quanto sia in USB 3.0 che 4.0 rimane comunque la retrocompatibilità per le versioni precedenti. Ciò significa che a livello hardware nella porta type-C sono presenti anche le linee D- e D+, utilizzate in caso venga negoziata la modalità full speed tra host e dispositivo. Questo è il caso più comune per una moltitudine di dispositivi che semplicemente non necessitano di velocità elevate per funzionare, come mouse o tastiera. Chiaramente questo non è il caso nel nostro particolare utilizzo, in cui la velocità di trasmissione è chiave.

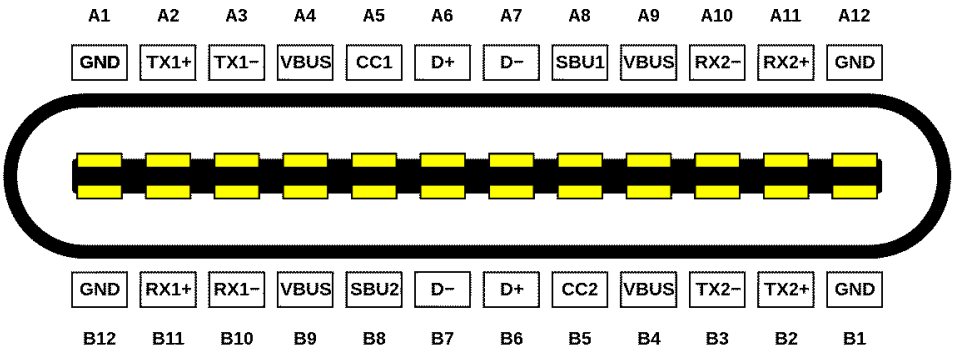


Figura 2.2: Schema della struttura dei pin in una porta USB type-c [32]

2.4.3 Pacchetti

Nelle comunicazioni USB, i dati sono trasmessi come pacchetti. Inizialmente, tutti i pacchetti sono inviati dall'host ai dispositivi, ma alcuni di questi pacchetti indicano a un dispositivo di inviare alcuni pacchetti di risposta.
Ogni pacchetto inviato è preceduto da un campo sync (sequenza di sincronizzazione) e seguito da un marcatore di fine pacchetto. Questo definisce l'inizio e la fine del pacchetto e permette anche al nodo ricevente di sincronizzarsi correttamente in modo che i vari elementi siano a posto temporalmente.

Poiché ci sono diversi tipi di dati che possono essere inviati, per ogni pacchetto, è richiesto un ID che indichi il tipo, e a volte viene anche restituita una conferma, sotto forma di acknowledgment.
In particolare, dopo il campo sync, tutti i pacchetti sono composti da byte di 8 bit, trasmessi a partire dal bit meno significativo (modalità little endian). Questi 8 bit costituiscono un byte identificatore del pacchetto (PID, Packet IDentifier). Il PID è in realtà di 4 bit, ma il byte consiste nel PID a 4 bit seguito dal suo complemento bit a bit. Questa ridondanza aiuta a rilevare gli errori.
Tutti i possibili PID sono divisi per categoria di utilizzo:

Tipo	Valore	byte trasmesso (lsb first)	Nome	Descrizione
Riservato	0000	0000 1111		
Token	1000	0001 1110	SPLIT	Transazione di split (USB 2.0)
	0100	0010 1101	PING	Controlla se l'endpoint può ricevere dati (USB 2.0)
Speciale	1100	0011 1100	PRE	Preambolo di transazione per bassa velocità
Handshake			ERR	Errore di transazione di split (USB 2.0)
	0010	0100 1011	ACK	Pacchetto accettato
	1010	0101 1010	NAK	Pacchetto rifiutato. Ritrasmettere
	0110	0110 1001	NYET	Dati non ancora pronti (USB 2.0)
	1110	0111 1000	STALL	Trasferimento impossibile, fare recupero dati
Token	0001	1000 0111	OUT	Indirizzo per trasmissione da host a dispositivo
	1001	1001 0110	IN	Indirizzo per trasmissione da dispositivo a host
	0101	1010 0101	SOF	Indicatore di inizio di frame
	1101	1011 0100	SETUP	Indirizzo per trasmissione di controllo da host a dispositivo
Data	0011	1100 0011	DATA0	Pacchetti di dati con numero pari
	1011	1101 0010	DATA1	Pacchetti di dati con numero dispari
	0111	1110 0001	DATA2	Pacchetti di dati per trasmissioni isocrone ad alta velocità (USB 2.0)
	1111	1111 0000	MDATA	Pacchetti di dati per trasmissioni isocrone ad alta velocità (USB 2.0)

Tabella 2.4: Lista dei possibili ID dei pacchetti (PID)

2.4.3.1 Pacchetti di Token

I pacchetti di token sono sempre i primi ad essere inviati in una transazione. Sono costituiti da un byte di PID seguito da due byte di payload: 11 bit di indirizzo e un 5 bit di CRC.
I token vengono inviati solo dall'host e mai da un dispositivo.
Di seguito sono riportati i token presenti a partire dal protocollo USB 1.0:

- **IN**: Il token presenta 7 bit di indirizzo del dispositivo seguito da 4 bit per il numero della funzione (utilizzato dai dispositivi multifunzione), e comanda il dispositivo di trasmettere pacchetti di dati.
  Dopo un token di IN, inviato dall'host, quest'ultimo si aspetta una risposta dal dispositivo, che può essere NAK, STALL, oppure direttamente i pacchetti di dati (nel caso la trasmissione sia stata ricevuta con successo). In quest'ultimo caso l'host risponde al dispositivo con un ACK.

- **OUT**: Come in IN, il token presenta 7 bit di indirizzo del dispositivo seguito da 4 bit per il numero della funzione (utilizzato dai dispositivi multifunzione), ma questa volta comanda il dispositivo di ricevere pacchetti di dati. Nelle trasmissioni OUT, siccome è l'host a dover trasmettere dati, questo li trasmette immediatamente dopo il token OUT. A seguito di queste operazioni il dispositivo che li riceve risponde con ACK, NAK, STALL o NYET, a seconda dei casi.

- **SETUP**: Il token di SETUP opera in maniera analoga al token di OUT, ma viene utilizzato solamente nella fase iniziale di setup del dispositivo. La sua trasmissione è seguita da un pacchetto di dati a 8 byte del formato DATA0.

- **SOF**: Ogni ms, viene inviato dall'host un pacchetto speciale di Start Of Frame (SOF), che contiene un numero incrementale che rappresenta il frame, al posto dell'indirizzo del dispositivo. Questo è utilizzato per sincronizzare le trasmissioni di tipo isocrono e a interrupt.
  Nel protocollo USB 2.0 questi pacchetti vengono invece inviati ogni "micro-frame" di durata pari a un ottavo di quella del frame, e vengono perciò inviati ogni 125 μs.

A partire da USB 2.0 sono stati introdotti anche i seguenti token:

- **PING**: Il token di ping serve all'host per chiedere ad un dispositivo se è pronto a ricevere una trasmissione di tipo OUT. L'utilizzo più comune è quello di polling da parte dell'host verso un dispositivo che di recente ha risposto con NAK o NYET, in questo modo si evita di inviare pacchetti di lunghezza elevata a un dispositivo che si sospetta sia non in grado di ricevere dati. Successivamente il dispositivo risponde con ACK, NAK o STALL, in base alla situazione.

- **SPLIT**: Le transazioni di SPLIT sono utilizzate dall'host per comunicare a un hub i dettagli di una connessione a velocità bassa che dovrà gestire, e poi restituire i risultati. Questo evita che un dispositivo a bassa velocità occupi il bus ad alta velocità con la trasmissione dati, diventando allora un cono di bottiglia per tutto il sistema.

  Come si può vedere in [33], a partire dal protocollo USB 3.0 i pacchetti di SOF sono stati sostituiti da pacchetti ITP (Isochronous Timestamp Packet), che trasportano le informazioni di tempo dell'host ai dispositivi, per allineare la sincronizzazione.

Il formato dei pacchetti di token è descritto dal diagramma sottostante:

2.4.3.2 Pacchetti di inizio frame

È utilizzato da USB 1.0 e 2.0 per allineare temporalmente gli orologi degli endpoint con quello dell'host. A questo pacchetto un dispositivo non risponde mai con un segnale di acknowledgment.
Come detto in precedenza, a partire da USB 3.0 questi pacchetti sono stati sostituiti da

Campo	Sync	PID	Address	Endpoint	CRC5	EOP
Bit		8	7	4	5	
Segnale	segnale di sync	PID specifico	indirizzo del dispositivo	indirizzo dell'endpoint	CRC a 5 bit	segnale EOP

Tabella 2.5: Diagramma dei bit in un pacchetto di token

pacchetti ITP, che servono lo stesso scopo. È interessante notare che questi pacchetti sono gli unici ad essere trasmessi in modalità broadcast sul BUS, in modo tale che tutti i dispositivi connessi ne possano beneficiare.

2.4.3.3 Pacchetti di dati

Un pacchetto di dati consiste nel PID, seguito dal payload del pacchetto, con dimensione variabile da 0 a 1024 byte, e seguito da un CRC a 16 bit.
Ci sono due diversi tipi di pacchetto di dati: DATA0 e DATA1. Questo perché un pacchetto di dati è sempre preceduto da un pacchetto di token di indirizzo, e di solito è seguito da un token di handshake da parte del ricevitore verso il trasmettitore. I due diversi tipi di DATA effettuano in tutti i sensi un stop-and-wait ARQ a un bit, determinato dal numero dopo DATA. In particolare, i dati vengono mandati dal trasmettitore con tipologie alternate (una volta DATA0, una volta DATA1), in questo modo il ricevitore sa che dovrebbe ricevere una sequenza alternata; se ciò non accade e si ricevono due pacchetti dello stesso tipo, il secondo viene ignorato, anche se viene comunque mandato un acknowledgment.

Questo metodo permette di distinguere i casi in cui il pacchetto venga perso nel tragitto (si ricevono due pacchetti dello stesso tipo), ai casi in cui si riceve il pacchetto giusto ma il CRC fallisce. In quest'ultimo caso non viene generato un ACK di risposta, che fa si che il pacchetto venga ritrasmesso.

Da USB 2.0 sono stati introdotti anche pacchetti di tipo DATA2 e MDATA, per permettere trasmissioni ad una maggiore velocità, maggiore di un pacchetto ogni micro-frame.
Il formato dei pacchetti di dati è descritto dal diagramma sottostante:

Campo	Sync	PID	Data	CRC16	EOP
Bit		8	0-8192	16	
Segnale	segnale di sync	PID specifico	payload	CRC a 16 bit	segnale EOP

Tabella 2.6: Diagramma dei bit in un pacchetto di dati

2.4.3.4 Pacchetti di Handshake

I pacchetti di handshake sono costituiti da un solo byte di PID e sono generalmente inviati in risposta ai pacchetti di dati. Il rilevamento degli errori avviene tramite la trasmissione del PID da 4 bit due volte, in forma complementare, per un totale di 8 bit (vedi tabella 2.4).
I tre tipi di base sono:

- **ACK**: indica che i dati sono stati ricevuti con successo;
- **NAK**: indica che i dati non possono essere ricevuti e devono essere ritentati;

- **STALL**: indica che il dispositivo ha una condizione di errore e non può trasferire dati fino a quando non si verifica un'azione correttiva (come l'inizializzazione del dispositivo).

USB 2.0 ha aggiunto due ulteriori pacchetti di handshake: NYET e ERR:

- **NYET**: indica che una transazione suddivisa non è ancora stata completata e che il ricevitore non è ancora pronto. Un secondo utilizzo del pacchetto NYET è quello di comunicare all'host che il dispositivo ha accettato un pacchetto di dati, ma non può accettarne altri a causa dei buffer pieni. Ciò consente all'host di passare all'invio di piccoli token PING per informarsi sulla disponibilità del dispositivo, piuttosto che inviare un intero pacchetto DATI indesiderato solo per ottenere un NAK.;

- **ERR**: indica che una transazione suddivisa è fallita.

L'unico pacchetto di handshake che l'host USB può generare è ACK. Se non è pronto a ricevere dati, non deve dare istruzioni a un dispositivo per l'invio.

2.4.3.5 Transazione OUT

Le transazioni sono raggruppamenti di pacchetti, che svolgono insieme un unico lavoro. Nel caso della transazione OUT, l'host vuole inviare dei dati al dispositivo.
Questa operazione si divide in 3 fasi:

1. Invio del token di OUT da parte dell'host. In questa fase si comunica al dispositivo di indirizzo specificato nel token di iniziare ad ascoltare per l'arrivo di dati nell'enpoint specificato ancora una volta, nel token;

2. Invio del pacchetto di dati effettivo da parte dell'host. Questo pacchetto è di tipo DATA (0 o 1), e contiene i dati da trasmettere;

3. Invio del pacchetto di ACK da parte del dispositivo. Questo pacchetto rappresenta l'handshake del dispositivo, che conferma la corretta ricezione del pacchetto.

Il protocollo USB 3.0 e successivi usano questi 3 passaggi in modo diverso, in particolare per le transazioni di OUT il token è incorporato nel pacchetto di dati. Inoltre, le versioni 3.0 e successive supportano la modalità dual simplex, che permette la comunicazione in entrambe le direzioni in simultanea su canali indipendenti, per questo motivo durante tutte le transazioni in questa versione, è possibile inviare più pacchetti di dati senza prima ricevere un handshake. Questa operazione è chiamata burst, ed è di durata variabile e di lunghezza dei pacchetti variabile.
Ulteriori differenze stanno nel fatto che USB 2.0 trasmette la tripletta di indirizzi (dispositivo, endpoint, direzione) in modo broadcast a tutti i dispositivi connessi, mentre USB 3.0 e superiori la trasmettono in modalità unicast, tranne che per il pacchetto ITP, che viene comunque trasmesso in modalità broadcast perché è utilizzato da tutti i dispositivi indipendentemente dal fatto che stiano comunicando o meno.

2.4.3.6 Transazione IN

La transazione IN, al contrario di quella OUT, raggruppa i pacchetti che si riferiscono all'invio di dati da parte del dispositivo verso l'host.
Anche questa operazione si divide in 3 passaggi:

1. Invio del token di IN da parte dell'host. In questa fase si comunica al dispositivo di indirizzo specificato nel token di iniziare ad inviare i dati memorizzate nel buffer dell'enpoint specificato ancora una volta nel token;

2. Invio del pacchetto di dati effettivo da parte del dispositivo. Questo pacchetto è di tipo DATA (0 o 1), e contiene i dati da trasmettere richiesti dall'host;

3. Invio del pacchetto di ACK da parte dell'host. Questo pacchetto rappresenta l'handshake, che conferma la corretta ricezione del pacchetto.

Il protocollo USB 3.0 e successivi usano questi 3 passaggi in modo diverso, in particolare per le transazioni di IN il token è rimpiazzato da un ulteriore handshake. In questo caso il dispositivo può rispondere con dati, STALL o NRDY.

2.4.3.7 Transazione SETUP

La transazione di SETUP viene utilizzata per l'enumerazione del dispositivo e la gestione della connessione. Informa il dispositivo che l'host desidera avviare uno scambio di trasferimenti di controllo.
Anche in questo l'operazione è suddivisa in 3 pacchetti:

1. Invio del token di SETUP da parte dell'host. In questa fase si comunica al dispositivo di indirizzo specificato nel token di iniziare la procedura di setup e di prepararsi a ricevere un pacchetto di dati;

2. Invio del pacchetto di dati effettivo da parte del dispositivo. Questo pacchetto è di tipo DATA=, e contiene i dati di setup inviati dall'host. Questi dati sono contenuti in un payload da 8 byte e contiene informazioni come il tipo di controllo da effettuare (enumerazione, richiesta descrittori, ecc.) e altri campi come la lunghezza aspettata del pacchetto di risposta da parte del dispositivo, il componente a cui si riferisce la richiesta (dispositivo, interfaccia o endpoint);

3. Invio del pacchetto di ACK da parte del dispositivo. Questo pacchetto rappresenta l'handshake, che conferma la corretta ricezione del pacchetto di setup all'host.

2.4.3.8 Trasmissione di controllo

La transazione di SETUP è il primo passaggio della trasmissione di controllo, ed è seguito dall'invio dei dati richiesti da parte del dispositivo verso l'host, successivamente il dispositivo entra in uno stato fantoccio che simboleggia la fine della trasmissione.

2.4.4 Enumerazione

L'enumerazione USB è il processo con cui un host rileva che è stato collegato un dispositivo USB, identifica il dispositivo collegato e quindi carica i relativi driver.
La specifica USB definisce sei stati del dispositivo. Durante l'enumerazione, un dispositivo passa attraverso quattro dei sei stati: Alimentato, Predefinito, Indirizzo e Configurato. (Gli altri stati sono Collegato e Sospensione). Si tratta di un mix di tecniche hardware, per rilevare la presenza di un dispositivo e software, per identificare ciò che è stato collegato.
Dopo aver rilevato la presenza di un dispositivo, l'host avvierà un trasferimento con esso per determinare di cosa si tratta. L'host esegue questa operazione richiedendo i "descrittori" del

dispositivo che ne definiscono la classe e i driver che devono essere caricati.

L'host USB ha l'indirizzo 0, e poi assegna gli indirizzi a ciascun dispositivo in ordine crescente. Quando un dispositivo non è ancora stato configurato oppure è appena stato resettato, torna ad avere indirizzo predefinito 0, e poi gli viene cambiato dall'host in un numero da 1 a 127, cioè tutti i valori non riservati esprimibili con 7 bit, ovvero la lunghezza standard del campo di indirizzo nei pacchetti.

2.4.5 Descrittori

I dispositivi vengono identificati con l'ausilio di descrittori; ogni descrittore contiene informazioni sul dispositivo nel suo complesso o su un elemento di esso.
I descrittori USB sono strutture di dati che che consentono all'host di conoscere le caratteristiche di un dispositivo. Ogni dispositivo (ad eccezione dei dispositivi composti) ha uno ed un solo descrittore che contiene informazioni su di esso e specifica il numero di configurazioni supportate.
Per ogni configurazione il dispositivo ha un descrittore di configurazione che fornisce informazioni sull'uso dell'alimentazione del dispositivo e sul numero di interfacce supportate da quella determinata configurazione.
Per ogni interfaccia, il dispositivo ha un descrittore di interfaccia che specifica il numero di endpoint. Ogni endpoint ha poi un descrittore endpoint che contiene le informazioni necessarie per comunicare con lo stesso.

I dispositivi SuperSpeed devono inoltre fornire un descrittore BOS (Binary device Object Store) e almeno due descrittori di capacità del dispositivo subordinati: un descrittore USB SuperSpeed e un descrittore di estensione USB 2.0.
Successivamente, ogni descrittore di endpoint SuperSpeed ha un descrittore di accompagnamento di endpoint SuperSpeed subordinato.
Quando si riceve una richiesta di descrittore di configurazione, il dispositivo deve restituire il descrittore di configurazione e tutti i dati relativi all'interfaccia, all'endpoint e agli altri descrittori subordinati della configurazione, fino al numero di byte richiesti.

Un descrittore può contenere una stringa di testo come il nome del fornitore o del dispositivo, oppure il numero di serie.

2.5 Altri protocolli di interesse

2.5.1 Wireless USB

L'evoluzione dello standard USB si concentra su due strade: la prima prevede un innalzamento della velocità massima di trasferimento, mentre la seconda strada, chiamata Wireless USB, prevede l'abbandono dei cavi per la comunicazione tramite onde radio. Una prima specifica Wireless USB è già stata standardizzata e alcuni dispositivi sono in commercio.

Entrambe queste opzioni sono rilevanti per la nostra applicazione: l'aumento della velocità massima di trasferimento permetterebbe un invio di dati maggiore, e quindi comporterebbe un possibile aumento della qualità audio e video; la seconda invece è rilevante in quanto la mobilità quando si indossa un visore VR è essenziale e l'essere ristretti dall'attacco con il cavo può rilevarsi fastidioso se non pericoloso.

Per questo motivo, qui di seguito vengono riportate alcune informazioni riguardo quest'ultima estensione del protocollo.

Wireless USB (o WUSB) è un'estensione senza fili per l'USB dotata di elevata ampiezza di banda, a corto raggio che combina la velocità dei dispositivi USB 2.0 e superiori con la praticità della tecnologia wireless. Tale applicazione definisce dei limiti di banda in base alla distanza tra il dispositivo e l'host, che partono dalla massima possibile, ovvero 480 Mb/s, entro i 3 metri di distanza, e scendono gradualmente fino a 10 metri di distanza, dove la larghezza di banda è stimata, in [34], a 110 Mb/s. Tale limite sulla distanza è abbastanza irrilevante per i nostri fini, in quanto l'utilizzo di un visore VR può essere facilmente confinato in una zona di qualche metro quadrato, e quindi all'interno della zona a massima larghezza di banda.

Wireless USB si basa sulla modulazione ultra wideband della WiMedia Alliance [35], con una banda passante teorica di 480 Mbit/s fino alla distanza di 3 metri e 110 Mbit/s fino a 10 metri. Per le nostre applicazioni questa velocità può rivelarsi sufficiente, sebbene sicuramente non soddisfacente.

Questa strada può sicuramente rivelarsi come soluzione al nostro problema, però richiederebbe ancora tempo per sviluppare una velocità di trasferimento almeno simile a quelle di USB 3.0 e superiori. Per questo motivo, almeno per il corto termine, si considera la versione cablata come opzione dominante, e su quest'ultima verrà centrata la nostra analisi.

2.5.2 USB 4.0

È in fase di sviluppo la tecnologia USB 4.0, che si basa sulle specifiche del protocollo Thunderbolt 3 [36], supporta 40 Gbit/s di throughput, è compatibile con Thunderbolt 3 e DisplayPort 2.0 Alt Mode, e ha una retrocompatibilità con USB 3.2 e USB 2.0.
L'architettura definisce un metodo per condividere dinamicamente un singolo collegamento ad alta velocità con più tipi di dispositivi che è più utile al trasferimento di dati per tipo e applicazione.

Usb 4.0 rimpiazza completamente USB 3.2, ove possibile e mantiene la retrocompatibilità con USB 2.0. La base architetturale per USB 4.0 rimane quella di Enhanched SuperSpeed definita da USB 3.2. La differenza tra i due sta nel fatto che USB 4.0 è connection oriented con architettura a tunnel fatta appositamente per combinare molteplici protocolli in un'unica interfaccia fisica, in modo da condividere dinamicamente la velocità totale e le performance del nuovo standard. USB 4.0 permette ai trasferimenti di dati di operare in parallelo a altri protocolli indipendenti specifici per video (Display Port protocol), caricamento e immagazzinamento dati (PCI express) e interfacce host-to-host.
In aggiunta USB 4.0 espande ulteriormente il limite di banda teoriche di USB 3.2 da 20 Gb/s a 40 Gb/s, seppur utilizzando la stessa architettura dual simplex a doppia corsia.
In [37] si evince che la nuova tecnologia USB 4.0 è attualmente in fase di sviluppo specialmente nel settore industriale, dove è nato il concetto di Industrial Augmented Reality (IAR).

Capitolo 3

Raccolta di dati sperimentali

3.1 Meta Quest 2

La raccolta di dati sperimentali è stata fatta utilizzando il visore Meta Quest 2, uno tra i più diffusi nel settore videoludico. Il visore è dotato di un display LCD a commutazione rapida diviso in due lenti regolabili, con una risoluzione di 1832x1920 per occhio. La frequenza di aggiornamento può essere impostata a 60, 72, 90 e 120 Hz. Nella raccolta dati effettuata si è cambiato questa impostazione, partendo da 90 e arrivando a 120 Hz.
Il visore è dotato di due controller, il cui uso però è opzionale ed è sostituibile dal tracciamento delle mani.
Il tracciamento di mani e giocatore avviene a 6 gradi di libertà (3 posizionali e 3 rotazionali), e permette al visore di captare ogni movimento di testa e corpo senza l'uso di nessuno strumento aggiuntivo esterno.
Il dispositivo è dotato di audio posizionale 3D integrato direttamente nel visore, eventualmente collegabile a delle cuffie tramite la porta audio da 3,5mm, oppure utilizzando le casse integrate.
Il visore è di tipo all-in-one avanzato, ovvero permette l'utilizzo solamente con controller e il visore stesso. Alternativamente il visore può essere connesso a un computer con il cavo Meta Link.
Il cavo è USB type-C versione 3.0 SuperSpeed 5 Gbps con collegamento di almeno 2 Gb/s [38].
La funzione all-in-one permette teoricamente di utilizzare solamente la potenza di calcolo presente nell'hardware del dispositivo, tuttavia questa è limitata rispetto a quella di un computer esterno, per ovvi motivi di spazio. Per questo motivo nella nostra raccolta e analisi dati abbiamo utilizzato il visore collegato al computer, che rappresenta comunque la metodologia di utilizzo più diffusa per il settore videoludico.

Figura 3.1: Visore Meta Quest 2. [39]

3.2 Procedura di raccolta dati

3.2.1 Apparato sperimentale

L'apparato sperimentale è costituito dal visore Meta Quest 2, collegato tramite il cavo Meta Link [40] ad un computer esterno Omen Obelisk - HP, dotato di processore Intel Core i7-9700K, scheda grafica NVIDIA 2080 Ti, 64GB di Ram DDR4, 6TB HDD, 2TB SSD e Windows 10.
Per la nostra raccolta dati abbiamo usato anche i controller inclusi con il visore.
L'utilizzo del visore è stato circoscritto ad una zona relativamente limitata di circa 1 metro quadrato.

È importante notare che il computer esterno è dove vengono eseguite le applicazioni e i calcoli, che poi vengono trasmessi via cavo al visore.
Questo apparato ci permette di intercettare e catturare le tracce USB trasmesse tra il computer e il visore, per poterle analizzare.

3.2.2 Wireshark

Le tracce USB sono state intercettate e catturate mediante il programma Wireshark [41], installato alla versione 3.6.6.
Wireshark è stato fatto partire prima dell'avvio delle applicazioni ed è stato fermato dopo la chiusura delle stesse. Questo ci ha permesso di catturare anche il traffico di controllo generato all'inizio. Inoltre, siccome al computer erano collegati altri dispositivi collegati tramite USB, con Wireshark abbiamo filtrato solamente il traffico derivante e direzionato all'interfaccia di nostro interesse.

3.2.3 USBPcap

Una importante considerazione va fatta sul software che ci ha permesso di catturare le tracce USB: USBPcap [42].

Il software permette di catturare le tracce USB, che altrimenti sarebbero invisibili a Wireshark.

Per come effettua la cattura dei pacchetti, il software aggiunge un header di preambolo ad ogni pacchetto, che però non impatta l'analisi dati, in quanto specifica solamente un diverso arrangiamento degli stessi dati che ci sarebbero normalmente. Di questo fenomeno si parlerà più avanti nell'analisi dei dati.

3.2.3.1 USB Request Blocks (URBs)

USBPcap cattura i dati degli USB Request Block (URB) che vengono trasportati all'interno degli I/O Request Packets (IRP). Wireshark presenta i pacchetti URB come frame. È necessario tenere presente che il pacchetto di USBPcap non è esattamente uguale a quello delle specifiche USB e che il frame di Wireshark è sicuramente diverso dal frame USB.
Questa differenza, sebbene sostanziale a livello dei pacchetti, non altera il senso generale dell'analisi, in quanto vista la numerosità dei frame catturati (circa 6 milioni), tutte le analisi statistiche vengono comunque molto fedeli alla realtà. Per quanto riguarda le analisi dei singoli pacchetti, queste verranno fatte invece sui singoli frame, ancora una volta questa differente pacchettizzazione dei dati non altera i valori di interesse.

3.3 Applicazioni utilizzate

Per questo studio abbiamo differenziato la cattura dei frame utilizzando 4 diverse applicazioni, tutte gratuite, prese sia da Steam che da Oculus Store.
L'utilizzo di 4 applicazioni diverse è inteso in modo da diversificare la provenienza delle tracce analizzate, per provare a definire delle differenze di traffico dovute alle diverse situazioni.

3.3.1 Spider-Man: Homecoming VR

La prima applicazione utilizzata è stata Spider-Man: Homecoming VR, si tratta di un breve trailer, tratto dall'omonimo film, con scopo di promozione.
Lo scenario è in cima a un palazzo in cui il movimento spaziale è non presente (perciò ci sono solo 3 gradi di libertà rotazionali), e dove il giocatore è chiamato a usare i controller per mirare a degli oggetti fluttuanti.
Vista la brevità del gioco stesso e la scarsa variabilità negli scenari, la qualità grafica è molto curata, con poca distinzione tra le parti cinematiche e quelle in motore di gioco.
Questo gioco è stato fatto girare a 90 Hz di frequenza di aggiornamento e grafica alta.

Figura 3.2: Schermata di gioco di Spider-Man: Homecoming VR [43].

3.3.2 Cactus Cowboy - Plants at War

La seconda applicazione utilizzata è Cactus Cowboy - Plants at War, un gioco uscito appena qualche giorno prima dell'acquisizione dati. Si tratta di un gioco non sviluppato da una grande azienda produttrice, bensì da pochi sviluppatori autonomi. La grafica si presenta un po' semplificata e basata su poligoni abbastanza evidenti, che però non diminuiscono la qualità dell'esperienza di gioco.
Il gioco si presenta come un buon mix di cinematiche ed esperienze di gioco, in cui il giocatore si trova a poter decidere dove andare e cosa fare, seppure sempre seguendo una linea di tutorial.
Il gioco si conclude in una battaglia a campo aperto dove il giocatore ha piena facoltà decisionale, e deve usare sia movimenti fisici che rotazionali, oltre ai comandi con i controller. Quest'ultima fase era anche piena di oggetti in movimento costante come proiettili, che sicuramente aumentano la difficoltà di calcolo richiesta.
Questo gioco è stato eseguito con le impostazioni grafiche al massimo, sebbene questa non sia, come già discusso, particolarmente pesante, e frequenza di aggiornamento del visore di 90 Hz.

Figura 3.3: Schermata di gioco di Cactus Cowboy - Plants at War [44].

3.3.3 Astral Slider

La terza applicazione utilizzata è Astral Slider, un gioco arcade in cui il giocatore si trova su una pista a circolo chiuso che si ripete per 3 volte. La pista è poi divisa in 8 binari, in cui sono sparsi degli ostacoli che il giocatore deve evitare, e dei blocchi che invece possono essere distrutti usando i controller che aumentano la velocità della pista.
Il giocatore non ha facoltà di movimento libero, se non quello di decidere di cambiare binario tramite i controller, ma rimane libero di guardare dove vuole con il visore, quindi anche in questo caso rimangono i 3 gradi di libertà rotazionali.
Il gioco si presenta come diverse "sfide" con piste diverse, ciascuna con grafica abbastanza minimale e geometrica, alternate da momenti di scelta al menù. Per questo motivo ci aspettiamo che le tracce mostrino questa differenza con un calo di traffico durante questi ultimi momenti.
Anche in questo caso il gioco è stato fatto partire con grafica al massimo e con una frequenza di aggiornamento del visore a 90 Hz.

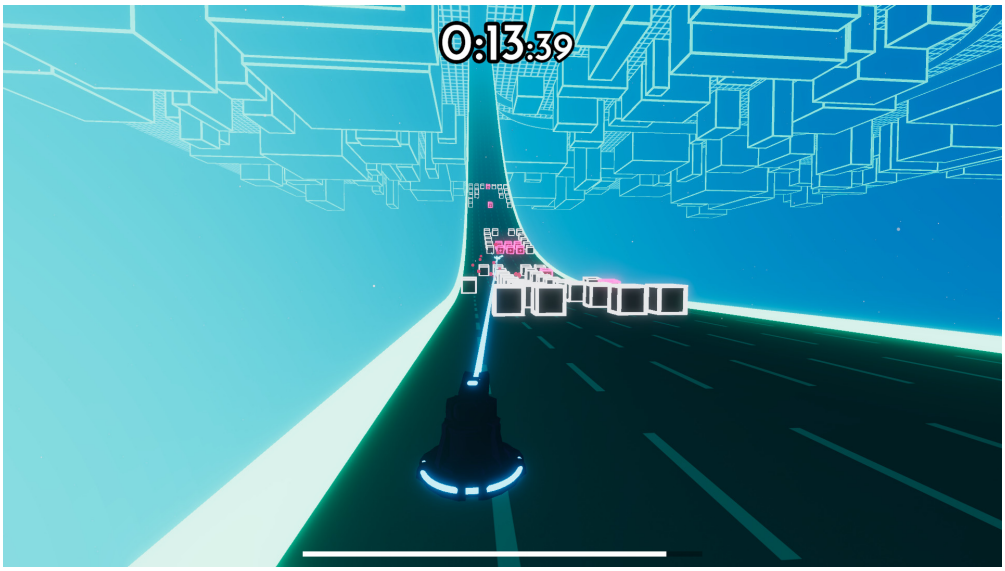


Figura 3.4: Schermata di gioco di Astral Slider [45].

3.3.4 Google Earth VR

L'ultima applicazione utilizzata rappresenta la versione VR della popolare applicazione di visualizzazione del mondo: Google Earth.
L'applicazione permette all'utente di essere un osservatore volante sopra al pianeta, a quasi ogni distanza da esso. Quando si è abbastanza vicini le immagini della Terra vengono proposte in modo tridimensionale, anche se queste sono semplicemente adattamenti delle immagini bidimensionali scattate dal satellite ad un modello altimetrico della Terra, e quindi la qualità di queste ultime risulta abbastanza scarsa in certi punti. Si nota anche che le immagini sono caricate ad una qualità migliore man mano che ci si avvicina alla visuale del giocatore, mentre diminuiscono in qualità con la maggiore distanza. Questo fenomeno alle volte si presenta anche nelle zone vicine non appena si cambia posto di visualizzazione, ma questo viene riportato alla normalità dopo qualche secondo, tempo in cui le immagini vengono caricate ad una qualità superiore.
Per cercare di migliorare le prestazioni abbiamo impostato la grafica al massimo e aumentato la frequenza di aggiornamento da 90 a 120 Hz, anche se questo non ha determinato un miglioramento sostanziale della qualità dell'esperienza, ma solo marginale.

L'applicazione permette anche di utilizzare la funzione Street View, in cui l'utente può visualizzare le zone stradali come se fosse un pedone e muoversi liberamente, seppure sempre attenendosi alla griglia stradale, e guardarsi attorno. In questo caso il movimento non è di tipo classico, ma è limitato allo spostamento in punti fissi tramite il controller, analogamente alla versione non VR dell'applicazione. In questa modalità le immagini presentate all'utente sono vere e proprie fotografie e quindi la qualità è ottima.

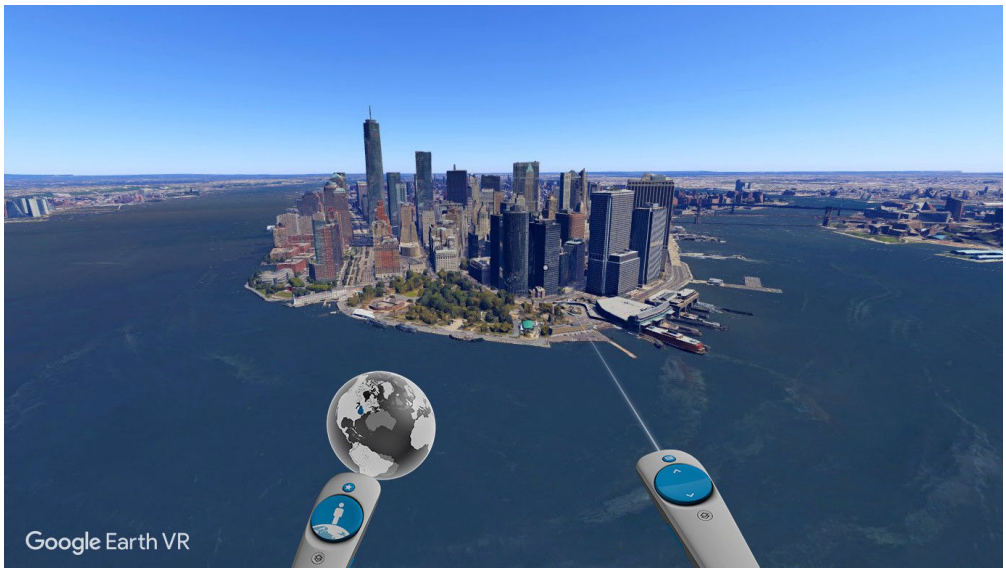


Figura 3.5: Schermata di Google Earth VR [46].

3.3.5 Tabella dei parametri

Di seguito viene riportata una tabella riassuntiva dei parametri utilizzati per ogni cattura, e dei rispettivi tempi.

Applicazione	Spider-Man: Homecoming VR	Cactus Cowboy - Plants at War	Astral slider	Google Earth VR
Durata	196 s	668 s	351 s	263 s
Frequenza di aggiornamento	90 Hz	90 Hz	90 Hz	90-120 Hz
Numero di frame catturati	840210	3095858	1594489	833311
Totale dei dati scambiati	3,071 GB	10,198 GB	5,781 GB	2,926 GB
Gradi di libertà	3	6	3	3
Tipologia di gioco	D'azione	FPS D'azione	Arcade	Simulazione

Tabella 3.1: Tabella dei parametri delle catture

Capitolo 4

Analisi dei dati sperimentali

In questa sezione verranno presentati i dati catturati, e successivamente analizzati con Wireshark e poi elaborati con MATLAB [47].
Di seguito viene riportata una schermata di Wireshark, con dei pacchetti catturati.

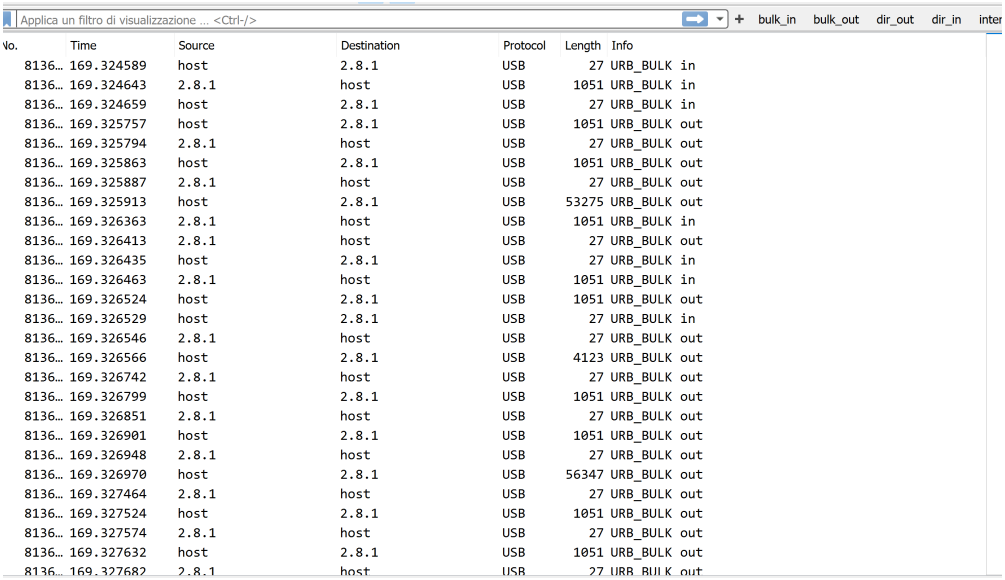
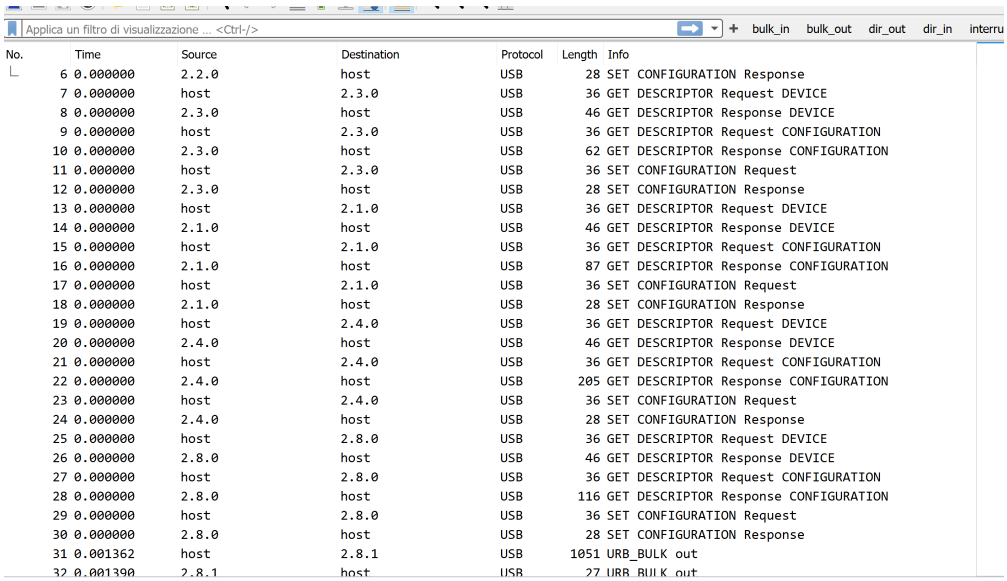


Figura 4.1: Schermata di Wireshark con alcuni pacchetti catturati

Si nota subito la grande quantità di dati scambiati, perché la schermata rappresenta un intervallo di solo 3,1 ms.
Wireshark presenta i dati ordinati temporalmente in base all'inizio della cattura, e quindi mantiene, per ogni pacchetto, il tempo passato dall'inizio, ma anche il tempo assoluto sotto forma di Unix Epoch Time. Successivamente si nota come Wireshark tenga conto anche del dispositivo che ha originato e di quello che ha ricevuto ogni pacchetto. In questo caso "host" rappresenta il computer mentre "2.8.1" è l'indirizzo di bus USB, dispositivo ed endpoint associati al visore.
La colonna successiva esplicita il protocollo utilizzato, che nel nostro caso è sempre USB, tranne nel caso di errori come malformazione di pacchetti, in cui viene usato il protocollo THRIFT.
La colonna length indica la dimensione dei frame catturati. Si nota subito che questi sono principalmente di dimensione 27 e 1051 byte, di questo se ne parlerà ampiamente nella sezione sulla lunghezza dei pacchetti.

L'ultima colonna rappresenta le informazioni su ogni frame catturato, notiamo che la maggioranza di essi è "USB_BULK in" e "USB_BULK out", che rappresentano come già discusso nel capitolo sul protocollo USB, il metodo più veloce per trasmettere grandi quantità di dati nelle versioni USB 3.0 SuperSpeed e superiori. La direzionalità si riferisce sempre all'host, cioè al computer. Quindi "USB_BULK in" si riferisce alle transazioni da visore a computer e "USB_BULK out" a quelle da computer a visore.
Queste due tipologie non rappresentano la totalità dei frame, infatti all'inizio della cattura sono presenti 30 frame di configurazione del dispositivo che rappresentano la transazione di setup, gestita tramite pacchetti di tipo di controllo. La figura sottostante rappresenta i pacchetti suddetti, come catturati da Wireshark.


Applica un filtro di visualizzazione ... <Ctrl-/>

bulk_in bulk_out dir_out dir_in interru

No.	Time	Source	Destination	Protocol	Length	Info
6	0.000000	2.2.0	host	USB	28	SET CONFIGURATION Response
7	0.000000	host	2.3.0	USB	36	GET DESCRIPTOR Request DEVICE
8	0.000000	2.3.0	host	USB	46	GET DESCRIPTOR Response DEVICE
9	0.000000	host	2.3.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
10	0.000000	2.3.0	host	USB	62	GET DESCRIPTOR Response CONFIGURATION
11	0.000000	host	2.3.0	USB	36	SET CONFIGURATION Request
12	0.000000	2.3.0	host	USB	28	SET CONFIGURATION Response
13	0.000000	host	2.1.0	USB	36	GET DESCRIPTOR Request DEVICE
14	0.000000	2.1.0	host	USB	46	GET DESCRIPTOR Response DEVICE
15	0.000000	host	2.1.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
16	0.000000	2.1.0	host	USB	87	GET DESCRIPTOR Response CONFIGURATION
17	0.000000	host	2.1.0	USB	36	SET CONFIGURATION Request
18	0.000000	2.1.0	host	USB	28	SET CONFIGURATION Response
19	0.000000	host	2.4.0	USB	36	GET DESCRIPTOR Request DEVICE
20	0.000000	2.4.0	host	USB	46	GET DESCRIPTOR Response DEVICE
21	0.000000	host	2.4.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
22	0.000000	2.4.0	host	USB	205	GET DESCRIPTOR Response CONFIGURATION
23	0.000000	host	2.4.0	USB	36	SET CONFIGURATION Request
24	0.000000	2.4.0	host	USB	28	SET CONFIGURATION Response
25	0.000000	host	2.8.0	USB	36	GET DESCRIPTOR Request DEVICE
26	0.000000	2.8.0	host	USB	46	GET DESCRIPTOR Response DEVICE
27	0.000000	host	2.8.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
28	0.000000	2.8.0	host	USB	116	GET DESCRIPTOR Response CONFIGURATION
29	0.000000	host	2.8.0	USB	36	SET CONFIGURATION Request
30	0.000000	2.8.0	host	USB	28	SET CONFIGURATION Response
31	0.001362	host	2.8.1	USB	1051	URB_BULK out
32	0.001390	2.8.1	host	USB	27	URB_BULK out

Figura 4.2: Schermata di Wireshark con i pacchetti di configurazione iniziali

È interessante vedere come questa procedura richieda esattamente 30 frame in ognuna della nostre catture, quindi si può notare come questa rappresenti una sequenza standardizzata. Altro dato interessante è il fatto che i 30 frame vengano scambiati in meno di 1μs, infatti il software non è in grado di misurare la durata effettiva della procedura con la precisione dell'etichetta temporale.
Durante lo scambio dei pacchetti si nota il fenomeno descritto nel capitolo 2 in cui il dispositivo prima ha indirizzo 0, poi viene enumerato, poi l'host richiede i descrittori per ogni livello (dispositivo, interfaccia, endpoint) e man mano che riceve le informazioni configura l'indirizzo completo del dispositivo, fino ad arrivare a "2.8.1".
Sono presenti anche alcuni frame gestiti a interrupt, ma questi rappresentano una forte minoranza e si ipotizza che gestiscano gli input derivanti dai controller.

Come già discusso in precedenza, questa sezione di analisi non ha lo scopo di andare nel dettaglio dei singoli pacchetti, ma si sofferma ad analizzare i frame catturati da Wireshark con USBPcap, questi rappresentano comunque un dato interessante perché rende l'idea della distribuzione statistica dei pacchetti reali.
L'analisi si soffermerà in alcuni punti chiave per capire il traffico scambiato:

- si parlerà innanzitutto di throughput, ovvero della quantità pura di dati scambiati dai dispositivi lungo il collegamento USB, differenziandolo poi per applicazione e per direzione;

- la seconda analisi si concentrerà sulla lunghezza dei pacchetti, soffermandosi sul concetto accennato nella sezione sul capitolo USB di bursting, funzionalità chiave e tipica dei protocolli USB 3.0 SuperSpeed e superiori, che ci permetterà di interpretare meglio i grafici della prima sezione dell'analisi sul throughput.
- la terza analisi si soffermerà sul tempo inter-frame, anche in questo caso si rimarcherà il concetto di bursting e si vedrà come la specifica SuperSpeed permette ai frame di essere inviati con una frequenza notevolmente maggiore rispetto agli standard precedenti.

4.1 Throughput e Input/Output

In questa sezione parleremo della quantità di dati scambiati.
Come detto prima divideremo poi i dati per applicazione e direzione: la differenziazione per direzione può essere fatta in due modi diversi:

- La prima è fatta suddividendo i frame in base a chi ha generato il frame e chi lo ha ricevuto;
- La seconda invece suddivide i frame in base al tipo di transazione. Come discusso nel capitolo sul protocollo USB, infatti, sappiamo che le transazioni di IN o OUT, che logicamente rappresentano un trasferimento dati in input o output rispetto all'host, comprendono in realtà anche pacchetti nell'altra direzione: per esempio nella transazione di input oltre a pacchetti di dati con direzione da dispositivo a host, sono presenti anche dei pacchetti che vanno dall'host al dispositivo come il primo token e l'acknowledgment finale. È bene notare che sebbene la numerosità di questi pacchetti con direzione contro intuitiva sia notevole (circa il 50%), questi non rappresentano una grossa mole di dati, in quanto i suddetti pacchetti sono solo di poche decine di Byte, in confronto alle migliaia dei pacchetti di dati, e quindi il loro contributo al throughput non sarà predominante.

4.1.1 Throughput

La seguente analisi ha lo scopo di visualizzare la grossa mole di dati scambiati tra visore e computer. Non essendo questa divisa in alcun modo, si può apprezzare solamente la diversa quantità di dati scambiata durante le fasi di gioco, nonché la differenza tra le varie applicazioni. Quest'ultimo punto verrà discusso alla fine quando si compareranno i dati delle 4 applicazioni in un unico grafico.

Di seguito vengono riportati i grafici relativi ad ogni singola cattura. In ciascuno sono riportati i dati originali finestrati ogni secondo, la loro regressione polinomiale di ordine 10 e la loro media totale. La scelta di fare una regressione polinomiale di ordine 10 è dettata dal fatto che i dati originali si presentano con molte oscillazioni, che rendono difficile la visione d'insieme. La regressione in questo senso aiuta a vedere l'andamento generale del throughput. Si noti che l'ordine della regressione è stato scelto empiricamente, in modo da rendere bene l'andamento senza allo stesso tempo seguire troppo le oscillazioni.
La media totale, invece, ci aiuta a confrontare le applicazioni tra loro, classificandole in base alla mole di dati che si scambia. Questo dato è significativo perché tutte le catture sono state fatte su un intervallo di tempo abbastanza ampio da riuscire a testare tutte le funzionalità di ogni applicazione, e quindi su un intervallo di tempo variabile dai 3 ai 12 minuti circa.

4.1.1.1 Spider-Man: Homecoming VR

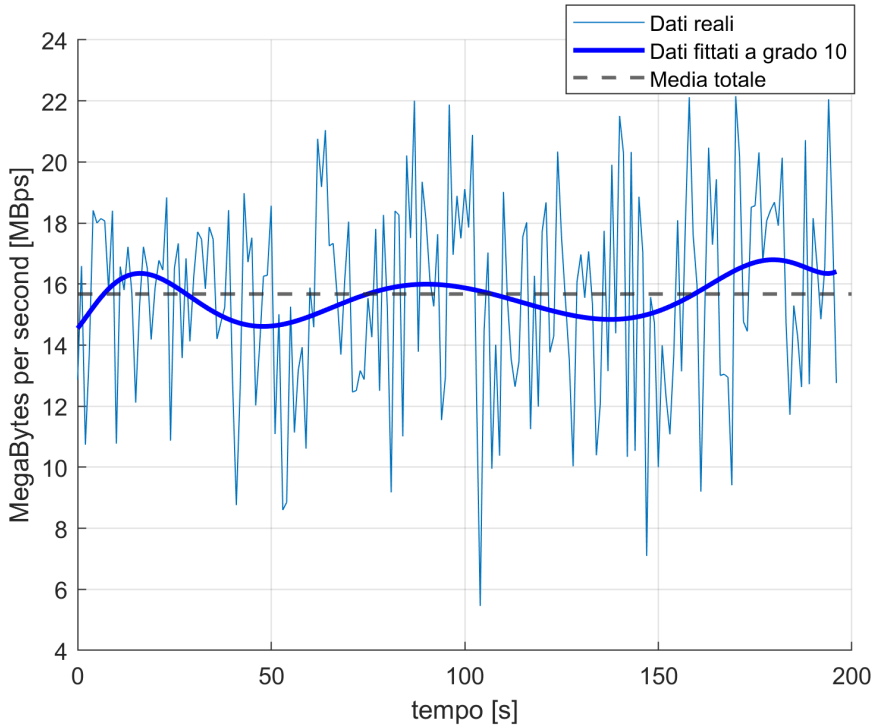


Figura 4.3: Throughput delle catture di Spider-Man: Homecoming VR

In questo grafico possiamo notare che il traffico è abbastanza costante, in media, con alcune variazioni maggiori in corrispondenza delle varie fasi di gioco: i tre picchi più alti sono riconducibili alle tre cinematiche che il gioco presenta al giocatore, prima all'inizio in cui si presenta un tutorial sui comandi di base; poi circa a metà in cui si presenta nuovamente un tutorial, questa volta su controlli più avanzati come spari di ragnatele e bombe; e in ultimo in corrispondenza della fine del gioco quando viene presentato il finale in sospeso che porta l'attenzione verso il film. Quest'ultima cinematica è leggermente più pesante delle altre in quanto ci sono molti oggetti in movimento e le scene in generale sono più complesse.

4.1.1.2 Cactus Cowboy - Plants at War

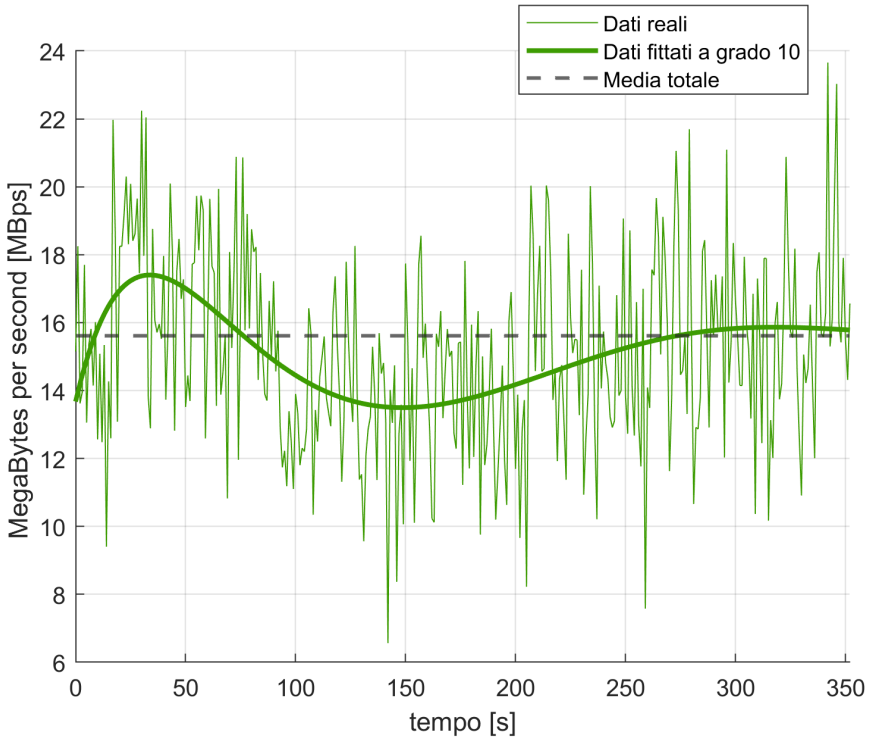


Figura 4.4: Throughput delle catture di Cactus Cowboy - Plants at War

Questa figura fa notare un picco iniziale dei dati trasmessi, seguito da un declino abbastanza sostanzioso per poi arrivare ad un ultimo leggero aumento alla fine.
Si ipotizza che il picco iniziale sia dovuto alla cinematica di presentazione del gioco, di durata circa un minuto. Questa cinematica è più pesante del gioco in sé perché è più curata dei dettagli.
Il seguente declino dei dati trasmessi può essere riconducibile all'inizio del gioco vero e proprio, in cui come detto in precedenza, la grafica non è molto pesante. Durante questa fase si attraversa un tutorial che man mano spiega i comandi al giocatore, le scene non sono quindi di complessità elevata e gli oggetti in movimento sono pochi.
Il lieve rialzo finale è invece riconducibile alla battaglia finale, in cui la scena diventa ben più complessa, e con un numero elevato di oggetti e particelle in movimento come proiettili, fuoco e fumo.

4.1.1.3 Astral Slider

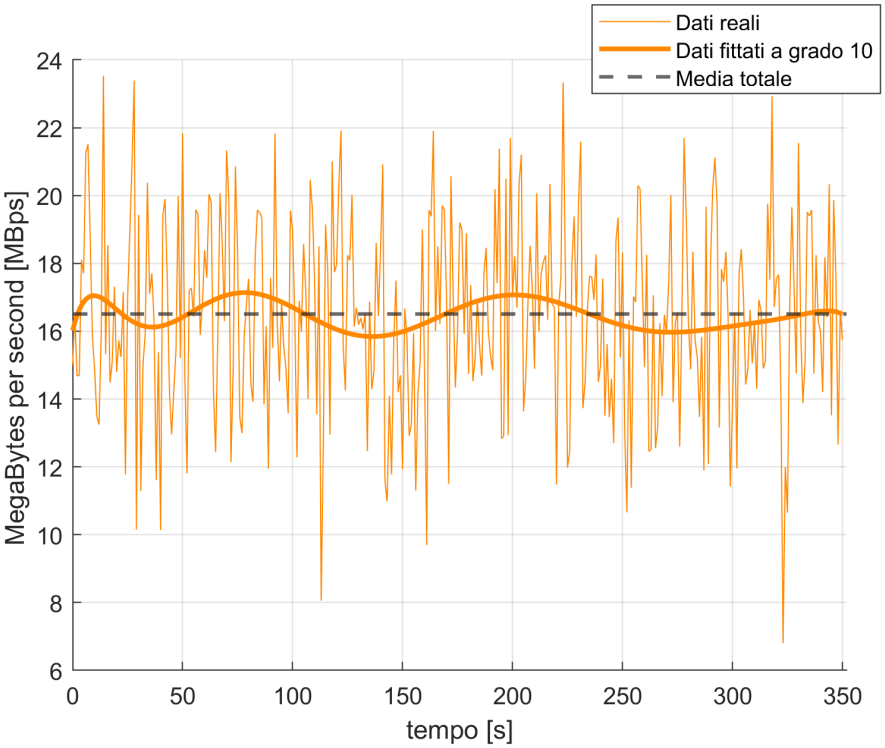


Figura 4.5: Throughput delle catture di Astral Slider

Nella figura si possono distinguere 3 momenti di picco e 3 momenti di conca, in cui il traffico è relativamente minore. Questo perché, nella terza applicazione sono stati effettuati tre "round" di gioco, ciascuno intervallato da una fase di scelta del livello al menù.
Si nota facilmente come i tre livelli corrispondano a delle scene di movimento, con blocchi che si distruggono e molte particelle, mentre le scene del menù erano scene statiche con un video demo di alcuni livelli in sottofondo. Questo video in background si ipotizza essere la causa del fatto che durante queste fasi il traffico non diminuisca di un fattore considerevole, come ci si potrebbe aspettare.
In generale in questo grafico ci stupisce il valore relativamente alto di dati inviati, in quanto come già discusso la grafica del gioco non è particolarmente dettagliata. Ciò potrebbe allora essere dovuto al come il programma ottimizza i dati, per esempio potrebbe dover caricare ogni pezzo di mappa ogni volta che entra nel campo visivo, invece che tenerla in memoria per i successivi 2 giri della pista.

4.1.1.4 Google Earth VR

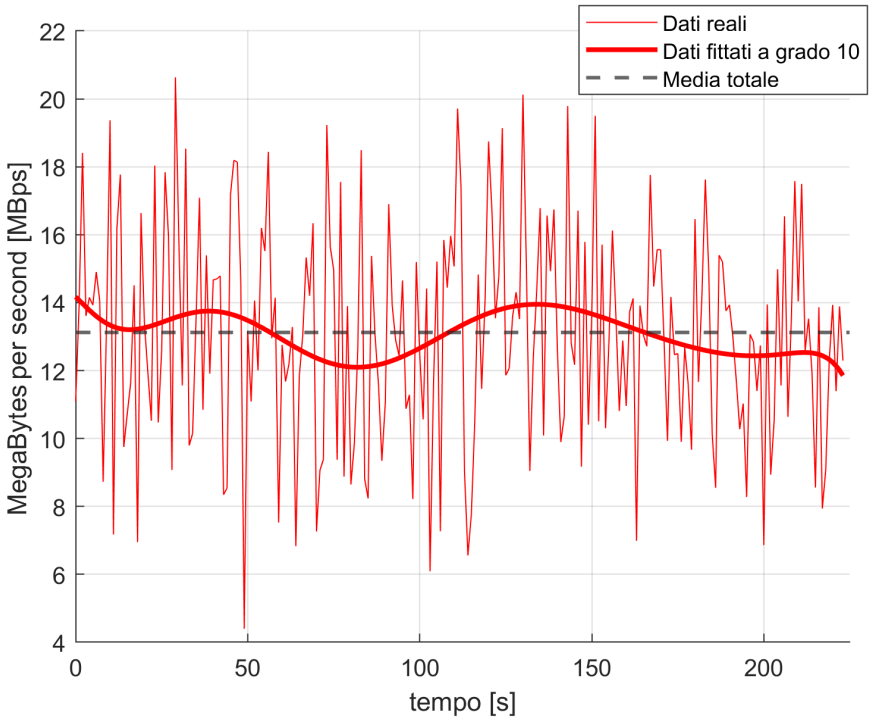


Figura 4.6: Throughput delle catture di Google Earth VR

Nella figura si notano delle oscillazioni iniziali, seguite da un calo temporaneo e poi seguito da un picco abbastanza notevole.
Si interpretano questi dati, ancora una volta associando i picchi ai momenti in cui si stavano caricando scenari, come immagini satellitari ad una distanza abbastanza vicina da poter identificare i singoli elementi come le case.
In questo caso la regressione polinomiale non aiuta ad apprezzare tutte queste fasi, ma mette bene in evidenza come la fase di StreetView, corrispondente alla parte di flusso dati minore (a circa 70 secondi), sia molto più leggera di quella di visione dall'alto.
Il picco principale (a 130 secondi), appare in corrispondenza del settaggio delle nuove impostazioni del visore: si passa da 90 a 120 Hz. Si nota subito come questo porti ad un notevole incremento del traffico, anche se questo tende a stabilizzarsi col passare del tempo.

4.1.1.5 Totale

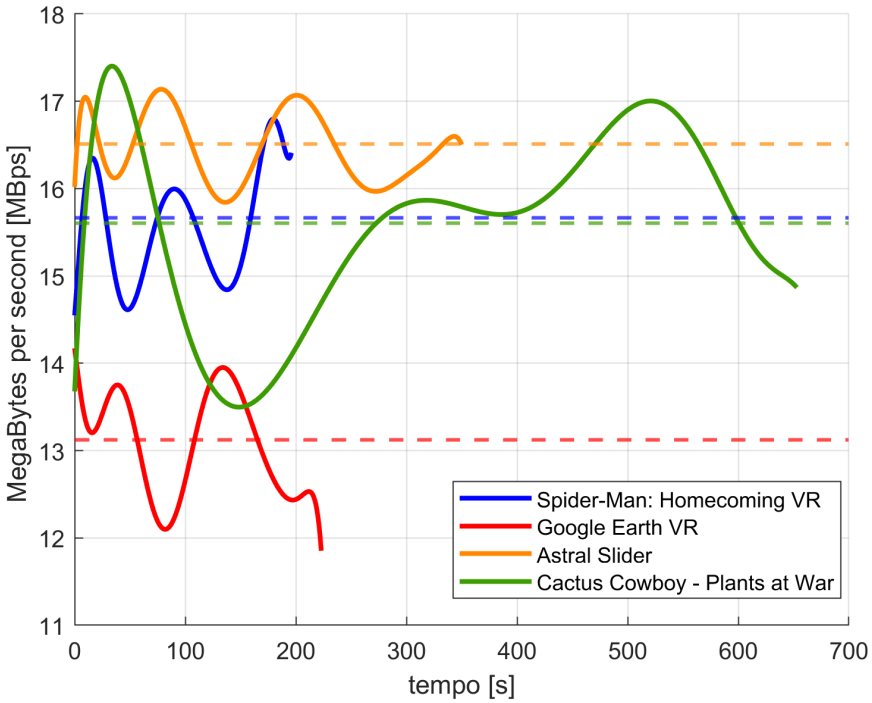
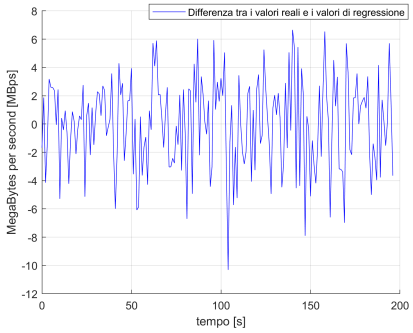


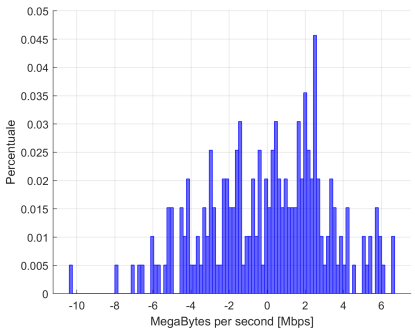
Figura 4.7: Confronto tra le regressioni e le medie delle varie applicazioni

La figura rappresenta il confronto tra le regressioni polinomiali dell'andamento del flusso dei dati per ogni applicazione, insieme alla media finale.
Si noti che, come detto in precedenza, le catture hanno durata differente, quindi quelle che durano meno sono semplicemente messe nel grafico fino alla loro fine.
Il grafico ci permette innanzitutto di classificare i vari programmi in base alla loro pesantezza: Astral slider risulta essere notevolmente superiore agli altri, con quasi 1MB/s in più rispetto al secondo e al terzo, nonostante la sua grafica semplice e geometrica. Per questo programma si nota però come il flusso sia più costante rispetto alle altre applicazioni, con una variazione tra massimo e minimo di poco più di 1MB, mentre negli altri casi siamo intorno ai 2MB e per Cactus Cowboy - Plants at War intorno a 3,5MB.
Si nota anche che le variazioni di flusso di quest'ultimo programma sono ben più marcate di quanto sembrasse dal grafico della singola applicazione, e si può veramente apprezzare la differenza tra i vari scenari (cinematiche, scena di gioco tutorial, scena di gioco complicata).
Un altro fattore che si può notare dal grafico è che Google Earth VR abbia una media di traffico dati decisamente minore delle altre applicazioni. Questo, probabilmente, è dovuto al fenomeno di caricamento graduale delle immagini, in cui vengono mostrate al giocatore con qualità man mano superiore nel giro di qualche secondo, ammortizzando il peso delle immagini ad alta risoluzione in un intervallo di tempo più alto. Questa opzione riduce notevolmente il traffico medio, anche se diminuisce leggermente la qualità dell'esperienza nei momenti intermedi, in cui le immagini sono caricate a qualità media.

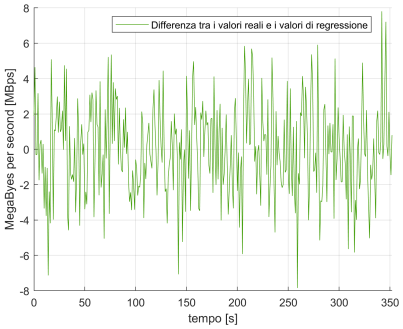
Di seguito vengono riportati i grafici che mostrano la differenza tra i dati reali e quelli della regressione (il residuo), e successivamente i relativi istogrammi per visualizzarne la distribuzione.



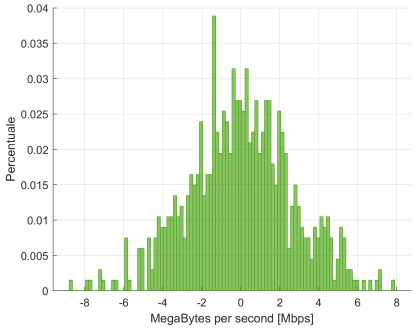
(a) Residuo di Spider-Man: Homecoming VR



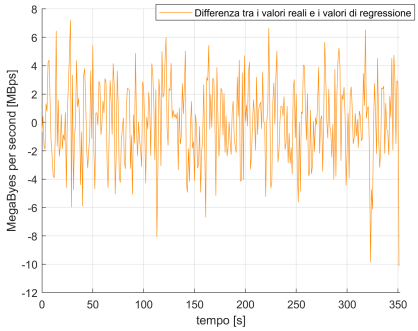
(b) Distribuzione di probabilità per il residuo di Spider-Man: Homecoming VR



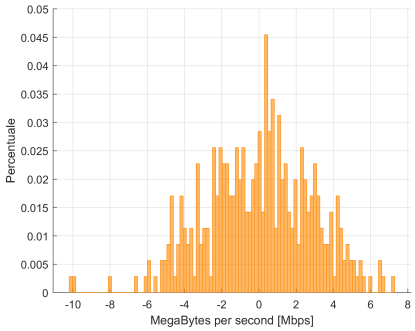
(c) Residuo di Cactus Cowboy - Plants at War



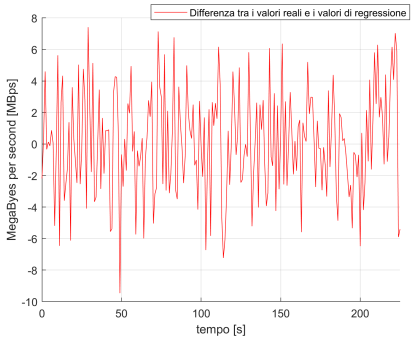
(d) Distribuzione di probabilità per il residuo di Cactus Cowboy - Plants at War



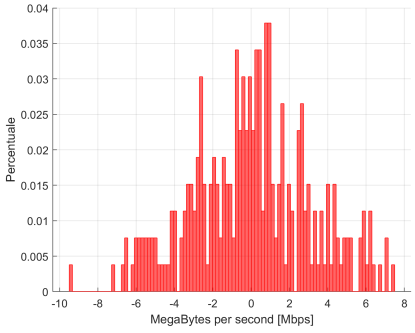
(e) Residuo di Astral Slider



(f) Distribuzione di probabilità per il residuo di Astral Slider



(g) Residuo di Google Earth VR



(h) Distribuzione di probabilità per il residuo di Google Earth VR

Figura 4.8: Analisi della differenza tra dati reali e quelli di regressione

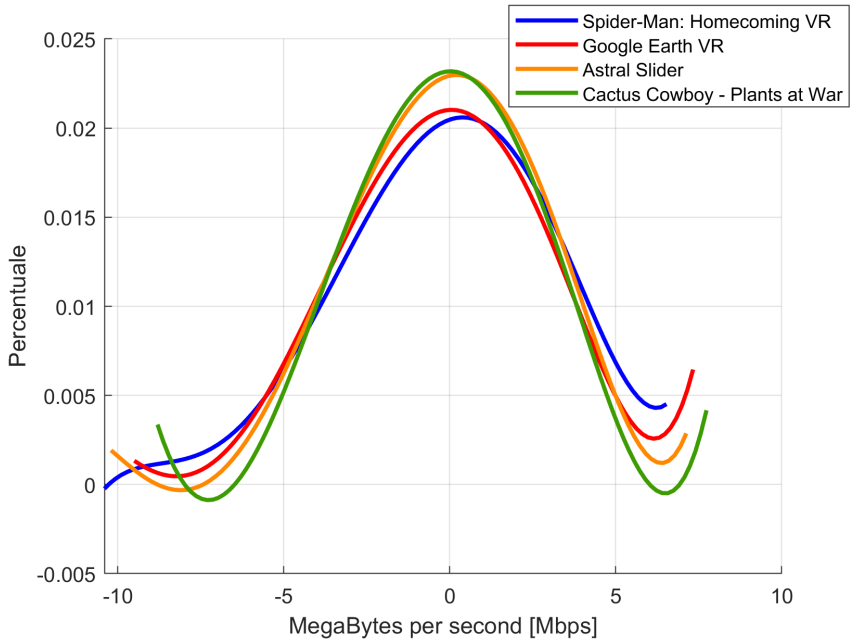
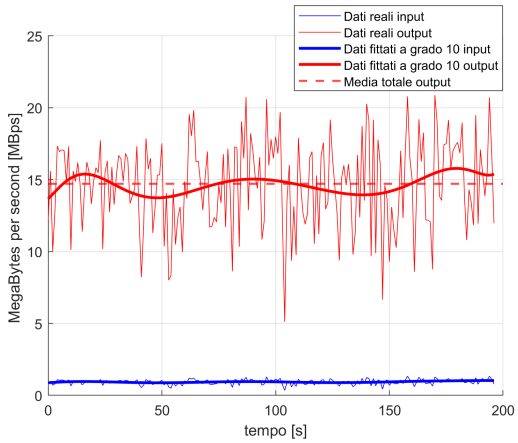


Figura 4.9: Confronto tra le regressioni di ordine 5 delle distribuzioni delle quattro applicazioni

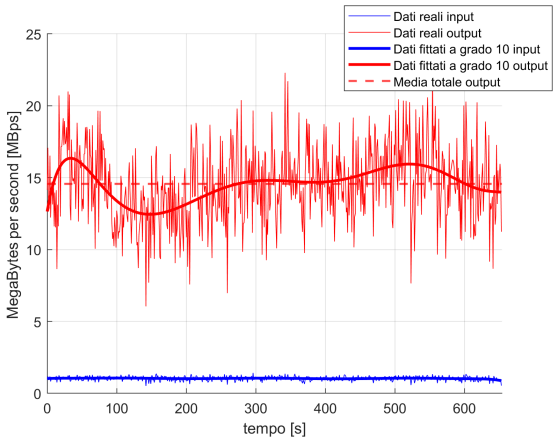
Le figure 4.8(a), 4.8(c), 4.8(e) e 4.8(g) evidenziano che la variazione continua ad avere un andamento circa costante e con la stessa distribuzione dei dati reali. Alcuni momenti di lunghezza di qualche secondo si presentano come tutti positivo o negativi, ovvero in quel intervallo temporale i dati reali sono tutti maggiori, o minori rispettivamente, dei valori della regressione. Questi momenti possono essere riconducibili ad alcuni momenti di gioco specifici.
Ad esempio nel caso della figura 4.8(a) tra 60 e 65 o tra 90 e 100 secondi circa, il picco positivo è riconducibile ad una cinematica che, come detto nel paragrafo precedente, aumenta il traffico totale. Lo stesso fenomeno lo si può individuare in figura 4.8(c) tra 20 e 30 o tra 210 e 220 secondi circa. Non ci sono intervalli positivi apparenti e riconducibili a questo fenomeno negli altri due programmi testati, perché questi ultimi non avevano al loro interno alcuna cinematica.
Al contrario gli intervalli in cui la variazione è negativa corrispondono a periodi in cui il traffico è particolarmente più basso rispetto al valore aspettato. Ad esempio l'intervallo da 320 a 325 secondi della figura 4.8(e), rappresenta un momento in cui si era ad un menù di gioco.
Gli istogrammi in figura 4.8(b), 4.8(d), 4.8(f) e 4.8(h), e il confronto in figura 4.9 dimostrano che in tutti e quattro i casi, la variazione dei dati è casuale e molto simile, infatti ha una distribuzione riconducibile ad una normale. Questo è dovuto alla forte variabilità dei dati e dal fatto che i momenti di gioco particolari come i menù o le cinematiche fossero pochi e di breve durata.
Si noti che questo non sembra essere vero da alcuni istogrammi come quelli in figura 4.8(b), che sembra leggermente spostato verso destra. In realtà il fenomeno è dovuto alla scarsità di dati: essendo la cattura lunga solamente qualche minuto, i valori possono non rispecchiare la statistica in modo preciso.
Si noti anche che l'incremento della probabilità nei valori estremi è dovuto al fit polinomiale, e non è rispecchiato dai valori reali negli istogrammi.

4.1.2 Input-Output direzionale

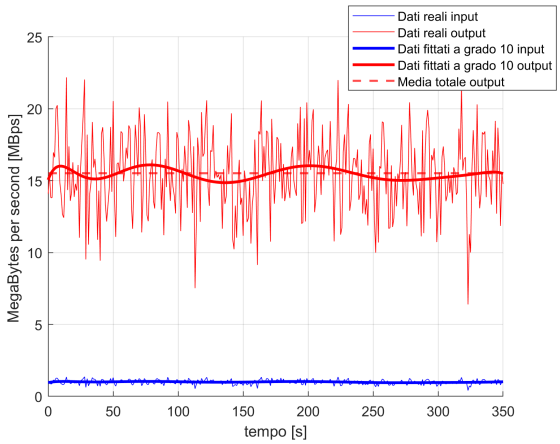
Questa sezione ha come scopo la differenziazione del traffico in base a chi lo ha generato. Per ottenere questo si sono filtrati i frame catturati da Wireshark con il filtro `usb.irp_info.direction`, che vale 0 se il traffico è da computer a visore, e 1 nel caso opposto.
L'analisi non si sofferma sul numero di frame scambiati, ma solo sulla dimensione di essi. Questo perché sarebbe inutile, visto che per ogni frame scambiato in input ne corrisponde uno in output, di acknowledgment. Questa considerazione segue direttamente dallo studio del funzionamento delle transazioni USB, in cui dallo standard 3.0 in poi, si può sempre eliminare la prima fase di scambio di token, che è incorporata nel pacchetto di dati, ma rimangono comunque il pacchetto di dati effettivo in una direzione e l'acknowledgment nella direzione opposta. Ne consegue che per ogni transazione sono presenti due frame: uno in input e l'altro in output, dove se si tratta di transazione IN, avremo il pacchetto dati in input e l'acknowledgment in output e viceversa se si tratta di una transazione OUT.



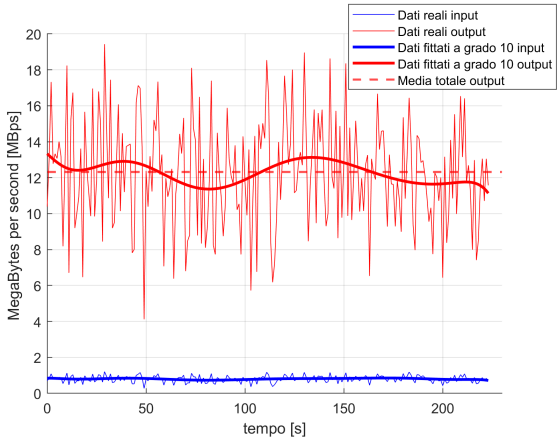
(a) Confronto tra Input e Output delle catture di Spider-Man: Homecoming VR



(b) Confronto tra Input e Output delle catture di Cactus Cowboy - Plants at War



(c) Confronto tra Input e Output delle catture di Astral Slider



(d) Confronto tra Input e Output delle catture di Google Earth VR

Figura 4.10: Confronto tra traffico in Input e traffico in Output nelle quattro applicazioni testate

Nei grafici possiamo notare come l'andamento del traffico di input (da visore a pc) sia di molto inferiore a quello di output (da computer a visore), inoltre il primo si presenta come

quasi costante durante tutto l'esperimento, mentre il secondo segue l'andamento del traffico totale (throughput), discusso nella sezione precedente.
In tutti e quattro i casi testati si sono presentate le stesse caratteristiche per il traffico di Input. In particolare, in tutti i casi questo si presenta come un traffico quasi costante di circa 1 MB/s.
Il traffico di output segue in tutti i casi l'andamento del throughput, e numericamente è diminuito di circa 1 MB/s, in quanto viene occupato dal traffico di input.
È inoltre interessante notare come la variazione della velocità di trasferimento rispetto da valore medio sia decisamente maggiore per il traffico di output, rispetto che al traffico di input, per il quale si presentano variazioni molto lievi.

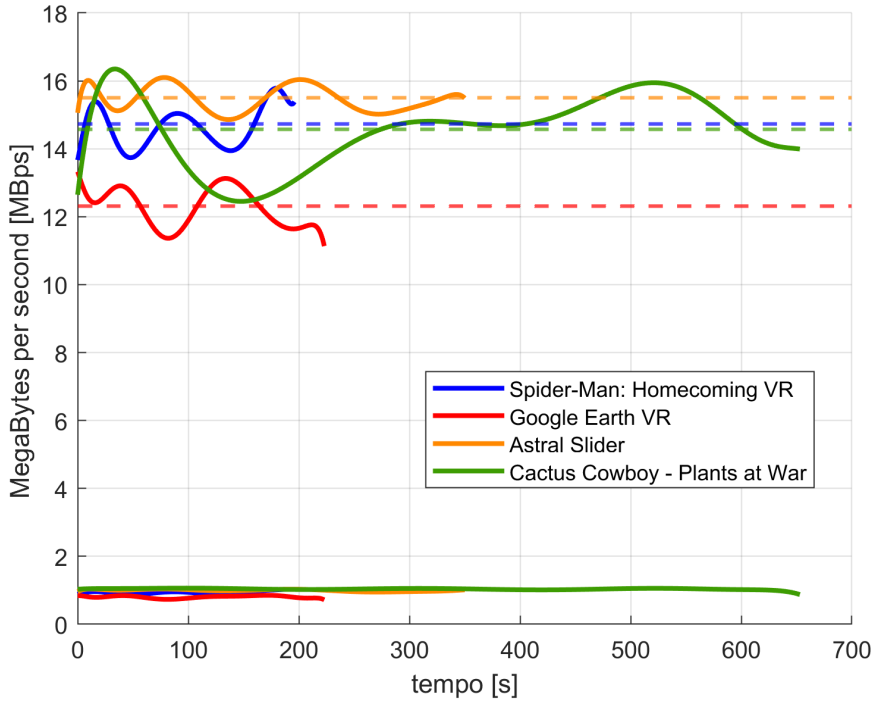


Figura 4.11: Confronto tra le regressioni di input e output e tra le medie delle varie applicazioni

Il grafico di confronto ci permette di confermare che il fenomeno osservato singolarmente di caso in caso, è identico per tutte e quattro le applicazioni. In particolare, per tutte il traffico di input rimane circa costante ad un valore di 1MB/s, mentre per il traffico di output valgono considerazioni analoghe a quelle fatte per il throughput.
Non è sorprendente notare che il traffico di input sia minore di quello di output, visto che il computer deve inviare al visore tutte le informazioni elaborate e di grande quantità, come tutte le immagini a video. Viceversa, dal visore al computer basta trasferire poche informazioni, principalmente per quanto riguarda la posizione del giocatore e la sua visuale, quindi i 6 gradi di libertà, e i comandi dei controller. Sebbene questa non sia una cosa inaspettata, la differenza tra i due traffici rappresenta comunque una differenza notevole, ben più alta di quella che ci si potrebbe aspettare.

4.1.3 Input-Output transazionale

Per quanto riguarda l'analisi del traffico, differenziandolo in base alle transazioni, non vengono riportati grafici, in quanto valgono considerazioni analoghe a quelle appena fatte. Le transazioni, come detto in precedenza, sono nel nostro caso composte da 2 frame: uno di dimensione elevata con i dati veri e propri; e uno di dimensione ridotta con l'acknowledgment. I pacchetti di acknowledgment sono di dimensione 27 byte, cioè la dimensione del solo header, mentre quelli di dati sono di dimensione variabile, a partire da 1051 byte (27 di header e 1024 di payload) fino a oltre 50000 byte. Per filtrare in base alle transazioni è sufficiente accorgersi di quello appena visto, cioè le transazioni di IN sono composte dai pacchetti di grandi dimensioni in direzione input e da quelli di dimensione 27 byte in direzione output, mentre per la transazione OUT si considerano solo i pacchetti di grandi dimensioni in direzione output e quelli di 27 byte in dimensione input. Per filtrare in base a questo con Wireshark è sufficiente legare le due condizioni, formando il comando `(usb.irp_info.direction == 1 && frame.len < 28) || (usb.irp_info.direction == 0 && frame.len > 1050)` per le transazioni bulk in e `(usb.irp_info.direction == 0 && frame.len < 28) || (usb.irp_info.direction == 1 && frame.len > 1050)` per le transazioni bulk out.
Per il motivo appena descritto, l'input-output transazionale si distingue da quello direzionale solamente per i pacchetti di 27 byte, che sono considerati da una parte di input direzionale e dall'altra di output transazionale e viceversa. Ne consegue quindi che questa differenza viene bilanciata dal fatto che la numerosità di questi pacchetti sia quasi uguale: circa il 60% dei pacchetti sono bulk out e il 40% di bulk in. La differenza di traffico in byte tra queste due divisioni è quindi irrisoria e non visibile nei grafici per colpa dell'ordine di grandezza.

4.2 Lunghezza dei pacchetti

In questa sezione ci concentriamo sull'analisi della lunghezza dei frame catturati, in particolare analizzeremo le statistiche di frequenza per ogni lunghezza e compareremo le probabilità di un frame di essere di una determinata lunghezza per ogni applicazione.

Come già accennato in precedenza, la maggioranza dei frame catturati rientra in due categorie: quelli lunghi 27 byte e quelli lunghi 1051 byte.
Tutti i frame catturati contengono l'header, di lunghezza 27 byte e con struttura descritta dall'immagine sottostante.

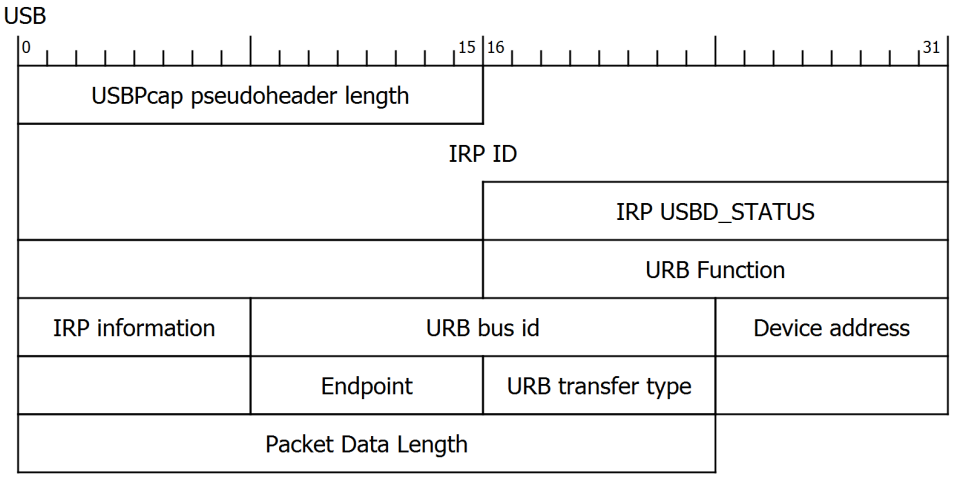


Figura 4.12: Struttura di un Frame USB catturato da Wireshark

Si nota subito che questo header è stato modificato dall'applicazione utilizzata per catturare i pacchetti, anche se al suo interno rimangono comunque tutte le informazioni che c'erano in precedenza, solo in ordine diverso. Quindi per la nostra analisi lo consideriamo come un frame normale non modificato.

Tutti i pacchetti non da 27 byte, e quindi che non contengono solamente l'header, contengono anche il payload di dati. Quest'ultimo è sempre di dimensione massima per ogni pacchetto, cioè di 1024 byte per la tecnologia SuperSpeed. I frame catturati non corrispondono ai pacchetti USB, bensì comprendono una moltitudine di essi: alle volte uno solo e altre fino a diverse centinaia. Questo fenomeno ci permette di visualizzare molto bene la tecnica di bursting, tipica di USB 3.0 SuperSpeed, e che permette velocità di trasferimento così elevate. Analizzando il traffico da Wireshark si nota infatti come ogni frame di dati sia intervallato da un frame da 27 byte di acknowledgment, ma il frame di dati stesso ha dimensione diversa. Questo fa capire che il protocollo non prevede un acknowledgment per ogni pacchetto di dati inviato (da 1024 byte), ma è permesso inviare più pacchetti di dati in contemporanea, per poi ricevere un unico acknowledgment. Questo è il fenomeno di bursting di cui si parlava nella sezione 2.4.3.5.
È chiaro allora che questo comportamento implica che i frame di dati siano sempre composti da multipli interi di pacchetti di dati da 1024 byte. Dai dati raccolti, si riesce facilmente a confermare questa cosa, guardando le lunghezze dei frame catturati su Wireshark, che si presentano sempre della forma `(27 + n*1024)` byte, dove `n` è un numero intero non negativo. Per ogni applicazione verificheremo poi, con un'analisi più approfondita, che questo sia valido effettivamente per ogni frame.

Nelle sezioni successive verranno presentati, per ogni applicazione testata, degli istogrammi, che riportano la frequenza di ogni pacchetto in base alla sua lunghezza. In questi istogrammi è stata scelta come grandezza di ogni intervallo 1024 byte, in modo che ciascun intervallo contenesse solo uno dei valori possibili delle lunghezze dei frame. In questo modo gli istogrammi ci forniscono un conto esatto dei frame per ogni lunghezza, senza alcuna incertezza sull'intervallo.
Per ogni applicazione poi verrà mostrato che il fenomeno descritto poco sopra sulla lunghezza esatta dei frame è valido, riportando un ingrandimento dell'istogramma delle frequenze, a cui è stata diminuita radicalmente la grandezza di ogni intervallo, per mostrare che non ci

sono frame con lunghezze intermedie.
Successivamente per ciascuna verrà estratta la probabilità per ogni frame di essere di una determinata lunghezza. Questo ci permetterà di confrontare alla fine le varie applicazioni senza essere influenzati dalla durata delle catture, in quanto questa aumenterà i dati sulle frequenze, ma non quelli sulle probabilità.

4.2.1 Spider-Man: Homecoming VR

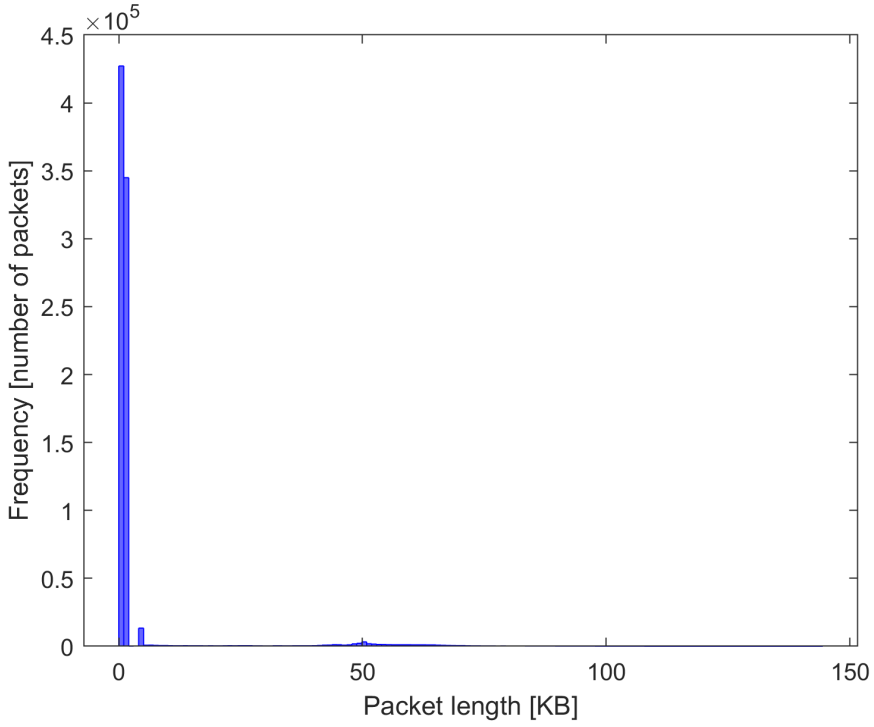


Figura 4.13: Istogramma delle frequenze dei frame catturati, in base alla loro lunghezza per Spider-Man: Homecoming VR

L'istogramma mostra chiaramente che i pacchetti da 27 e 1051 byte sono di gran lunga i più comuni. Infatti le due barre che li rappresentano sono notevolmente più alte delle altre, che quasi non si vedono.
Si nota però che le successive frequenza maggiori sono a circa 5KB e 50KB, dimostrando immediatamente che non c'è una distribuzione di tipo esponenziale, come ci si potrebbe aspettare.

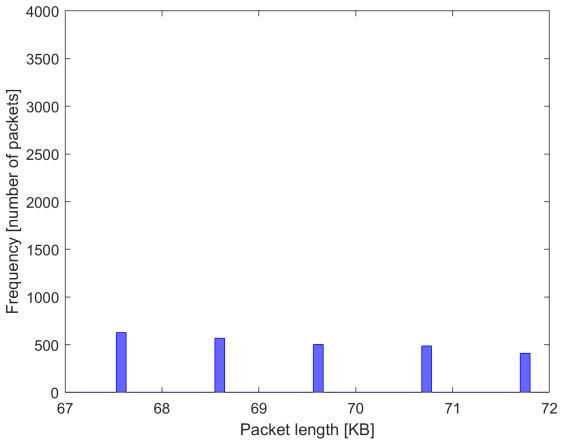
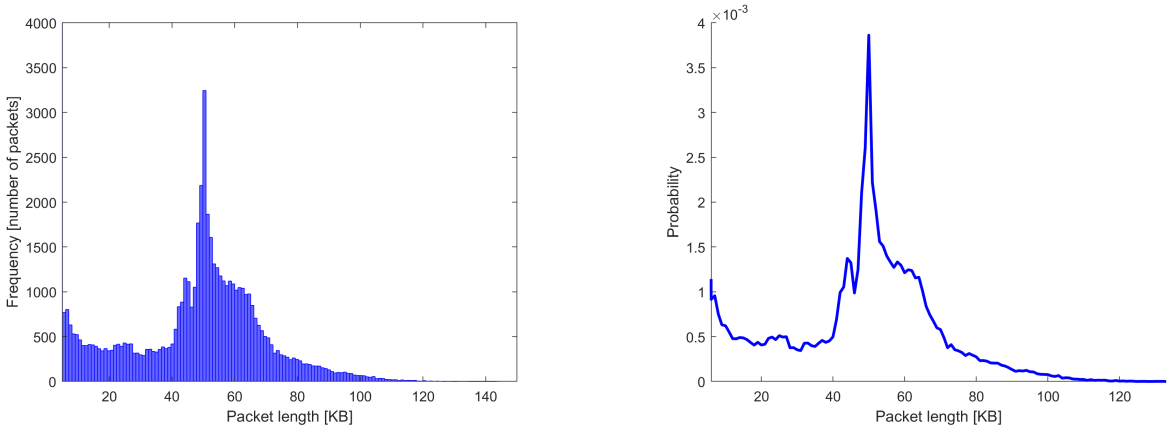


Figura 4.14: Dettaglio dell'istogramma precedente, con larghezza delle classi diminuita

La precedente immagine rappresenta l'istogramma delle frequenze sopra mostrato, con attenzione in un intervallo scelto casualmente. Nell'istogramma sono state ridotte le larghezze delle classi, di un fattore maggiore dei 10.
La figura serve a dimostrare che in questo caso non ci sono pacchetti di lunghezza intermedia ai valori ipotizzati all'inizio di questa sezione. In particolare, nella figura si notano picchi solamente in posizione relativa alle lunghezze, in byte, di 67611, 68635, 69659, 70683 e 71707, corrispondenti ad un valore di `n` di 66, 67, 68, 69 e 70.



(a) Istogramma delle frequenze, a partire da 5 KB

(b) Distribuzione di probabilità per le lunghezze dei pacchetti

Figura 4.15: Dettaglio dell'istogramma delle frequenze, troncato per eliminare i tre picchi principali

La figura 4.15(a) ha lo scopo di mostrare la distribuzione dei frame catturati, in base alla loro lunghezza, in cui sono stati eliminati i primi tre picchi per riuscire a vedere in dettaglio il resto del grafico.
La prima figura mostra quello che si intuiva dall'istogramma completo, ovvero che l'altro valore più comune per la lunghezza dei frame è intorno ai 50 KB, con una distribuzione tutt'altro che morbida. Si noti anche che i frame con lunghezze molto superiori, cioè oltre ai 120 KB, sono molto rari, e spesso la loro frequenza è contenuta in una decina per tutta la cattura.

La figura 4.15(b) mostra la distribuzione di probabilità per ogni lunghezza. In altre parole per ogni intervallo si è diviso il valore della classe per il numero totale di frame catturati, in modo da ottenere una percentuale. Come è normale aspettarsi, questo secondo grafico mantiene le esatte proprietà di forma del primo, riscalandolo solamente sull'asse y, in modo da essere normalizzato rispetto a 1.

Di seguito viene riportata una tabella per i valori di probabilità delle parti tagliate dal grafico della figura 4.15(b).

Lunghezza [byte]	Percentuale dei pacchetti
27	50,84%
1051	41,05%
2075	0,01%
3099	0,03%
4123	1,59%
5147 e superiori	6,48%

Tabella 4.1: Distribuzione in percentuale dei pacchetti per le misure di Spider-Man: Homecoming VR

Si nota quindi che il grafico in dettaglio della figura 4.15 rappresenta la distribuzione di appena il 6,5% dei pacchetti. Inoltre, si può notare come il terzo picco più alto sia in corrispondenza di 4123 byte.

4.2.2 Cactus Cowboy - Plants at War

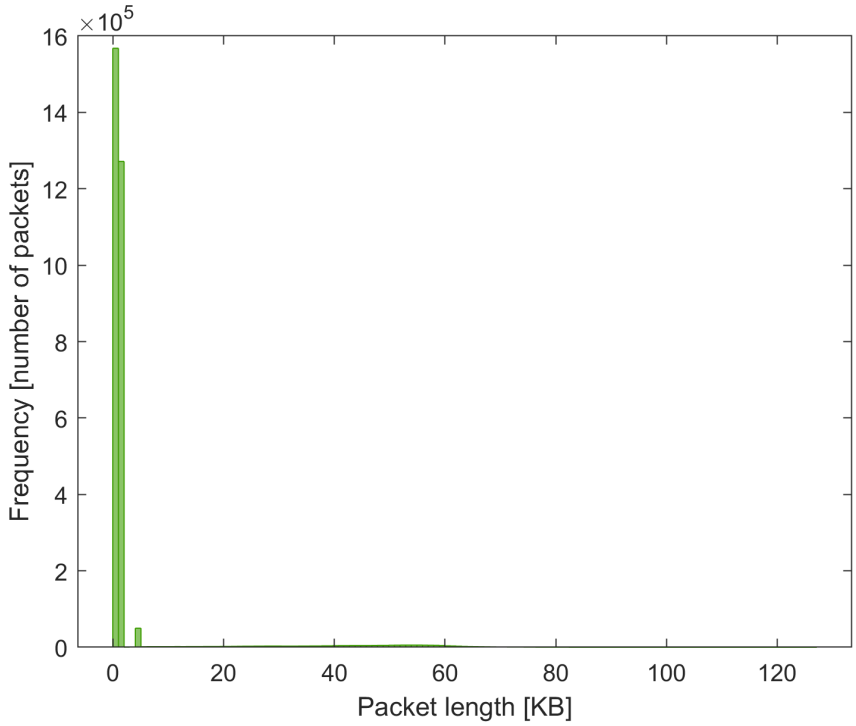


Figura 4.16: Istogramma delle frequenze dei frame catturati, in base alla loro lunghezza in KB per Cactus Cowboy - Plants at War

Anche in questo caso si nota immediatamente che la vasta maggioranza dei pacchetti è contenuta nelle 3 classi da 27, 1051 e 4123 byte, mentre gli altri pacchetti sono distribuiti in maniera più uniforme rispetto all'applicazione precedente, in un intervallo che va da 20 a 65 KB.

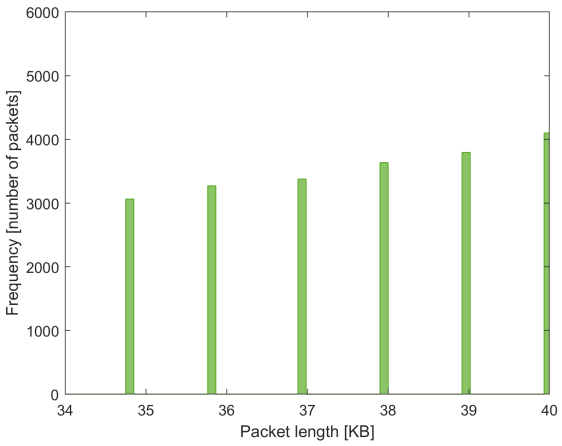
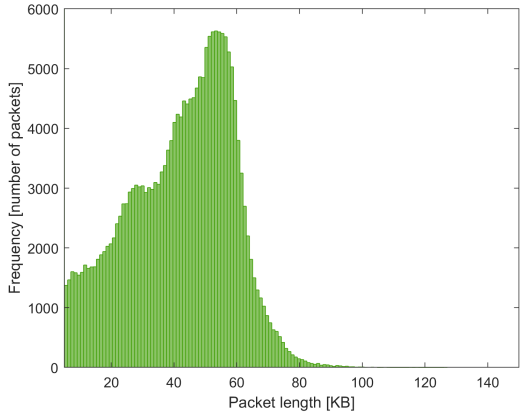


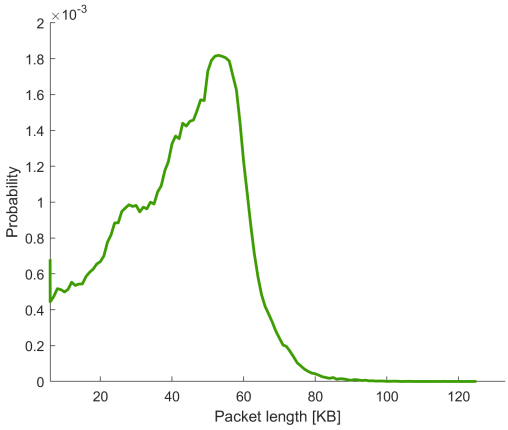
Figura 4.17: Dettaglio dell'istogramma precedente, con larghezza delle classi diminuita

Anche per questa applicazione è stato scelto un intervallo casuale per lo studio in dettaglio della distribuzione dei pacchetti. Anche in questo caso la larghezza degli intervalli delle classi è stata ridotta di un fattore superiore a 10. Si vede immediatamente che nemmeno in

questo studio sono presenti pacchetti di lunghezza intermedia, oltre a quelli di dimensione, in byte, di 34843, 35867, 36891, 37915 e 38939, corrispondenti a valori di `n` di 34, 35, 36, 37 e 38.



(a) Istogramma delle frequenze, a partire da 5 KB



(b) Distribuzione di probabilità per le lunghezze dei pacchetti

Figura 4.18: Dettaglio dell'istogramma delle frequenze, troncato per eliminare i tre picchi principali

La figura 4.18(a) mostra il dettaglio dell'istogramma delle frequenze, dove è stata troncata la parte relativa ai primi tre picchi, in modo da poter visualizzare comodamente l'andamento del resto del grafico. Si nota subito che la distribuzione dei frame questa volta è molto più morbida, anche se presenta comunque una notevole discesa dopo il picco più alto, intorno ai 50KB. Frame di lunghezza 100KB e superiore molto rari, quasi singolarità che appaiono con una frequenza di poche unità.
La figura 4.18(b) rappresenta la distribuzione di probabilità della figura 4.18(a). Per ogni classe la frequenza è stata divisa per il numero di elementi totali catturati in modo da presentare una percentuale. In questo modo l'andamento del grafico è preservato ma l'asse y è normalizzato a 1.

Di seguito viene riportata una tabella per i valori di probabilità delle parti tagliate dal grafico della figura 4.18(b).

Lunghezza [byte]	Percentuale dei pacchetti
27	50,64%
1051	41,08%
2075	0,01%
3099	0,02%
4123	1,61%
5147 e superiori	6,64%

Tabella 4.2: Distribuzione in percentuale dei pacchetti per le misure di Cactus Cowboy - Plants at War

Anche in questo caso si nota che il terzo picco corrisponde alla lunghezza di 4123 byte, mentre i grafici della figura 4.18 corrispondono solamente al 6,64% dei frame catturati.

4.2.3 Astral Slider

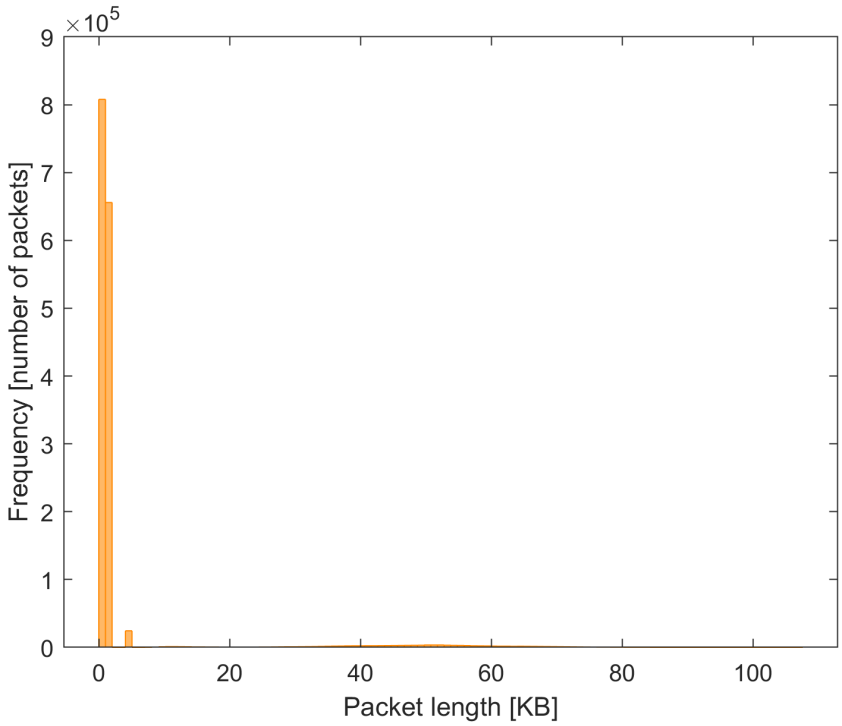


Figura 4.19: Istogramma delle frequenze dei frame catturati, in base alla loro lunghezza in KB per Astral Slider

Ancora una volta, si nota subito che la maggioranza dei frame catturati rientra nelle classi da 27, 1051 e 4123 byte, con gli altri che sono distribuiti intorno ai 50 KB.

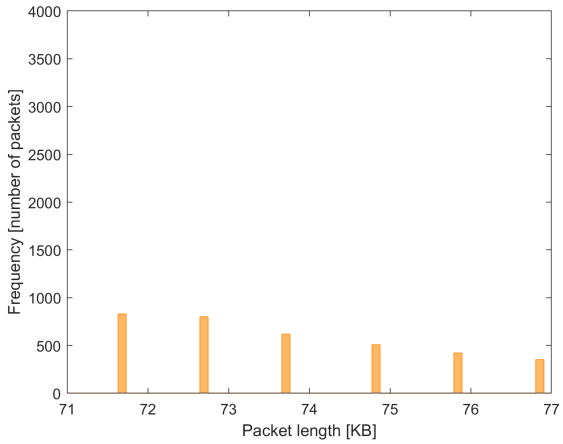
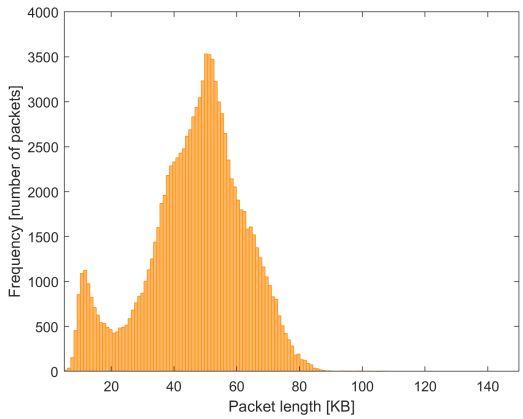


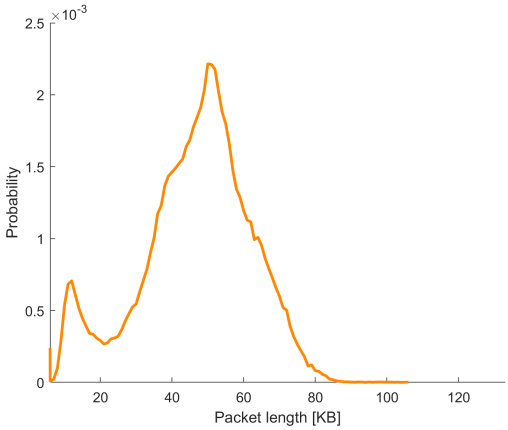
Figura 4.20: Dettaglio dell'istogramma precedente, con larghezza delle classi diminuita

Anche per Astral Slider è stato scelto un intervallo casuale per lo studio in dettaglio della distribuzione dei pacchetti, riducendo la larghezza degli intervalli delle classi di un fattore

superiore a 10. Si vede immediatamente che nemmeno in questo studio sono presenti pacchetti di lunghezza intermedia, ma gli unici presenti dono quelli di dimensione, in byte, di 71707, 72731, 73755, 74779, 75803 e 76827, corrispondenti a valori di `n` di 70, 71, 72, 73, 74 e 75.



(a) Istogramma delle frequenze, a partire da 5 KB



(b) Distribuzione di probabilità per le lunghezze dei pacchetti

Figura 4.21: Dettaglio dell'istogramma delle frequenze, troncato per eliminare i tre picchi principali

Nella figura 4.21(a) vengono riportate le frequenze dei frame per le lunghezze da 5 KB in poi, in questo caso la distribuzione è molto particolare: è composta da due distribuzioni quasi triangolari, una più piccola centrata intorno a 13 KB e una più ampia centrata intorno a 55 KB. Contrariamente ai casi precedenti, questa distribuzione sembra essere molto definita, il ché potrebbe suggerire che il programma gestisca l'invio dei pacchetti seguendo una distribuzione particolare specifica definita dagli sviluppatori del gioco.
La figura 4.21(b) è ancora una volta ottenuta dividendo, per ogni classe, il numero di elementi per quello dei frame totali catturati. In questo modo otteniamo una misura di probabilità (o percentuale) per ogni intervallo, riscalando ogni ampiezza in modo da essere normalizzata a 1.

Di seguito viene riportata una tabella per i valori di probabilità delle parti tagliate dal grafico della figura 4.21(b).

Lunghezza [byte]	Percentuale dei pacchetti
27	50,67%
1051	41,13%
2075	0,00%
3099	0,00%
4123	1,53%
5147 e superiori	6,67%

Tabella 4.3: Distribuzione in percentuale dei pacchetti per le misure di Astral Slider

Questa applicazione presenta una distribuzione un po' più particolare, come già visto nella figura 4.21(b). La tabella evidenzia come gli intervalli che contengono i pacchetti lunghi 2075 e 3099 siano praticamente vuoti, mentre per le applicazioni precedenti presentavano diversi ordini di grandezza in più di elementi. Si noti che il valore 0,00% non vuol dire che i frame di tale lunghezza siano 0, bensì che essi rappresentano meno del 0,005% dei pacchetti totali. Come al solito, la tabella dice anche che i grafici presentati in figura 4.21 rappresentano solamente il 6,67% dei pacchetti totali catturati.

4.2.4 Google Earth VR

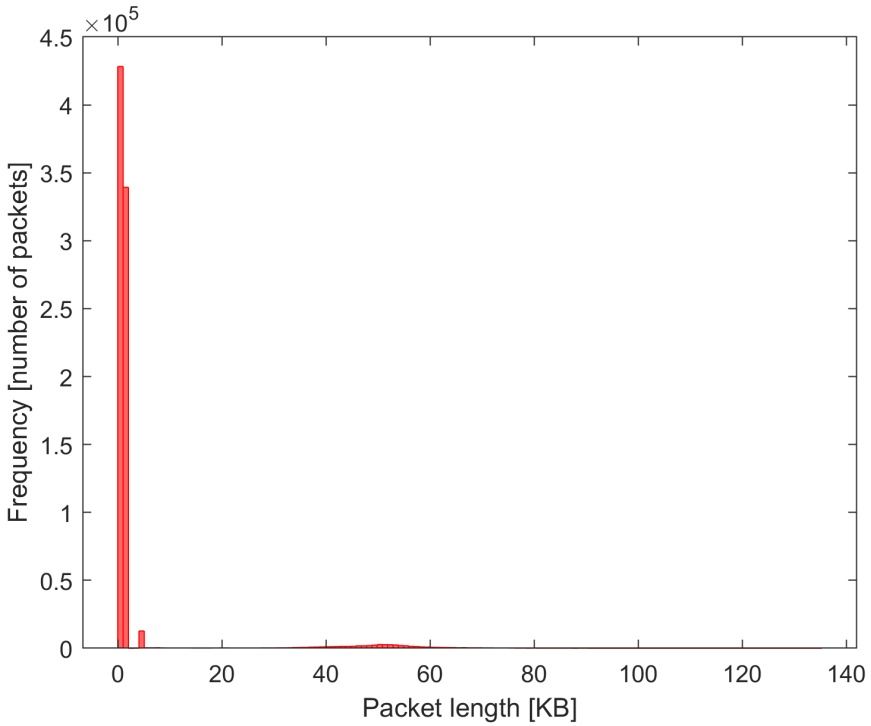


Figura 4.22: Istogramma delle frequenze dei frame catturati, in base alla loro lunghezza in KB per Google Earth VR

Anche nell'ultimo caso, le frequenze sono concentrate principalmente intorno alla lunghezza di 27, 1051 e 4123 byte. Questa volta però già da questo grafico si può intuire che la distribuzione degli altri pacchetti non sembra essere molto uniforme.

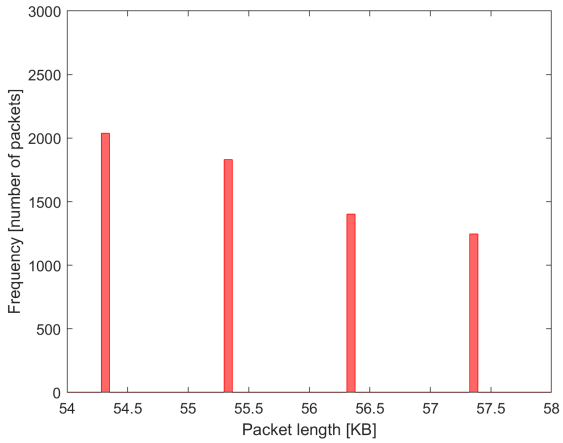
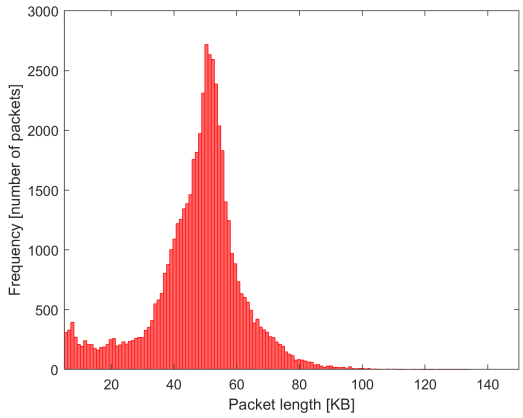
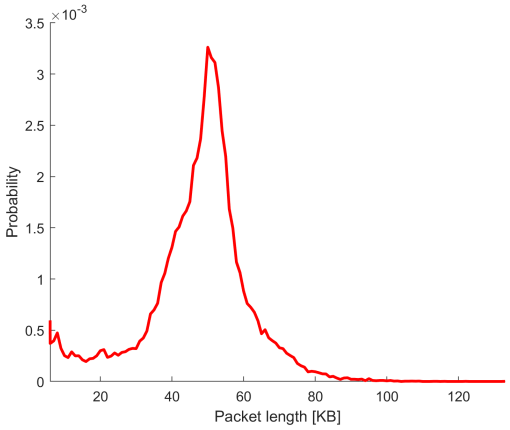


Figura 4.23: Dettaglio dell'istogramma precedente, con larghezza delle classi diminuita

Anche l'ultimo caso conferma l'ipotesi che tutti i pacchetti siano di lunghezza definita dalla formula sopracitata. In questo caso particolare è stato preso un intervallo casuale dell'istogramma a cui sono state diminuite di un fattore almeno 10 del larghezze delle classi.
Si vede immediatamente che nemmeno in questa ultima ispezione sono presenti pacchetti di lunghezza intermedia, ma gli unici presenti dono quelli di dimensione, in byte, di 54299, 55323, 56347 e 57371, corrispondenti a valori di `n` di 53, 54, 55 e 56.



(a) Istogramma delle frequenze, a partire da 5 KB



(b) Distribuzione di probabilità per le lunghezze dei pacchetti

Figura 4.24: Dettaglio dell'istogramma delle frequenze, troncato per eliminare i tre picchi principali

La figura 4.24(a) conferma quanto ipotizzato guardando l'istogramma totale, in quanto la distribuzione della lunghezza dei frame è decisamente poco uniforme, e anzi si presenta come un picco abbastanza marcato centrato nuovamente intorno a 50 KB.
La figura 4.24(b) è stata ottenuta anche questa volta dividendo il valore di ogni classe per il numero totale di frame catturati, ottenendo una misura di probabilità. In questo modo si riscala il grafico nell'asse y, normalizzandolo a 1.

Di seguito viene riportata una tabella per i valori di probabilità delle parti tagliate dal grafico

della figura 4.24(b).

Lunghezza [byte]	Percentuale dei pacchetti
27	51,39%
1051	40,72%
2075	0,00%
3099	0,01%
4123	1,52%
5147 e superiori	6,36%

Tabella 4.4: distribuzione in percentuale dei pacchetti per le misure di Google Earth VR

Ancora una volta la tabella rimarca la vasta concentrazione dei pacchetti attorno ai primi due valori, seguiti dal terzo a 4123 byte. La tabella mostra che i grafici in figura 4.24 rappresentano solamente il 6,36% dei frame totali.

4.2.5 Totale

La seguente sezione ha lo scopo di confrontare la distribuzione della lunghezza dei pacchetti delle quattro applicazioni testate, con l'obiettivo di evidenziare le potenziali simiglianze e differenze.
Il grafico è stato creato sovrapponendo le quattro distribuzioni di probabilità, in questo modo rendiamo le misure confrontabili, in quanto insensibili alle diverse durate delle catture, che aumentano l'ampiezza dei singoli grafici.

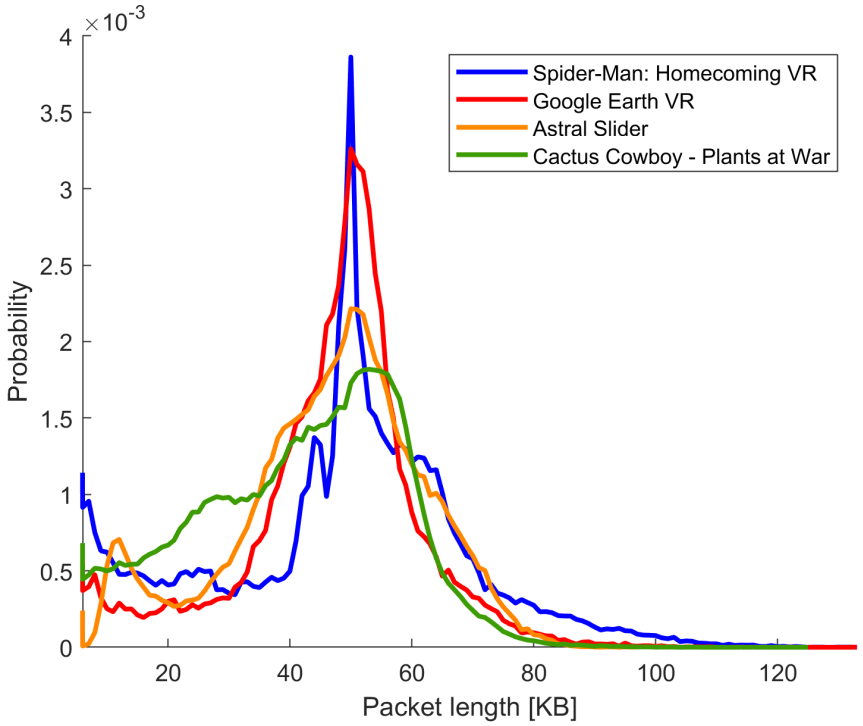


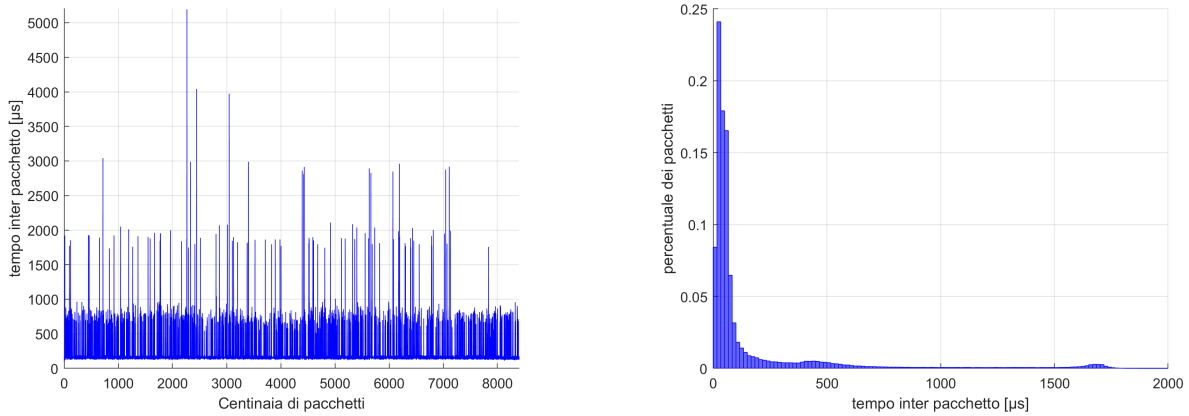
Figura 4.25: Confronto delle distribuzioni di probabilità delle lunghezze dei frame nelle quattro applicazioni

Si nota subito che in tutti i quattro casi la distribuzione presenta un picco principale intorno ai 50 KB, con un andamento simile per tutte le applicazioni.
Le differenze stanno nelle leggere differenze di valori: in alcuni casi come Spider-Man: Homecoming VR e Google Earth VR, il picco risulta più marcato e ci sono meno frame di dimensione dai 10 ai 40 KB. In altri casi come in Cactus Cowboy - Plants at War il picco è decisamente meno marcato, con ampiezza di circa la metà rispetto ai precedenti due casi, ma con la presenza di molti più pacchetti di lunghezza compresa tra i 10 e i 40 KB.
Nonostante queste leggere differenze, che comunque rappresentano solamente un 6,5% dei pacchetti in media, in tutti e 4 i casi di studio si evidenzia un andamento decisamente simile per quanto riguarda la distribuzione dei pacchetti in base alla loro lunghezza. Questo è probabilmente dovuto alle impostazioni del visore oppure al fatto che il software stesso sia programmato per favorire pacchetti di dimensione pari a circa 50 KB, rispetto ad altri, nel caso di burst e di dimensione 1051 byte nel caso normale.

4.3 Tempo inter pacchetto

Nella seguente sezione analizzeremo i tempi inter pacchetto per ogni applicazione utilizzata. Questa parte ha lo scopo di verificare se ci sia una relazione tra le tempistiche di ritardo tra un pacchetto ed il successivo.
Per ogni applicazione verrà presentato un grafico con il tempo inter pacchetto di ogni frame. Siccome plottare milioni di valori in un grafico relativamente piccolo faceva si che non si vedessero le linee chiaramente, sono state graficate invece le medie dei tempi, fatte ogni 100 pacchetti consecutivi.

4.3.1 Spider-Man: Homecoming VR



(a) Tempo inter pacchetto medio ogni 100 frame, in microsecondi

(b) Istogramma della distribuzione dei frame, in base al tempo inter pacchetto

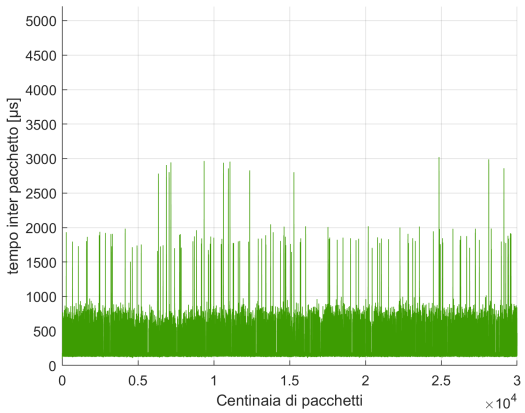
Figura 4.26: Analisi dei tempi inter pacchetto per Spider-Man: Homecoming VR

La figura 4.26(a) ci permette di visualizzare, con una certa accuratezza, la distribuzione dei tempi inter pacchetto nel tempo. Si nota subito che non ci sono momenti particolari in cui i tempi aumentano o diminuiscono, ma essi sono abbastanza uniformi, per la maggior parte intorno al millisecondo. Le punte più alte, di circa 2 millisecondi e le altre anomalie sono

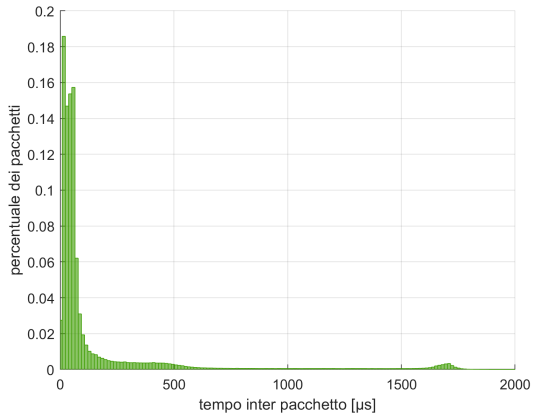
distribuite regolarmente. Questo ci fa pensare che non ci sia alcuna relazione tra la scena e il tempo inter pacchetto, cosa che non sorprende visto che scene più complesse vengono gestite tramite burst, e quindi tramite l'invio di frame più grandi e non tramite l'aumento del numero di frame trasmessi, che diminuirebbe il tempo inter pacchetto.
Si noti anche che avendo mediato i valori, il grafico ammortizza in maniera abbastanza significativa le anomalie, ma questo non ci preoccupa visto che la nostra analisi non si concentra sui casi patologici.

La figura 4.26(b), invece, rappresenta l'istogramma della distribuzione dei frame, in base al tempo inter pacchetto. Si noti che anche in questo caso, ogni intervallo è stato rappresentato come percentuale sul totale dei pacchetti. Questo ci permetterà di confrontare le quattro applicazioni alla fine della sezione.
L'istogramma evidenzia anche che i tempi per la maggior parte seguono un andamento esponenziale negativo, con dei punti in cui tendono a concentrarsi di più, come a circa 500 e 1700 μs.
La quasi totalità dei frame, cioè più del 90%, rientra nella categoria di tempo inter pacchetto inferiore a 250 μs, il che corrisponde a circa 4000 frame al secondo, che è circa la media totale dell'esperimento.
Il grafico è stato volutamente troncato al valore di 2 ms sull'asse x, questo perché un rapido sguardo alla figura 4.26(a) permette di realizzare che quasi tutti i frame rientrano in questa categoria, anche se nel caso di Spider-Man: Homecoming VR, il tempo inter pacchetto massimo della cattura è stato di 504,4 ms. Non è chiaro se questo valore rappresenti una anomalia oppure un valore falsato dal ritardo di fermata della cattura rispetto alla chiusura del gioco.

4.3.2 Cactus Cowboy - Plants at War



(a) Tempo inter pacchetto medio ogni 100 frame, in microsecondi



(b) Istogramma della distribuzione dei frame, in base al tempo inter pacchetto

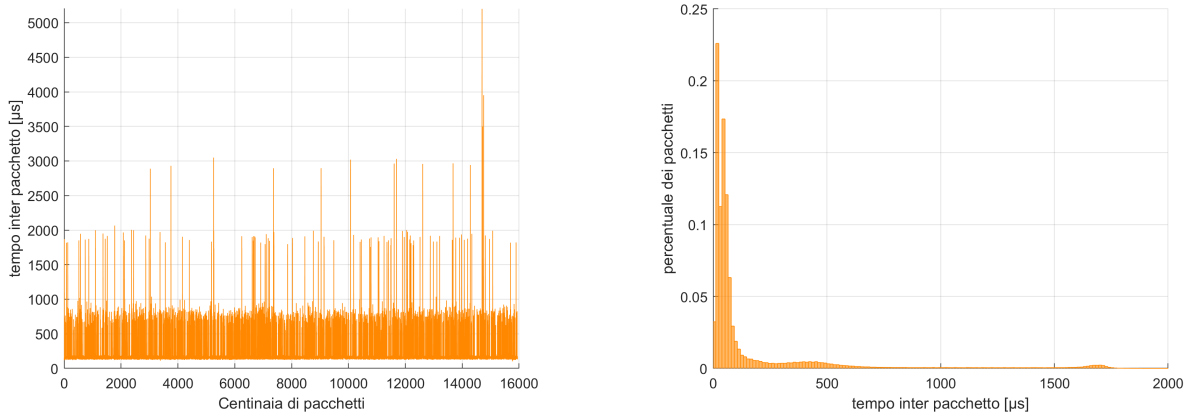
Figura 4.27: Analisi dei tempi inter pacchetto per Cactus Cowboy - Plants at War

La figura 4.27(a) ci permette di vedere la distribuzione dei tempi inter pacchetto nel tempo. Si nota subito che, anche in questo caso, non ci sono momenti particolari in cui i tempi aumentano o diminuiscono, ma essi sono abbastanza uniformi, per la maggior parte intorno al millisecondo. Le punte più alte, di circa 2 millisecondi e le altre anomalie sono distribuite

regolarmente, anche se queste rimangono comunque sempre sotto la soglia dei 3 ms. Questo ci fa pensare che, ancora una volta, non ci sia alcuna relazione tra la scena e il tempo inter pacchetto.
Si noti anche che avendo mediato i valori, il grafico ammortizza in maniera abbastanza significativa le anomalie, il ché potrebbe spiegare come mai tutti i valori stiano sotto a 3 ms.

Il grafico in figura 4.27(b) rappresenta l'istogramma della distribuzione dei frame, in base al tempo inter pacchetto. Si noti che anche in questo caso, ogni intervallo è stato rappresentato come percentuale sul totale dei pacchetti. Questo ci permetterà di confrontare le quattro applicazioni alla fine della sezione.
L'istogramma evidenzia anche che i tempi per la maggior parte seguono un andamento esponenziale negativo, con dei punti in cui tendono a concentrarsi di più, come nell'intervallo da 200 a 500 μs e a circa 1700 μs.
La quasi totalità dei frame, cioè più del 95%, rientra nella categoria di tempo inter pacchetto inferiore a 500 μs, il che corrisponde a circa 2000 frame al secondo, anche se comunque di essi, la maggioranza ha un tempo inter pacchetto inferiore a 200 μs.
Il grafico è stato volutamente troncato al valore di 2 ms sull'asse x, questo perché un rapido sguardo alla figura 4.27(a) permette di realizzare che quasi tutti i frame rientrano in questa categoria, anche se nel caso di Cactus Cowboy - Plants at War, il tempo inter pacchetto massimo della cattura è stato di 282,2 ms. Non è chiaro se questo valore rappresenti una anomalia oppure un valore falsato dal ritardo di fermata della cattura rispetto alla chiusura del gioco, anche se rappresenta comunque un valore del tutto accettabile per le fasi in cui le informazioni da scambiare non sono tante, come in un menù di gioco.

4.3.3 Astral Slider



(a) Tempo inter pacchetto medio ogni 100 frame, in microsecondi

(b) Istogramma della distribuzione dei frame, in base al tempo inter pacchetto

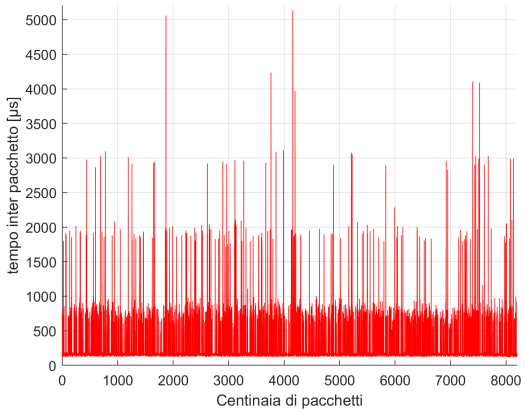
Figura 4.28: Analisi dei tempi inter pacchetto per Astral Slider

La figura 4.28(a) rappresenta la distribuzione dei frame in base al loro tempo inter pacchetto. Anche nel terzo caso è stato riscalato in modo tale da normalizzare l'asse y a 1, rappresentando quindi la percentuale dei frame sul totale.
Come nello studio precedente, anche in questo la maggioranza dei frame ha un tempo inter pacchetto inferiore a 3 ms, con la maggioranza concentrata verso il valore di 1 ms.

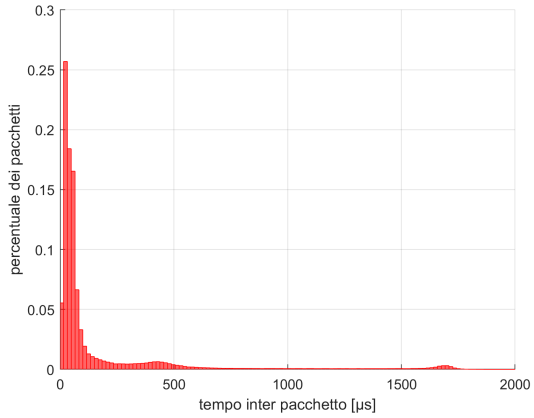
La distribuzione dei valori è nuovamente regolare, suggerendo l'indipendenza di essi dalla scena di gioco.

Il grafico in figura 4.28(b) rappresenta l'istogramma dei tempi inter pacchetto. Anche in questo caso i valori sembrano distribuiti in maniera esponenziale, con i punti a circa 500 e 1700 μs che hanno una maggiore concentrazione.
Oltre il 90% dei frame rientra nella categoria con tempo inter pacchetto minore di 250 μs.
Anche questa volta il grafico è stato troncato al valore di 2 ms sull'asse x perché guardando la figura 4.28(a) si vede che quasi tutti i frame rientrano in questa categoria.
Il tempo inter pacchetto massimo della cattura è stato di 678,6 ms, anche se ancora una volta questo dato potrebbe essere falsato dai momenti finali della cattura.

4.3.4 Google Earth VR



(a) Tempo inter pacchetto medio ogni 100 frame, in microsecondi



(b) Istogramma della distribuzione dei frame, in base al tempo inter pacchetto

Figura 4.29: Analisi dei tempi inter pacchetto per Google Earth VR

Guardando la figura 4.29(a), anche nell'ultimo caso si può notare che la maggioranza dei frame ha un tempo inter pacchetto intorno al millisecondo, alcuni di circa 2 ms e pochi di circa 3 ms, con qualche anomalia che arriva fino a 5 ms.
In confronto alle altre applicazioni analizzate, questa sembra essere quella con la maggiore quantità di tempi maggiori di 1 ms. Questo potrebbe essere dovuto al fatto che l'applicazione è relativamente leggera rispetto ad un gioco.
Anche in questo ultimo caso la distribuzione non rappresenta variazioni durante la cattura, suggerendo l'indipendenza dall'uso delle diverse funzioni dell'applicazione.

Come negli altri casi, l'istogramma in figura 4.29(b) evidenzia un andamento generale di tipo esponenziale con i soliti due punti di concentramento a circa 500 e 1700 μs.
Per lo stesso motivo delle altre analisi, l'istogramma è troncato al valore di 2 ms sull'asse x.
Oltre il 90% dei frame ha un tempo inter pacchetto inferiore a 250 ms, anche se in questa analisi, il tempo massimo è stato di 3,389 s. Anche se questa applicazione è molto leggera e contiene momenti in cui si sta completamente fermi, questo dato è decisamente più alto di un risultato verosimile, ed è perciò considerato come errore dovuto alla fine della cattura.

4.3.5 Totale

Tutti i casi analizzati hanno presentato una distribuzione simile dei tempi inter pacchetto. Analizzando graficamente i grafici delle figure 4.26, 4.27, 4.28 e 4.29 sembrerebbe che la maggior parte dei tempi sia circa multiplo di 1 ms. Sebbene non sia del tutto errata, questa assunzione viene smentita dai relativi istogrammi, che evidenziano sicuramente dei punti di concentramento costanti nei quattro casi di interesse, ma non multipli di 1 ms. Non è chiaro se il fatto che i tempi tendano a concentrarsi intorno a dei valori predeterminati sia dovuto a come è programmato il visore, oppure a come si tende a trasferire i dati nel protocollo USB 3.0 SuperSpeed, ma questo fatto è sicuramente di interesse.
Il fatto che tutti gli istogrammi presentino una distribuzione piuttosto simile tra loro e di forma circa esponenziale non è decisamente sorprendente: è normale aspettarsi che i tempi tendano ad essere più bassi con maggiore probabilità; inoltre, come già accennato, i tempi sembrano essere indipendenti dalla scena, e quindi dal gioco stesso, in quanto le differenze vengono gestite tramite la lunghezza dei frame nella procedura di bursting, e non tramite l'aumento della velocità di trasferimento dei pacchetti stessi.

Capitolo 5

Conclusioni

5.1 Risultati

Dall'analisi sono emersi alcuni risultati aspettati, come quelli riguardo al tempo inter pacchetto, ed altri inaspettati.
In questa sezione si intendere riassumerli brevemente: per quanto riguarda il throughput abbiamo notato che, per ogni applicazione, tende a rimanere abbastanza costante, con alcune deviazioni dovute alle diverse scene. Il confronto tra le applicazioni ha permesso di evidenziare quali fossero più o meno pesanti e perché. In particolare dei giochi con cinematiche ad alta qualità si sono rivelati più pesanti di applicazioni ottimizzate come Google Earth VR.
L'analisi di input-output ci ha fatto capire quanto grossa sia effettivamente la differenza del traffico in output rispetto a quello in input. Sebbene fosse normale aspettarsi una differenza, non ci si aspettava che fosse così marcata. Questo dato può essere utile in futuro, in particolare se si dovesse decidere di utilizzare una connessione o un protocollo non simmetrico.
L'analisi sulla lunghezza dei pacchetti, invece, ci ha fatto capire che il protocollo USB 3.0 SuperSpeed riesce a gestire una grossa mole di dati trasmessi, come nei casi studiati, utilizzando la tecnica specifica del bursting. In questo caso è interessante notare che ci sia una preferenza per i frame di lunghezza pari a circa 50 KB e questo dato non è sembrato cambiare né tra le applicazioni, né alterando le impostazioni grafiche e quelle di aggiornamento del visore.
L'analisi sul tempo inter pacchetto ha evidenziato una indipendenza tra un frame e il successivo e l'applicazione utilizzata. In particolare, nemmeno all'interno dello stesso programma si sono verificati mutamenti significativi durante le diverse scene di gioco. I tempi sembrano seguire una distribuzione circa esponenziale negativa con alcuni ammassamenti vicini a dei valori costanti per tutti e quattro gli studi.

5.2 Sviluppi futuri

L'obiettivo di questa tesi era quello di porre delle fondamenta solide per sviluppi futuri riguardo all'argomento. È chiaro che l'analisi delle quattro applicazioni si è limitata solamente a qualche parametro di interesse, senza tuttavia verificare le relazioni tra di loro e cercando di capire come questi valori variano se si cambiano le circostanze.
Gli sviluppi futuri si potrebbero sicuramente concentrare sull'analisi degli stessi parametri, ma studiando diverse situazioni, in cui l'unica variabile sia la frequenza di aggiornamento del visore, la grafica dell'applicazione stessa, impostando il traffico desiderato oppure cercando di isolare l'utilizzo delle funzioni del visore: prima muovendo solo i controller, poi solo la

visuale, poi muovendosi solo nello spazio, ecc.
Cambiare tutte queste variabili e provare diverse combinazioni tra di esse permetterebbe di riuscire a capire le possibili correlazioni che ci sono, permettendo al modello di traffico di essere più predittivo. Basterebbe allora sapere a priori i valori delle variabili per stimare il traffico durante l'utilizzo.
Un modello derivato da studi successivi più accurati e specifici potrebbe sicuramente aiutare a sviluppare dei protocolli e mezzi di trasmissione specifici per questa applicazione, se non protocolli di compressione che aiuterebbero a diffondere l'utilizzo di questa tecnologia, adattandola alle architetture che si possono trovare normalmente nelle case di tutti.

Bibliografia

[1] R. Bagheri, "Virtual reality: The real life consequences," *UC Davis Bus. LJ*, vol. 17, p. 101, 2016.

[2] E. A.-L. Lee and K. W. Wong, "A review of using virtual reality for learning," *Transactions on edutainment I*, pp. 231–241, 2008.

[3] F. Pallavicini, L. Argenton, N. Toniazzi, L. Aceti, and F. Mantovani, "Virtual reality applications for stress management training in the military," *Aerospace medicine and human performance*, vol. 87, no. 12, pp. 1021–1030, 2016.

[4] A. C. Firu, A. I. Tapîrdea, A. I. Feier, and G. Drăghici, "Virtual reality in the automotive field in industry 4.0," *Materials Today: Proceedings*, vol. 45, pp. 4177–4182, 2021.

[5] J. Beck, M. Rainoldi, and R. Egger, "Virtual reality in tourism: a state-of-the-art review," *Tourism Review*, vol. 74, no. 3, pp. 586–612, 2019.

[6] R. Kulpa, F. Multon, and F. Argelaguet, "Virtual reality & sport," in *ISBS-Conference Proceedings Archive*, 2015.

[7] T. S. Perry, "Virtual reality goes social," *IEEE Spectrum*, vol. 53, no. 1, pp. 56–57, 2015.

[8] J. Falah, S. Khan, T. Alfalah, S. F. M. Alfalah, W. Chan, D. K. Harrison, and V. Charissis, "Virtual reality medical training system for anatomy education," in *2014 Science and Information Conference*, pp. 752–758, 2014.

[9] S. Schutt, D. Holloway, D. Linegar, and D. Deman, "Using simulated digital role plays to teach healthcare 'soft skills'," in *2017 IEEE 5th International Conference on Serious Games and Applications for Health (SeGAH)*, pp. 1–6, 2017.

[10] J. Bates, "Virtual reality, art, and entertainment," *Presence: Teleoperators & Virtual Environments*, vol. 1, no. 1, pp. 133–138, 1992.

[11] https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game, Ultimo accesso: 01.06.2022.

[12] M. Lecci, A. Zanella, and M. Zorzi, "An ns-3 implementation of a bursty traffic framework for virtual reality sources," in *Proceedings of the Workshop on ns-3*, pp. 73–80, 2021.

[13] M. Lecci, M. Drago, A. Zanella, and M. Zorzi, "An open framework for analyzing and modeling xr network traffic," *IEEE Access*, vol. 9, pp. 129782–129795, 2021.

[14] M. Lecci, F. Chiariotti, M. Drago, A. Zanella, and M. Zorzi, "Temporal characterization of xr traffic with application to predictive network slicing," *arXiv preprint arXiv:2201.07043*, 2022.

[15] S. Zhao, H. Abou-zeid, R. Atawia, Y. S. K. Manjunath, A. B. Sediq, and X.-P. Zhang, "Virtual reality gaming on the cloud: A reality check," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2021.

[16] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7573–7586, 2019.

[17] D. Abdlkarim, M. Di Luca, P. Aves, S.-H. Yeo, R. C. Miall, P. Holland, and J. M. Galea, "A methodological framework to assess the accuracy of virtual reality hand-tracking systems: A case study with the oculus quest 2," *bioRxiv*, 2022.

[18] J. Kelly, T. Doty, M. Ambourn, and L. Cherep, "Distance perception in the oculus quest and oculus quest 2," 2022.

[19] Y. Y. Dyulicheva, D. A. Gaponov, and O. Y. Poleshchuk, "About the features of the virtual simulators development and their usage in dental education," 2021.

[20] C. M. Craig, J. Stafford, A. Egorova, C. McCabe, and M. Matthews, "Can we use the oculus quest vr headset and controllers to reliably assess balance stability?," *Diagnostics*, vol. 12, no. 6, p. 1409, 2022.

[21] USB Implementers Forum, "USB 2.0 Specification," tech. rep., USB Implementers Forum, 2021.

[22] USB Implementers Forum, "Universal Serial Bus Revision 3.1 Specification," tech. rep., USB Implementers Forum, 2016.

[23] USB.org, "SuperSpeed USB Consumer Cert Final 2," tech. rep., USB.org, 2012.

[24] USB Implementers Forum, "USB 3.2 Specification," tech. rep., USB Implementers Forum, 2022.

[25] USB Implementers Forum, "USB 4.0 Specification," tech. rep., USB Implementers Forum, 2022.

[26] USB.org, "USB Promoter Group USB4 Specification," tech. rep., USB.org, 2019.

[27] hp.com, "USB 3.0 Technology," tech. rep., hp.com, 2014.

[28] USB.org, "USB Power Delivery Specification 1.0," tech. rep., USB.org, 2015.

[29] "Universal serial bus." https://wiki.osdev.org/Universal_Serial_Bus, 2011, Ultimo accesso: 01.06.2022.

[30] Acroname, "Usb architecture." https://acroname.com/blog/whats-difference-between-usb-hub-usb-switch-what-multiplexer, 2017, Ultimo accesso: 02.07.2022.

[31] P. P. Tom Mainelli, "Usb 2.0's real deal." https://web.archive.org/web/20101205151115/http://www.pcworld.com/article/82005/news_and_trends_usb_20s_real_deal.html, 2002, Ultimo accesso: 01.06.2022.

[32] WIkipedia, the Free Encyclopedia, "Usb type-c pinout," 2016, Ultimo accesso: 02.07.2022.

[33] C. H. Lim, B. A. bin Rosdi, and C. F. Yap, "Synchronization of multiple USB 3.0 devices using Isochronous Timestamp Packet," *Computer Standards Interfaces*, vol. 49, pp. 22–33, 2017.

[34] N. Leavitt, "For wireless usb, the future starts now," *Computer*, vol. 40, no. 7, pp. 14–16, 2007.

[35] D. Leenaerts, R. van de Beek, J. Bergervoet, H. Kundur, and G. van der Weide, "Wimedia uwb technology: 480mb/s wireless usb," in *2007 IEEE International Workshop on Radio-Frequency Integration Technology*, pp. 8–12, 2007.

[36] C.-M. Stan and D. Neacşu, "Role of usb communication in modem engineering education," in *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, pp. 1–4, IEEE, 2021.

[37] P. Fraga-Lamas, T. M. Fernández-Caramés, O. Blanco-Novoa, and M. A. Vilar-Montesinos, "A review on industrial augmented reality systems for the industry 4.0 shipyard," *Ieee Access*, vol. 6, pp. 13358–13375, 2018.

[38] M. Inc., "Meta quest 2." https://store.facebook.com/it/quest/products/quest-2/tech-specs#tech-specs, 2022, Ultimo accesso: 27.06.2022.

[39] L. Bianchi, "Meta quest 2." https://www.smartworld.it/tecnologia/oculus-quest-2-si-mostra-a-nudo-in-un-teardown.html, 2022, Ultimo accesso: 02.07.2022.

[40] Meta Inc., "Meta link cable." https://store.facebook.com/it/quest/accessories/quest-2/link-cable/, 2022, Ultimo accesso: 02.07.2022.

[41] T. W. team. https://www.wireshark.org/, 2022, Ultimo accesso: 27.06.2022.

[42] T. Moń, "Usbpcap." https://desowin.org/usbpcap/index.html, 2020, Ultimo accesso: 27.06.2022.

[43] G. Baiunco, "Spider-man: Homecoming vr." https://www.vigamusmagazine.com/82264/spider-man-homecoming-vr-ecco-mostrato-il-gameplay/, 2017, Ultimo accesso: 02.07.2022.

[44] SteamDB, "Cactus cowboy - plants at war." https://steamdb.info/app/1973110/, 2022, Ultimo accesso: 02.07.2022.

[45] Steam, "Astral slider." https://store.steampowered.com/app/1916930/Astral_Slider/, 2022, Ultimo accesso: 02.07.2022.

[46] M. Morselli, "Come usare google earth vr su oculus rift." https://www.vr-italia.org/come-usare-google-earth-vr-su-oculus-rift/, 2016, Ultimo accesso: 02.07.2022.

[47] MathWorks, "Matlab." https://it.mathworks.com/products/matlab.html, 2022, Ultimo accesso: 02.07.2022.

[48] S. D. Dharkar, G. R. Bamnote, P. S. Ingle, and S. E. Lokhande, "Super speed data traveller usb 3.0," *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)*, vol. 2, no. 6, p. 13, 2013.