

ANALISI DI UNA RETE AUTOMOTIVE ETHERNET CON L'UTILIZZO DEL PROTOCOLLO IEEE 802.1Q

Alberto Provenzano

provenzano.alberto21@gmail.com

Giuseppe Leocata

peppeleocata@gmail.com

1. INTRODUZIONE

Si vuole simulare una rete con le seguenti specifiche:

- Collegamenti Ethernet a 100Mbps
- Mappatura del traffico nelle classi SR A, SR B e best-effort (Credit Based Shaper di Audio Video Bridging)

per un'applicazione automotive con traffico cross-domain.

La rete è composta da 18 end-node e due switch e trasporta dati per i domini:

- ADAS (Advanced Driver Assistance Systems)
- Multimedia/infotainment

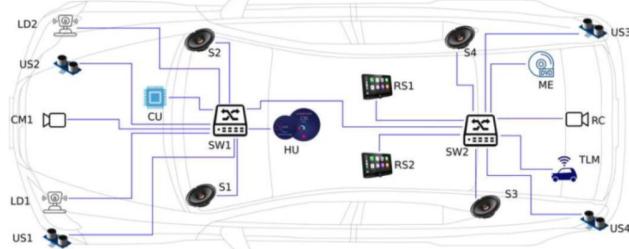


Figura 1: Schema scenario implementato

I sistemi Multimedia/Infotainment sono soft Real-Time ma non Safety-Critical, mentre i sistemi ADAS sono hard Real-Time e Safety-Critical.

1.1. Descrizione flussi

È stato implementato il Credit Based Shaper dello standard IEEE 802.1Q allo scopo di valutare le seguenti metriche:

- End-to-end delay
- Jitter Assoluto
- Numero di frame nella coda di trasmissione

Ogni nodo, inoltre, esegue un numero di applicazioni pari alla somma dei flussi in trasmissione e in ricezione. I flussi scambiati dai vari end-node sono riportati nella tabella 1.

2. SCENARIO

Il sistema ADAS prevede l'utilizzo di quattro sensori ad ultrasuoni (US1, US2, US3, US4), due sensori lidar (LD1, LD2) e di due

videocamere: una anteriore (CM1) e una posteriore (RC). Queste ultime, trasmettono i flussi video al monitor della Head Unit (HU) per fornire assistenza visuale al guidatore. Tutti i sensori, invece, trasmettono i dati campionati ad una CU (Control Unit), la quale si occupa di elaborare i dati ed inoltrare avvisi al conducente tramite l'HU.

Il sistema multimediale è composto da quattro altoparlanti (S1-S4) e due Rear Seat Entertainment (RS1, RS2). Il nodo Multimedia Entertainment (ME) trasmette un flusso video per ogni Rear Seat (RS1, RS2) e un flusso audio ad ogni altoparlante (S1-S4).

Vi è, infine, anche un sistema telematico che trasmette informazioni all'HU e alla CU come avvisi sul traffico o dati GPS.

Le caratteristiche relative ai flussi nello scenario iniziale sono riportate nella Figura 2. A seguito di numerose considerazioni (che verranno discusse in seguito) si è scelto di trasmettere tutti i flussi video a 15fps e la configurazione dei flussi è stata modificata come si può osservare dalla Tabella 1.

Flow Id	Src	Dst	Periodo	Deadline Rel.	Payload
1	LD1, LD2	CU	1.4 ms	1.4 ms	1300 byte
2	ME	S1, S2, S3, S4	250 us	250 us	80 byte
3	US1, US2, US3, US4	CU	100 ms	100 ms	188 byte
4	CU	HU	10ms	10 ms	10500 byte
5	CM1	HU	16.66 ms	16.66 ms	178500 byte
6	ME	RS1, RS2	33.33 ms	33.33 ms	178500 byte
7	TLM	HU, CU	625 us	625 us	600 byte
8	RC	HU	33.33 ms	33.33 ms	178000 byte

Figura 2: caratteristiche iniziali dei flussi

2.1. Configurazione della simulazione

A causa dei collegamenti Ethernet a 100Mbps è stato necessario intervenire sui flussi video per diverse ragioni. In primo luogo, si è calcolato il Workload relativo al collegamento tra lo Switch 1 ed HU (in quanto su questo viaggiano i flussi più pesanti). Il calcolo è stato eseguito secondo la seguente formula:

$$\text{Workload} = \frac{(\text{payload} + \text{overhead}_\text{eth} \times \text{n_frames}) \times 8}{T} \quad (2.1)$$

dove il payload e l'overhead ethernet sono espressi in byte (per questo si moltiplica al numeratore per 8, cioè per ottenere il numero di bit), l'overhead ethernet è di 42 byte e T rappresenta il periodo del flusso in considerazione. Infine, n_frames è ottenuto come:

$$\text{n_frames} = \left\lceil \frac{\text{payload}}{\text{MTU}} \right\rceil \quad (2.2)$$

Il workload così calcolato è relativo ad un singolo flusso; pertanto, per valutare quello riferito al collegamento si sono sommati i

Flow Id	Src	Dest	Periodo	Deadline rel.	Payload	Burst
1	LD1,LD2	CU	1.4ms	1.4ms	260 byte	5
2	ME	S1,S2,S3,S4	250μs	250μs	80 byte	1
3	US1,US2,US3,US4	CU	100ms	100ms	188 byte	1
4	CU	HU	10ms	10ms	250 byte	42
5	CM1	HU	66.66ms	66.66ms	400 byte	447
6	ME	RS1,RS2	66.66ms	66.66ms	700 byte	255
7	TLM	HU,CU	625μs	625μs	600 byte	1
8	RC	HU	66.66ms	66.66ms	400 byte	445

Tabella 1: Caratteristiche flussi di rete

workload dei flussi 4,5,7,8 ottenendo un valore di 148.891Mbps, quindi superiore al massimo consentito di 100Mbps. Allo stesso modo, nel collegamento tra ME e lo Switch 2, il workload totale relativo ai flussi generati da ME è di 103.7Mbps, anche in questo caso oltre il limite supportato.

In secondo luogo, data la necessità di garantire la consegna delle frame real-time e di mappare i flussi nelle classi SR A ed SR B previste da AVB (Audio Video Bridging), il calcolo dell'Idle-Slope (velocità con cui aumenta il credito associato ad una classe di traffico SR) da impostare nelle diverse porte Ethernet ha portato a diverse problematiche dovute al limite dei collegamenti di 100Mbps. In particolare, dopo aver calcolato tale valore nelle varie porte mediante l'uso della seguente formula:

$$\text{idleSlope} = \frac{(\text{MTU} + \text{overhead_ethernet}) \times 8}{\text{CMI}} \quad (2.3)$$

dove MTU (*Maximum Transmission Unit*) è la dimensione massima della frame trasmessa e CMI (*ClassMeasurementInterval*) è un valore definito nello standard IEEE 802.1Q per entrambe le classi A e B (125 μs per la classe SR A e 250 μs per SR B); è stato constatato che, per riuscire a mappare i flussi video nelle classi SR, è necessario usare valori più bassi di MTU, ma questo implica due problematiche: la prima consiste nel fatto che il workload aumenta a causa dell'overhead (in quanto il flusso viene suddiviso in un maggior numero di frame), la seconda è legata al delay relativo alla trasmissione dell'ultima frame di un flusso, dovuto al CMI (le frame non vengono trasmesse tutte in una volta). La formula è:

$$\text{Delay} = (\text{n_frames}-1) \times \text{CMI} \quad (2.4)$$

Di conseguenza, è necessario trovare un compromesso tra la dimensione dell'MTU e il Delay che ne consegue. Si prenda come esempio il flusso 5 prodotto dalla videocamera anteriore: scegliendo un valore di MTU di 800 byte si ottiene un valore di idleSlope molto alto e, considerando ad esempio il collegamento SW1->HU, si ha un valore complessivo (di idleSlope) di 53.88Mbps se si sceglie di mapparlo in SR A e di 26.94Mbps se si opta per la mappatura in SR B, con un delay di, rispettivamente, 27.87ms e 55.75ms (in entrambi i casi ampiamente oltre la deadline relativa). Va sottolineato, inoltre, che tali ritardi sono relativi alla sola trasmissione e non tengono conto della latenza, che è comunque certamente limitata (2ms per la classe A e 50ms per la classe B); alla luce di ciò, l'unico "parametro" modificabile per ovviare a tutte le problema-

tiche appurate è il periodo dei flussi video. Nello specifico, sono stati tutti portati a 66.66ms (15 fps).

2.2. Priorità assegnate

Si è scelto di assegnare le priorità come mostrato nella seguente tabella:

Src	Dst	DeadLine rel.	Priorità
LD1,LD2	CU	1.4ms	7 (SR A)
ME	S1,S2,S3,S4	250μs	7 (SR A)
US1,US2,US3,US4	CU	100ms	6 (SR B)
CU	HU	10ms	7 (SR A)
CM1	HU	66.66ms	7 (SR A)
ME	RS1,RS2	66.66ms	6 (SR B)
TLM	HU,CU	625μs	5 (BE)
RC	HU	66.66ms	7 (SR A)

Nello specifico, si è deciso di assegnare al traffico best-effort solamente il flusso relativo al gps, in quanto soft Real-Time. I messaggi di controllo generati dalla control unit sono stati assegnati alla classe SR A in quanto il flusso 4 ha una deadline di 10ms e quindi necessita di un limite di latenza più basso, come quello fornito dalla classe A (2ms). Un ragionamento analogo è stato fatto per i flussi generati dai lidar. I flussi video relativi alla videocamere posteriore e anteriore sono stati mappati nella classe SR A poiché il delay calcolato per essi con la formula 2.4 risulta elevato, pertanto si sono ritenute più appropriate le garanzie date dalla classe A. Anche per quanto riguarda i flussi indirizzati agli speaker si è scelta la classe A: questa decisione è stata presa congiuntamente a quella di mappare i flussi video (generati sempre da ME) nella classe B perché si è preferito dare maggiore priorità ai primi in quanto hanno una deadline più stringente. I flussi generati dai sensori ad ultrasuoni sono stati mappati nella classe B in quanto hanno un periodo molto grande (100ms) e quindi le garanzie di latenza fornite da tale classe sono sufficienti.

2.3. Configurazione idleSlope

Gli idleSlope sono stati settati per tutte le porte Ethernet di trasmissione dei diversi end-node e switch attraversati da flussi SR. In particolare, in ciascuna porta, la somma degli idleSlope dei flussi SR che la attraversano deve essere minore di 100Mbps.

I valori di idleSlope impostati sono riportati nella seguente tabella:

Parametri	idleSlope
Datarate della porta	100Mbps
LD[1-2] -> SW1 idleSlope(A)	19.328Mbps
CU -> SW1 idleSlope(A)	18.688Mbps
US[1-2] -> SW1 idleSlope(B)	7.36Mbps
CM1 -> SW1 idleSlope(A)	28.288Mbps
SW1 -> S[1-2] idleSlope(A)	7.808Mbps
SW1 -> CU idleSlope(A)	57.344Mbps
SW1 -> CU idleSlope(B)	29.44Mbps
SW1 -> HU idleSlope(A)	75.264Mbps
ME -> SW2 idleSlope(A)	31.232Mbps
ME -> SW2 idleSlope(B)	47.488Mbps
US[3-4] -> SW2 idleSlope(B)	7.36Mbps
SW2 -> RS[1-2] idleSlope(B)	23.744Mbps
SW2 -> S[3-4] idleSlope(A)	7.808Mbps
SW2 -> SW1 idleSlope(A)	43.904Mbps
SW2 -> SW1 idleSlope(B)	14.72Mbps
RC -> SW2 idleSlope(A)	28.288Mbps

2.4. Dettagli implementativi

La rete implementata interconnette 18 end-node attraverso l'uso di 2 switch (SW1, SW2). Ogni end-node, inoltre, possiede un numero di applicazioni pari al numero di flussi che devono inviare/ricevere e ciascuna applicazione è di tipo *EthernetApplication*. In particolare, ogni nodo può gestire flussi in uscita e in entrata settando *true* i parametri booleani, rispettivamente, *hasOutgoingStreams* *hasIncomingStreams*.

La rete è descritta dalle seguenti classi:

- **EthernetApplication:** Simple Module che astrae ciascuna applicazione associata ad un nodo; ovvero, la classe rappresenta un'applicazione che invia o riceve dati sulla rete, può agire sia come trasmettitore che come ricevitore a seconda del valore del parametro *startTime* (per convenzione, se impostato a -1s l'applicazione funge solamente da ricevitore). Tale astrazione permette all'applicazione di interagire con la rete senza dover gestire direttamente i dettagli di basso livello. Nello specifico, si occupa della generazione delle frame Ethernet e del calcolo statistiche (ritardo end-to-end e jitter). Tramite il parametro *priority*, si può assegnare la priorità ai diversi flussi di traffico, questo valore viene utilizzato per impostare il *pcp* nel TAG, sulla base del quale i vari flussi vengono mappati nella coda relativa alla corrispondente classe di traffico. Un altro parametro fondamentale è l'**MTU** (Maximum

Transmission Unit) utilizzato per il calcolo dell'idleSlope relativo al Credit Based Shaper (CBS).

I diversi flussi generati da ciascuna applicazione vengono identificati attraverso il parametro *name* tramite il quale, assieme al parametro *remoteAddress* ciascun flusso viene associato alla relativa applicazione a cui è destinato.

- **TSNSocket:** Classe che eredita da *SocketBase*, la quale funge da interfaccia tra *EthernetApplication* e *TSNNetLayer*. Utilizza un'interfaccia di callback (la "ICallback") per notificare eventi relativi al socket, tra cui l'arrivo dei dati, gli errori e la chiusura del socket. Si occupa di associare un nome di flusso (*flowname*) all'applicazione durante la fase di *bind*. Questo nome di flusso è cruciale per il *TSNNetLayer* per indirizzare correttamente i pacchetti all'applicazione corretta.
- **TSNNetLayer:** Modulo che nasce come conseguenza alla mancanza che ha Ethernet: di base, non ha un livello di trasporto con cui poter mappare direttamente un messaggio con l'applicazione. Agisce come un livello di rete (fittizio) che gestisce i socket e l'instradamento dei pacchetti in base al nome del flusso. Si registra come protocollo TSN; ovvero, i *message dispatcher*, quando ricevono un messaggio che ha protocol type TSN, lo inviano a tale modulo. Mantiene una mappa che associa gli ID socket ai nomi dei flussi: quando riceve un pacchetto, lo esamina per trovare il socket corrispondente sulla base del nome del flusso e lo indirizza di conseguenza. Inoltre, gestisce la registrazione del protocollo TSN con i dispatcher di messaggi, i quali aggiungono ai frame un tag di protocollo.

2.5. Metriche valutate

Le metriche valutate (misurate nell'application layer) sono:

- **end-to-end delay** (per ogni flusso): tempo totale impiegato da un pacchetto per viaggiare dalla sorgente alla destinazione, includendo anche eventuali ritardi intermedi.

$$\text{e2eDelay} = \text{RxTime} - \text{GenTime}$$

Nella formula, RxTime fa riferimento all'istante di ricezione del pacchetto a livello applicativo mentre con GenTime si fa riferimento all'istante di generazione del pacchetto da parte dell'applicazione.

- **jitter assoluto** (flussi con payload $\leq 1500B$): metrica attraverso cui si identifica la differenza temporale più ampia tra il momento in cui il pacchetto con il tempo di fine più lungo ha raggiunto la sua destinazione e quello in cui è arrivato il pacchetto con il tempo di fine più breve.

$$\text{AbsJitter} = \max(\text{e2eDelay}) - \min(\text{e2eDelay})$$

- **numero di frame in coda** (di trasmissione, per ogni coda)

3. RISULTATI SIMULAZIONI

Per valutare le prestazioni della rete è stata effettuata una simulazione di 115s, assumendo che tutti i nodi della rete inizino a trasmettere nello stesso momento, ovvero i flussi sono in fase tra loro (*startTime* = 1s), si è scelto quindi di valutare la rete nel caso peggiore di massima interferenza.

Per quanto riguarda la statistica relativa al numero di frame in coda nelle code di trasmissione, si può osservare che nessuna di

queste raggiunge un numero elevato: ciò è dovuto alla scelte fatte relativamente alla mappatura dei flussi; infatti, tutti i flussi per i quali si ha un numero elevato di frame sono stati assegnati ad una classe SR. Pertanto, per ognuno di essi è stato impostato un valore di *interarrivalTime* sulla base dei valori di CMI definiti dallo standard IEEE802.1Q (125 µs per la classe SR A e 250 µs per SR B). Di conseguenza, le diverse frame dei flussi con i payload più grandi non arrivano in una coda tutti insieme e ciò fa sì che nelle diverse code di trasmissione si accodino poche frame. Quanto appena spiegato è riscontrabile nei grafici seguenti, i quali mostrano le code interessate da più flussi (per questioni di leggibilità, si riportano solo i primi 6s).

Per quanto concerne il nodo ME (Figura 3), questo genera quattro flussi audio verso gli speaker (flusso 2) e due flussi video verso i tablet posteriori (flusso 6). Com'è possibile osservare in Figura 4, nell'istante in cui parte la simulazione, il numero di frame in coda è 6 in quanto abbiamo quattro frame relative al flusso 2 e due relative al flusso 6. La stessa situazione si ripete ogni 250 µs in quanto tale valore corrisponde sia al periodo del flusso 2 che all'*interarrivalTime* del flusso 6.

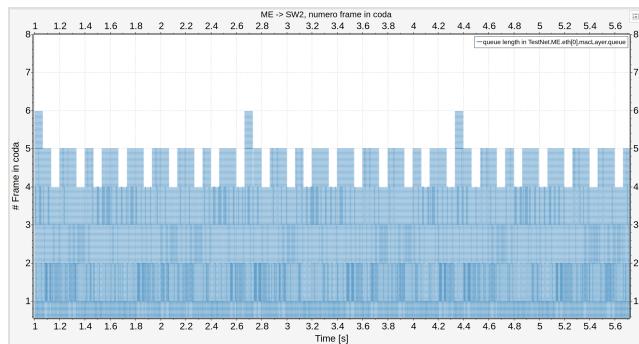


Figura 3: Frame in coda ME -> SW2

A seguire vi è una figura che mostra lo stesso andamento in una finestra temporale molto ristretta:

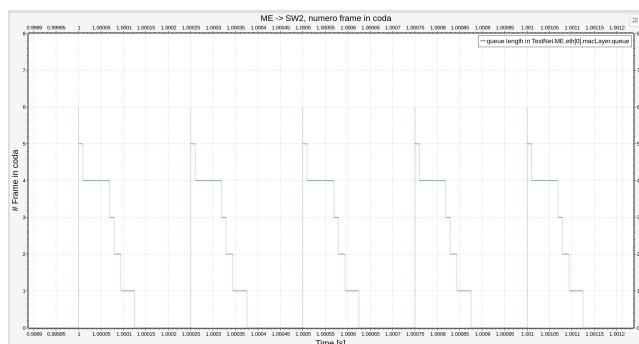


Figura 4: zoom frame in coda ME -> SW2

Per quanto riguarda, invece, il collegamento che va dallo Switch 1 alla Control Unit, esso è attraversato dai flussi 1,3 e 7. Anche qui, come si può osservare dalla Figura 5, il numero delle frame in coda si mantiene basso (max 4) grazie al fatto che vengono trasmesse frame con payload molto piccoli (per i valori di mtu scelti

per i singoli flussi) e anche perché, per ogni flusso, le frame non arrivano mai a burst ma una alla volta. Analizzando il grafico in Figura 5 si può notare che, periodicamente, nella coda si accumulano al massimo quattro frame (fenomeno visibile con maggiore chiarezza nella Figura 6). In particolare, questa situazione si verifica quando vengono trasmesse le frame relative ai flussi 1 e 3 a causa del fatto che le frame del flusso 3 in quell'istante hanno un credito negativo e quindi rimangono temporaneamente "bloccate" nella relativa coda (come è possibile osservare nella Figura 7).

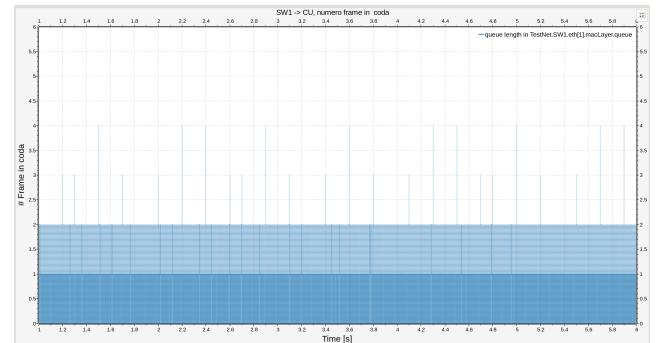


Figura 5: Frame in coda SW1 -> CU

Anche in questo caso, di seguito, viene mostrato lo stesso andamento in una finestra temporale molto ristretta:

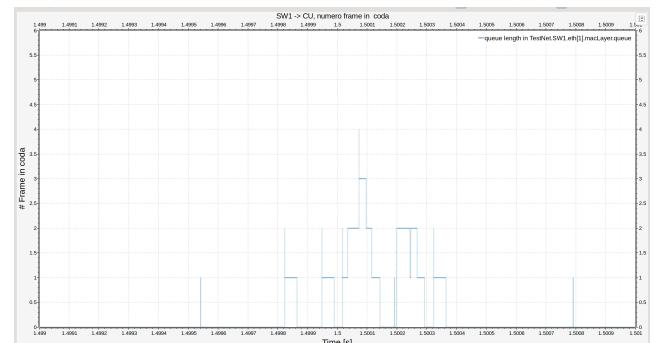


Figura 6: zoom frame in coda SW1 -> CU

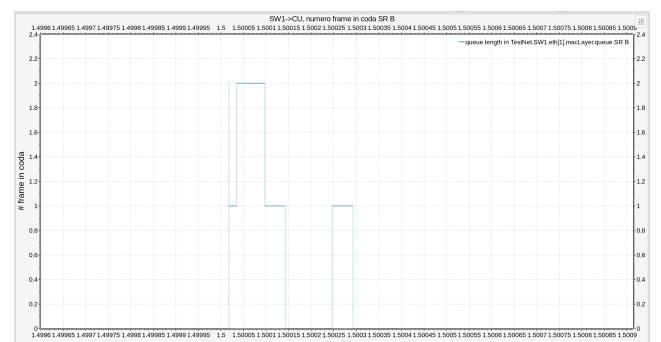


Figura 7: Frame nella coda SR B di (SW1 -> CU)

Anche nel caso dei collegamenti SW2 -> SW1 e SW1 -> HU, i quali sono attraversati da diversi flussi, il comportamento è simile a quello mostrato per il collegamento SW1 -> CU. L'andamento generale è mostrato nelle Figure 8 e 9.

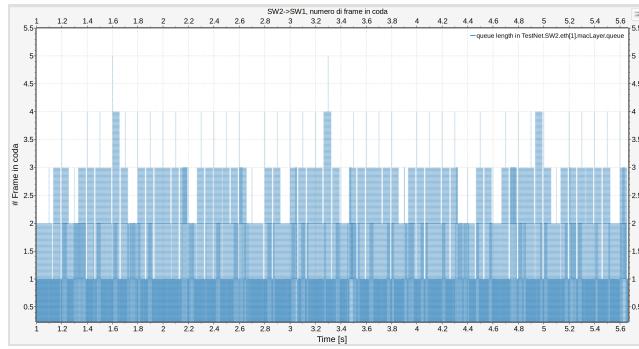


Figura 8: Frame in coda SW2 -> SW1

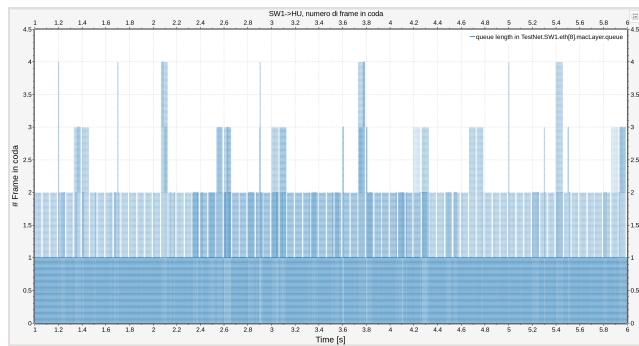


Figura 9: Frame in coda SW1 -> HU

Nell'analizzare i risultati della simulazione condotta con OM-NeT++ relativi all'end-to-end delay e al jitter, è importante sottolineare che la raccolta dei dati è stata eseguita a livello applicativo, in modo da avere una visione diretta del modo in cui le prestazioni della rete influenzino l'esperienza dell'utente finale.

Partendo dall'analisi dei flussi 2 (generati da ME ed indirizzati agli speaker), si è deciso di mappare tali flussi nella classe SR A e, allo stesso tempo, di mappare i flussi 6 (flussi video generati da ME e destinati ai tablet posteriori) nella classe SR B; questo affinché i primi, avendo una priorità maggiore dei secondi, subiscano meno ritardi dovuti all'accodamento delle frame che si verifica nella coda di trasmissione ME->SW2 (Figura 3). A seguito di tale scelta, si è riuscito a garantire il rispetto delle deadline relative per i flussi audio, come è possibile vedere nelle Figure 10, 11, 12, 13. Inoltre, si può notare il jitter per i flussi 2 è piuttosto variabile, questo perché i messaggi di tale flusso sono trasmessi in singole frame.

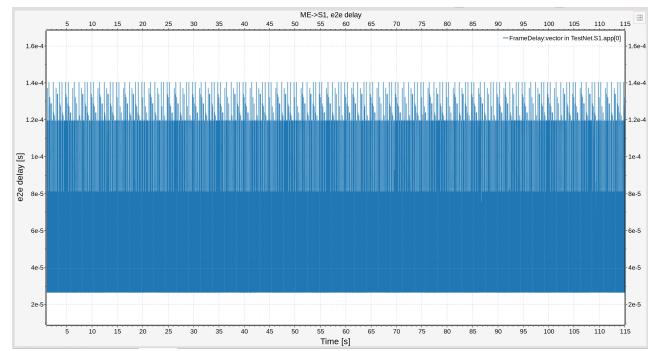


Figura 10: E2ED ME-> S1

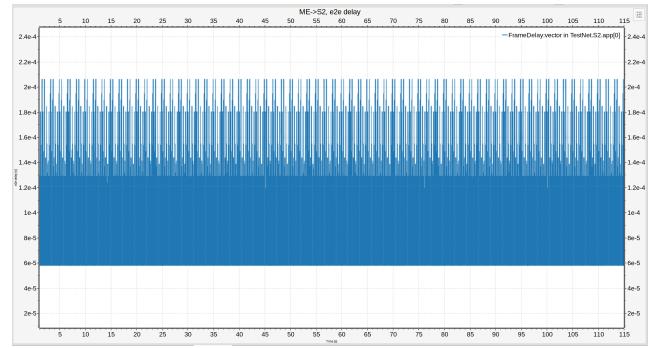


Figura 11: E2ED ME-> S2

Confrontando i grafici in Figura 10,11 con quelli in Figura 12,13 è possibile notare che gli ultimi hanno una minor "ampiezza" dei valori, la quale si traduce in un miglior jitter: ciò si verifica perché i flussi diretti agli speaker S1 ed S2 attraversano il collegamento SW2->SW1 il quale è attraversato anche da altri flussi (flusso 3, flusso 8, flusso 7), mentre i flussi diretti agli speaker posteriori (S3,S4) no.

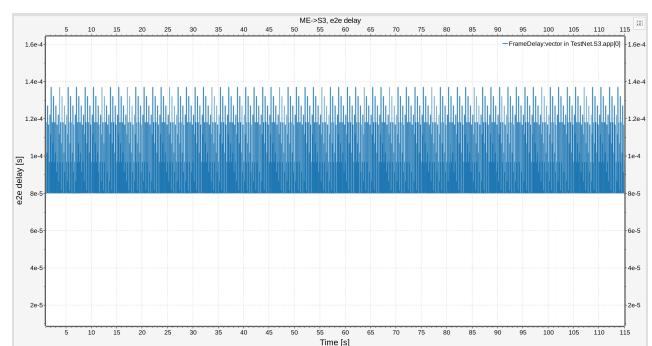


Figura 12: E2ED ME-> S3

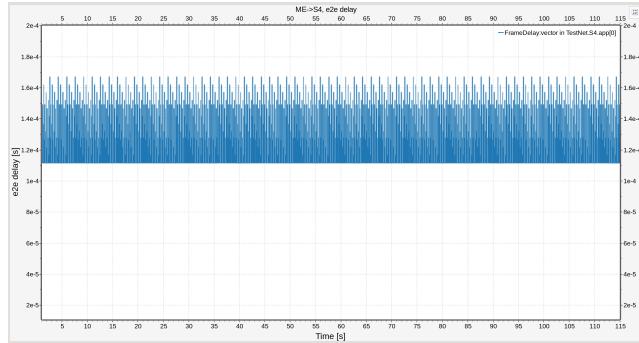


Figura 13: E2ED ME-> S4

Nonostante la scelta di mappare i flussi 6 nella classe SR B, si è riusciti a rispettare comunque la deadline relativa dei due flussi (come si osserva nelle Figure 14,15). Il motivo di ciò è correlato alla scelta del valore di MTU; specificamente, si è scelto un valore tale per cui, sfruttando l'equazione 2.4 (delay relativo alla trasmissione dell'ultima frame), si è ottenuto un valore di 63.5ms, al quale va aggiunto il tempo necessario a trasmettere l'ultima frame di tale flusso e, mettendoci nel caso peggiore, anche quello necessario a trasmettere 4 frame del flusso 2 (le quali avendo maggiore priorità e materializzandosi insieme nella relativa coda vengono trasmesse prima). I tempi di trasmissione sono stati calcolati utilizzando la seguente formula:

$$T = \frac{(\text{payload} + \text{overhead_ethernet}) \times 8}{\text{bitrate}} \quad (3.1)$$

In particolare, si è considerata come bitrate l'idleSlope relativo al flusso (47.488Mbps) ottenendo un tempo di trasmissione di 125 μs per quanto riguarda l'ultima frame, e un tempo pari a 125 μs relativo al tempo totale di trasmissione delle 4 frame del flusso 2 (qui si considera come bitrate 31.232Mbps); per un totale pari a 250 μs. Dunque, sommando 250 μs a 63.5ms, si ottiene 63.75ms, da intendere come limite superiore, il quale è abbondantemente sotto la deadline relativa. Si puntualizza che tale verifica è stata eseguita in quanto la classe SR B garantisce una latenza massima di 50ms e, quindi, non era sufficiente a scongiurare la miss della deadline.

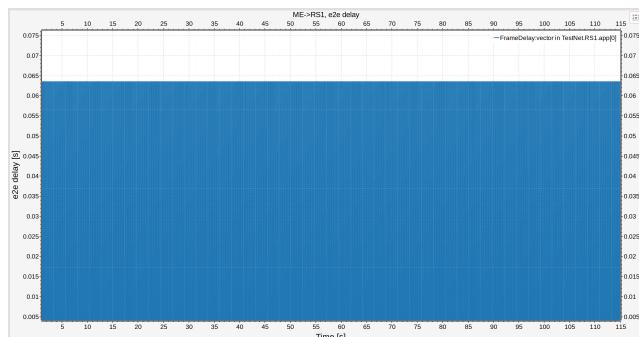


Figura 14: E2ED ME-> RS1

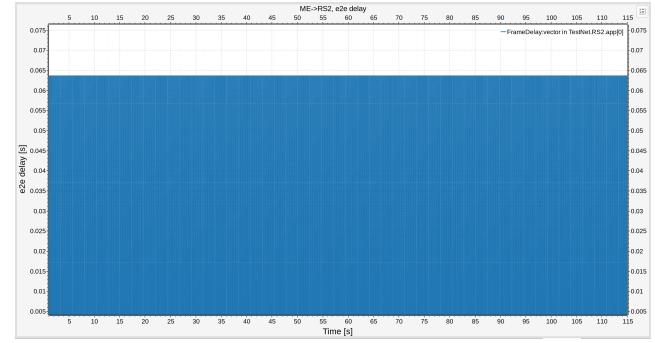


Figura 15: E2ED ME-> RS2

Si è scelto di mappare il flusso 8 assegnando un MTU pari a 400 byte, e si è calcolato con l'equazione 2.4 che l'ultima frame viene trasmessa dopo 55.5ms; la classe SR A garantisce fino a 7 hop una latenza massima di 2ms e, pertanto, come è possibile vedere in Figura 16, la deadline viene rispettata anche in questo caso.

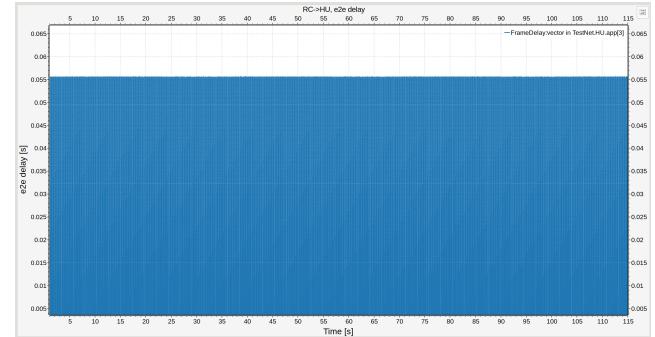


Figura 16: E2ED RC-> HU

Per quanto riguarda i sensori ad ultrasuoni, sono stati mappati nella classe SR B in quanto generano traffico audio con una deadline relativa abbastanza ampia (100ms), per cui le garanzie offerte dalla classe SR B (latenza massima di 50ms) sono più che sufficienti. Inoltre, come si può osservare dalle Figure 17,18,19,20, i valori di end-to-end delay ottenuti sono molto bassi (dell'ordine dei micro secondi) questo grazie al piccolo numero di frame nella coda di trasmissione SW1-> CU (Figura 6).

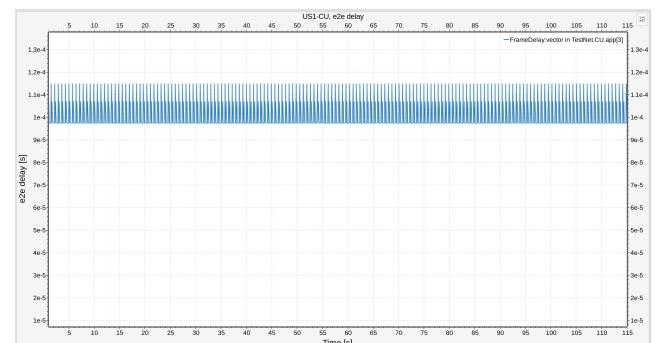


Figura 17: E2ED US1 -> CU

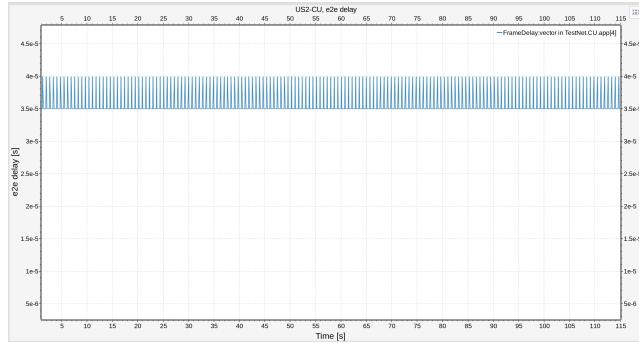


Figura 18: E2ED US2 -> CU

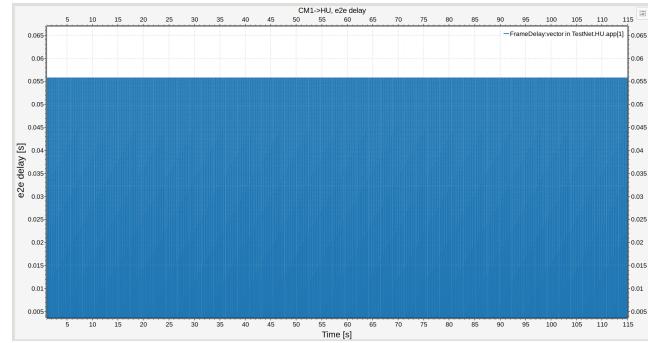


Figura 21: E2ED CM1 -> HU

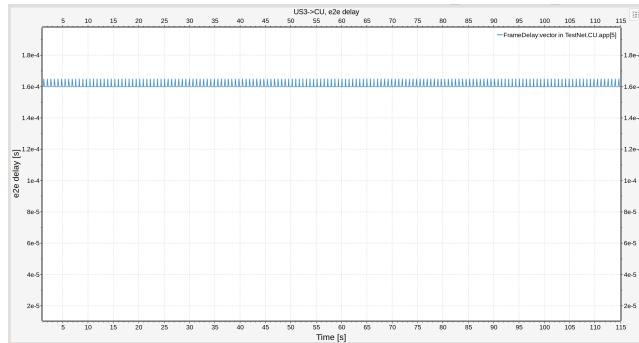


Figura 19: E2ED US3 -> CU

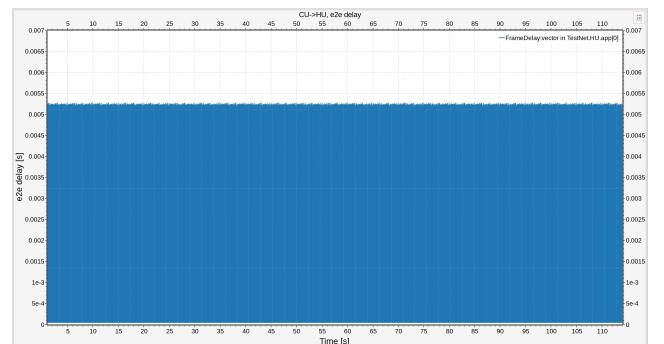


Figura 22: E2ED CU -> HU

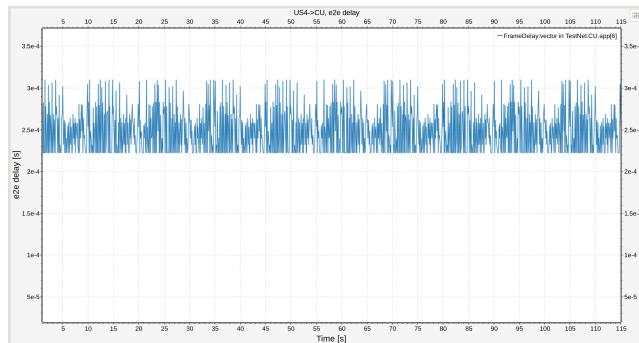


Figura 20: E2ED US4 -> CU

Anche per quanto riguarda il flusso 5 (generato dalla telecamera anteriore e diretto ad HU) è stato seguito lo stesso ragionamento discusso per il flusso 8. Qui, con un valore di MTU pari a 400 byte si è ottenuto un delay di 55.75ms, il quale, sommato al limite superiore di latenza garantito dalla classe A, rientra nella deadline relativa di 66.66ms (come si vede in figura 21).

Per il flusso 4 (generato dalla Control Unit) si è scelta la classe SR A in quanto, essendo un flusso Real-time safety critical, necessita di garanzie sulla latenza ed, avendo un periodo di 10ms, si sono ritenute necessarie le più strette garanzie di latenza (offerte appunto dalla classe A), infatti in questo caso applicando la formula 2.4 si è ottenuto un delay relativo all'ultima trasmissione di 5.125ms e come si vede in Figura 22 l'end-to-end delay è al massimo di 5.269ms.

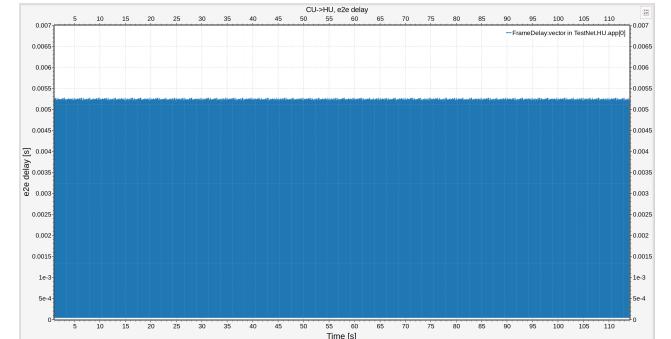


Figura 23: zoom E2ED CU -> HU

Inoltre, allargando l'asse temporale relativo al grafico dell'end-to-end delay del flusso 4 (Figura 23), è evidente come per questo flusso (così come per gli altri flussi che sono segmentati in numerose frame, Figure 14,15,16,21) il jitter si mantenga piuttosto costante rispetto a quello relativo ai flussi trasmessi in singole frame.

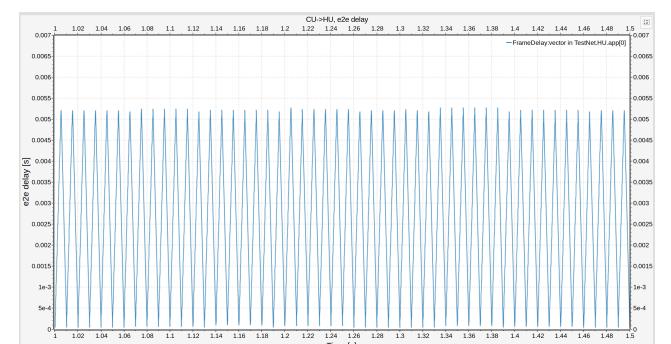


Figura 24: zoom E2ED CU -> HU

I lidar sono stati mappati nella classe SR A. Qui, è stato usato un MTU di 260 byte poiché, se si fosse mantenuto il payload originale, avremmo avuto un valore di idleSlope di 85.88Mbps per un singolo flusso (troppo alto). Si è ottenuto un ritardo di trasmissione (dell'ultima frame) pari a 0.5ms e, applicando l'equazione 3.1, si ha un tempo di trasmissione pari a 24.16 μ s. Applicando lo stesso ragionamento fatto per quanto concerne i flussi 6 generati da ME, qui otteniamo un *upper bound* pari a 912.77 μ s, quindi si rispetta la deadline (come osservabile nelle Figure 24, 25).

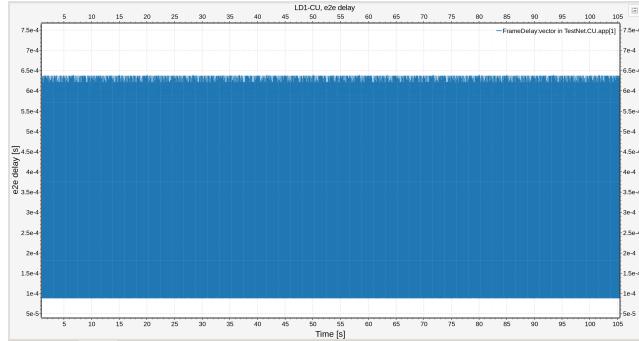


Figura 24: E2ED LD1 -> CU

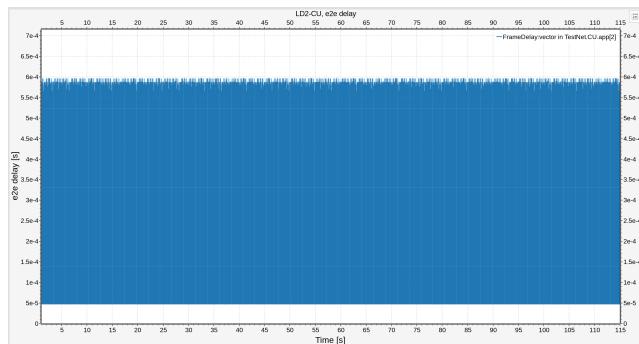


Figura 25: E2ED LD2 -> CU

Il flusso 7 (generato dal TLM) è stato mappato nella classe best-effort in quanto soft Real-Time; nonostante questa scelta, anche in questo caso non vi sono deadline miss (come si osserva nelle Figure 27,26), grazie al fatto che le code di trasmissione dei collegamenti SW2->SW1, SW1 -> CU e SW1 -> HU non raggiungono mai un numero elevato in quanto tutti gli altri flussi sono stati mappati nelle classi SR.

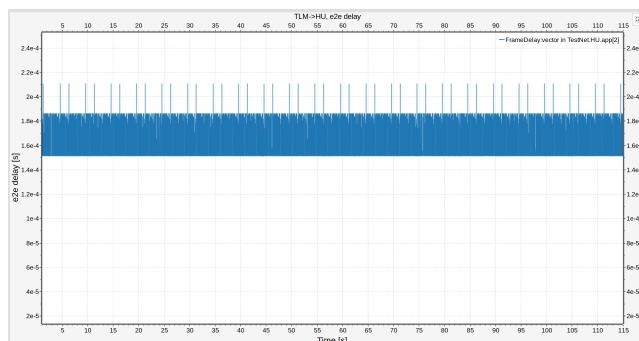


Figura 26: E2ED TLM -> HU

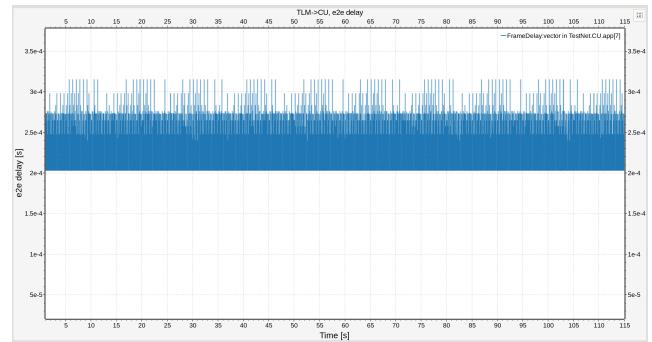


Figura 27: E2ED TLM -> CU

Di seguito una tabella nella quale, per ogni flusso con payload inferiore a 1500 byte, è riportato il valore di jitter assoluto e quello del massimo end-to-end delay:

Flusso	MAX E2EDDelay (μ s)	Jitter Assoluto (μ s)
LD1 -> CU	639	50
LD2 -> CU	597	50
ME -> S1	140	114
ME -> S2	206	148
ME -> S3	137	57
ME -> S4	167	56
US1 -> CU	115	17
US2 -> CU	40	5
US3 -> CU	165	5
US4 -> CU	310	87
TLM -> CU	315	113
TLM -> HU	211	60

È possibile osservare (come ci si aspettava dato il basso numero di frame in coda, Figure 8 e 9) che, per i diversi flussi, sono stati registrati valori di jitter molto piccoli (nell'ordine del micro secondo) ed, inoltre, i valori massimi di end-to-end delay si mantengono al di sotto delle rispettive deadline relative; in aggiunta, si può notare che i valori di jitter sono piuttosto variabili: ciò accade perché i flussi sono trasmessi in singole frame ed il meccanismo del CBS è più efficiente con flussi suddivisi in più frame. Il CBS tenta, infatti, di compensare il ritardo per una frame (appartenente a una classe SR) attraverso il meccanismo del credito (idleSlope): più frame vengono trasmesse consecutivamente (il valore del credito aumenta) se una frame subisce un ritardo a causa della trasmissione in corso di una frame non SR. Se, invece, un messaggio ha un payload di dimensioni tali da stare in una singola frame Ethernet, il meccanismo CBS è meno efficace nel mitigare il ritardo. Come è possibile osservare nelle Figure 14, 15,16, 21, 22 ed in particolar modo dalla Figura 23, i flussi che vengono segmentati in numerose frame hanno un jitter meno variabile rispetto a quelli che vengono trasmessi in singole frame (Figure 10, 11,12,13,17,18,19,20).