UNIVERSITÀ degli STUDI di CATANIA

Corso di Industrial and Automotive Real-Time Networks

Elaborato OMNeT++

Scenario 1 Variante 2

Giorgia Arena (1000069961),
Giorgia Grazia Mucciarella (1000069228),
Alessio Tornabene (1000069970).

I. Introduzione

Si vuole simulare una rete Switched Ethernet a 1 Gbps per un'applicazione automotive con traffico cross-domain. La rete e` composta da 18 end-node e due switch e trasporta dati per i domini:

- ADAS (Advanced Driver Assistance Systems);
- Multimedia/Infotainment;

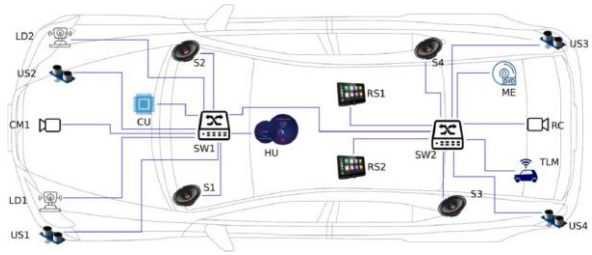

Figura 1: Schema scenario implementato

A. *Descrizione flussi*

I flussi scambiati dai vari end-node sono riportati nella tabella I. Nella seguente trattazione sono state implementate due varianti del protocollo TSN:

- **Caso 1** : Mappare il traffico nelle classi SR (A o B) o Best-effort.
- **Caso 2** : Mappare il traffico utilizzando solo Strict Priority Selection(Deadline Monotonic)

Sono state valutate le seguenti metriche:

- ***Absolute Jitter:*** è il massimo scostamento tra il momento previsto di arrivo di un pacchetto e l'istante di arrivo effettivo:

$$absJitter = \max(e2eDelay) - \min(e2eDelay)$$

- ***End -to-end Delay:*** è il tempo di trasmissione, misurato a livello applicazione, tra il nodo sorgente e il nodo destinazione:

$$e2eDelay = RxTime - GenTime$$

- ***Numero massimo di frame in coda:***

rappresenta il numero massimo di frame che possono essere temporaneamente immagazzinati in una coda. Questa misura è fondamentale per comprendere la capacità della rete di gestire il traffico senza perdita di dati o congestione.

Tutte le metriche sopra elencate sono state misurate per ogni flusso di rete.

II. Scenario

Lo scenario considerato, mostrato in Figura 1, consiste di due differenti domini funzionali automotive:

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

I sistemi ADAS possono essere *Camera Based,* rappresentati dai nodi CM1 e RC , rispettivamente camera frontale e posteriore, oppure *Sensor Based*, rappresentati dai sensori ad ultrasuoni US1, US2, US3 e US4 e dai sensori lidar LD1 e LD2.
I dati raccolti dalle telecamere vengono inviati al dispositivo HU chiamato Head Unit, in modo da poter essere osservati dal guidatore. I dati raccolti dai sensori ad ultrasuoni e dai sensori lidar vengono invece inoltrati all'unità di controllo CU che si occupa di processare le informazioni ed inoltrarle al guidatore tramite il nodo HU.
Nello scenario considerato, la videocamera anteriore (CM1) opera ad un frame rate pari a 60 fotogrammi al secondo (fps) e genera una frame di 178500 byte con periodo pari a 16.66ms. La camera posteriore(RC), usata durante le fasi di manovra, opera ad un frame rate pari a 30 fps e genera una frame di 178000 byte con un periodo che risulta essere pari a 33.33ms.
Lo scenario include, inoltre, un sistema multimediale composto da quattro altoparlanti (S1, S2, S3, S4) e due *Rear Seat Entertainment* (RS1, RS2).
Il nodo *Multimidia Entertainment* (ME) trasmette un flusso video per ogni *Rear Seat* (RS1, RS2) a 30 fps e un flusso audio ad ogni altoparlante (S1-S4) con periodo 250µs.
Lo scenario prevede anche un *sistema di telematica* (TLM) che trasmette informazioni all'HU e alla CU quali: avvisi sul traffico, dati GPS, mappe, etc.
I sistemi Multimedia/infotainment e telematica sono real-time ma non safety-critical.

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

Tabella I: Caratteristiche flussi di rete

B. *Configurazioni in esame*

Sono state simulate le seguenti configurazioni:

- **Caso 1: Mappare il traffico nelle classi SR (A o B) o Best-effort.**

In questa configurazione viene utilizzato lo *Stream Reservation*(SR) definito nello standard IEEE 802.1 Qat e il *Credit Based Shaping* (CBS) definito nello standard IEEE 802.1 Qav. Nello standard IEEE 802.1Q, che include il CBS e l'SR, ogni porta Ethernet fornisce un massimo di otto code ordinate in base alle priorità crescenti. Ogni volta che il canale è libero, l'algoritmo Strict Priority seleziona per la trasmissione la frame in testa alla coda a più alta priorità. Questo meccanismo consente fino a 8 priorità di frame.
Il traffico SR è sottoposto al traffico shaping, poiché il

Credit-Based Shaping (CBS) viene applicato alle porte di uscita di entrambi gli switch e dei nodi finali. Lo standard IEEE 802.1Q prevede più classi di traffico :

- ***Classe A***, che fornisce una latenza massima di 2 ms su 7 hop.
- ***Classe B,*** che fornisce una latenza massima di 50 ms su 7 hop.

E a queste va aggiunta un' ulteriore classe di traffico a bassa priorità chiamata ***Best-Effort*** (BE), cioè gestita senza garanzie di tempi e di consegna. Il traffico BE non è sottoposto a traffic shaping.
Secondo il CBS, ad ogni classe di traffico SR è associato un parametro di credito. Le frame contenute nelle code SR possono essere trasmesse solo quando il credito associato è non negativo. Durante la trasmissione del messaggio, il valore del credito diminuisce alla velocità *sendSlope* definita per la classe, mentre il credito aumenta alla velocità costante *idleSlope* definita per la classe quando le frame di quella classe sono in attesa di trasmissione su un canale occupato o quando le frame non sono in attesa, ma il credito è negativo. Se il credito non è negativo, ma non ci sono frame della classe in attesa di trasmissione, il credito viene azzerato automaticamente. La formula per calcolare l'*idleSlope* è la seguente:

$$idleSlope_f = \frac{(MFS + PFO) \times MIF}{CMI}$$

Formula 1: idleslope

Dove *MFS* (Max Frame Size) è la dimensione massima presunta del payload, PFO è l'overhead Ethernet di 42byte, MIF (MaxIntervalFrames) è un valore intero che rappresenta il massimo numero di frame di un flusso che può essere trasmesso in un CMI(dove CMI è 125us per la classe A e 250us per la classe B)

- **Caso 2: Mappare il traffico utilizzando solo Strict Priority Selection(Deadline Monotonic):**

In questa configurazione viene utilizzato un algoritmo di schedulazione statica, *Deadline Monotonic*, che usa le deadline relative per stabilire le priorità mediante la seguente relazione:

$$P_i \propto \frac{1}{D_i}$$

Si è scelto di utilizzare *Deadline Monotonic* perché offre maggiore flessibilità per gestire cambiamenti nei requisiti di temporizzazione; è più robusto a modifiche future dove le deadline potrebbero differire dai periodi e inoltre è più efficiente poichè gestisce meglio le priorità in presenza di flussi con scadenze diverse dai loro periodi.
Le priorità ottenute risultano essere descritte nella Tabella II:

SRC	DST	Deadline Relativa	Priorita`
LD1, LD2	CU	1.4 ms	5
ME	S1, S2, S3, S4	250 μs	7
US1, US2, US3, US4	CU	100 ms	0
CU	HU	10 ms	4
CM1	HU	16.66 ms	3
ME	RS1, RS2	33.33 ms	1
TLM	HU, CU	625 μs	6
RC	HU	33.33 ms	2

Tabella II: Priorità DM

C. *Dettagli implementativi*

La rete, risulta essere composta da 18 end-node visualizzabile nel file *TestNet.ned*, interconnessi tramite due switch chiamati SW1 e SW2 (vedi Figura 2). All' interno degli end-node, chiamati TsnDevice in OMNeT++, si possono trovare i seguenti parametri della rete, settati nel file *omnet.ini :*

- *hasIncomingStreams:* è un parametro che indica se un nodo ha flussi di dati in ingresso ed è utilizzato per configurare e gestire le applicazioni o i componenti del nodo in modo appropriato a seconda che ci siano flussi di dati in ingresso. Se impostato a *true*, il nodo è configurato per ricevere dati da altri nodi della rete.
- *hasOutgoingStreams:* indica se un nodo ha flussi di dati in uscita. Se impostato a *true*, il nodo è configurato per generare e trasmettere dati ad altri nodi della rete.
- *hasEgressTrafficShaping:* indica se un'interfaccia di rete è configurata per eseguire il traffic shaping sui flussi di dati in uscita. Il traffic shaping in uscita è associato agli standard di rete come IEEE 802.1Qav (Credit-Based Shaper), che è utilizzato per garantire la qualità del servizio per i flussi di dati AV (Audio-Video) nelle reti TSN (Time-Sensitive Networking). Quindi se è impostato a *true*, l'interfaccia di rete esegue il traffic shaping sui pacchetti in uscita.

- *startTime:* è utilizzato per specificare il momento in cui un'applicazione o un modulo deve iniziare la propria attività all'interno della simulazione. Nell'elaborato in questione il parametro viene settato a 1s.
- *period* e *payload:* sono rispettivamente il periodo di una frame e la lunghezza di una frame. In questo caso il periodo è uguale alla deadline relativa, motivo per cui si è trascurato questo parametro.
- *priority:* è utilizzato per assegnare una priorità specifica alle applicazioni o ai pacchetti. Questa priorità è fondamentale per determinare l'ordine di gestione del traffico, sia nei moduli di rete che negli algoritmi di schedulazione come il *Credit-Based Shaping* (CBS) e il *Deadline Monotonic Scheduling.* Nel primo caso il CBS utilizza la priorità per classificare il traffico in diverse classi di servizio, come Classe A e Classe B e Best-Effort. E' importante notare che lo standard 802.1Q mette la priorità con valore 1 maggiore di quella con 0:
  -Il valore 0 corrisponde alle frame Ethernet non taggate VLAN ,
  -il valore 1 lo considera come traffico di background che ha una priorità minore di quella best effort ma è una questione di mapping di default.
  Quindi si è deciso nell'elaborato di scambiare la priorità 0 con 1.
  Nell' AVB i pacchetti con priorità alta , come i flussi video e audio sono mappati in classe A e possono essere trasmessi quando il credito è sufficiente. I flussi multimedia e infotainment sono stati assegnati alla classe B, mentre il traffico telematico (TLM) è stato collocato nella classe Best-Effort. In seguito nella Tabella III vi è la configurazione delle priorità del caso 1:

SRC	DST	**Classe**	Priorita`
LD1, LD2	CU	A	7
ME	S1, S2, S3, S4	A	7
US1, US2, US3, US4	CU	A	7
CU	HU	B	6
CM1	HU	A	7
ME	RS1, RS2	B	6
TLM	HU, CU	BE	0
RC	HU	A	7

Tabella III: Configurazione priorità CBS

Nel caso del DM, invece i task con scadenze più brevi ricevono priorità più alta (vedi Tabella II).

- *interarrivalTime :* la frequenza con cui i pacchetti vengono generati e trasmessi in una rete. Questo parametro aiuta a modellare e analizzare il carico di rete, la latenza e la larghezza di banda. Nei meccanismi di *traffic shaping* come il CBS, questo parametro assume diversi valori a seconda della classe di appartenenza : per la classe A questo valore risulta essere pari a 125us, invece per la classe B è settato a 250us.

- *remoteAddress:* viene utilizzato per specificare l'indirizzo della destinazione a cui un'applicazione o un nodo di rete deve inviare i propri dati. Questo parametro è fondamentale per la configurazione delle comunicazioni di rete, consentendo di indirizzare correttamente i pacchetti verso il nodo di destinazione desiderato.

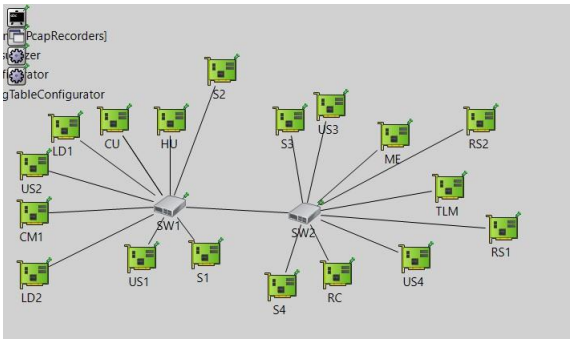

Figura 2: TestNet.ned-Configurazione di rete

- *mtu:* è un parametro di rete che specifica la dimensione massima, in byte, di un singolo frame di dati che può essere trasmesso su una rete. Quindi permette di frammentare una frame con un payload troppo grande >= 1500B(default).Nel caso 2 ovvero quello con il Credit Based Shaping , tale valore viene settato nel seguente modo:

  1) LD1,LD2-->CU , le frame di tali flussi in trasmissione vengono settate con un valore di mtu pari a 1300B; calcolando il numero di frame trasmesse come Num.Frame=Payload/MTU=1 e moltiplicando tale valore per l'interarrivalTime della classe A ovvero 1*125us questo risulta essere minore della deadline relativa che per questi flussi è pari a 1.4 ms.
  2) ME-->S1,S2,S3,S4 , le frame di tali flussi in trasmissione vengono settate con un valore di mtu pari a 80B; calcolando il numero di frame trasmesse come Num.Frame=Payload/MTU=1 e moltiplicando tale valore per l'interarrivalTime della classe A ovvero 1*125us questo risulta essere minore della deadline relativa che per questi flussi è pari a 250us.
  3) US1,US2,US3,US4-->CU , le frame di tali flussi in trasmissione vengono settate con un valore di mtu pari a 188B; calcolando il numero di frame trasmesse come Num.Frame=Payload/MTU=1 e moltiplicando tale valore per l'interarrivalTime della classe A ovvero 1*125us questo risulta essere minore della deadline relativa che per questi flussi è pari a 100 ms.

  Le frame degli altri flussi (A e B) hanno un valore di mtu pari al default(1500B) e rispettano le loro deadline, nonostante abbiano un payload molto grande.

- *idleslope:* questo parametro determina la velocità di trasmissione dei dati alla quale il traffico sarà limitato. Utilizzando la Formula 1 dell' idleslope si ricavano questi valori per i flussi in classe A, B e per le varie porte degli Switch:

  1) LD1,LD2-->CU (classe A)
  LD1: *idleslope=(1300+42)*8*8000= 85,88 Mbps <1Gbps*
  LD2: *idleslope=(1300+42)*8*8000= 85,88 Mbps<1Gbps*

  2) ME-->S1,S2,S3,S4 (classe A)
  ME: *idleslope=(80+42)*8*8000*4Flussi = 31.232 Mbps<1Gbps*

  3) *US1,US2,US3,US4-->CU (classe A)*
  US1*: idleslope=(188+42)*8*8000= 14.72 Mbps<1Gbps*
  US2*: idleslope=(188+42)*8*8000= 14.72 Mbps<1Gbps*
  US3: *idleslope=(188+42)*8*8000= 14.72 Mbps<1Gbps*
  US4*: idleslope=(188+42)*8*8000= 14.72 Mbps<1Gbps*

4) CU->HU (classe B)
CU: *idleslope=(1500+42)*8*4000= 49.344 Mbps<1Gbps*

5) CM1->HU (classe A)
CM1: *idleslope=(1500+42)*8*8000= 98.688 Mbps<1Gbps*

6) *ME->RS1,RS2(classe B)*
ME*: idleslope=(1500+42)*8*4000*2 flussi= 98.688 Mbps<1Gbps*

7) *RC->HU(classe A)*
RC: *idleslope=(1500+42)*8*8000= 98.688 Mbps<1Gbps*

Dato l'utilizzo degli Switch, che sono dispositivi che permettono la micro segmentazione, ogni link di 1Gbps è totalmente dedicato ad ogni singolo nodo della rete. Si noti come tutti i calcoli sopra non superano la grandezza di ogni singolo link della rete.
In Figura 3 troviamo l'implementazione del codice nel file *omnet.ini:*

```
# SR A
*.LD1.eth[*].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 85.88Mbps
*.LD2.eth[*].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 85.88Mbps
*.ME.eth[*].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 31.232Mbps
*.US*.eth[*].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 14.72Mbps
*.CM1.eth[*].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 98.688Mbps
*.RC.eth[*].macLayer.queue.transmissionSelectionAlgorithm[7].idlesSope= 98.688Mbps

#SR B
*.CU.eth[*].macLayer.queue.transmissionSelectionAlgorithm[6].idleSlope= 49.344Mbps
*.ME.eth[*].macLayer.queue.transmissionSelectionAlgorithm[6].idleSlope= 98.688Mbps
```

Figura 3: implementazione idleslope dei nodi

L'idleslope di ogni porta di uno Switch si trova sommando gli l'idleslope dei flussi ricevuti dai singoli nodi e notiamo anche qui che la velocità non supera quella del link ovvero 1Gbps.

SW1->CU
(SW1 invia i flussi LD1,LD2,US1,US2 di classe A su CU)
SW1: *idleslope=85,88*2+14.72*2= 201.2Mbps<1Gbps*

SW1->S1(SW1 invia il flusso ME di classse A su S1)
SW1*: idleslope=31.32/4= 7.808Mbps<1Gbps*

SW1->S2(SW1 invia il flusso ME di classse A su S2)
SW1: idleslope=31.32/4= 7.808Mbps<1Gbps

SW1->S3(SW1 invia il flusso ME di classse A su S3)
SW1: idleslope=31.32/4= 7.808Mbps<1Gbps

SW1->S4(SW1 invia il flusso ME di classse A su S4)
SW1: idleslope=31.32/4= 7.808Mbps<1Gbps

SW2->SW1(SW2 invia i flussi US3,US4,RC,ME*2flussi di classse A su SW1)
*SW2: idleslope=14.72*2+98.688+2*7.808= 143.744Mbps<1Gbps*

SW1->HU(SW1 invia i flussi CM1, RC di classse A e di classe B CU su HU)

*SW1: idleslope=98.688*2= 197.376Mbps(Classe A) <1Gbps*
SW1: idleslope=49.344Mbps(Classe B)<1Gbps

SW2->RS1(SW2 invia il flusso ME di classse B su RS1)
SW2: idleslope=98.688/2= 49.344Mbps<1Gbps

SW2->RS2(SW2 invia il flusso ME di classse B su RS2)

SW2: idleslope=98.688/2= 49.344Mbps<1Gbps

```
*.SW1.eth[2].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 201.2Mbps    #SW1->CU
*.SW1.eth[7].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 7.808Mbps    #SW1->S2
*.SW1.eth[8].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 7.808Mbps    #SW1->S1
*.SW1.eth[4].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 197.376Mbps  #SW1->HU


*.SW2.eth[1].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 143.744Mbps  #SW2->SW1
*.SW2.eth[2].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 7.808Mbps    #SW2->S3
*.SW2.eth[3].macLayer.queue.transmissionSelectionAlgorithm[7].idleSlope= 7.808Mbps    #SW2->S4

*.SW1.eth[4].macLayer.queue.transmissionSelectionAlgorithm[6].idleSlope= 49.344Mbps  #SW1->HU
*.SW2.eth[6].macLayer.queue.transmissionSelectionAlgorithm[6].idleSlope= 49.344Mbps  #SW2->RS2
*.SW2.eth[7].macLayer.queue.transmissionSelectionAlgorithm[6].idleSlope= 49.344Mbps  #SW2->RS1
```

Figura 4 :implementazione idleslope degli switch

In Figura 4, troviamo l'implementazione del codice nel file omnet.ini

III. RISULTATO SIMULAZIONI

Per valutare le prestazione della rete, sono state effettuate due simulazioni, una per il Caso 1: Mappare il traffico nelle classi SR (A o B) Best-effort e una per il Caso 2: Mappare il traffico utilizzando solo Strict Priority Selection(Deadline Monotonic) , ognuna di 10s. Sono state eseguite nel caso peggiore di massima interferenza dove tutti i flussi sono in fase (startTime= 1s).
Durante le simulazioni sono state valutate le seguenti metriche, così come descritto nell'introduzione:

1) End to End Delay
2) Absolute Jitter
3) Numero massimo di frame in coda

a)Simulazione Caso 1: End-to-End Delay

Il ritardo end-to-end viene calcolato come la differenza tra il tempo di ricezione del messaggio (RxTime), ovvero il momento in cui il messaggio viene ricevuto dal livello di applicazione della destinazione e il tempo di generazione del messaggio (GenTime), ovvero il momento in cui il messaggio è stato generato dall'applicazione. Il ritardo end-to-end (E2EDelay) è calcolato come:

$$e2eDelay = RxTime - GenTime$$

Nella rete valutata, i ritardi medi più bassi sono stati ottenuti nei flussi mappati in classe A come si nota in Tabella IV:

Flow name	E2E delay (s)	Absolute Jitter (s)
LD1_CU	0.000021	0.000035
LD2_CU	0.000075	0.000084
ME_S1	0.000019	0.000114
ME_S2	0.000027	0.000121
ME_S3	0.000005	0.000027
ME_S4	0.000006	0.000028
US1_CU	0.000004	0.000004
US2_CU	0.000013	0.000013
US3_CU	0.000057	0.000129
US4_CU	0.000068	0.000138
CU_HU	0.000777	0.001536
CM1_HU	0.007404	0.014821
ME_RS1	0.014768	0.029474
ME_RS2	0.014780	0.029487
RC_HU	0.007421	0.014806
TLM HU	0.000016	0.000041
TLM_CU	0.000021	0.000033

Tabella IV: Metriche medie dei flussi nel caso 1

Abbiamo preso in esame i casi migliori e peggiori dei vari nodi mostrati in Tabella IV per la metrica End to End Delay, per fare un confronto tra i vari ritardi delle classi A, B e Best-Effort. Come si evince dal Grafico I, che rappresenta il End to End Delay del flusso US1.app[0]->CU.app[2] di classe A, notiamo che il ritardo ha un valore infinitesimo prossimo allo 0. Infatti i flussi di traffico in classe A hanno la massima priorità. Ciò significa che, quando il credito è sufficiente, i pacchetti di classe A sono i primi a essere trasmessi, riducendo significativamente il tempo di attesa in coda.

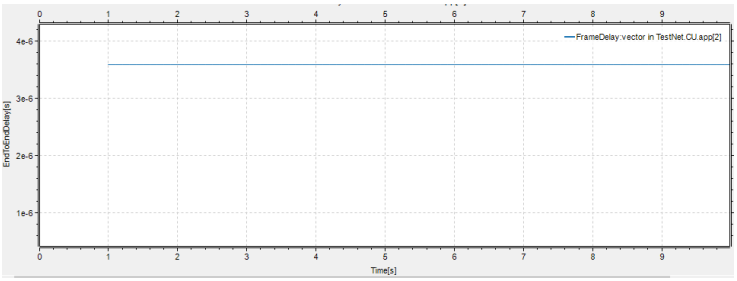

Grafico I: End-to End Delay di US1.app[0]->CU.app[2]

Invece, per quanto riguarda i flussi in classe B come ME.app[4]->RS1.app[0] e ME.app[5]->RS2.app[0] , troviamo che il valore del End to End Delay è più alto in entrambi casi poichè in un secondo vengono schedulate 4000 frame piuttosto che 8000 frame.Infatti, i flussi di traffico in classe B hanno una priorità inferiore rispetto alla classe A. Questo significa che i pacchetti di classe B devono aspettare che i pacchetti di classe A vengano trasmessi, aumentando il tempo di attesa in coda.

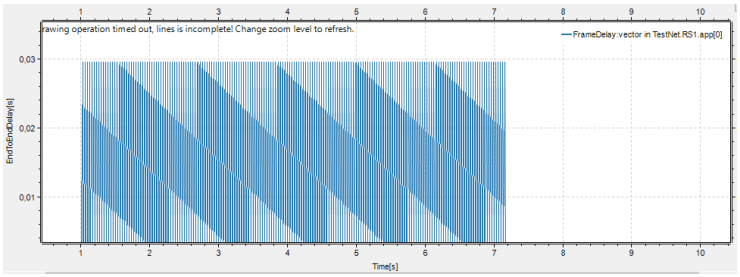

Grafico II: End-to End Delay di ME.app[4]->RS1.app[0]

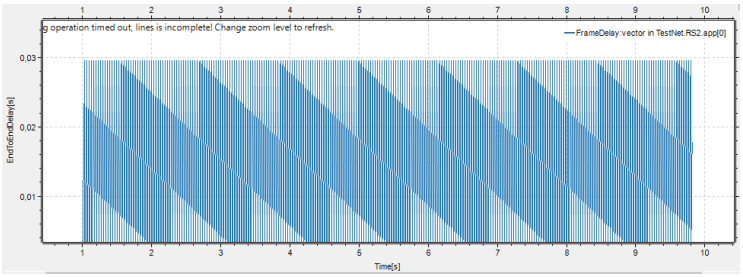

Grafico III: End-to End Delay di ME.app[5]->RS2.app[0]

I flussi di traffico Best Effort hanno la priorità più bassa e non sono soggetti a traffic shaping come le classi A e B. Quindi, i pacchetti Best Effort non beneficiano di meccanismi che garantiscono tempi di consegna specifici. Tuttavia, in assenza di traffico di alta priorità, possono essere trasmessi senza significativi ritardi, come si evince dal Grafico IV e dai rispettivi valori in Tabella IV.

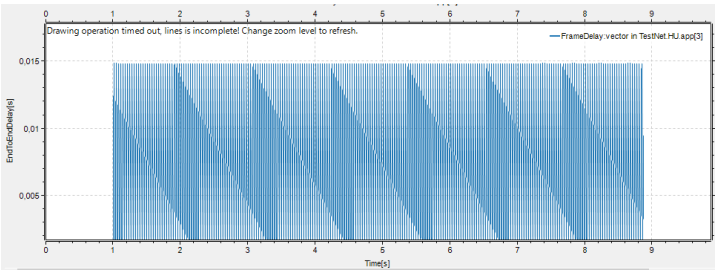

Grafico IV: End-to End Delay di TLM.app[0]->HU.app[3]

Simulazione Caso 1: Absolute Jitter

Indica il massimo scostamento tra il momento previsto di arrivo di un pacchetto e l'istante di arrivo effettivo si calcola come segue:

$$absJitter = \max(e2eDelay) - \min(e2eDelay)$$

Nell'EthernetApplication.cc la seguente metrica viene calcolata in questo modo:

```
void EthernetApplication::socketDataArrived(TSNSocket *socket, Packet *pkt) {
    /* Messaggio dalla rete */
    EV << "Arrivato pacchetto: " << pkt->getName() << endl;
    auto pay = pkt->peekData<EthernetDataPayload>();
    int bs = pay->getBurst_size();
    int num = pay->getFrame_num();
    simtime_t e2ed;
    simsignal_t sig;
    simsignal_t absjitter;

    if((bs-1) == num) {
        e2ed = simTime()-pay->getGenTime();
        sig = registerSignal("BurstDelay");
        emit(sig, e2ed);
    }


    e2ed = simTime()-pay->getGenTime();
    sig = registerSignal("FrameDelay");
    emit(sig, e2ed);
    simtime_t max=e2eDelayMax(e2ed);
    simtime_t min=e2eDelayMin(e2ed);
    absjitter = registerSignal("JitterAssoluto");
    emit(absjitter, max-min);

    delete pkt;


}

simtime_t EthernetApplication ::e2eDelayMax(simtime_t e2eM) {
    if (e2eM > e2eMax) {
        e2eMax = e2eM;
    }

    return e2eMax;
}

simtime_t EthernetApplication ::e2eDelayMin(simtime_t e2eM) {
    if (e2eM < e2eMin) {
        e2eMin = e2eM;
    }
```

Come si evince dalla Tabella IV e dai grafici possiamo notare che gli idleSlope delle varie classi sono configurati in modo tale da favorire rapidamente i pacchetti di classe A, mantenendo basso il Jitter. Gli idleSlope della classe B sono più grandi rispetto alla classe A, ciò comporta ad una maggiore variabilità nei tempi di arrivo dei pacchetti. Questo può causare un Jitter più elevato. Invece, per quanto per quanto riguarda i pacchetti Best Effort dipende molto dalle condizioni di congestione della rete. In una rete meno congestionata, esso può essere relativamente basso, ma in una rete molto congestionata, può aumentare significativamente. Di seguito sono riportati dei grafici dei flussi di classe A , classe B e Best-Effort che risentono del Jitter :

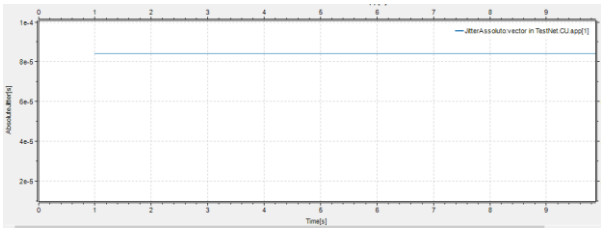

Grafico V: Absolute Jitter di US1.app[0]->CU.app[2]

Si nota che nel Grafico V del flusso di classe A, il jitter si assesta a un valore pari a 0.000004s.

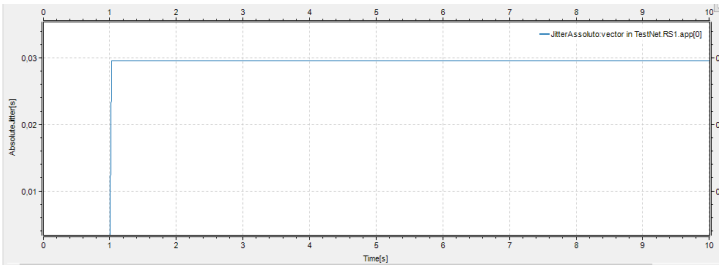

Grafico VI: Absolute Jitter di ME.app[4]->RS1.app[0]

Si nota che nel Grafico VI del flusso di classe B, il jitter si assesta a un valore pari a 0.029474s, e già si evince che è piu alto rispetto a quello di classe A.

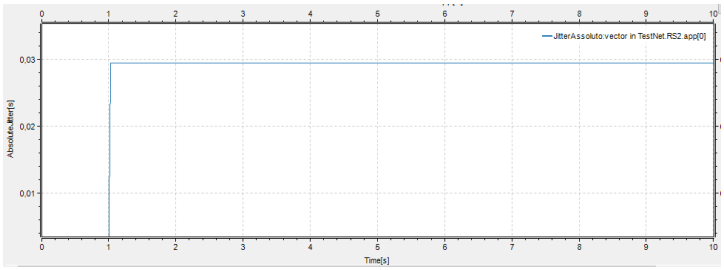

Grafico VII : Absolute Jitter di ME.app[5]->RS2.app[0]

Si nota che nel Grafico VII del flusso di classe B, il jitter si assesta a un valore pari a 0.029487s , simile a quello del grafico VI ma sempre minore rispetto a quello di classe A .

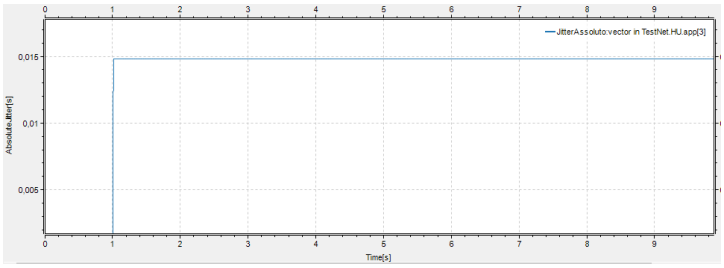

Grafico VIII : Absolute Jitter di TLM.app[0]->HU.app[3]

Si nota che nel Grafico VIII del flusso di classe Best-Effort, il jitter si assesta a un valore pari a 0.000041s ,ma essendo una rete meno congestionata, risulta essere relativamente basso.

Simulazione Caso 1: Numero massimo frame in coda

rappresenta il numero massimo di frame che possono essere temporaneamente immagazzinati in una coda.

CBS	#0	TestNet.SW2.eth[1].macLayer.queue.queue[7]	4 pk
CBS	#0	TestNet.SW1.eth[2].macLayer.queue	4 pk
CBS	#0	TestNet.SW1.eth[4].macLayer.queue.queue[7]	3 pk
CBS	#0	TestNet.SW1.eth[4].macLayer.queue	3 pk
CBS	#0	TestNet.SW1.eth[2].macLayer.queue.queue[7]	3 pk
CBS	#0	TestNet.SW2.eth[1].macLayer.queue.queue[1]	2 pk

Grafico IX : numero massimo frame in coda

Dal grafico IX , dato che il traffico che va dal nodo ME verso la porta eth[1] dello SW2 risulta piu affollato , si nota che il numero massimo di frame in coda è pari a 4 . I parametri di configurazione del CBS sono stati scelti con attenzione per bilanciare latenza e throughput, assicurando che il numero di frame in coda sia stato gestito efficacemente.

a) Simulazione Caso 2: End-to-End Delay

Nella seguente tabella sono riportati i seguenti valori trovati dalla simulazione utilizzando solo lo Strict Priority Selection con il Deadline Monotonic:

Flow **name**	*E2ED* **(s)**	*Absolute* **Jitter (s)**
LD1_CU	0.000021	0.000021
LD2_CU	0.000032	0.000037
ME_S1	0.000134	0.002936
ME_S2	0.000135	0.002937
ME_S3	0.000135	0.002937
ME_S4	0.000136	0.002938
US1_CU	0.000004	0.000004
US2_CU	0.000005	0.000005
US3_CU	0.001023	0.001458
US4_CU	0.001041	0.001486
CU_HU	0.000066	0.000115
CM1_HU	0.000767	0.001581
ME_RS1	0.000752	0.001481
ME_RS2	0.002220	0.002949
RC_HU	0.000016	0.000038
TLM_HU	0.001191	0.003053
TLM_CU	0.000021	0.000037

Tabella V : Metriche medie dei flussi nel caso 2

Abbiamo preso in esame i casi migliori e peggiori dei vari nodi mostrati in Tabella V per la metrica End to End Delay :

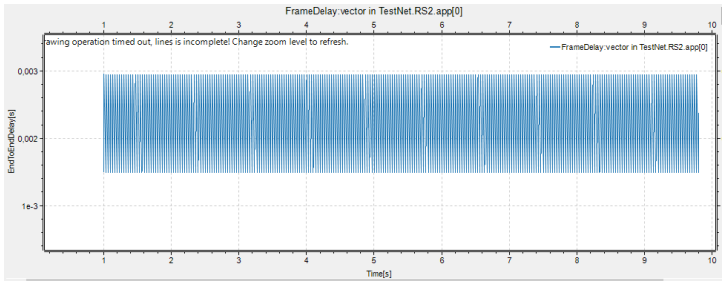

Grafico X : End-to End Delay di ME.app[5]->RS2.app[0]

Nel grafico X, si nota che il traffico generato dal nodo ME risulta avere un end-to End Delay più alto rispetto a tutti gli altri nodi , dovuto al fatto che oltre ad avere un payload abbastanza grande (178500 byte) risulta anche essere di priorità più bassa (1) .

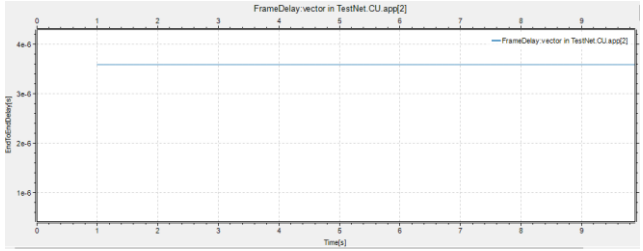

Grafico XI : End-to End Delay di US1.app[0]->CU.app[2]

Nel grafico XI , si nota il traffico generato dal nodo US1 risulta avere un End-to End Delay piu basso rispetto a tutti gli altri nodi, dovuto al fatto che nonostante sia di priorità piu bassa(0) abbia un payload abbastanza piccolo (188byte) e ciò significa che la frame occupa meno tempo di trasmissione effettiva.

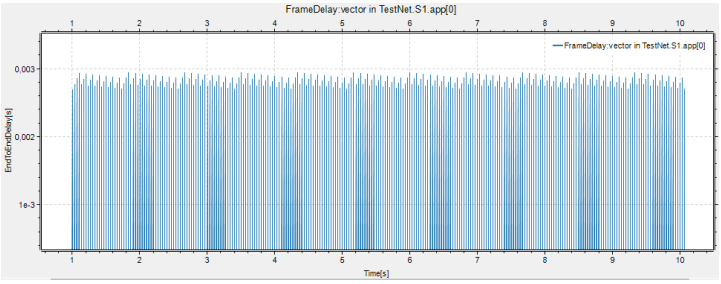

Grafico XII : End-to End Delay di ME.app[0]->S1.app[0]

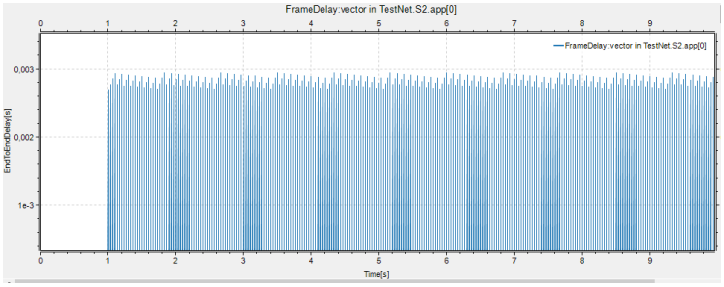

Grafico XIII : End-to End Delay di ME.app[1]->S2.app[0]

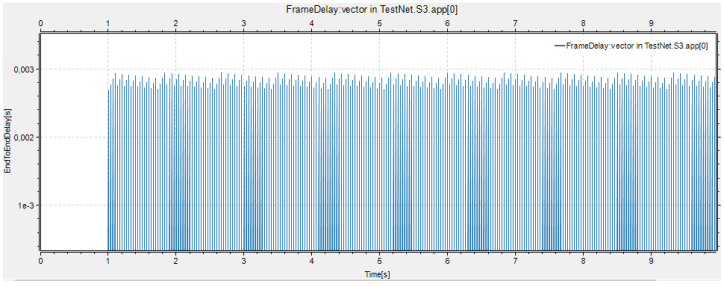

Grafico XIV : End-to End Delay di ME.app[2]->S3.app[0]

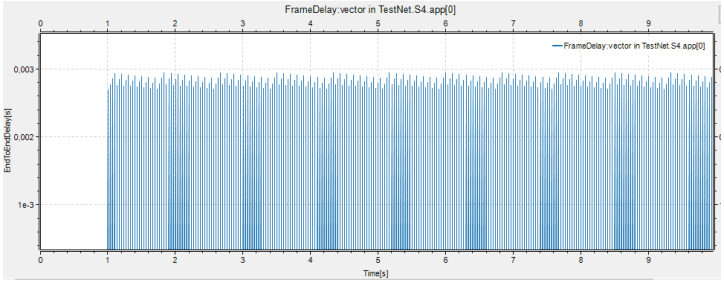

Grafico XV : End-to End Delay di ME.app[3]->S4.app[0]

Nei seguenti grafici abbiamo riportato gli end-To-EndDelay del nodo ME Che rappresenta quello con più alta priorità (7) nella rete.

Simulazione Caso 2: Absolute Jitter

Abbiamo considerato i grafici dei vari nodi che generano dei flussi con i valori di jitter assoluto piu alti e bassi della rete.

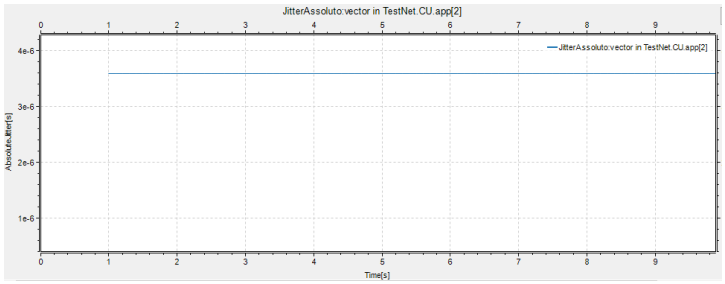

Grafico XVI : Absolute Jitter di Us1.app[0]->CU.app[2]

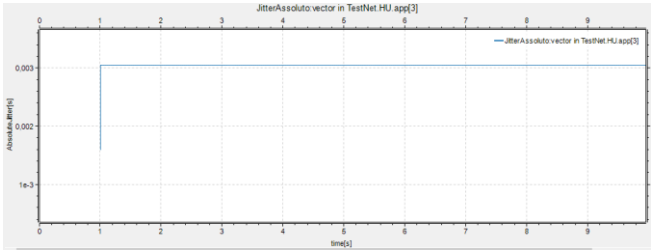

Grafico XVII : Absolute Jitter di TlM.app[0]->HU.app[3]

Si nota che nelle reti il Jitter che si riferisce alla variazione del ritardo ed è strettamente legato all'End-To-End Delay.
I nodi della rete, inoltre, mostrano un Jitter che diventa costante dopo pochi secondi di stallo.

Simulazione Caso 2: Numero massimo frame in coda

rappresenta il numero massimo di frame che possono essere temporaneamente immagazzinati in una coda.

General	#0	TestNet.SW1.eth[4].macLayer.queue	125 pk
General	#0	TestNet.SW2.eth[1].macLayer.queue	6 pk

Grafico XVIII : numero massimo frame coda

Dal grafico XVII , dato che il traffico che va dal nodo CU verso la porta eth[4] dello SW1 risulta abbastanza affollato , si nota che il numero massimo di frame in coda è pari a 125 , mentre dal nodo ME verso lo SW2 nella porta eth[1] si crea una coda di 6 pacchetti.