POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Informatica

Tesi di Laurea Magistrale

Backbone Automotive Ethernet

Relatore
prof. Massimo Violante

Candidato
Andrea Giuliano

A.A. 2017/2018

Indice

Elenco delle figure

Capitolo 1

Introduzione

Tutte le compagnie e le attività implicate nella fabbricazione di autoveicoli, inclusa la componentistica come motori e chassie, vengono raggruppate in un unico settore: l'Automotive.
L'evoluzione del settore manufatturiero dell'autoveicolo ha avuto un grosso impatto sull'industrializzazione del ventesimo secolo. Anche se le origini dell'auto, risalenti alla fine del diciannovesimo secolo, sono europee, il mercato della prima parte del novecento è stato maggiormente dominato dagli Stati Uniti grazie all'invenzione della produzione di massa. Nella seconda parte del secolo scorso invece, l'Europa e il Giappone sono diventati i maggiori produttori ed esportatori di autoveicoli [8]. Se si concentrasse l'attenzione su gli ultimi due decenni, si noterebbe come il settore dell'Automotive stia subendo un avanzamento tecnologico non indifferente. Motori elettrici, guida autonoma e sistemi di infotainment sono esempi di come le aziende operanti in questo campo inizino ad avere sempre più bisogno di un supporto dell'ingegneria informatica ritenuta, in questo settore, marginale fino agli inizi degli anni 2000.

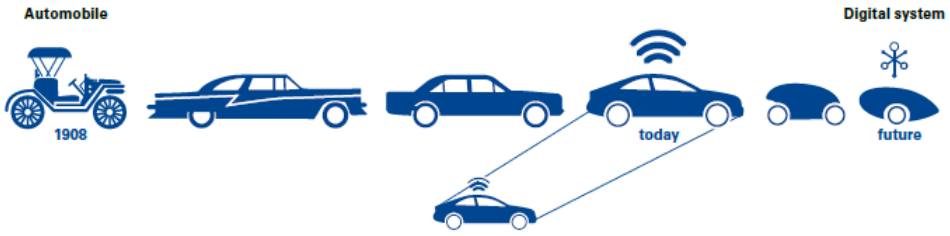

Figura 1.1: Evoluzione delle automobili [Figure credit: KPMG, Global Automotive Executive Survey, 2017].

Attualmente il mercato riguardante il campo dell'Automotive è in continua crescita, infatti, dal 1995 al 2015, il suo valore è passato da 7 miliardi a 30 miliardi di

dollari e si ipotizza, che entro il 2020, il valore degli scambi commerciali destinati ad autoveicoli arriverà a toccare i 40 miliardi. Questo settore rappresenta una grande fonte di reddito per le aziende produttrici di dispositivi elettrici.
Nel grafico riportato in figura 1.2 si può notare quanto incida la produzione della componentistica destinata agli autoveicoli. Nel dettaglio si osserva che le vendite di dispositivi CAN, al 2014, erano circa il triplo di quelle delle porte Ethernet o anche che i controller Wi-Fi insieme alle porte Ethernet riuscivano a raggiungere i numeri della produzione di nodi CAN [11].

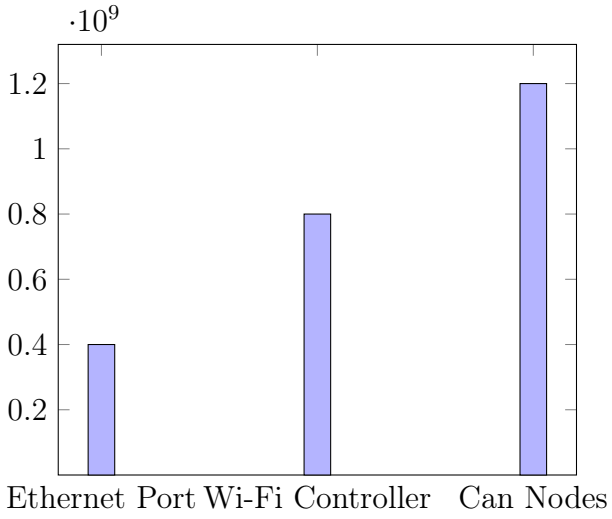

Figura 1.2: Situazione delle vendite nel 2014 [11].

Con la crescente richiesta di nuovi sistemi multimediali, dell'introduzione di sistemi per la guida autonoma ed anche, al momento in parte minore, dell'IoT applicato agli autoveicoli, le unità di controllo elettronico (Electronic Control Unit o ECU), hanno iniziato ad aver bisogno di ricevere sempre più dati da elaborare per sapere ciò che accade all'interno ed all'esterno del veicolo per poter intervenire opportunamente.

1.1 Contesto

Le attuali reti di un veicolo per le comunicazioni interne sono ormai obsolete, si pensi ad esempio al protocollo maggiormente diffuso: il CAN-Bus, sviluppato agli albori degli anni '80 da Bosch ed aggiornato agli inizi degli anni '90.

Si può suddividere l'elettronica di un autoveicolo in domini, ognuno dei quali ha controlli indipendenti[1], ciascuno con esigenze diverse.
In generale, questi sono:

- **Powertrain**: il gruppo di componenti che genera energia e ne usufruisce per il movimento del veicolo. Queste parti dell'auto, come motore e trasmissione, hanno bisogno di controllori molto complessi per gestire i sensori e per massimizzare l'efficienza e minimizzare i consumi. Non ha bisogno di grosse moli di dati ma necessita di una bassa latenza;

- **Chassis**: include i dispositivi (freni, volante e sospensioni) di supporto alla categoria powertrain. Anche qui si ha la necessità di una latenza minima con piccole moli di dati;

- **Body e Comfort**: includono tutti quei dispositivi non indispensabili al controllo del veicolo (cristalli, riscaldamento e A/C,..). Non è indispensabile alta velocità di trasferimento per i pochi dati generati da questi dispositivi;

- **Assistenza alla guida**: il dominio più critico perché include tutti quei componenti progettati per la sicurezza del guidatore, dei passeggeri e dei pedoni e per l'assistenza alla guida. I controllori ricevono una grossa mole di dati derivante, per esempio, dalle telecamere ed ad una latenza minima per poter intervenire prontamente.

Soddisfare certi requisiti[2], soprattutto quelli richiesti dall'assistenza alla guida, non è sempre facile per le reti presenti al giorno d'oggi nei veicoli. Automotive Ehernet, come spiegato nel sottocapitolo successivo, può essere di supporto.

1.2 I motivi di Automotive Ethernet, vantaggi e opportunità

Anche se Ethernet esiste da circa 30 anni non era stato possibile applicare questo protocollo nel campo Automotive per vari problemi:

1. Non rispettava i requisti riportati nella tabella 1.1;
2. Non garantiva latenza inferiore a pochi microsecondi;
3. Non aveva un strumento per controllare l'allocazione della banda;
4. Non aveva un modo per sincronizzare i dispositivi della rete.

[1] Possono essere controlli meccanici, elettrici o veri e propri microcontrollori.
[2] Riportati per semplicità nella tabella 1.1.

Dominio	**Latenza** (µs)	**Banda richiesta**
Powertrain	< 10	Bassa
Chassis	< 10	Bassa
Body e Comfort	< 10000	Bassa
Assistenza alla guida	< 250	10-100 Mbps per camera

Tabella 1.1: Riepilogo in dettaglio dei requisiti dei domini.

Con l'aggiornamento del documento IEEE 802.3 e l'aggiunta di nuovi protocolli, di cui si parlerà nei prossimi capitoli, Ethernet è divento uno standard perfetto da poter usare nelle reti interne degli autoveicoli. Il cablaggio è al terzo posto per costo della componentistica di un autoveicolo e si stima che per il 2020 il costo dell'elettronica interna arriverà al 40% del totale (32% oggi).
Con Authomotive Ethernet diverrebbe semplice anche la trasformazione della struttura della rete, passando da una struttura centralizzata ad una struttura ad albero (figura 1.3), riducendo in questo modo il costo e il peso del cablaggio e aumentando la facilità di cooperazione tra i componenti presenti sulla rete.

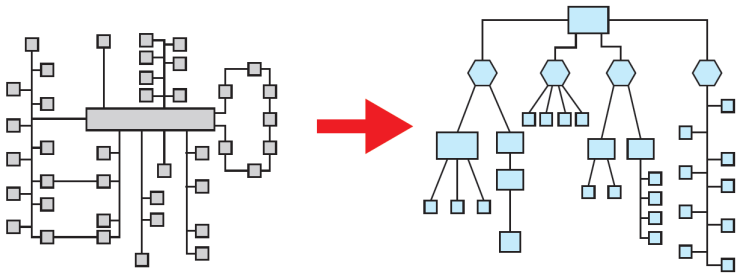

Figura 1.3: Possibile cambiamento della struttura della rete.

1.3 Obiettivo della Tesi

Come precedentemente descritto nel paragrafo 1.1, in concomitanza al progresso dei sistemi informatici ed elettronici a bordo, è sorto il bisogno impellente di realizzare un aggiornamento delle tecnologie per la comunicazione intra-veicolare.
Sulla base di queste premesse, durante il periodo di tesi, presso la sede di Torino di Teoresi S.p.A.(1.3.1), è stata progettata e sviluppata una prima implementazione di una backbone basata su Automotive Ethernet.
In particolare, l'obiettivo è stato quello di modellare una rete Automotive Ethernet,

che unisse più CAN-bus leggendo i dati in transito su di essi e, rispettando tutti i protocolli dettati da Automotive Ethernet, trasferisse tutti i segnali.
Risulta quindi interessante l'idea di sviluppare una backbone Automotive Ethernet, graficamente schematizzata in figura 1.4, in quanto potrebbe essere utilizzata come primo step per concretizzare un graduale passaggio dalle reti attuali ad una rete interamente basata su questo protocollo.

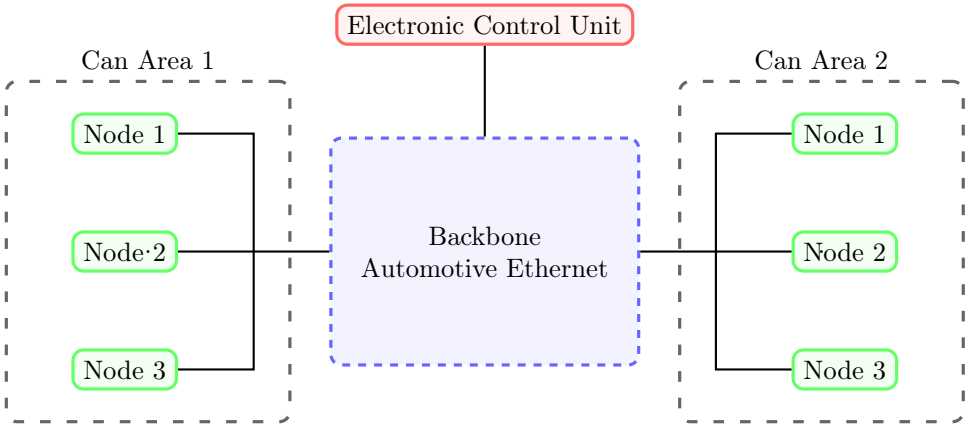

Figura 1.4: Rappressentazione schematica di una Backbone Automotive Ethernet.

1.3.1 Teoresi S.p.A.

Teoresi, società a carattere internazionale con sede principale a Torino, si propone ai propri clienti quale partner qualificato per favorirne lo sviluppo, avvalendosi di tecnologie innovative. Forte di una competenza globale in ambito engineering, è in grado di offrire servizi di progettazione di ingegneria, sviluppo e consulenza qualificata. Teoresi ha iniziato la propria attività nel 1987, ampliando la sua presenza nel mercato grazie alla distribuzione di un insieme di software tecnico-scientifici riconosciuti come prodotti leader a livello mondiale e vitali per il mondo ingegneristico, con focalizzazione sui temi della modellazione e simulazione, dei controlli e dello sviluppo software.
La crescita costante di aspettative in termini di servizi da parte del mercato, ha indotto Teoresi a compiere un percorso di riorganizzazione aziendale che ha portato alla nascita del Gruppo Teoresi, una realtà focalizzata sulla consulenza ingegneristica e sui servizi di systems integration, con un ampio e ben strutturato portafoglio di offerta. Per ottimizzare il proprio processo di crescita, Teoresi Group ha consolidato partnership già collaudate e vincenti assumendo il controllo diretto di due altre società: ALTO sistemi e Altemedia [1].

Capitolo 2

Automotive Networks

Nonostante i notevoli vantaggi, in parte descritti in precedenza, derivanti dall'uso di Automotive Ethernet, bisogna precisare che il progressivo sviluppo di questa tecnologia non potrà soppiantare le reti attuali, essendo queste dotate di una serie di caratteristiche in grado di renderle competitive e vantaggiose, ancora per molti anni. Tra queste vale sicuramente la pena citare:

- il basso costo;
- la robustezza;
- l'essere state collaudate per parecchi anni;
- abbastanza banda a disposizione per le applicazioni che non necessitano grandi prestazioni.

L'obiettivo di Automotive Ethernet, dunque, non è sostituire interamente il cablaggio e le reti presenti negli autoveicoli, ma fornire un supporto più efficiente per quelle applicazioni che necessitano di tre requisiti fondamentali: prestazioni, latenza e un'ingente mole di dati da trasferire.

Per fornire una visione più analitica del panorama attuale, nei prossimi capitoli saranno innanzitutto descritte le reti automotive attualmente presenti sul mercato, mettendone in luce i punti di forza e di debolezza.

2.1 Controller Area Network (CAN)

Il CAN (Controller Area Network) è un protocollo basato su comunicazione seriale sviluppato da Bosch agli inizi degli anni '80. Definisce uno standard per una comunicazione real-time tra sensori, attuatori, controllori e altri nodi, efficiente ed affidabile. Il primo sviluppo di CAN fu maggiormente supportato dall'industria dell'autoveicolo [12], infatti per più di 30 anni, CAN è stata la rete dominante in questo settore. Citando alcuni punti di forza di questa rete:

- CAN è robusta e collaudata. Per decenni, la rete CAN è stata la scelta primaria per la maggior parte dei produttori di autoveicoli [11].
- CAN è uno standard aperto, definito nell'ISO 11898, che spinge molti produttori a sviluppare una gran varietà di controllori CAN, creando concorrenza e riduzione dei costi da parte degli acquirenti [11].
- CAN, nell'Automotive, è il protocollo più vicino al concetto di "plug and play": i nodi possono essere aggiunti e rimossi senza aver bisogno di riconfigurare i parametri e gli altri nodi della rete [11].
- Esistono sul mercato di molti strumenti, come ValueCAN, che consentono un facile sviluppo ed analisi della rete [11].

2.1.1 Topologia

CAN è stato implementato come un bus condiviso in cui ogni nodo presente sulla rete è fisicamente collegato alla linea di comunicazione. Il punto di forza di questo tipo di implementazione è la semplicità, che ne rappresenta al contempo anche una debolezza, poiché più nodi non potranno trasmettere in contemporanea senza che avvengano collisioni. Questo rende necessario l'inserimento di un metodo per l'arbitraggio delle comunicazioni.

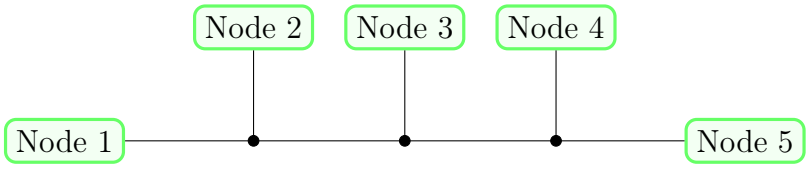

Figura 2.1: CAN-bus condiviso da 5 nodi

2.1.2 Formato del Frame

I nodi di un CAN-bus sono in grado, in qualsiasi momento, di avviare la trasmissione di un frame. Il compito di far prevalere il messaggio proveniente da un nodo piuttosto che da un altro, è svolto dal metodo di arbitraggio, chiamato Media Access Control (MAC). CAN usa un metodo molto astuto per evitare che la trasmissione contemporanea di messaggi crei collisioni e quindi che il nodo con piu‘ priorita‘ debba ritrasmettere il messaggio [10].
Nel momento in cui un nodo vuole avviare l'invio di un messaggio, manda inizialmente il suo ID. La rete è configurata come un AND logico, in cui i bit ad 1 sono recessivi, e quelli a 0 sono dominanti. Nel momento in cui più nodi stanno inviando contemporaneamente un messaggio, e tutti trasmettono 1, il bus rimane settato a

1. Qualora uno dei nodi invece trasmetta uno 0, il bus andrà a 0, consentendo al nodo da cui era partito di continuare l'invio del messaggio e l'interruzione di tutti gli altri nodi.

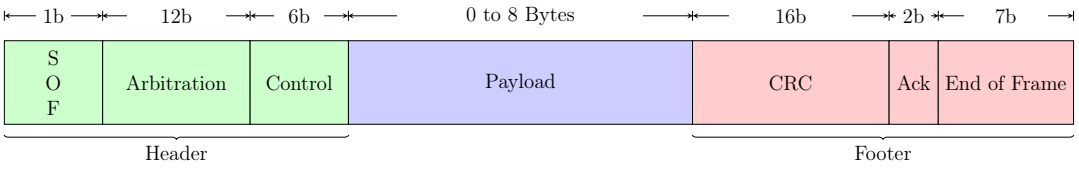

Figura 2.2: Standard CAN Frame Format

Il formato del frame di un messaggio CAN, riportato in Figura 2.2, è costituito da:

- **SOF (Start Of Frame)**: un solo bit dominante (0) mandato dal nodo vuole iniziare una trasmissione e indica l'inizio di un messaggio CAN;
- **Arbitration**: campo costituito da dodici bit. I primi undici costituiscono l'ID per l'arbitraggio, descritto precedentemente, il dodicesimo è il bit RTR (Remote Transfer Request), ormai quasi completamente in disuso, che serve per forzare l'invio di dati da parte della sorgente;
- **Control**: indica la lunghezza del Payload su quattro bit, più due bit riservati per il protocollo;
- **Payload**: i dati trasmessi. Questo campo può avere una lunghezza variabile che va da un minimo di zero byte ad un massimo di otto;
- **CRC (Cyclical Redundancy Check**: campo costituito da sedici bit, di cui i primi quindici vengono calcolati dalla sorgente e dalla destinazione, con un algoritmo che elabora il Payload, e sono usati per un controllo sulla trasmissione. Il sedicesimo indica la fine di questo campo;
- **Ack**: usato dalla destinazione per indicare che il campo CRC corrisponde, e quindi la trasmissione è avvenuta con successo;
- **EOF (End Of Frame)**: sette bit recessivi (1) che indicano la fine del frame e quindi della trasmissione da parte del nodo sorgente.

2.1.3 Vantaggi e Svantaggi

Il più grande punto di forza del CAN, come già citato, è indubbiamente la flessibilità che introduce nella rete. Essendo un sistema multi-master, non ha bisogno di nodi special purpose come switch, permettendo ai designer, nel momento di aggiunta o

rimozione di un nodo, di non dover ridisegnare completamente la rete o apportare significativi cambiamenti agli altri dispositivi della rete [11].

Lo svantaggio di questa rete è invece legato alle performance: la velocità di trasferimento infatti può arrivare fino ad un massimo di 1 Mb/s o, se consideriamo la sua evoluzione, fino a 10 Mb/s.
Con lo sviluppo degli ultimi anni, una velocità di trasferimento così bassa sta rappresentando un limite sempre più insormontabile per i campi di applicazione delle reti CAN.
Un altro punto debole della rete CAN e' la sua bassa efficienza. Un payload di massimo 8 Byte porta ad un overhead sulla rete perche' viene trasmessa quasi la stessa quantita' di bit appartenenti ad header e footer e di bit appartenenti al payload [11].
Un problema da non sottovalutare è anche la topologia a bus condiviso. Si potrebbe verificare, in presenza di un timing errato, nell'invio dei messaggi, che il MAC porti ad uno stato dove il nodo con bassa priorità non riesce ad inoltrare messaggi poiché trova il bus sempre occupato da un nodo con più alta priorità.

2.2 Local Interconnect Network (LIN)

Verso la fine degli anni '90, alcune grandi aziende del settore automobilistico hanno avuto l'esigenza di ridisegnare alcune parti delle reti automotive per poterne ridurre complessità, costo e peso.
Su queste necessità è nata la LOCAL INTERCONNECT NETWORK (LIN) che, al contrario di altre reti, è sviluppata per quelle applicazioni in cui le prestazioni della rete CAN non sono necessarie, riuscendo anche a ridurne i costi di sviluppo. Essa Infatti, ad oggi, è ancora largamente diffusa negli autoveicoli per la gestione di funzionalità secondarie come il controllo dei cristalli, l' accensione e lo spegnimento delle luci all'interno dell'abitacolo e altre funzioni non direttamente collegate alla guida o alla sicurezza delle persone[13].

LIN e Automotive Ethernet si posizionano agli estremi di un grafico dove gli assi rappresentano, rispettivamente, performance e costo[1], motivo per cui LIN sara' una di quelle reti che continuerà ad essere largamente utilizzata anche dopo l'introduzione di Automotive Ethernet.
Si può riassumere il funzionamento LIN in tre punti chiave:

- è una rete costituita da un singolo cavo con una velocità massima di 10 kb/s, che permette un basso costo e flessibità nel cablaggio [11];
- può essere implementata ed usata anche dai controllori ad 8-bit meno prestanti presenti sul mercato perché non ha bisogno di un supporto particolarmente complesso da parte del microcontrollore [11];

[1] Descritto nel dettaglio nel capitolo successivo.

- il suo livello fisico necessita di un'alimentazione a 12 V, ovvero non ha bisogno di transceiver per essere compatibile con l'alimentazione di un autoveicolo [11].

2.2.1 Topologia

LIN è un bus basato su un'architettura di tipo master-slave (Figura 2.3). In ogni implementazione di questo bus è sempre presente un solo master e un massimo di sedici altri nodi identificabili come slave, ognuno dei quali usa lo stesso mezzo fisico che ha lunghezza massima di 40 m. Essendo un mezzo condiviso, un solo messaggio alla volta può essere trasmesso contemporaneamente.

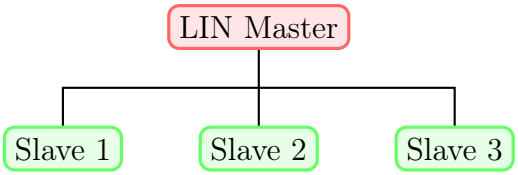

Figura 2.3: LIN bus con un Master e tre Slave

2.2.2 Formato del Frame

La comunicazione su LIN può essere definita time-triggered, un sistema di trasmissione dei dati di tipo temporizzato, controllato esclusivamente dal nodo Master. Questo tipo di formato implica che il Master sia l'unico può iniziare una comunicazione e impedendo ogni possibilità di collisione che invece costituiva un limite nel sistema CAN.
Il Master conserva internamente una tabella di schedulazione che permette di tracciare, tramite l'utilizzo di un ID corrispondente ed univoco, quale messaggio, sia stato mandato e quando ciò è avvenuto. Questa tabella può supportare fino ad un massimo di 64 elementi distinti.
Gli header dei messaggi sono creati dal Master, basandosi su suddetta tabella, anche se i dati sono stati generati da uno Slave.

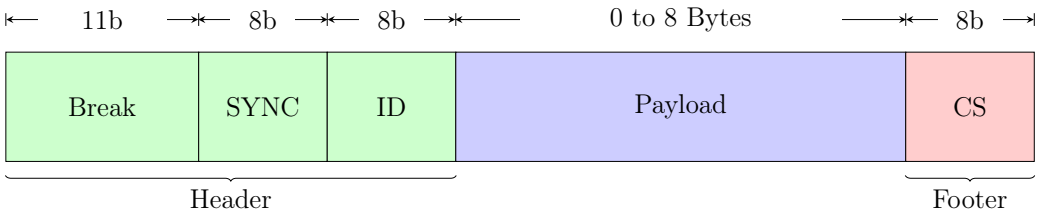

Figura 2.4: Standard LIN Frame Format

Nel dettaglio, il frame di un messaggio LIN (Figura 2.4), è costituito da:

- **Break**: campo costituito da tredici bit dominanti (0) sempre trasmessi dal master che indicano, a tutti i nodi della rete, che sta per essere trasmesso un messaggio;
- **SYNC**: sequenza di otto bit alternati, che permette la sincronizzazione degli Slave con il Master;
- **ID**: sequenza di sei bit che rappresentano l'identificativo (ID) del messaggio. . La dimensione di questa porzione limita l'unicità degli ID ad un massimo di 64 come precedentemente esposto, a cui si aggiungono due bit di parità per la rilevazione di errori di trasmissione;
- **Payload**: campo che, a differenza di quelli precedentemente descritti che vengono creati e mandati dal Master, viene mandato dallo Slave selezionato tramite l'ID;
- **CS (Checksum)**: otto bit per un rilevamento di errori.

2.2.3 Vantaggi e Svantaggi

I costi contenuti di LIN, i più bassi su questo mercato, rendono possibile l'implementazione di una rete automotive meno dispendiosa. Purtroppo, come già descritto, il basso costo è legato alle basse prestazioni garantite dalla rete, che possono raggiungere al massimo 20 Kbit/s.
Per questa ragione il sistema LIN non è compatibile con quelle applicazioni in cui il Master necessita di iniziare una comunicazione in modo immediato, poichè i messaggi vengono schedulati nel tempo, come descritto nella sezione 2.1.2, e dispone di un meccanismo di rilevazione degli errori molto semplice [11].

2.3 Media Oriented Serial Transport (MOST)

Nella seconda metà degli anni novanta, i sistemi di infotainment a bordo del veicolo erano già abbastanza diffusi e avanzati, specialmente nella fascia dei veicoli di lusso, imcludendo infatti sistemi per la navigazione e display all'avanguardia. Questo trend, sempre più crescente, ha spinto quattro grandi aziende del settore automotive a unire le proprie conoscenze e creare un team, chiamato *Most Coperation* che potesse sviluppare un nuovo protocollo, e quindi una nuova rete, per affrontare la concorrenza e la crescente domanda di sistemi di infotainment a bordo del veicolo [7].
Most Cooperation aveva il compito di creare un nuovo tipo di rete che andasse a colmare le lacune delle altre reti automotive, non adatte per i sistemi di infotainment perché non ottimizzate per il trasporto di dati riguardanti audio e video. MOST è stata creata senza uno standard per lo strato fisico in modo tale che potesse essere

adattata a vari tipi di protocolli[2]. In pratica, a livello più basso, MOST è una rete sincrona pilotata da un Master che la rende capace di trasmettere stream di dati audio e video.
I punti chiave di questa rete sono:

- canali incorporati per lo streaming di dati audio e video, essenziali per le applicazioni di infotainment;
- banda molto superiore a quella delle altre reti automotive che può raggiungere i 150 Mb/s [11];
- possibilità di cablaggio con fibra ottica, che permette di filtrare i disturbi elettromagnetici.

2.3.1 Topologia

La topologia usata in una rete MOST è identificabile in una struttura ad anello unidirezionale, come mostrato in Figura 2.5, in cui un singolo Master controlla il clock, abilitando la comunicazione.

L'unidirezionalità della comunicazione prevede che, quando il Master inizia la

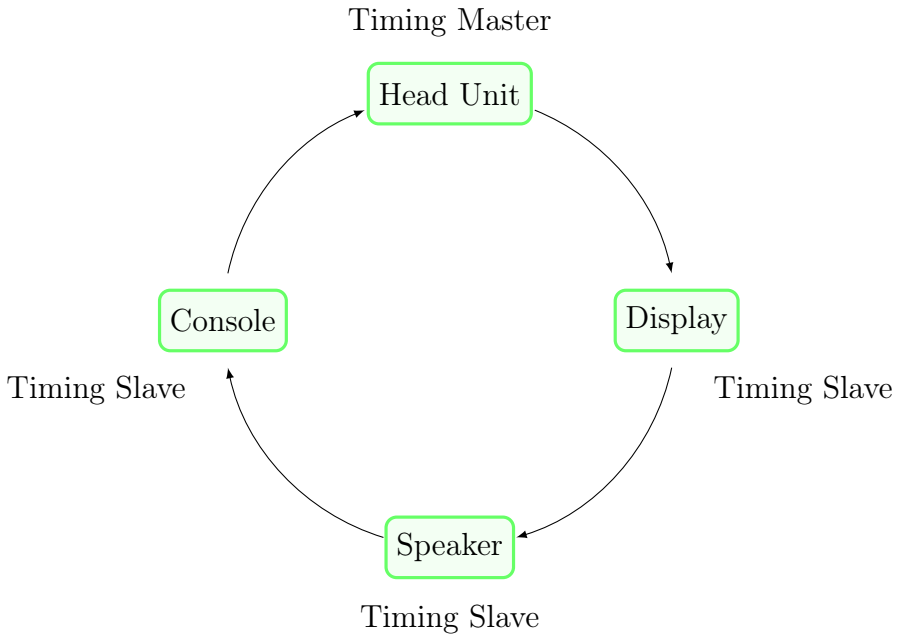

Figura 2.5: Topologia ad anello di MOST: un timing Master e tre timing Slave.

trasmissione, invia un frame sulla sua porta di uscita e verso quella di entrata del

[2]Sincroni, asincroni, ad eventi, tools per lo sviluppo, ecc..

nodo successivo, il quale, a sua volta, lo trasmetterà al seguente nello stesso analogo fino al raggiungimento del nodo a cui è destinato il messaggio. Sulla rete è possibile trovare anche i messaggi per la sincronizzazione, generati dal Master e propagati dagli Slave, i quali però in questo sistema, sono in grado di aggiornare il frame, tramite l'inserimento dei propri dati.

2.3.2 Formato del Frame

La struttura del frame è influenzata dalla caratteristica di MOST di avere una trasmissione sincrona di dati multimediali. Il frame viene diviso in canali (Figura 2.6: il primo è destinato alla trasmissione sincrona di streaming di dati[3], il secondo è asincrono e permette la trasmissione di pacchetti dato e il terzo per la trasmissione di messaggi di controllo [7].

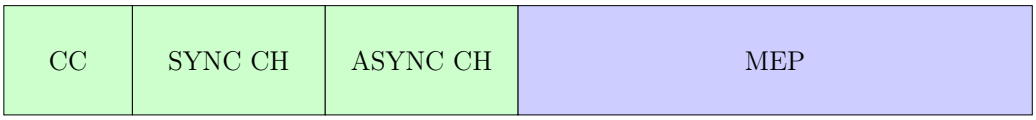

Figura 2.6: Formato del frame di MOST150.

Analizzando il frame di MOST150, variante di MOST più recente e performante, i canali presenti sono:

- **CC (Control Channel)**: contiene informazioni per il controllo delle operazioni che avvengono sulla rete. Usato anche per scambiare messaggi sugli stati della rete;
- **SYNC CH (Synchronous Channel)**: questa sezione viene usato per scambiare tra le varie sorgenti e destinazioni dati audio e video sincroni. Questo canale viene diviso dinamicamente per permettere di inserire e prelevare dati da più nodi contemporaneamente;
- **ASYNC CH (Asynchronous Channel)**: usato per la trasmissione di pacchetti asincroni;
- **MEP (MOST Ethernet Packets)**: canale introdotto recentemente, è stato sviluppato per trasportare dati IP (Internet Protocol). Riesce a trasferire fino a 120 Mb/s.

[3]Solo per la variante MOST150, quella presa in esame in questo capitolo.

2.3.3 Vantaggi e Svantaggi

Come già accennato nell'introduzione di questa sezione (2.3), il vantaggio di MOST è quello di avere a disposizione dei canali integrati per la trasmissione di dati multimediali, caratteristica indispensabile per seguire il trend del mercato automotive che si muove in una direzione in cui i sistemi di infotainment a bordo sono sempre più complessi e richiedono alte prestazioni (si pensi ad ADAS, il sistema per la guida autonoma).
La topologia ad anello unidirezionale caratterizzante MOST, può essere considerata sia un vantaggio perché è meno restrittiva rispetto ad una topologia ad albero, ma allo stesso tempo ha una tolleranza ai guasti estremamente bassa. Per tale ragione un problema su una linea di trasmissione o ad un nodo porta alla completa irraggiungibilità di tutti i nodi posizionati dopo il guasto.

Due dettagli che non permettono una veloce diffusione di MOST sono: l'essere prodotta da una sola azienda di semiconduttori ed il brevetto su di essa. Ciò sta spingendo i produttori di autoveicoli a trovare soluzioni alternative [11].

Capitolo 3

Da Ethernet ad Automotive Ethernet

Ethernet è una tecnologia basata su cavo ed attualmente è quella che presenta la più ampia diffusione, con più di 3 miliardi di interfacce in tutto il mondo, di cui oltre un miliardo solo negli U.S. e oltre 3 miliardi in tutto il mondo [9].
Si tratta di una famiglia di tecnologie per reti locali, sviluppata inizialmente in via sperimentale agli inizi degli anni '70 presso i laboratori di Xerox PARC, e aggiornata negli anni seguenti subendo delle trasformazioni che ne hanno standardizzato le specifiche tecniche del livello *fisico* e *data link* del modello ISO/OSI.
L'obiettivo primordiale dell'esperimento era ottenere un protocollo che su cavo coassiale raggiungesse una velocità di trasferimento dati pari a 3 Mb/s e che fosse al contempo affidabile in caso di traffico, mediante un MAC che regolasse l'accesso al mezzo.
I motivi principali che hanno reso Ethernet la tecnologia per LAN più diffusa sono:

- la sua nascita avvenuta agli inizi dell'era di Internet, che ne ha favorito la diffusione e offuscato le tecnologie concorrenti (FDDI[1], ATM[2]);
- il basso costo e la facilità d'uso;
- la bassa probabilità d'errore.

Queste caratteristiche hanno portato IEEE a creare più team tecnici per implementare ed ottimizzare gradualmente questo standard. Tra questi, uno dei più importanti è il IEEE Project 802, nato con l'obiettivo di sviluppare gli standard per le reti locali LAN (Local Area Network) e metropolitane MAN (Metropolitan Area Network). Fanno parte del IEEE 802 Project diverse commissioni che si occupano di standardizzare livelli diversi del modello di riferimento [11].

[1]Fiber distributed data interface: è un particolare tipo di rete ad anello token ring basata sull'uso delle fibre ottiche come mezzo trasmissivo.

[2]Asynchronous Transfer Mode: protocollo di rete di livello data link, a commutazione di circuito virtuale e trasmissione di cella di lunghezza fissa anziché in pacchetti a lunghezza variabile.

Il modello sviluppato da IEEE 802 Project, specificatamente inerente all'Ethernet, è descritto nello standard 802.3. Fin dalla sua nascita, 802.3 è stato lo standard che ha subito più aggiornamenti da parte di IEEE 802 Project.
Come già accennato, lo standard 802.3 è andato a definire le specifiche tecniche per implementare lo strato fisico e data link del modello ISO/OSI (Figura 3.1).

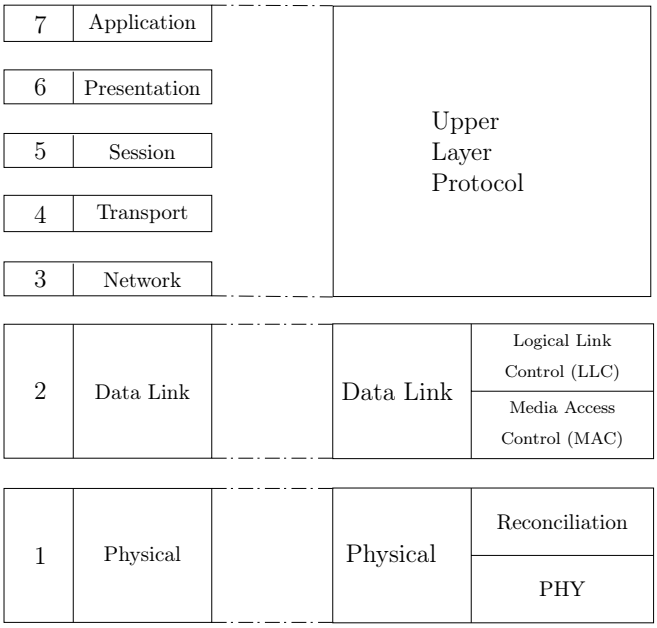

Figura 3.1: Confronto con il modello ISO/OSI (sinistra) e il modello IEEE 802 (destra).

In breve, dal basso verso l'alto:

- **Physical Medium**: connettori e cavi sui quali i segnali viaggiano;
- **Physical Layer (PHY)**: un insieme di sublayer che definiscono segnali e codifiche;
- **Reconciliation SubLayer (RS)**: sublayer che permette di mappare e tradurre segnali per MAC;
- **Media Access Control (MAC) Sublayer**: la parte logica di Ethernet responsabile dell'implementazione del CSMA/CD e pacchettizzazione;
- **Logical Link Control (LLC) Sublayer**: (optional) sublayer che viene implementato quando Ethernet fa uso anche dello standard IEEE 802.2 Logical Link Control.

3.1 Ethernet

Se dovessimo citare le tre caratteristiche che dalla sua creazione, avvenuta nei primi anni '70, hanno fatto crescere Ethernet fino a renderla, ad oggi la rete LAN più diffusa al mondo, dovremmo sicuramente tenere in considerazione il basso costo, la velocità e la semplicità.
Infatti, nell'ordine si consideri che:

- il tipo di cablaggio, ovvero l'UTP (Unshielded Twisted Pair), è molto meno costoso di altri tipi che possono raggiungere le stesse prestazioni e provvede anche all'isolamento elettrico;
- ha dimostrato di adattarsi alla domanda del mercato. . Nel corso dei suoi 40 anni di vita Ethernet ha subito cambiamenti che ne hanno, per esempio, comportato l'aggiunta di nuove caratteristiche o il miglioramento delle prestazioni;
- la sua topologia può essere facilmente allargata al crescere del numero di nodi della rete.

Un altro importante componente caratterizzante Ethernet è il MAC, unico nel suo genere, denominato CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Esso descrive nel dettaglio come i nodi accedono alla rete o, più nello specifico in che modo i dispositivi della rete decidono le tempistiche di trasmissione ed i metodi per rilevare collisioni e nel caso ritrasmettere i frame.
Se una collisione viene rivelata durante una trasmissione, tutte le stazioni in quel momento attive nell'invio dei dati, attendono in modo casuale prima di riprovare a trasmettere di nuovo.

3.1.1 Topologia

Nella costruzione di una rete Ethernet è utile identificare, innanzitutto, i vari tipi di dispositivi, programmi e concetti ad essa appartenenti e necessari al suo funzionamento, in particolare:

- Network Devices;
- Network Interconnection Devices;

Network Devices (End Device, Host)

Fanno parte di questa categoria tutti quei dispositivi che necessitano di una rete per essere connessi tra di loro e si possono far coincidere con gli utenti della rete, chiamati anche host. In un veicolo possiamo identificare questi dispositivi, per esempio, nella centralina.
In genere includono:

- un *microcontrollore*;
- *porte* appropriate per connettersi alla rete, dette interfacce di rete;
- un *protocol stack* che implementa la tecnologia LAN da usare;
- un insieme di applicazioni che permettono di usare i dispositivi elencati per interfacciarsi con gli altri host.

Network Interconnection Devices

Si possono raggruppare in un'unica categoria tutti i dispositivi facenti parte della rete ma non identificabili come host, grazie ai quali è possibile, per esempio, interfacciare più reti.
I principali sono:

- **Ripetitori**: un rudimentale apparato o che connette fisicamente 2, e grazie all'uso di due porte replica ciò che riceve dalla prima sulla seconda;
- **Hub**: un ripetitore a più porte. I segnali entranti da una porta vengono replicati su tutte le altre;
- **Bridge**: simile ad un ripetitore, opera al secondo livello del modello ISO/OSI;
- **Switch**: un bridge multi-porta ma con la caratteristica di inoltrare i segnali ricevuti solo sulle uscite in cui essi sono richiesti.

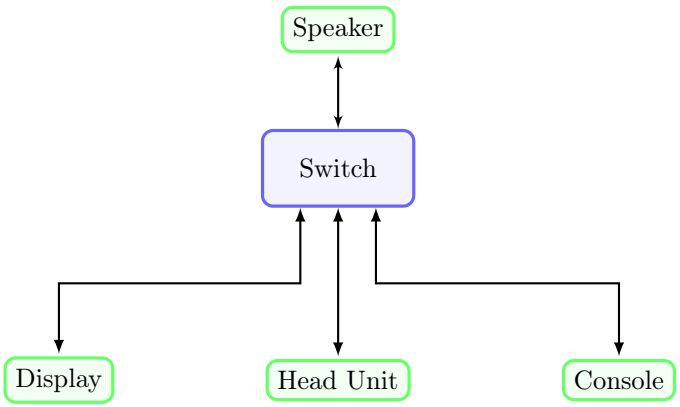

Figura 3.2: Topologia di una rete Ethernet applicata all'automotive.

3.1.2 Formato del Frame

Ethernet ha a sua disposizione vari tipi di formato del frame, ognuno adatto ad uno specifico tipo di applicazioni, ma tutti molto simili.
Il tipo di frame più diffuso è quello presente nella seconda versione, riportato graficamente in Figura 3.3.

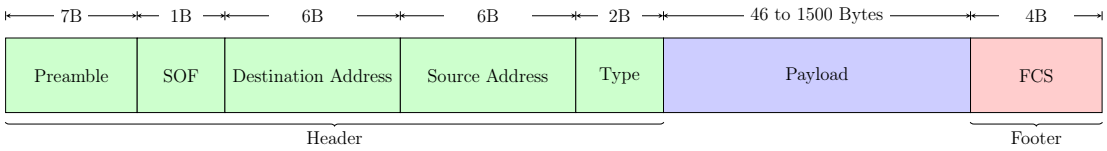

Figura 3.3: Seconda versione dell'Ethernet Frame.

Nel dettaglio esso di compone di:

- **Preamble**: stringa di 7 Byte di 0 e 1 alternati, che indica l'inizio della trasmissione di un nuovo frame e permette la sincronizzazione tra sorgente e destinazione;
- **Start of Frame (SOF)**: 1 Byte, formato da una sequenza predefinita, per indicare l'inizio del frame;
- **Destination and Source Address**: due campi che indicano l'indirizzo della sorgente e quello della destinazione, ognuno composto da 6 Byte ciascuno e fondamentale per l'addressing di Ethernet. Insieme, questi due campi, si possono considerare la sostanziale differenza con le reti automotive. L'Ethernet infatti, al contrario di CAN e LIN, ha la particolarità che permettere di stabilire il destinatario o i destinatari dei frame;
- **Type**: campo, formato da 2 byte, necessario per differenziare le tipologie di frame Ethernet che, come anticipato, possono essere molteplici;
- **Payload**: il campo dei dati trasmessi, ha una dimensione minima di 46 Byte ed una massima di 1500;
- **FCS (Frame Check Sequence)**: contiene 4 Byte per il CRC (Cyclic Redundancy Check), ovvero un campo che permette di controllare eventuali errori durante la trasmissione. Viene calcolato e scritto dalla sorgente basandosi sul Payload da inviare e viene confrontato dalla destinazione che lo ricalcola con ciò che ha ricevuto.

3.2 Automotive Ethernet

Ethernet 802.3 è nata come rete a bus condiviso con controllo dell'accesso tramite CSMA/CD, e aveva l'obiettivo di recapitare i frame "*as fast as possible*", senza offrire nessuna garanzia circa la consegna, la congestione, la latenza e la disponibilità della banda.
Può infatti accadere che, nel caso di congestione, le prestazioni si abbassino esponenzialmente.
Essendo spesso utilizzata con IP, rete a commutazione di pacchetto il cui controllo di flusso, quando richiesto, è demandato al livello superiore, l'Ethernet, ha mantenuto questa caratteristica "*best effort*" in tutte le sue evoluzioni. Per questo motivo, Ethernet non poteva essere considerata una rete di per sé adatta alla comunicazione real-time o multimediale, o in generale utile per applicazioni che richiedevano un ritardo massimo ed una banda garantiti, a meno che non si fossero trovate soluzioni proprietarie che avessero rotto la compatibilità con Ethernet standard.
Sulla base di queste prerogative, nel 2005 l'IEEE ha creato un gruppo di lavoro nell'802.1, denominato "Audio Video Bridging Task Group" (AVB Task Group). Lo scopo del gruppo era creare degli standard open per l'uso di Ethernet per queste applicazioni, in modo compatibile ed interoperabile, con un'iniziale attenzione alla comunicazione audio e video.
Nel tempo l'efficacia e l'utilità di questi standard non si sono limitate alle sole applicazioni audio e video, ma hanno spaziato verso tutte quelle applicazioni che richiedono flussi di dati a bassa latenza e sincronizzati, ed infatti, nel Novembre 2012, il gruppo di lavoro è stato rinominato "Time-Sensitive Networking Task Group".
AVB è anche il nome con cui si indica l'insieme degli standard pubblicati da AVB Task Group. Nello specifico essi sono:

- **Stream Reservation Protocol (SRP)**: descritto nello standard IEEE 802.1Qat "*IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)*";
- **Forwarding and Queuing for Time-Sensitive Streams (FQTSS)**: descritto nello standard IEEE 802.1Qav "*IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*";
- **generalized Precision Time Protocol (gPTP)**: descritto nello standard IEEE 802.1AS "*IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*";

- **Audio Video Bridging (AVB) System**: descritto nello standard IEEE 802.1BA "*IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems*".

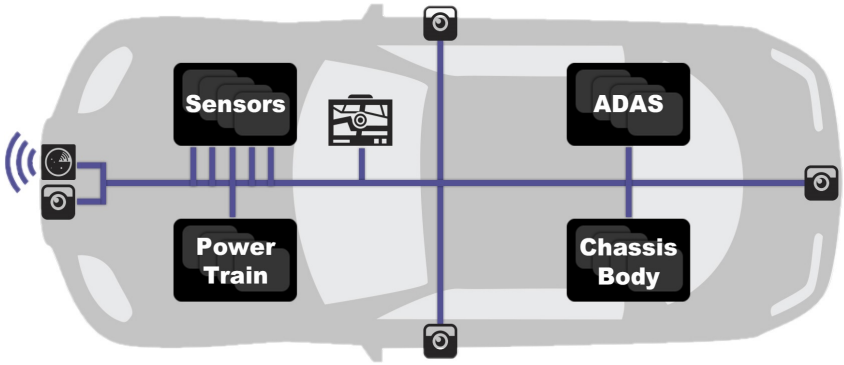

Figura 3.4: Applicazioni Automotive Ethernet [Figure credit: Avnu Alliance].

L'introduzione di questi standard, che verranno descritti nel dettaglio nel Capitolo 4, ha aperto ad Ethernet le porte anche nel settore automotive in diverse applicazioni:

- l'infotainment;
- il sistema di guida autonoma ADAS;
- backbone per supportare le attuali reti automotive.

Capitolo 4

Audio Video Bridging (AVB) Time-Sensitive Networking (TSN)

Gli standard che verranno esposti in questo capitolo giocano un ruolo fondamentale nel fornire ad Ethernet quelle caratteristiche specifiche per adattarsi al mondo automotive.
IEEE 802.Qat (Sezione 4.1) permette di riservare della banda a degli stream preimpostati, mediante l'utilizzo di messaggi di controllo.
IEEE 802.1AS (Sezione 4.2), basandosi su IEEE 1588 Precision Time Protocol, permette una sincronizzazione accurata tra i nodi della rete grazie, utilizzando un nodo Master che comunica le informazioni a tutti gli altri.
IEEE 802.1Qav (Sezione 4.3) gestisce la priorità di allocazione degli stream aggiungendo delle informazioni agli header Ethernet.
Infine, IEEE 802.BA (Sezione 4.4) mette a disposizione le funzionalità per identificare i profili AVB e i nodi all'interno della rete.
Come già detto, l'obiettivo iniziale dello standard AVB era riferito esclusivamente alla regolazione del traffico audio e video, ma ben presto fu chiaro che il suo potenziale poteva essere sfruttato anche per altri tipi di applicazione, nelle quali era necessaria una trasmissione sincrona.
L'interesse di poter applicare AVB in altri domini portò il team di IEEE ad apportare alcune modifiche e miglioramenti allo standard in modo da poterne facilitarne l'ingresso in campi diversificati, quali l'automotive e l'industriale. Ciò ha spinto le più grandi aziende di autoveicoli ad incentivare economicamente questo progetto [5].

4.1 Stream Reservation Protocol

Stream Reservation Protocol, o SRP, si basa su dei sotto-protocolli di segnalazione per stabilire, tra i nodi di una rete, degli stream riservati.

SRP fa affidamento al Multiple Stream Registration Protocol (MSRP) come protocollo di segnalazione, che dispensa alle "end station" l'abilità di riservare risorse di rete, per garantire la trasmissione e la ricezione di stream di dati.
Le "end station" vengono divise in Talker, dispositivi che producono gli stream, e Listener, dispositivi che leggono gli stream [3]. Nella rete sono anche presenti dei dispositivi intermedi, detti Bridge.
I Talker dichiarano i loro attributi che definiscono le caratteristiche degli stream, in modo che i Bridge abbiano a disposizione le informazioni necessarie nel momento in cui debbono allocare le risorse per uno stream. I Bridge propagano le informazioni lungo tutto il percorso dai Talker ai Listener, associando degli identificativi agli stream, chiamati StreamID, in modo da allocare il giusto numero di risorse nel momento delle richieste.

4.1.1 Comportamento delle End Station

Talker

Per annunciare quali stream possono essere inviati e le loro caratteristiche, i Talker usano una richiesta di *Join* per creare la "Talker Declaration"; qualora, invece, volesser disabilitare un determinato stream, essi inviano una richiesta di *Leave* per modificare le proprie informazioni presenti sulla rete.
Le Talker Declaration sono propagate sulla rete grazie a MSRP affinché i Listener e i Bridge siano avvisati della presenza di ogni Talker e degli stream che essi possono fornire [3].
Le Talker Declaration sono divise in:

- **Talker Advertise**: una comunicazione che avvisa che lo stream non ha incontrato limitazioni di banda o altri intoppi lungo il percorso che parte dal Talker;
- **Talker Failed**: un avviso che lo stream richiesto dal Listener non è disponibile o per limitazioni della banda o altri problemi lungo il percorso che lo congiunge al Talker.

Listener

Per annunciare quali stream vogliono ricevere, i Listener usano una richiesta di *Join* per creare la "Listener Declaration". Se invece volessero disabilitare la ricezione di un determinato tipo di stream, inviano una richiesta di *Leave* per modificare le proprio informazioni presenti sulla rete.

Le Listener Declaration trasmettono sulla rete anche le inforazioni riguardo banda e risorse allocate [3].
Le Listener Declaration sono divise in:

- **Listner Ready**: uno o più Listner hanno richiesto la ricezione di uno stream e c'è banda a sufficienza per soddisfare le richieste;
- **Listner Ready Failed**: due o più Listener hanno richiesto la ricezione di uno stream e c'è banda a sufficienza per soddisfare solo una richiesta;
- **Listner Asking Failed**: uno o più Listner hanno richiesto la ricezione di uno stream e non c'è banda a sufficienza per soddisfare le richieste.

4.1.2 Comportamento dei Bridge

Grazie a MSRP, i Bridge, sono capaci di registrare e cancellare le varie Listener e Talker Declaration. Qualsiasi cambio degli stati viene propagato su tutta la rete.
In generale, le Talker Declaration sono propagate a tutti gli altri Bridge presenti sulla rete. Al contrario, le Listener Declaration, sono trasmesse solo verso Bridge a cui i Talker sono associati [3].

4.2 generalized Precision Time Protocol

gPTP Uno degli aspetti fondamentali per la consegna di dati attendibili è la sincronizzazione tra le sorgenti e le destinazioni, dato che in applicazioni real-time, un ritardo di 10 ms può essere notevole, quindi è essenziale avere un meccanismo di sincronizzazione affidabile.

Nel Novembre del 2000, venne istituita una commissione IEEE allo scopo di creare lo standard IEEE 1588, ovvero Precision Time Protocol (PTP), atto a fornire un sistema per la sincronizzazione sulle reti. Precisamente l'obiettivo era creare un protocollo che:

- fornisse un'accuratezza ed una precisione di qualche microsecondo o meno;
- fosse usabile sia da dispositivi di fascia alta che bassa.

Questi obiettivi portarono lo sviluppo di un protocollo molto articolato.PTP, infatti, definisce vari tipi di clock e modelli per la sincronizzazione in base alle specifiche della rete, ed è inoltre uno standard definito come network-agnostic, cioè capace di funzionare senza conosccere il layout della rete.

Per creare qualcosa di più maneggevole per applicazione time-sensitive, nel 2006, IEEE ha rilasciato lo standard 802.1AS, gPTP (generalized Precision Time Protocol) [11]. Questo standard ha permesso di colmare le lacune di PTP riguardo alla facilità d'uso.

4.2.1 Dominio gPTP

Un dominio gPTP consiste in uno o più sistemi sincronizzati e collegamenti che rispettano le caratteristiche del protocollo IEEE 802.3AS [4].
In esso, ogni clock è identificato in modo esclusivo da una *ClockIdentity* e da una o più porte identificate dal *PortNumber*. Il binomio *ClockIdentity-PortNumber* costituisce la *PortIdentity* utilizzata come indirizzo logico [11]. Ogni dominio gPTP possiede un solo nodo, definito Grandmaster, usato come riferimento da tutto il dominio. Il GrandMaster può essere sia scelto automaticamente, che pre-assegnato, come avviene nell'ambito automotive, in modo da velocizzare l'avvio.

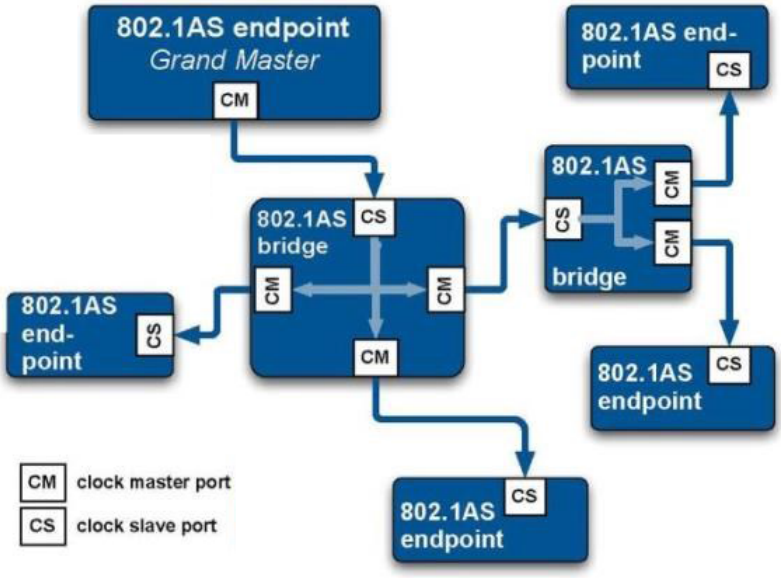

Figura 4.1: Dominio gPTP [Figure credit: Michael Johas Teener].

Durante l'inizializzazione della rete ne viene determinata anche l'estensione, considerandola come un albero che ha il GrandMaster come root. In ogni coppia di dispositivi collegati, uno si comporta come ClockMaster e uno da ClockSlave, e questa relazione Master/Slave viene propagata per tutto il dominio, in modo gerarchico (Figura 4.1).

4.2.2 Misurazione del Link Delay

Tutti i dispositivi sulla rete che ricevono il clock di riferimento, devono ovviamente fare delle piccole correzioni, aggiungendo ad esso il tempo di propagazione del messaggio trasmesso dal Master.
Questo tempo, definito come Link Delay, su una rete Ethernet viene misurato come mostrato in Figura 4.2

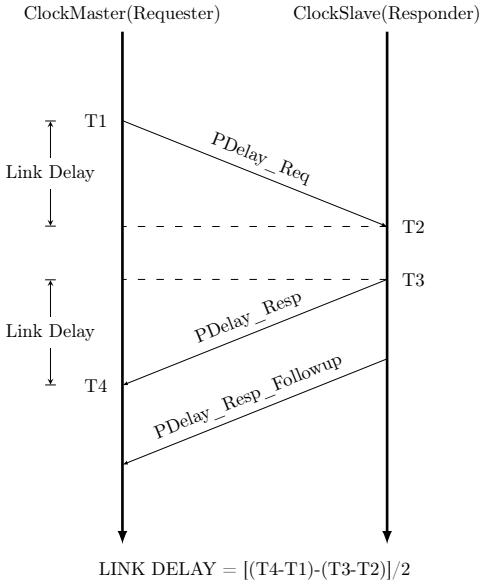

Figura 4.2: Calcolo del Link Delay.

1. Il richiedente schedula un *PDelay_Req*;
2. All'uscita del messaggio dal PHY, il tempo T1 viene catturato usando il ClockMaster;
3. Il tempo T2 viene catturato appena il messaggio *PDelay_Req* raggiunge il PHY dello Slave;
4. Il ricevitore (ClockSlave) schedula un *PDelay_Resp*;
5. Vengono salvati T3 e T4;
6. Il *PDelay_Resp_Followup* contiene i tempi T2 e T3.

4.2.3 Sincronizzazione del Clock

Ad intervalli definiti ma configurabili [1] avviene la sincronizzazione dei clock (Figura 4.3):

- Il ClockMaster schedula un messaggio *SYNC*;
- All'uscita del PHY, il tempo T1 viene salvato usando il clock del ClockMaster;

[1] lo standard de facto è fissato a 125 ms.

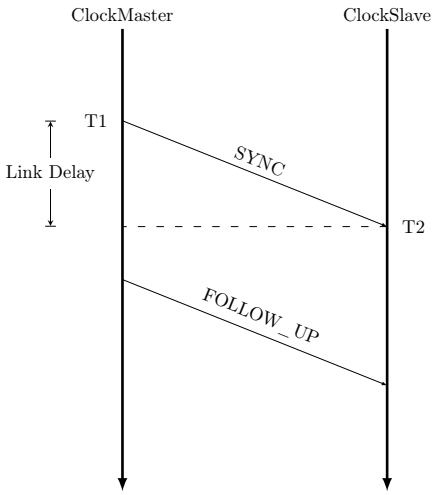

Figura 4.3: Sincronizzazione dei Clock.

- T2 vine catturato, usando il clock dello slave, appena passa dal PHY del ClockSlave;
- Un messaggio *FOLLOW_UP* trasporta T1 al ClockSlave.

Quando i messaggi di Sync attraversano la rete gPTP, vengono registrati vari valori:

- **Residence Time**: è la durata che un messaggio di Sync impiega per attraversare un Bridge; più è accurata questa misurazione, più sarà accurato il time di gPTP;
- **rateRatio**: una media delle differenze di frequenze tra nodi e GrandMaster.

4.3 Forwarding and Queuing of Time Sensitive Streams

Forwarding and Queuing of Time Sensitive Streams (FQTSS) è uno standard che consente ai Bridge di garantire prestazioni nelle comunicazioni real-time e time-sensitive, riguardanti dati audio e video [2]. In un Endopoint AVB, alcuni stream sono riservati alla trasmissione prioritaria su una rete best-effort e quste priorità vengono associate grazie a FQTSS.
FQTSS va a definire una specifica per creare code di messaggi, che in seguito vengono inoltrati basando l'ordine di invio sulla priorità, definita come *Credit-Based Shaper (CBS)*.

4.3.1 Credit-Based Shaper

CBS è un algoritmo per la trasmissione basato su uno schema a crediti (*credit*) che ricorda vagamente il funzionamento del *leaky bucket*, un algoritmo utilizzato per controllare che le trasmissioni di pacchetto dati siano ben delimitate in banda e velocità di trasmissione.

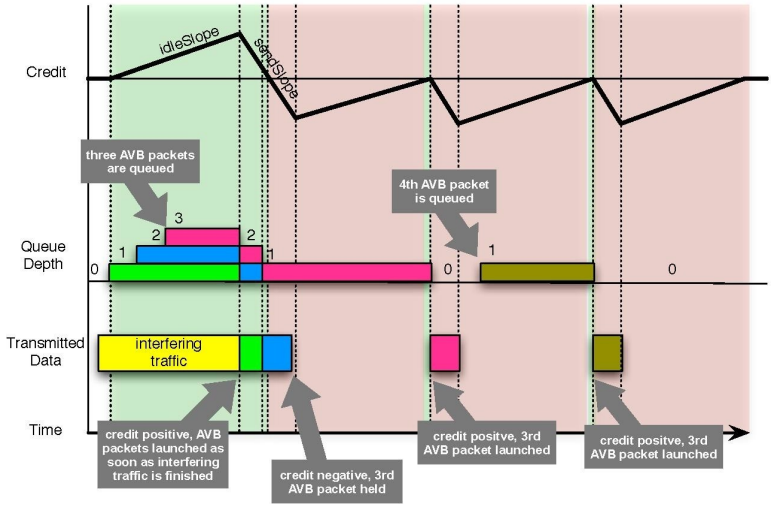

Figura 4.4: FQTSS e CBC diagramma [Figure credit: IEEE].

I frame di una determinata classe (descritte nella Sezione successiva 4.3.2) sono trasmessi solo se il valore di credit di tale classe risulta maggiore o uguale a zero. Se il credit è negativo, i frame di quella classe non verranno inviati, garantendo il trasferimento comunque di una buona percentuale di messaggi non prioritari. Nella Figura 4.4 si può notare un esempio del funzionamento di FQTSS e CBC. Il credit della classe, durante la trasmissione di traffico non prioritario, viene accumulato e si crea al contempo una coda di pacchetti, che saranno rilasciati non appena il traffico interferente finisce, perché ci si ritrova in una situazione di credit positivo. Alla fine dell'invio del secondo pacchetto, il credit negativo non permette di inoltrarne un terzo rimasto in coda, che invece dovrà attendere che il credit torni nuovamente a un valore (maggiore o) uguale a zero [11].
Il credit viene calcolato per classe e per porta e basandosi su due parametri: *idleSlope*, che rappresenta il rapporto con cui vengono accumulati i credit e il *sendSlope*, il rapporto con cui i credit vengono consumati durante l'invio di messaggi. Sono calcolati come:

$$idleSlope = \frac{reservedBytes}{classMeasurementInterval}$$

$$sendSlope = idleSlope - portTransmitRate$$

Alcuni regole sui credit sono:

1. se non ci sono stream AVB da mandare, il credit viene settato a zero;
2. durante l'invio di uno stream AVB il credit si riduce in accordo con il *sendSlope*;
3. se non ci sono stream AVB da mandare e il credit è negetivo, quest'ultimo cresce in accordo al *idleSlope*;
4. se uno stream AVB è bloccato perché uno non-AVB è in trasmissione, i crediti vengono accumulati seguendo sempre il parametro *idleSlope*, portando il credit ad un valore positivo.

4.3.2 Classi AVB

Il traffico AVB è classificato sulla velocità di trasferimento, ovvero ogni quanto, in media, i pacchetti vengono mandati sulla rete. Con un'alta velocità di trasferimento si ha meno latenza per ogni hop ma ciò porta un sovraccarico delle CPU [11].
I nodi che supportano FQTSS devono implementare obbligatoriamente anche 2 code AVB distinte, che possono essere usate per la stessa classe o per due classi diverse. Le classi base di FQTSS sono due:

- **Classe A**: gli stream di questa sono trasmessi ad una velocità di 8000 pacchetti al secondo, uno ogni 125 µs. È una classe in genere usata per quelle applicazioni dove si ha una necessità molto elevata di una bassa latenza (applicazioni real-time).
- **Classe B**: in questa classe, invece, gli stream sono trasmessi ad una velocità di 4000 pacchetti al secondo, uno ogni 250 µs, la metà della classe A.

È possibile, anche, che un utente possa definire le proprie classi, andando a settare la velocità di trasmissione desiderata, un identificativo ed una priorità. Le caratteristiche di queste classi sono riportate nella Tabella 4.1, dove si può anche notare un esempio di classe definita dall'utente.

SR Class	SR ID	SR Priority	Measurament Interval (µs)	Packets per second
A	6	3	125	8000
B	5	2	250	4000
User Defined	4	2	1333	750

Tabella 4.1: Riepilogo Classi SR.

4.4 Audio Video Bridging (AVB) System

Per avere una rete funzionante nel trasporto di dati time-sensitive, è indispensabile definire dei profili di funzionamento in modo da poter unire i vari standard descritti fin ad ora.
IEEE 802.1BA mette a disposizione dei profili che selezionano funzioni, opzioni, configurazioni, protocolli e procedure per i Bridge e per le End Station della LAN, e che sono indispensabili per costruire una rete capace di trasportare i dati per quelle applicazioni che necessitano di basse latenze [5].
In particolare, questo standard va a definire le caratteristiche e le opzioni che supportano le funzionalità dei livelli 1 e 2 del modello ISO/OSI, in modo tale da precisare i requisiti per una rete LAN.
Gli obiettivi di questo standard, che vedremo nel dettaglio nelle prossime sezioni, sono:

- descrivere i componenti che possono essere usati per creare una rete AVB e come possono essere combinati;
- descrivere le conseguenze e le limitazioni portate dall'uso di stream non AVB in una rete AVB;
- definire funzioni addizionali non descritte negli altri standard della famiglia;
- fornire dei requisiti sulla latenza per ottenere una rete AVB che funzioni correttamente;

4.4.1 Architettura di una rete AVB

Una rete AVB è formata da vari componenti che partecipano alla comunicazione e rispettano gli standard da essa dettati. Tali dispositivi sono:

- End Station identificabili come Talker, Listener o entrambi;

- Bridge che supportano i requisiti di AVB;
- LAN che interconnettono Bridge, Talker e Listener;

Ovviamente, è possibile avere delle reti ibride, cioè costituite da dispositivi AVB, assieme a dispositivi non-AVB che fanno parte del sistema ma che non partecipano alla comunicazione, poiché non ricevono e non trasmettono un traffico time-sensitive. La Figura 4.5 mostra questo tipo di situazione.

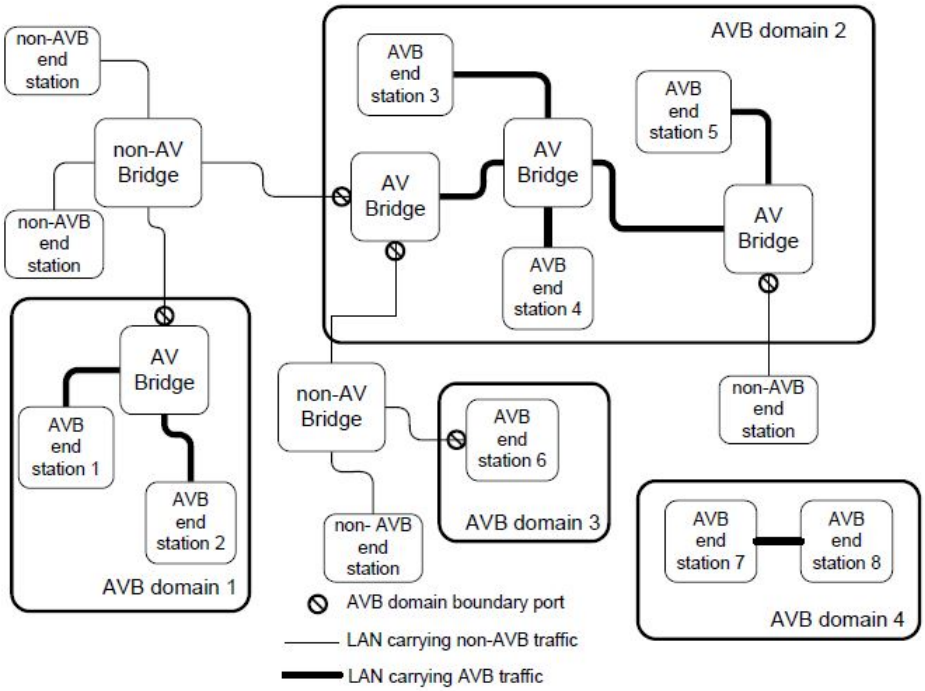

Figura 4.5: Domini AVB [Figure credit: IEEE].

Si notano quattro domini AVB (Sezione 4.4.2). Le stazioni 1 e 2 facenti parte del dominio AVB 1 possono comunicare tra di loro creando degli stream AVB tramite il Bridge AV. Situazione analoga avviene nel dominio 2 con le End Staion 3, 4 e 5, nel dominio 4 con le End Station 7 e 8. Nel dominio 3, invece, la sola End Station 6 non potrà creare AVB stream con altre stazioni, essendo essa l'unica presente. I quattro domini non potranno però scambiare stream, essendo collegati tra loro da zone sulle quali non è possibile usare AVB e perciò non sarà possibile avere garanzie sulla priorità e sulla latenza, poiché esse non possono essere garantite al di fuori di un dominio AVB [5].

4.4.2 Domini AVB

Un dominio AVB è l'intersezione di un dominio SRP e un dominio gPTP.
IEEE 802.1Q definisce un dominio SRP come un insieme di dispositivi collegati su una LAN che supporta SRP, aventi tutti la stessa priorità per classe di appartenenza.
IEEE 802.1AS, invece, definisce un dominio gPTP come un insieme di dispositivi connessi che supportano gPTP.
La disponibilità contemporanea di SRP e gPTP è considerata una condizione necessaria per il supporto delle operazioni di AVB.

Lo standard AVB estende la definizione di dominio SRP per prendere in consderazione anche le caratteristiche della rete e le informazioni acquisite tramite gPTP:

- se la porta non supporta gPTP, o se la rete collegata a quella porta non consente l'uso di AVB, allora:
  1. il valore di del parametro di SRP che definisce se una porta è di confine o no (*SRPdomainBoundaryPort*) viene settato a TRUE;
  2. da quella stessa porta viene propagato un segnale (*Talker Failed*) per indicare che non è possibile usare in uscita il protocollo AVB su quella porta.
- se la porta supporta gPTP, le viene attribuito un valore (associato a *SRPdomainBoundaryPort*) per indicare le classi SR supportate [5].

4.4.3 Requisiti per i Bridge

Per essere considerati AVB Bridge e quindi poter supportare gli stream di questo standard, ci sono alcuni requisiti che questi devono possedere:

- tutte le porte devono supportare almeno uno stream, associato ad una classe che supporti FQTSS;
- tutte le porte devono supportare la registrazione, la dichiarazione e la propagazione dei messaggi MSRP;
- tutte le porte devono supportare gPTP;
- devono supportare la Classe B;
- se hanno più di una porta, tutte queste devono supportare anche la classe A [5].

4.4.4 Requisiti per i Talker

I requisiti per i Talker, invece, sono:

- devono essere capaci di trasmettere almeno uno stream;
- devono essere capaci di dichiarare gli attributi di MSRP, associati ad ogni stream;
- le porte devono supportare gPTP;
- devono poter essere capaci di operare da GrandMaster [5].

4.4.5 Requisiti per i Listener

I requisiti per i Listener, invece, sono:

- devono essere capaci di ricevere almeno uno stream;
- devono essere capaci di registrare gli attributi di MSRP associati ad ogni stream;
- le porte devono supportare gPTP [5].

Capitolo 5

Sviluppo

Durante il periodo di Tesi è stato condotto uno studio per la costruzione di una Backbone Automotive Ethernet con l'intento di verificare la sua applicabilità all'interno di un autoveicolo.
Una Backbone è una porzione di una grossa rete che ne unisce delle zone, garantendo alta velocità di trasmissione, bassa latenza e poche collisioni.
Applicando questa idea al campo automotive, si è arrivati a definire la possibilità di creare un graduale passaggio dalle attuali tipologie di rete automotive, ormai, per certi aspetti, obsolete ed inadeguate, all'Automotive Ethernet, una tecnologia robusta ed innovativa che riesce a rispettare molte restrizioni dettate dall'industria dell'autoveicolo.

L'implementazione del sistema si è articolata in diverse fasi: in una fase iniziale è stato condotto uno studio del campo automotive, in particolare relativo ai protocolli necessari per lo sviluppo e alle limitazioni dettate dall'industria dell'autoveicolo; una seconda fase è stata invece dedicata alla creazione delle applicazioni: un talker per leggere un messaggio alla volta e un listener per scriverlo su un CAN-bus; si è poi proceduto ad implementare la comunicazione tra queste applicazioni, e a verificare l'ordine dei messaggi ricevuti e la loro identità rispetto a quelli inviati;

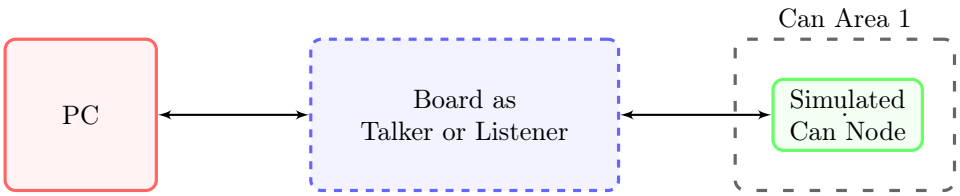

Figura 5.1: Prima fase: lettura o scrittura con un nodo simulato.

Alla realizzazione del primo step, ne è seguito un secondo, volto a creare una coda sull'applicazione talker, necessaria a poter trasmettere molteplici messaggi in

contemporanea. Infine, il sistema è stato fatto interagire con una reale applicazione automotive, testandolo con un computer di bordo allo scopo di testarne il corretto funzionamento.

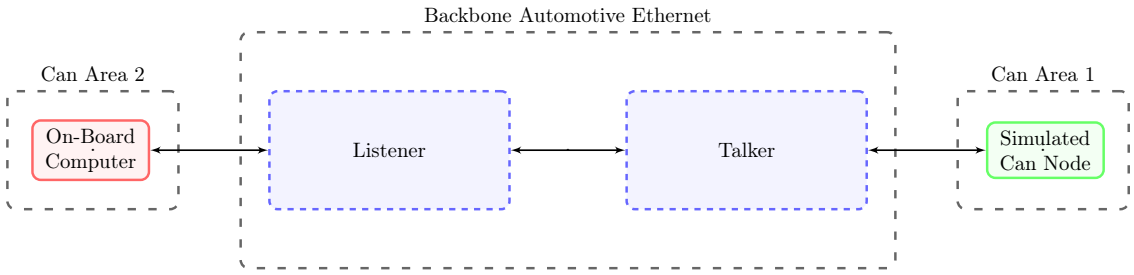

Figura 5.2: Schema seconda fase: testing con una reale appicazione.

Per lo sviluppo del sistema, oltre ai protocolli descritti fin ad ora, si è considerato un altro standard, IEEE 1722-2016 (*Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks*) (descritto nella Sezione 5.1), che traccia le linee guida per l'incapsulamento dei dati e per le procedure di sincronizzazione da usare con Ethernet, pur nel rispetto degli standard di IEEE 802 precedentemente descritti(Sezione 5.1).
Le librerie usate come base per lo sviluppo sono state create da Avnu Alliance, e saranno trattate nel paragrafo dedicato al software utilizzato (Sezione 5.3), mentre l'hardware utilizzato per il testing (Sezione 5.2) è stato fornito da Teoresi S.p.A..

5.1 AVTP

Audio/Video Transport Protocol (AVTP) è uno standard creato per facilitare l'interoperabilità tra le End Station che trasmettono dati time-sensitive e, andando a definire nuovi formati di pacchetti, è in grado di garantire la sincronizzazione, restrizioni sulla latenza e servizi di rete
Nel dettaglio, questo standard, va a specificare un protocollo per dati audio, video e controllo su una rete time-sensitive, una rete quindi in grado di supportare TSN. AVTP è stato sviluppato per poter usare, su una rete TSN, il supporto offerto da gPTP, SRP, e FQTSS.
Gli stream creati secondo lo standard AVTP possono essere trasportati da vari protocolli di rete, come, appunto, 802.3 o 802.11 o addirittura incapsularli in IP [6].
Tutte i nodi del dominio AVTP, che mandano o ricevono audio, video o comunque dati time-sensitivi devono necessariamente supportare questi tre protocolli: gPTP, SRP, FQTSS.
AVTP, come già accennato, si appoggia sui servizi offerti da questi standard:

- **gPTP**: per provvedere a tutta la rete le informazioni riguardo la sincronizzazione;
- **SRP** e **FQTSS**: per garantire la consegna dei dati, dai Talker ai Listener, con una latenza limitata;

AVTP, descrive i protocolli per pacchettizzare dati audio, video o di controllo. Nella prossima sezione si andrà a descrivere solo il formato dei pacchetti per la comunicazione sincrona e il relativo metodo di trasmissione per l'inoltro di dati di controllo, in quanto è ciò che è stato utilizzato nel progetto.

5.1.1 AVTP Control Format (ACF)

AVTP Control Format (ACF) fornisce un framework flessibile per la trasmissione di vari messaggi di controllo sia in maniera sincrona che non, usando TSN.
Grazie ad AVTP è possibile costruire un messaggio chiamato AVTP Protocol Data Unit(AVTPDU) con un header, che può rappresentare o una trasmissione sincronizzata o una non sincronizzata, e un payload, in grado invece di inglobare uno o più messaggi ACF. La differenza sostanziale dei due tipi di header è l'uso di alcuni campi della struttura atti ad inserire in essa le informazioni riguardati il timestamp. L'header d'interesse per lo sviluppo di questa tesi è stato il *Time-Synchronous Control Format header*, ovvero quello per la trasmissione sincronizzata, mostrato in figura 5.3.

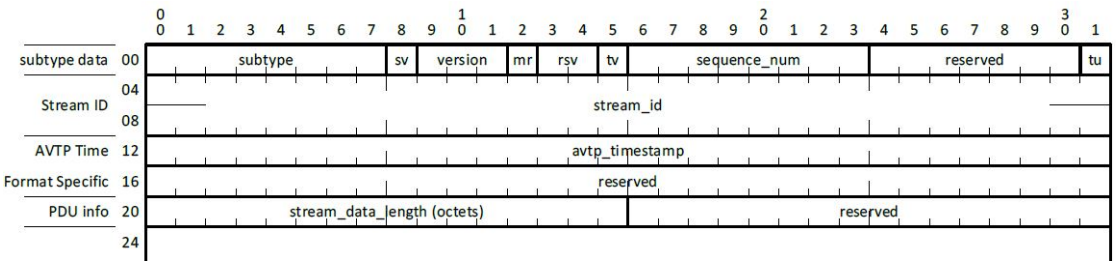

Figura 5.3: Time-Synchronous Control Format header.

Mentre per quanto riguarda il Payload, si prenderà in considerazione il *CAN ACF message*, mostrato in Figura 5.4, utilizzato per trasportare i dati dei messaggi CAN.

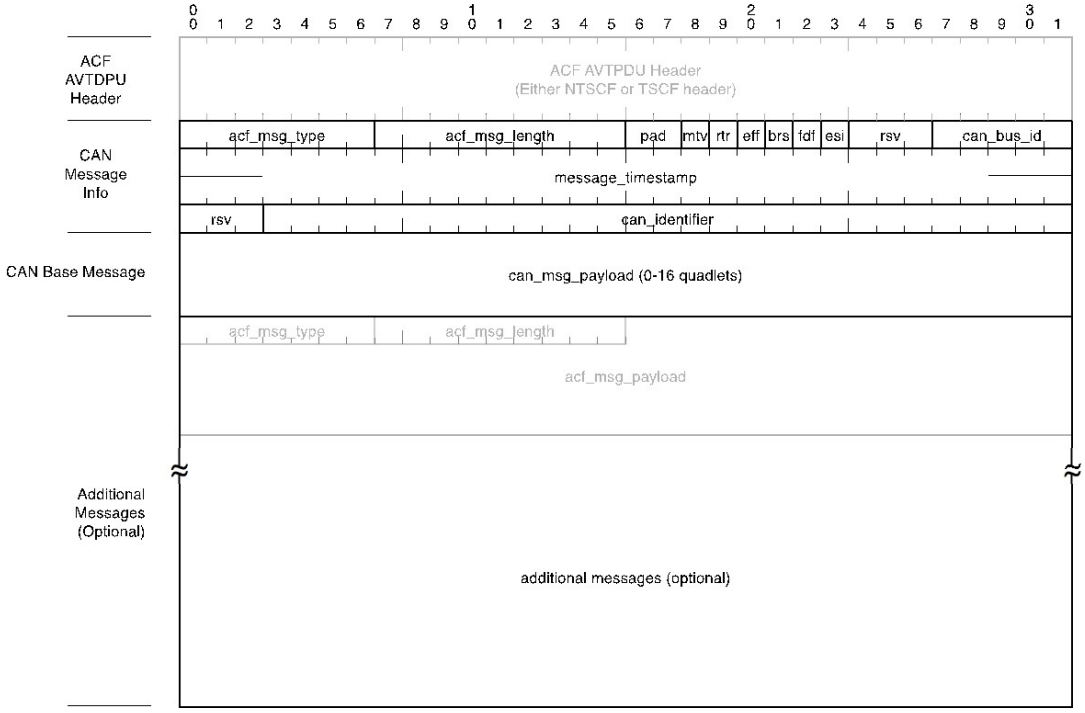

Figura 5.4: CAN ACF message.

5.2 Hardware utilizzato

Durante lo sviluppo del progetto sono stati utilizzati vari dispositivi hardware. Nello specifico per la costruzione della Backbone Automotive Ethernet sono state utilizzate due board, precisamente delle Nit6_SoloX di Boudary Devices (Sezione 5.2.1) come talker e listener, collegate da un cavo Ethernet. Le reti CAN messe in comunicazione erano: da un lato simulata, dall'altro, inizialmente un ValueCAN3 (Sezione 5.2.2), collegato ad un PC, per il confronto dei dati e in seguito un computer di bordo per poter testare la corretta comunicazione in una reale applicazione automotive.

5.2.1 Nit6_SoloX

La Nit6_SoloX (Figura 5.5) è una board basata su i.MX6 SoloX, un processore multi-purpose, facente parte della famiglia degli i.MX6, che utilizza sia un ARM Cortex-A9 che un ARM Cortex-M4 in supporto del primo.
Tra le caratteristiche principali di queste board troviamo, innanzitutto, la possibilità di poter usare il protocollo FQTSS, supportato perfettamente da queste schede, due porte Gigabit Ethernet integrate e 2 porte per il CAN bus.

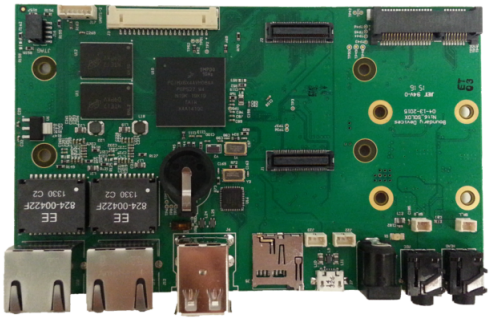

Figura 5.5: Nit6_SoloX

5.2.2 ValueCAN3

ValueCAN3 è un dispositivo semplice e di alta qualità, che offre la possibilità di collegare un PC ad un CAN bus, con la possibilità di simulare un nodo CAN. Questo dispositivo, abbinato ad un software adeguato, rende possibile scrivere e leggere su un CAN bus.

Figura 5.6: ValueCAN3

5.3 Software utilizzato

L'implementazione del sistema è avvenuta con l'ausilio di sistemi operativi, driver, framework e software, inizialmente necessari per lo sviluppo degli applicativi e, successivamente, per il testing del risultato prodotto.
Nel dettaglio, i supporti utilizzati sono stati:

- **Sistemi operativi**: Debian, Windows 10;
- **Driver**: SocketCAN;

- **Framework**: AVTP-Pipeline;
- **Software**: BUSMASTER.

Di seguito si andranno ad evidenziare le caratteristiche e il ruolo giocato da SocketCAN (Sezione 5.3.1), AVTP-Pipeline (Sezione 5.3.2), e BUSMASTER (Sezione 5.3.3) nello sviluppo del progetto.

5.3.1 SocketCAN

SocketCAN è un set di driver Open Source, usati per sfruttare il CAN in ambiente Linux, dal momento che il suo impiego rende possibile leggere e scrivere su un CAN-bus collegato al dispositivo ospitante questo sistema operativo.
Le librerie fornite nel package sono costruite basandosi sullo stack di rete di Linux, ciò vuol dire che sono estremamente simili al protocollo TCP/IP, garantendo allo sviluppatore, che ha familiarità con la programmazione di rete in ambiente Linux, un'alta facilità d'uso.

Durante lo sviluppo della Backbone Automotive Ethernet, queste librerie, sono risultati indispensabili per interfacciarsi con i CAN-bus presenti sulle board e per testare le applicazioni prodotte.Infatti, tra le funzioni a disposizione in SocketCAN, alcune permettono di inviare messaggi CAN con i dati del Payload generati casualmente.

5.3.2 AVTP-Pipeline

AVTP-Pipeline è un framework, scritto in linguaggio C, contenente i necessari componenti per l'implementazione di applicazioni che fanno uso di AVB. Il framework prende parte di un progetto open source, denominato OpenAvnu, sviluppato da Avnu Alliance, un'organizzazione che promuove l'uso di AVB su Ethernet.
AVTP-Pipeline mette a disposizione strutture, code, funzioni e callback per poter interfacciarsi con il livello fisico di Ethernet e per poter accedere ai dati acquisiti con gPTP. OpenAvnu, su Linux, utilizza un daemon gPTP per la sincronizzazione della comunicazione tra Talker e Listener e delega all'hardware l'implementazione del protocollo FQTSS.
In Figura 5.7 si può notare la struttura di una end station e il particolare del flow degli stream:

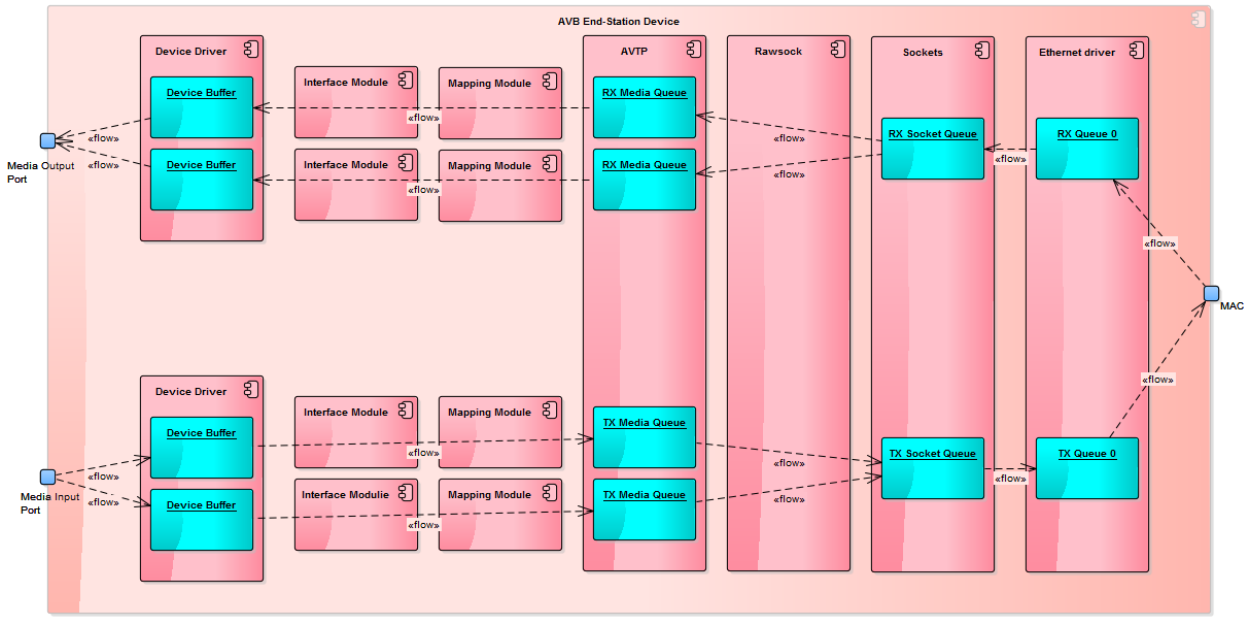

Figura 5.7: AVTP Data Flow [Figure credit: Avnu Alliance].

Nel sistema sviluppato, "*Media Input Port*" e "*Media Output Port*", all'estrema sinistra della figura, sono le porte CAN presenti sulla Nit6_SoloX, grazie alle quali è possibile collegarsi ai CAN-Bus; "*Device Driver*" è identificabile in SocketCAN, precedentemente descritto, necessario per scrivere e leggere sul CAN-Bus; "*Interface Module*" e "*Mapping Module*" sono le applicazioni sviluppate durante il periodo di Tesi, sia per il Talker che per il Listener; il modulo "*AVTP*" è la parte di OpenAvnu sfruttata per collegare il software sviluppato con il livello fisico di Ethernet, raffigurato nella parte a destra di questo modulo.

5.3.3 BUSMASTER

BUSMASTER è un software per sistemi Microsoft open source e di facile utilizzo, che permette di monitorare, analizzare i CAN-bus e di simulare messaggi CAN.
Utilizzando un dispositivo USB CAN, è possibile monitorare anche più bus contemporaneamente.
Ha 2 modi di funzionamento:

- **Attivo**: influenza il bus, per esempio simulando un nodo che invia messaggi;
- **Passivo**: rimane invisibile sul bus, facendo semplicemente packet sniffing.

Per lo sviluppo del progetto, BUSMASTER, abbinato a ValueCAN3, si è dimostrato utile nel testare il corretto funzionamento del Listener e il suo corretto funzionamento, ovvero l'inoltro dei messaggi CAN ricevuti sotto forma di messaggi ACF.

5.4 Software sviluppato

La fase di sviluppo delle applicazione software, Listener e Talker, è stata affrontata seguendo degli step progressivi, con l'intento di incrementare passo dopo passo le loro funzionalità.

5.4.1 Prima fase di sviluppo

Durante la prima parte, sono state sviluppate due applicazioni per la lettura e scrittura su CAN-bus. Poiché, come già descritto, i drive SocketCAN, rispecchiano lo stile di programmazione dei socket generici di Linux, è bastato associare dei file descriptor alle porte dei CAN-bus per avere l'accesso alle loro informazioni e ai loro dati. La trasmissione e la ricezione sono supportate dalla struttura, molto intuitiva, fornita dai driver stessi:

```
struct can_frame {
   canid_t can_id; /* 32 bit CAN_ID + EFF/RTR/ERR flags */
   __u8    can_dlc; /* frame payload length in byte (0 .. 8) */
   __u8    __pad;  /* padding */
   __u8    __res0; /* reserved / padding */
   __u8    __res1; /* reserved / padding */
   __u8    data[8] __attribute__((aligned(8)));
};
```

Listing 5.1: Struttura can_frame di SocketCAN

Si noti, nello specifico, il campo `can_id` costituito da 32 bit con il quale è possibile garantire la compatibilità con l'evoluzione di CAN, ovvero CANFD, che ha un id di 29 bit anziché di 11. Gli ulteriori 3 bit rappresentano 3 flag propri dei messaggi CAN. In `__pad` è presente il numero di byte che serve per ottene una word completa (4 Byte) nel campo `data`, utile per l'allineamento di più messaggi CAN.
Si è quindi proceduto a testare, con l'ausilio di BUSMASTER e ValueCAN3, l'assenza di problematiche durante la comunicazione:

- per il Listener sono stati generati messaggi CAN con un nodo simulato da BUSMASTER ed inoltrati tramite ValueCAN3, in modo che venissero visualizzati a video, su PC, grazie ad una connessione SSH con la board;
- per il Talker, invece, sono stati prodotti dalla board stessa i messaggi CAN, poi visualizzati con l'ausilio di ValueCAN3 e BUSMASTER in Passive Mode.

5.4.2 Seconda fase di sviluppo

Il secondo step nella realizzazione del progetto di tesi è stato sviluppare due nuove applicazioni, usando in questo caso il framework AVTP-Pipeline, per implementare

la connessione tra le due board nel rispetto di tutti i protocolli previsti per Automotive Ethernet. Si è proseguito in seguito ad unire le applicazioni sviluppate nella fase precedente per ottenere nel concreto la realizzazione della Backbone Automotive Ethernet tra due reti CAN.
Durante questa fase si è usato lo stack implementativo di AVTP-Pipeline che ha permesso di costruire una comunicazione, attraverso la divisione dell'applicazione in due moduli: *Mapping Module* e *Interface Module*.

Interface Module

L'Interface Module rappresenta la porzione di applicazione che rende possibile la comunicazione con le interfacce del CAN-Bus.
La prima parte dello sviluppo del progetto ha condotto alla stesura di funzioni che hanno permesso durante questa fase, di aprire e chiudere i socket sul CAN-Bus, e anche di leggere e scrivere messaggi CAN.
Per quanto riguarda il Talker, l'utilizzo di una callback, chiamata a ciascuno scatto di un timer a sua volta impostato su un valore dipendente dalla classe SRP scelta (es. Classe B, Timer di 125 µs), consente la lettura di un messaggio CAN. Il framework inoltre fornisce una coda che favorisce il fluire delle informazioni del messaggio CAN, presenti nella struttura `can_frame` 5.1, verso il Mapping Module che a sua volta, dopo le opportune trasformazioni, li inoltrerà su Ethernet.
Per il Listener, invece, viene chiamata una callback appena un elemento è presente nella suddetta coda. Questa callback si occuperà di ricostruire il messaggio CAN e inoltrarlo sulla rete CAN collegata al Listener.

Mapping Module

Il Mapping Module consiste nella parte dell'applicazione che permette la connessione al livello fisico di Ethernet. In questa porzione di codice viene effettuato l'inserimento delle informazioni dei messaggi CAN in un ACF-Message, per quanto riguarda il Talker, e l'estrapolazione delle informazioni da esso, per quanto riguarda il Listener.
Nel Talker, avviene la realizzazione di un messaggio ACF, poi inoltrato su Ethernet grazie ad una struttura interna. Questo tipo di costruzione è resa possibile grazie ad una callback collegata alla coda, in cui sono presenti i messaggi CAN provenienti dall'Interface Module.
La stessa struttura permette al Listener di essere risvegliato all'arrivo di un nuovo messaggio ACF. La callback del Listener si occuperà in questo caso di estrapolare le informazioni presenti nel messaggio ricevuto ed inserirle nella struttura `can_frame` 5.1, prima di essere poi passate all'Interface Module.

5.4.3 Terza fase di sviluppo

Durante la terza parte dello sviluppo si è dato rilievo all'ottimizzazione della comunicazione e delle procedure di lettura e scrittura.
Alla conclusione della seconda parte del lavoro, il Talker era in grado di leggere, incapsulare e inviare un messaggio alla volta, quindi in realtà veniva occupata solo una piccola porzione del payload di un messaggio Ethernet.
Per questo motivo, al fine di ottimizzare le applicazioni, si è quindi andati a modificare la porzione di codice che si occupava della lettura dal CAN-bus. La lettura di un singolo messaggio è diventata, grazie alle modifiche apportate, una lettura che garantisce il riempimento del payload del messaggio Ethernet fino al 99.2% dello spazio disponibile, in condizioni ideali, cioè una disponibilità di messaggi CAN da leggere infinita. In assenza del massimo ideale di messaggi che permette di raggiungere questa percentuale, un timer, associato a quello principale presente nel Talker, interrompe l'operazione di lettura, in modo che l'applicazione può procedere all'invio dei messaggi su Ethenet.
Per quello che riguarda il perfezionamento del Listner, le modifiche apportate durante questa fase sono state relativamente piccole e hanno riguardato solo l'estrapolazione delle informazioni di tutti i messaggi CAN, presenti nell'ACF-message. L'invio su CAN-bus, di contro, è rimasto essenzialmente uguale, poiché i messaggi CAN ricostruiti sono inoltrati nello stesso ordine con il quale sono stati letti.
Alla fine di questa fase di ottimizzazione, il sistema è stato testato collegandolo ad un reale computer di bordo fornito da Teorsi S.p.A..
I messaggi CAN generati con BUSMASTER, e scelti da un database installato su di esso, sono stati inviati al Talker tramite ValueCAN3, che, a sua volta li trasformava in messaggi ACF, e li inoltrava al Listener. Successivamente, il Listener, ricostruiti i messaggi CAN nell'ordine che erano stati generati da BUSMASTER, venivano mandati al computer di bordo, il quale, riconoscendoli, eseguiva i comandi abbinati a questi messaggi.

Capitolo 6

Conclusioni

La concorrenza nel mercato automobilistico, in concomitanza con l'evolversi del processo tecnologico spinge I produttori di autoveicoli a ricercare e sviluppare, continuamente, prodotti sofisticati per soddisfare requisiti e legislazioni riguardati prestazioni, sicurezza e consumi. Allo stesso tempo, l'interesse principale rimane ovviamente quello di sviluppare un aumento delle prestazione dei dispositivi elettronici a bordo del veicolo, fenomeno che ha assistito ad una repentina spinta, specialmente nelle ultime due decadi.
Questo incremento di sistemi elettronici a bordo comporta, da una parte un'estrema tecnologizzazione dell'autoveicolo, che si accompagna però anche a complessi problemi durante la loro fase di progettazione e di sviluppo.
L'idea alla base della Tesi, è nata, proprio, dalla necessità sempre più imminente delle industrie del settore automotive di poter progredire, sviluppando nuovi sistemi o incrementando le potenzialità di quelli già esistenti, senza le restrizioni imposte dal mezzo di trasmissione, attualmente basato su protocolli obsoleti, e senza ulteriori difficoltà nella progettazione.
Inserendo una Backbone Automotive Ethernet all'interno di un autoveicolo è possibile creare un passaggio graduale all'Automotive Ethernet, senza sconvolgere l'attuale processo di sviluppo e produzione di autoveicoli. Questo lavoro di Tesi, infatti, non si è limitato solo alla progettazione di una Backbone Automotive Ethernet, ma ha voluto evidenziare anche la possibilità di poter integrare, con i mezzi attualmente a disposizione, questo tipo di rete in un sistema già esistente, senza necessità di cambiamenti radicali dell'attuale processo produttivo.
Il progetto ha potuto dimostrare che è possibile quindi l'inserimento di una Backbone Automotive Ethernet, senza sconvolgere gli attuali progetti di cablaggio di un autoveicolo, poiché essa risulta del tutto invisibile alle reti CAN.
Questo permetterà alle aziende dell'industria automotive di iniziare una transizione per rinnovare la tecnologia classica di rete all'interno dell'auto veicolo, con una più innovativa, tramite la quale sarà possibile ottenere una comunicazione più efficiente al suo interno. Questo potrebbe condurre ad una nuova fase di crescita, concretizza-

bile con l'inserimento nell'autoveicolo di dispositivi che, sebbene richiedano alla rete di comunicazione alte prestazioni, permettono di sviluppare, potenziare o ottimizzare le applicazioni per l'autoveicolo senza le restrizioni dettate dalle reti attualmente esistenti, ma ormai considerate obsolete.

Bibliografia

[1] Teoresi in breve. http://www.teoresigroup.com/chi-siamo/in-breve/?lang=it.

[2] Ieee standard for local and metropolitan area networks - virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pages C1–72, Jan 2009.

[3] Ieee standard for local and metropolitan area networks—virtual bridged local area networks amendment 14: Stream reservation protocol (srp). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pages 1–119, Sept 2010.

[4] Ieee standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks. *IEEE Std 802.1AS-2011*, pages 1–292, March 2011.

[5] Ieee standard for local and metropolitan area networks–audio video bridging (avb) systems. *IEEE Std 802.1BA-2011*, pages 1–45, Sept 2011.

[6] Ieee standard for a transport protocol for time-sensitive applications in bridged local area networks. *IEEE Std 1722-2016 (Revision of IEEE Std 1722-2011)*, pages 1–233, Dec 2016.

[7] Grzemba Andreas. *MOST : the automotive multimedia network; from MOST25 to MOST150*. Franzis, Poing, 2011.

[8] Encyclopedia Britannica, Alan K. Binder and John Bell Rae. Automotive industry. https://global.britannica.com/technology/automotive-industry, July 2017. [Online; accessed 23-Feb-2018].

[9] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro. Ieee 802.3az: the road to energy efficient ethernet. *IEEE Communications Magazine*, 48(11):50–56, November 2010.

[10] Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller area network (can) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, Apr 2007.

[11] C.M. Kozierok, C. Correa, R.B. Boatright, and J. Quesnelle. *Automotive Ethernet - the Definitive Guide*. Intrepid Control Systems, Incorporated, 2014.

[12] Renjun Li, Chu Liu, and Feng Luo. A design for automotive can bus monitoring system. In *2008 IEEE Vehicle Power and Propulsion Conference*, pages 1–5, Sept 2008.

[13] M. Ruff. Evolution of local interconnect network (lin) solutions. In *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, volume 5, pages 3382–3389 Vol.5, Oct 2003.