Vehicular communications

Ollari Ischimji Dmitri

17 ottobre 2023

Indice

1 Introduction to Vehicular Communications **5**
- 1.1 Principles and challenges . . . 5
- 1.2 Standardization and open issues . . . 5
- 1.3 ITS Architecture . . . 5
- 1.4 ITS Applications . . . 5
- 1.5 Autonomous driving . . . 5

2 Telecomunicacion network basics **6**
- 2.1 The OSI and Internet models . . . 6
  - 2.1.1 Application Layer . . . 6
  - 2.1.2 Presentation Layer . . . 6
  - 2.1.3 Session Layer . . . 6
  - 2.1.4 Transport Layer . . . 7
  - 2.1.5 Network Layer . . . 7
- 2.2 Communication models . . . 7
- 2.3 Delimitation . . . 7
- 2.4 Sequence control . . . 7
- 2.5 Error management . . . 7
  - 2.5.1 Complement sum . . . 7
  - 2.5.2 Error correction . . . 8
- 2.6 Error recovery . . . 9

3 Intra-vehicle Communications **12**
- 3.1 Bus Systems . . . 12
  - 3.1.1 Perchè usare i bus? . . . 12
  - 3.1.2 Casi d'uso per intra-vehicle communications . . . 12
  - 3.1.3 Classificazione: On-board-communcation . . . 13
  - 3.1.4 Classificazione: Off-board-communication(OBD connector) . . . 13
  - 3.1.5 Classificazione per casi d'uso e importanza . . . 13
  - 3.1.6 Classificazione SAE(Society of Automotive Engineers) . . . 13
  - 3.1.7 Network Topologies . . . 13
- 3.2 Bit coding . . . 14
  - 3.2.1 Reducing ElectroMagnetic Interference(EMI) . . . 14
  - 3.2.2 Clock drift . . . 14
  - 3.2.3 Bit stuffing . . . 14
- 3.3 Classification according to bus access . . . 14
  - 3.3.1 Deterministic . . . 15
  - 3.3.2 Random . . . 15
  - 3.3.3 Typical structure of an ECU . . . 16
- 3.4 Protocols . . . 16

3.4.1 K-Like Bus 16
3.4.2 CAN - Controller Area Network 16

Elenco delle figure

2.1 Architettura OSI 6
2.2 Complement sum 8
2.3 Repetition code 8
2.4 Esempio comunicazione stop and wait senza SQN 9
2.5 Esempio comunicazione stop and wait con SQN 10
2.6 Sliding window base 11
2.7 Sliding window go back N 11
2.8 Sliding window selective repeat 11

3.1 Classification according to bus access 14
3.2 Struttura ECU 16
3.3 Formato dei dati CAN 18
3.4 Esempio di arbitrato bit per bit 18
3.5 Esempio di arbitrato bit per bit 19
3.6 Registri di filtraggio CAN 19
3.7 Data format 20
3.8 ISO-TP 20

Elenco delle tabelle

3.1 Classificazione per casi d'uso e importanza 13
3.2 Classificazione SAE 13

Capitolo 1

Introduction to Vehicular Communications

1.1 Principles and challenges

1.2 Standardization and open issues

1.3 ITS Architecture

1.4 ITS Applications

1.5 Autonomous driving

Capitolo 2

Telecomunicacion network basics

2.1 The OSI and Internet models

L'architettura **Open System Interconnetion(OSI)** punta a collegare sistemi eterogenei fra di loro, la sua specifica è la **ISO 7498** ed è un modello costituito di 7 *strati*.

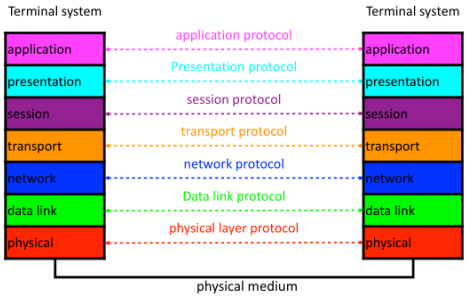

Figura 2.1: Architettura OSI

2.1.1 Application Layer

Livello del modello OSI dove le applicazioni accedono ai servizi di rete, permette ad esempio di trasferire un file, connettersi a database, uso di mail, ecc.

2.1.2 Presentation Layer

Livello del modello OSI adibito alla trasmissione di dati, traduce differenti formati di dati presenti nell'Application Layer in uno standard per gli strati inferiori.

Fornisce servizi per la trasmissione sicura ed efficiente dei dati come:

- data encryption
- data compression
- altre

2.1.3 Session Layer

Livello del modello OSI che permette due computer diversi di **creare**, **usare** e **finire** una sessione, utile per trasmissione di dati e accesso in remoto.

Introduce:

- **Controllo di dialogo**: per regolare la trasmissione e la durata delle stesse
- **Gestione dei token e sincronizzazione**

2.1.4 Transport Layer

Livello del modello OSI adibito alla gestione dei pacchetti da trasmettere:

- Divide pacchetti grandi in più piccoli
- Riordina i pacchetti nell'ordine correto all'arrivo

Gestisce inoltre il riconoscimento degli errori e il loro recupero:

- Ricezzione di pacchetti di confermo dell'arrivo (ACK)
- Reinvia pacchetti persi

2.1.5 Network Layer

Livello del modello OSI adibito alla gestione dell'instradamento dei dati attraverso le sottoreti:

2.2 Communication models

2.3 Delimitation

2.4 Sequence control

2.5 Error management

Il controllo dell'errore ha 3 possibili soluzioni:

- **Error detection**: rilevazione dell'errore
- **Error correction**: correzione dell'errore
- **Error recovery**: recupero dell'errore

2.5.1 Complement sum

Quando si riceve il pacchetto, si calcola il **checksum** dei dati ricevuti(come in Figura 2.2) e lo si confronta al checksum allegato al pacchetto ricevuto, nel caso di checksum differente si deve ritrasmettere il pacchetto.

Other codes

Polynomial codes conosciuti anche come **Cyclic Redundancy Check(CRC)**, usano moltiplicazioni tra polinomi per effettuare il checksum.

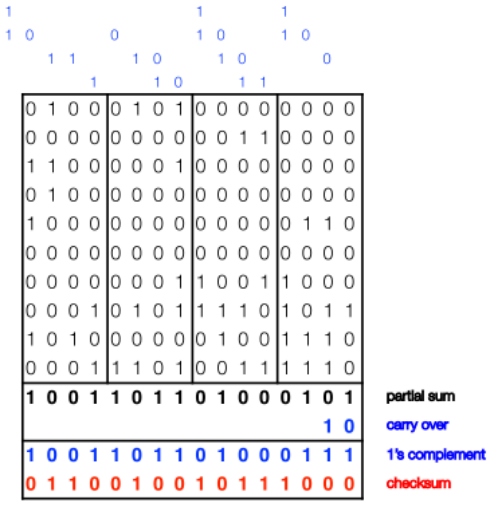

Figura 2.2: Complement sum

2.5.2 Error correction

Con la **block parity check** si possono recuperare errori ma solo se presente un errore di 1 bit.

Vengono quindi introdotte tecniche **Forward Error Correction(FED)**(ad esempio Algoritmo di Viterbi) che permettono di capire la presenza di un errore mediante algoritmi di ricostruzione.

Con FED si ricorre a ridondanza per eliminare errori(pochi in numero), non sono necessari messaggi di corretta ricezione, che torna molto utile nel caso di comunicazione unidirezionale.

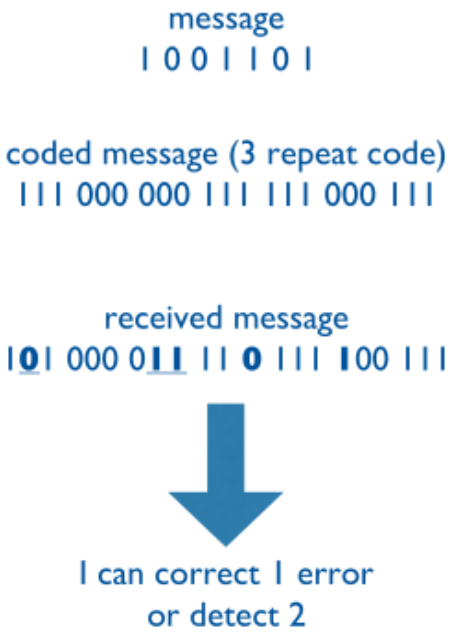

Figura 2.3: Repetition code

2.6 Error recovery

Quando si parla di comunicazione in reti di comunicazioni, si ricade nel richiedere automaticamente un pacchetto che non risulta corretto al ricevitori, esistono approcci automatici come **Automatic Repeat Request, ARQ**.

Esistono inoltre differenti meccanismi di ritrasmissione che come punto focale hanno:

- Error detection
- Acknowledgements
- timers
- IU identifiers

Le procedure ARQ cambiano in base alla dimensione delle finestra:

- **Stop and wait**: finestra di dimensione 1, si attende l'ack prima di inviare il pacchetto successivo
- **Sliding window, go-back-N**: finestra di dimensione N, si inviano N pacchetti prima di attendere l'ack(non ha un selettore per il resending e invia tutto il blocco)
- **Sliding window, selective repeat**: finestra di dimensione N, si inviano N pacchetti prima di attendere l'ack(ha un selettore per il resending e invia solo il pacchetto corrotto)

Stop and Wait

Il pacchetto **ACK(acknowledgement)** solitamente è molto corto per evitare correzzioni nel pacchetto che conferma la corretta ricezione.

È necessario stabilire un tempo limite entro il quale si da per scontato la *scomparsa* del pacchetto, solitamente si basa sul **Round Trip Time(RTT)** che dipende dalla congestione della rete e ne misura i ritardi per arrivare da punto A a punto B.

Altro fattore chiave è capire quali dati sono stati inviati e quali no, per evitare duplicazioni. Per questo problema di è scelto di indicizzare i pacchetti con una sequenza che prende il nome di **SeQuence Number(SQN)** per identificare univocamente quali pacchetti da ritrasmettere.

Si può parlare anche di ACK comulativi mediante l'uso di SQN consecutivi.

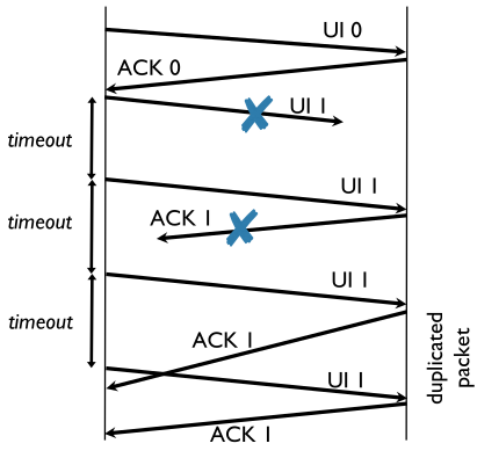

Figura 2.4: Esempio comunicazione stop and wait senza SQN

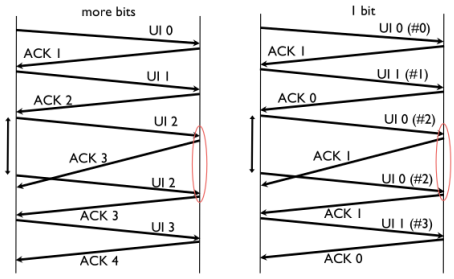

Figura 2.5: Esempio comunicazione stop and wait con SQN

Stop and wait performance

I tempi considerati sono:

- T_U: tempo di trasmissione di un pacchetto, misurato in s/IU
- T_P: tempo di propagazione di un pacchetto, misurato in s/IU
- T_A: tempo di trasmissione di un ACK, misurato in s/IU

Il tempo totale per inviare un'unità informativa(caso ideale):

$$T_{tot} = T_U + 2T_P + T_A \tag{2.1}$$

Il massimo grado di utilizzo di un canale di comunicazione nel caso di **assenza di errore**:

$$\rho_0 = \frac{T_U}{T_{tot}} \tag{2.2}$$

$$= \frac{T_U}{T_U + 2T_P + T_A} \tag{2.3}$$

$$= \begin{cases} \frac{1}{2+2\frac{T_P}{T_U}} & \text{se} \quad T_U = T_A \\ \frac{1}{2\frac{T_P}{T_U}+1} & \text{se} \quad T_U >> T_A \\ 0 \quad \text{se} & T_P >> T_U \end{cases} \tag{2.4}$$

Nel caso di **presenza di errore**, non viene ricevuto l'ACK dal trasmettitore, devo fare alcune assunzioni:

- Indipendenza statisticamente dei pacchetti informativi
- perdita di pacchetti ACK

Indico con p la probabilità di perdita del pacchetto.
Il tempo per l'arrivo di un pacchetto con presenza di errori diventa:

$$\bar{T_1} = (N_t - 1)T_0 + T_1 \tag{2.5}$$

$$= \frac{p}{1-p}T_0 + T_1 \tag{2.6}$$

$$\simeq \frac{T_1}{1-p} \tag{2.7}$$

Dove N_t è desscritta come:

$$N_t = \sum_{k=1}^{\infty} kp^{k-1}(1-p) = \frac{1}{1-p} \tag{2.8}$$

Quindi l'utilizzazione del canele diventa:

$$\rho = (1-p)\rho_0 \tag{2.9}$$

Sliding window

La ricezione con **sliding window** può essere non sequenziale ma non può superare il **timeout** che invalida il pacchetto.

Esistono due strategie per il reinvio dei dati persi con la tecnica dello sliding window:

- **go-back-N**: torna indietro di un numero N di unità informative
- **selective repeat**: reinvia solo i pacchetti effetivamnete persi

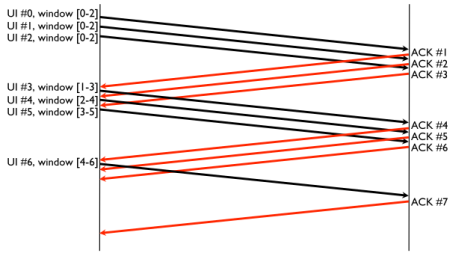

Figura 2.6: Sliding window base

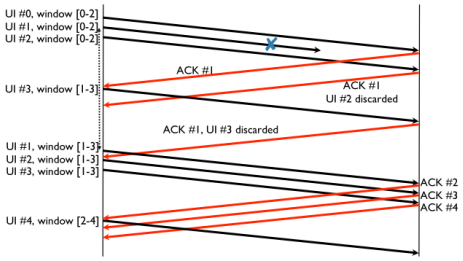

Figura 2.7: Sliding window go back N

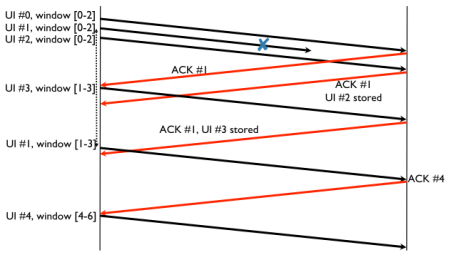

Figura 2.8: Sliding window selective repeat

Capitolo 3

Intra-vehicle Communications

3.1 Bus Systems

3.1.1 Perchè usare i bus?

Un bus che collega tutti i componenti al posto di avere una topologia a grafo completo ha i seguenti vantaggi:

- **Riduzione dei costi**: meno cavi e meno connettori
- **Riduzione del peso**: meno cavi
- **Riduzione del volume**: meno cavi
- **Alta modularità**: modifica veicoli
- **Alta modularità**: cooperazione con OEM
- **Modularità**: riuso di moduli
- **Standardizzazione**: standardizzazione dei componenti e dei protocolli (meno errori scemi)

3.1.2 Casi d'uso per intra-vehicle communications

- Driveline: Engine and transmission control
- Active Safety: Electronic Stability Programme (ESP)
- Passive Safety: Air bag, belt tensioners
- Comfort: Interior lighting, A/C automation
- Multimedia and Telematics: Navigation system, CD changer

La geolacaliuazione è fornita da protocolli di navigazione satellitare. I più famosi sono:

- GPS: USA
- Galileo: EU
- Glonass: Russia
- Beidou: China

Altre soluzioni sono RTK(Real Time Kinematic)!!

3.1.3 Classificazione: On-board-communcation

- Complex control and monitoring tasks: trasmissione dei dati tra ECUs(Engine Control Unit) e MMI(Man Machine Interface simile a HMI che sta per human machine interface)
- Simplification of wiring: rimpiazzare i fili di rame con bus per ridurre la complessità dei cablaggi
- Multimedia bus systems: Trasmette un sacco di dati per i sistemi di intrattenimento

3.1.4 Classificazione: Off-board-communication(OBD connector)

- Diagnostics: diagnosi del veicolo
- Flashing: aggiornamento del software
- Debugging: debug del software

3.1.5 Classificazione per casi d'uso e importanza

Application	Message Length	Message rate	Data rate	Latency	Robustness	Cost
Control and monitoring		2	2	3	3	2
Simplified wiring				1	2	1
Multimedia	1	2	3	1	1	3
Diagnosis						1
Flashing	2		2		1	
Debugging		1	1	2		

Tabella 3.1: Classificazione per casi d'uso e importanza

3.1.6 Classificazione SAE(Society of Automotive Engineers)

3.1.7 Network Topologies

- **Repeter**: amplificazione del segnale a livello fisico
- **Bridge**: medium/timing adaptation, unfiltered forwarding a livello data link
- **Router**: medium/timing adaptation, filtered forwarding a livello network
- **Gateway**: medium/timing adaptation, filtered forwarding, protocol translation a livello application

Class	Data rate	vantaggio	Dispositivi
A	$10kBit/s$	Economico	Diagnosi
B	$64kBit/s$	Correzione errori	Networking ECUs
C	$1MBit/s$	Comunicazione in tempo reale	Drive train
D	$10MBit/s$	Bassa latenza	X-By-Wire

Tabella 3.2: Classificazione SAE

3.2 Bit coding

Esistono due tipologie di encoding dell'informazione:

- Non Return to Zero (NRZ)
- Manchester

Nella tipologia **NRZ** il valore logico 0 è caratterizzato da un segnale basso, mentre il segnale logico 1 è identificato da un segnale alto.

Nell'encoding di tipo **Manchester** si pone attenzione al cambio di livello per attribuire il valore logico, il valore logico 0 è identificato dal passaggio da basso livello ad alto livello e il valore logico 1 è identificato dal passaggio di stato da alto livello a basso livello.

3.2.1 Reducing ElectroMagnetic Interference(EMI)

- Aggiungere schermatura ai fili
- usare fili twistati per coppie di fili(annullano effetti di elettromagnetismo a vicenda)
- Ridurre la ripidità del segnale
- usare usare NRZ che ha pochi cambi di stato

3.2.2 Clock drift

Il clock drift è causato dalla costruzione fisica del quarzo usato per il clock, che diffferensce leggermente da altri clock, questo fenomeno causa **desincronizzazione**.

3.2.3 Bit stuffing

Il problema associato all'utilizzo della codifica NRZ è che inviando una serie di bit costanti, in presenza di piccoli ritardi, i dati vengono ricevuti in maniera sbagliata.

Una soluzione proposta è quella del **Bit Stuffing** ed inserisce un bit extra dopo n bit consecutivi.

- Se ci sono 3 uni di fila, agiunge uno zero
- se ci sono 3 zeri di fila, aggiunge un uno

3.3 Classification according to bus access

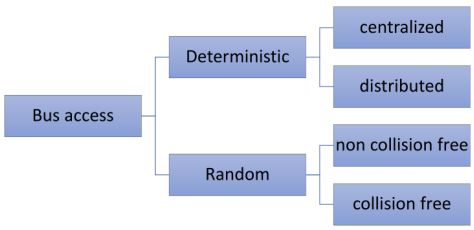

Figura 3.1: Classification according to bus access

3.3.1 Deterministic

Centralized

Accesso al bas di tipo **master-slave**(similare ad un sistema pooling)

Decentralized

Protocolli basati su **token**, tutti collegati a cerchio e si invia il messaggio con un token (che è tipo il bastone della parola) al ricevitore, il ricevitore manda il suo messaggio dopo nella catena insieme altoken e cosi via.

Solo il nodo con il token poò inviare il suo pacchetto di informazioni.

L'altro approccio è il **TDMA**(Time-division multiple access), si identificano i client attaccati al mezzo di comunicazione e si divide la comunicazioni a slot temporali e si può capire chi invia guardando il riferimento al clock

Grande problema di TDMA è la sincronizzazione.

3.3.2 Random

Non Collision Free

CSMA/CA(Carrier Sense Multiple Access)/(Collision Avoidance) misura l'energia del bus e invia quando l'energia è sotto un certo *livello di riferimento*.

CSMA senza CA, se sente il canale occupato, seleziona un tempo random che chiama backoff e aspetta, dopo di che riprova ad ascolare il bus.

CSMA con CA ogni nodo conta il tempo che il noda che sta comunicando finisca, i nodi posso comunicare in qualsiasi momento e quindi ogni conteggio sarà diverso, dopo che il nodo ha finito di comunicare, ogni nodo aspetta il tempo che ha contato il precedenza.

se mentre i nodi stanno aspettando il tempo contato per trasmettere uno dei nodi finisce, si salva il tempo avanzato agli altri nodi e si utilizza quello per il prossimo check di chi tocca.

Questo sistema non è *giusto* e può portare pacchetti a rimanere nella coda per tempi lunghi.

CSMA/CD(Carrier Sense Multiple Access)/(Collision Detection) se più nodi comunicano l'energia sul bus è maggiore del solito e i nodi si rendono conto del problema.

I nodi si fermano(per risparmiare risorse) e mandano un segnale **jamming** che è una sequenza di bit di alto livello per comunicare agli altri nodi il problema e di non comuniicare per un po.

Si applicano ai nodi dei tempi di backoff mediante strategie di backoff e si riprova.

CSMA/CR(Carrier Sense Multiple Access)/(Collision Resolution):

1. **Arbitration phase**: si compete per avere il canale
2. **data**: il nodo che ha vinto il canale comunica

E si itera questo processo ogni volta che un nodo vuole comunicare.

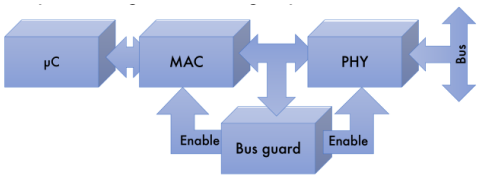

Figura 3.2: Struttura ECU

3.3.3 Typical structure of an ECU

3.4 Protocols

3.4.1 K-Like Bus

Si concentra su **Layer fisico** e **Layer data link** ed è un bus bidirezionale con comunicazione su di un filo.

Principalmente usato per connettere:

- ECU to Tester
- ECU to ECU

Lo **zero logico** ha un valore di energia inferiore al 20% del massimo del bus, mentre il **uno logico** è rappresentato quando il segnale supera l'80% del valore massimo.

Questo protocollo è compatibile con **UART**(Universal Asynchronous Receiver Transmitter).

3.4.2 CAN - Controller Area Network

Protocollo realizato nel 1986 da Bosh, la topologia di rete è quella del **Bus**.

Il protocollo riesce a mantenere contemporaneamente fino a 110 nodi e i segnali possibili sono 2:

- **LOW**: segnale dominante
- **HIGH**: segnale recessivo

CAN ha una lunghezza massima del BUS di massimo $500m$ ad una velocità di $125kBit/s$.
Successivamente nello standard **ISO 11898** si è descritto due velocità:

- LOW speed CAN: fino a $125kBit/s$
- High speed CAN: fino a $1MBit/s$

Il protocollo CAN interessa solo i layer 1 e 2 dell'ISO/OSI stack e le sue caratteristiche principali sono:

- random access
- collision free
- message oriented
- no indirizzi(solo broadcast o multicast)

Layer fisico

Il protocollo CAN può usare due configurazioni:

- **HIGH SPEED CAN**:
  - fino a $500kBit/s$
  - 2 cavi arrotolati per ridurre interferenze da campi elettromagnetici
  - cavi di collegamento ai nodi massimo di $30m$ (branch)
  - Segnale tra 0 e $2V$
  - resistore terminale di 120Ω
  - l'errore deve essere scoperto entro il tempo di 1 BIT, quindi la lunghezza è vincolata a Equazione 3.2
- **LOW SPEED CAN**:
  - fino a $125kBit/s$
  - 2 cavi per ridurre le interferenze elettromagnetiche
  - nessuna restrizione sui cavi di branch che collegano i nodi al bus
  - segnale tra 0 e $5V$
- **SINGLE WIRE CAN**:
  - velocità fino a $83kBit/s$
  - 1 solo cavo e massa come riferimento
  - segnale tra 0 e $5V$

Avendo il la velocità alla quale si vuole trasmettere i dati, si possono ricavare i metri massimi del cavo, ad esempio avendo una velocità di $R = 500kBit/s$ i metri massimi del filo sono calcolabili come:

$$\frac{1}{R} \geq 2l \cdot 10^{-8} \tag{3.1}$$

$$l \leq \frac{1}{2R \cdot 10^{-8}} \tag{3.2}$$

CAN in Vehicular Networks

Comunicazione senza indirizzo

- I messaggi portano un identificativo del messaggio di 11 bit (CAN 2.0A) o 29 bit (CAN 2.0B).
- Le stazioni non hanno un indirizzo, e i frame non ne contengono uno.
- Le stazioni utilizzano l'identificativo del messaggio per decidere se un messaggio è destinato a loro.
- L'accesso al mezzo avviene utilizzando CSMA/CR con arbitrato bit per bit.
- Il livello di collegamento utilizza 4 formati di frame: Dati, Richiesta (Remote), Errore, Sovraccarico (controllo di flusso).

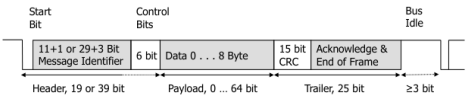

Figura 3.3: Formato dei dati CAN

- Il formato dei dati nel protocollo CAN segue lo schema Figura 3.3

CSMA/CR con arbitrato bit per bit

- Evita collisioni tramite l'accesso al bus controllato dalla priorità.
- Ogni messaggio contiene un identificativo corrispondente alla sua priorità.
- L'identificativo codifica "0" dominante e "1" recessivo: la trasmissione simultanea di "0" e "1" produce un "0".
- Riempimento dei bit: dopo 5 bit identici, viene inserito un bit di riempimento invertito (ignorato dal ricevitore).
- Quando nessuna stazione sta trasmettendo, il bus legge "1" (stato recessivo).
- La sincronizzazione avviene a livello di bit, rilevando il bit di inizio della stazione trasmittente.
- Attendere la fine della trasmissione corrente.
- Attendere 6 bit recessivi consecutivi.
- Inviare l'identificativo (mentre si ascolta il bus).
- Osservare una discrepanza tra il livello di segnale trasmesso e rilevato.
- Questo indica che si è verificata una collisione con un messaggio di priorità superiore.
- Ritirarsi dall'accesso al bus e riprovare in seguito.
- Realizzazione di uno schema di priorità non pre-emptive.
- Garanzie in tempo reale per i messaggi con priorità più alta, ad esempio, messaggi con il più lungo prefisso "0".

Esempio di arbitrato bit per bit(Figura 3.4):

Figura 3.4: Esempio di arbitrato bit per bit

In questo caso il client 2 si rende conto che c'è un problema e fa **back off** lasciando libero il bus.

In Figura 3.5 il client 1 si rende conto del bus impegnato da una comunicazione con priorità maggiore e fa **backoff** che porta a mantenere sul bus la comunicazione con priorità maggiore(client 3) e poi segue la trasmissione del dato che il client 3 deve trasmettere.

Figura 3.5: Esempio di arbitrato bit per bit

TTCAN(Time triggered CAN)

un problema del CAN è quello della temporizzazione e del mantenimento dei clock fra i client.

Per risolvere questo problema è stato creato il TTCAN, che ha un nodo dedicato che prende il nome di **time master** è che periodicamente manda un segnale **basic cycles** a tutti i nodi.

Con **basic cycle** si intende un numero definito di slot occupabili che vengono popolati da una fase di organizzazione per priorità all'inizio della stessa.

CAN non si può usare per real time communications.

Message Filtering

I messaggi vengono accettati in base al loro identificativo mediante l'uso di due registri:

Bit	10	9	8	7	6	5	4	3	2	1	0
Acceptance Code Reg.	0	1	1	0	1	1	1	0	0	0	0
Acceptance Mask Reg.	1	1	1	1	1	1	1	0	0	0	0
Resulting Filter Pattern	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**X**	**X**	**X**	**X**

Figura 3.6: Registri di filtraggio CAN

Data Format

Il formato di dati segue:

- NRZ
- bit stuffing
- il frame inizia con un bit significativo(0)
- il message idenfier è 11 Bit(CAN 2.0A) oppure adesso 29Bit (CAN 2.0B)
- Bit di controllo idenficano il tipo di messaggio e la lunghezza
- payload: massimo 8 Byte, trasmesso a $500kBit/s$
- i dati interessanti del frame sono pochi e quindi il data rate effettivo è $30kBit/s$

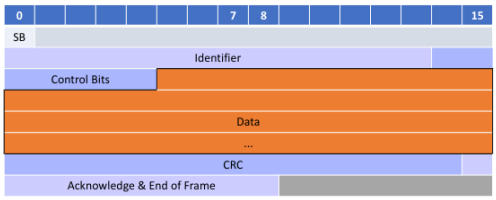

Figura 3.7: Data format

Error detection LOW LEVEL

- Il mittente verifica la presenza di livelli di segnale inattesi sul bus.
- Tutti i nodi monitorano i messaggi sul bus.
- Tutti i nodi verificano la conformità del protocollo dei messaggi.
- Tutti i nodi controllano il riempimento dei bit.
- Il ricevitore verifica il CRC.
- Se uno qualsiasi dei nodi rileva un errore, trasmette un segnale di errore. (6 bit dominanti senza bit stuffing)
- Tutti i nodi rilevano il segnale di errore e scartano il messaggio.

Error detection HIGH LEVEL

- Il mittente verifica la ricezione di un riconoscimento. (Il ricevitore trasmette un bit dominante "0" durante il campo di ACK del messaggio ricevuto)
- Ripetizione automatica delle trasmissioni fallite.
- Probabilità residua di fallimento circa 10^{-11}.

Transoport Layer

Utile alla gestione del flusso, e gestioen dei frammenti di un singolo messaggio divisi in più data frames.

I due protocolli sono:

- ISO-TP
- TP 2.0

ISO-TP

Figura 3.8: ISO-TP

L'**HEADER** introduce un byte opzionale per l'adressing e da 1 a 3 byte **PCI**(Protocol Control Information) che identificano il tipo di messaggio e byte specifici al messaggio.

Single-frame è identificato da 1 byte PCI, l'high-niggle(4bit) è 0, e il low-nibble identifica i bytes in payload, non è possibile fare controllo di flusso.

First-frame: identificato da 2 bytes PCI, high-nibble è 1 e low-nibble con aggiunta di 1 byte definisce la dimensione del payload.

Dopo il primo frame il sender aspetta il **flow control frame**.

Consecutive-frame: 1 byte PCI, high nibble è 2, low nibble è un numero d isequenza **SN** che parte da 1

Flow control-frame: 3 bytes pci, high nibble è 3 e low nibble specifica lo stato del flusso **FS**

- FS 1: cear send
- FS 2: wait

Il byte 2 specifica la block size BS e il byte 3 indica ST(separation time).

TP 2.0

Protocollo simile a TCP:

- connection oriented
- comunicazione basata su canali
- specifica fasi di: setup, configurazione, trasmissione e chisura
- ogni ECU ha un'indirizzo

Broadcast

- ripetizione per 5 volte per evitare perdita
- Byte 0: adress of desctination ECU
- Byte 1: operation code(broadcast response or request)
- Byte 2, 3, 4: Service ID(SID)
- Byte 5, 6: Response (si alterna 0x5555 e 0xAAAA per dire che non si vuole aspettare risposta)

Channel setup

- Byte 0: adress destination ECU
- Byte 1: Operation code(Channel request, positive response e negative response)
- Byte 2, 3: RX ID(presente se validity nibble di Byte 3 è 0 altrimenti non settato)
- Byte 4, 5: TX ID(presente se validity nibble di Byte 5 è 0 altrimenti non settato)
- Byte 6: application type