Vehicular communications

Ollari Ischimji Dmitri

10 ottobre 2023

Indice

Elenco delle figure

Elenco delle tabelle

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*.

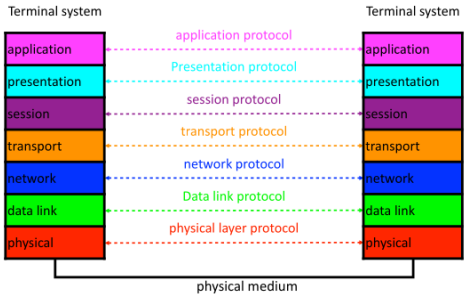

Figura 2.1: Architettura OSI

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.

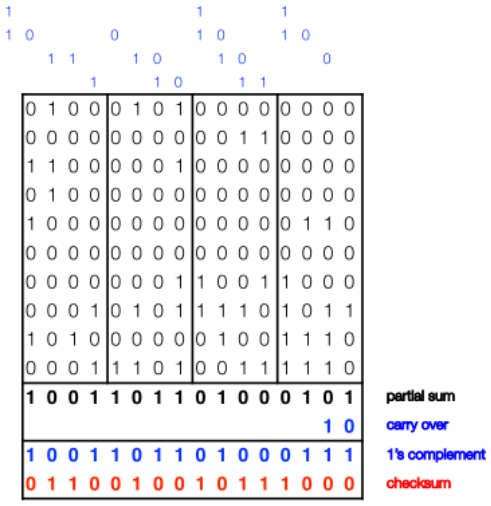

Figura 2.2: Complement sum

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.

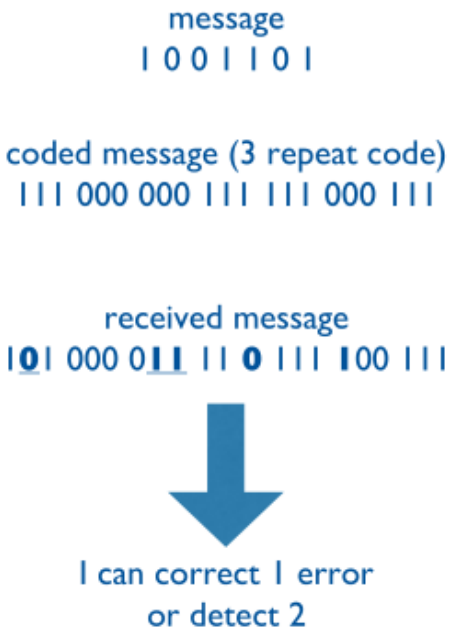

Figura 2.3: Repetition code

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.

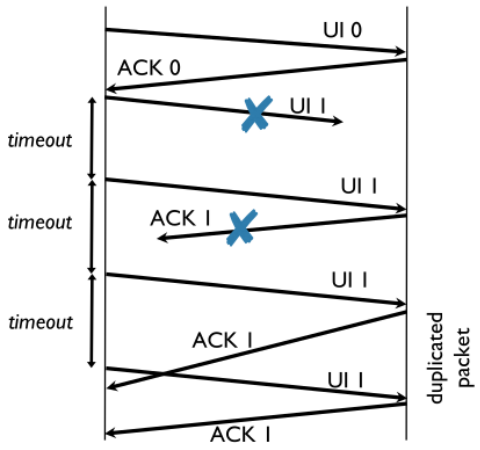

Figura 2.4: Esempio comunicazione stop and wait senza SQN

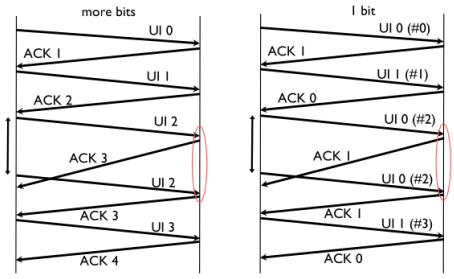

Figura 2.5: Esempio comunicazione stop and wait con SQN

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

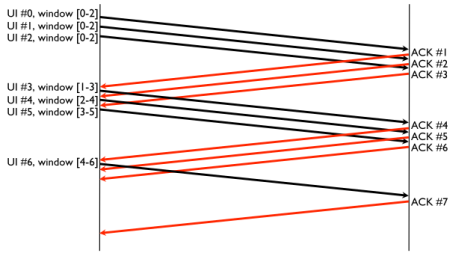

Figura 2.6: Sliding window base

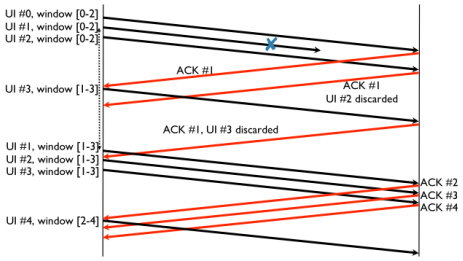

Figura 2.7: Sliding window go back N

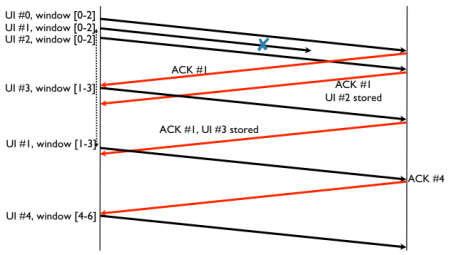

Figura 2.8: Sliding window selective repeat

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)