



# UNIVERSITÀ DI PARMA

---

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE  
Corso di Laurea Triennale in Informatica

## RETI REAL-TIME: PROTOCOLLI PER L'AUTOMAZIONE INDUSTRIALE

Real-Time Networks:  
Protocols for Industrial Automation

Candidato:  
**Nicolò Toscani**

Relatore:  
**Prof. Roberto Alfieri**



# Indice

<b>1 Reti di comunicazione</b>	<b>2</b>
1.1 Il modello OSI	3
1.2 Ethernet	4
1.2.1 Livello fisico	5
1.2.2 Il sottolivello MAC	5
1.2.2.1 Indirizzo	6
1.2.2.2 Trasmissione e ricezione	6
1.2.2.3 Frame	6
1.2.2.4 Gestione delle collisioni	7
1.3 Topologia di rete	8
1.4 IP: Internet Protocol	8
1.4.1 Indirizzo IP	9
1.4.2 Pacchetto IPv4	9
1.5 TCP: Transmission Control Protocol	11
1.5.1 Il segmento TCP	11
1.5.2 Connessione	13
1.5.3 Affidabilità della comunicazione	13
1.6 UDP: User Datagram Protocol	13
1.6.1 Datagramma UDP	14
<b>2 Comunicazione industriale</b>	<b>16</b>
2.1 Il flusso dei dati	17
2.2 Algoritmi di scheduling per reti Real-Time	20
2.3 I bus di campo	22
2.4 Determinismo	23
2.4.1 Fast Ethernet	24
2.4.1.1 Trasmissione Full-Duplex	24
2.4.1.2 Priorità dei messaggi	24
2.4.2 Switch	25
2.4.2.1 Switched Ethernet	26
2.4.3 Sincronizzazione: IEEE 1588	26
2.5 Industrial Ethernet	28
2.5.1 Real-Time Ethernet	29
2.5.1.1 Normative	31

2.5.1.2	Indicatori di prestazione . . . . .	32
2.5.2	Aspetti implementativi . . . . .	33
2.5.2.1	Categoria A: Soluzione basata su TCP/IP . . . . .	33
2.5.2.2	Categoria B: Ethernet MAC . . . . .	33
2.5.2.3	Categoria C: Ethernet modificata . . . . .	33
<b>3</b>	<b>EtherNet/IP . . . . .</b>	<b>35</b>
3.1	CIP: Common Industrial Protocol . . . . .	36
3.1.1	Orientamento ad oggetti . . . . .	36
3.1.1.1	Messaggistica . . . . .	39
3.2	Communication Objects . . . . .	40
3.3	Libreria degli oggetti . . . . .	42
3.3.0.1	Identity Object (Class ID: 0x01) . . . . .	43
3.3.0.2	Parameter Object (Class ID: 0x0F) . . . . .	44
3.3.0.3	Assembly Object (Class ID: 0x04) . . . . .	45
3.4	Device Profiles . . . . .	46
3.4.1	Descrizione del dispositivo . . . . .	46
3.4.2	Routing . . . . .	48
3.4.3	Gestione dei dati . . . . .	48
3.5	EtherNet/IP: protocollo CIP su Industrial Ethernet . . . . .	49
3.5.1	Livello fisico . . . . .	50
3.5.2	Sruttura dei frame . . . . .	50
3.5.2.1	Incapsulazione dati . . . . .	52
3.5.3	Connessione rapida . . . . .	54
3.6	Classi di dispositivi . . . . .	54
3.7	Requisiti per il supporto TCP/IP . . . . .	55
3.8	Architettura Ethernet . . . . .	56
3.8.1	Topologia di rete . . . . .	57
3.9	Oggetti EtherNet/IP aggiuntivi . . . . .	59
3.10	Indirizzi di rete . . . . .	60
3.11	EIPPI: EtherNet/IP Performance Indicators . . . . .	60
<b>4</b>	<b>EtherCAT . . . . .</b>	<b>64</b>
4.1	Livelli ISO/OSI . . . . .	64
4.1.1	Livello fisico . . . . .	65
4.1.2	Livello di collegamento . . . . .	66
4.1.3	Livello Applicazione . . . . .	67
4.1.3.1	Profili dispositivi . . . . .	68
4.2	Il protocollo . . . . .	69
4.2.1	Dati di processo "on the fly" . . . . .	70
4.2.2	Comunicazione aciclica . . . . .	71
4.2.3	Trasferimento dei dati . . . . .	72
4.2.3.1	Struttura del frame . . . . .	73
4.2.3.2	Indirizzamento . . . . .	75

4.2.3.3	Comandi . . . . .	76
4.2.3.4	Working Counter . . . . .	77
4.2.3.5	SyncManager . . . . .	78
4.2.3.6	Clock distribuito . . . . .	79
4.3	Topologia di rete . . . . .	79
4.4	Prestazioni . . . . .	81
4.4.1	Metodi di analisi per le prestazioni . . . . .	81
4.5	Sicurezza funzionale . . . . .	83
4.6	Implementazione master . . . . .	85
4.7	Implementazione slave . . . . .	86
4.8	Conformità e certificazione . . . . .	87
<b>5</b>	<b>Analisi dei protocolli . . . . .</b>	<b>88</b>
5.1	Diagnostica con EtherCAT . . . . .	88
5.1.1	EtherCAT con Wireshark . . . . .	89
5.1.2	Struttura dello slave . . . . .	90
5.1.3	Analisi frame EtherCAT . . . . .	91
5.2	Diagnostica con EtherNet/IP . . . . .	98
5.2.1	EtherNet/IP con Wireshark . . . . .	98
5.2.2	Profilo AC Drives 0x02 . . . . .	98
5.2.2.1	Assembly Object ID 21: Extended Speed Control Output . . . . .	99
5.2.2.2	Assembly Object ID 71: Extended Speed Control Input . . . . .	99
5.2.3	Analisi frame EtherNet/IP . . . . .	100
	<b>Conclusioni . . . . .</b>	<b>103</b>
5.3	Sviluppi futuri . . . . .	103
	<b>Bibliografia . . . . .</b>	<b>105</b>

# Elenco delle figure

1.1	Livelli del modello OSI . . . . .	3
1.2	Suddivisione del livello di collegamento . . . . .	5
1.3	Frame Ethernet . . . . .	7
1.4	Pacchetto IPv4 . . . . .	9
1.5	Segmento TCP . . . . .	12
1.6	Intestazione UDP . . . . .	14
2.1	Il flusso dei dati in un sistema produttivo industriale . . . . .	17
2.2	Esempio di sincronizzazione dei clock tra due dispositivi . . . . .	28
2.3	Requisiti dei jobs . . . . .	30
2.4	Tre approcci per l'implementazione di protocolli industriali su Ethernet . . . . .	34
3.1	Common Industrial Protocol . . . . .	35
3.2	Classe di oggetti . . . . .	37
3.3	Esempio di nodo nella rete CIP . . . . .	38
3.4	Esempio di indirizzamento all'interno di un nodo . . . . .	39
3.5	Connessione CIP bidirezionale . . . . .	39
3.6	Connessione I/O implicita . . . . .	41
3.7	Connessione con messaggistica esplicita . . . . .	41
3.8	Oggetti generici . . . . .	42
3.9	Oggetti rete . . . . .	42
3.10	Modello ad oggetti tipico di un dispositivo EtherNet/IP . . . . .	43
3.11	Mappatura oggetti Assembly tipica di un dispositivo . . . . .	45
3.12	Profili dispositivi definiti dallo standard . . . . .	46
3.13	Relazione tra frame CIP e Ethernet . . . . .	51
3.14	Pacchetto CIP incapsulato nel frame Ethernet . . . . .	52
3.15	Esempio di Common Packet Format . . . . .	53
3.16	Incapsulazione messaggio I/O . . . . .	54
3.17	Relazione tra protocollo CIP e i tipici protocolli Ethernet . . . . .	56
3.18	Topologia di rete lineare . . . . .	57
3.19	Topologia di rete ad anello con DLR . . . . .	58
3.20	Parametri e prestazioni della configurazione DLR . . . . .	59
3.21	Configurazione della rete in analisi . . . . .	63
4.1	Modello ISO/OSI modificato dal protocollo EtherCAT . . . . .	65

4.2	Macchina a stati EtherCAT . . . . .	67
4.3	Profili e protocolli supportati da EtherCAT . . . . .	69
4.4	Telegramma in transito sulla rete . . . . .	70
4.5	Ritardi introdotti dagli stack . . . . .	71
4.6	Dati di processo "on-the-fly" . . . . .	72
4.7	Telegramma EtherCAT . . . . .	73
4.8	Intestazione EtherCAT . . . . .	74
4.9	Datagramma EtherCAT . . . . .	74
4.10	Indirizzamento EtherCAT . . . . .	76
4.11	Gestione del Working Counter . . . . .	78
4.12	Topologia flessibile: linea, albero o stella . . . . .	80
4.13	Safety Integrity Level . . . . .	84
4.14	Fail Safe over EtherCAT . . . . .	85
4.15	Implementazione di un dispositivo master . . . . .	86
5.1	File di descrizione del dispositivo slave in formato XML . . . . .	89
5.2	Struttura del dispositivo slave in esame . . . . .	90
5.3	Configurazione rete in esame . . . . .	91
5.4	Comandi di movimento inviati allo slave . . . . .	92
5.5	Cattura frame inviato dal master verso la rete . . . . .	94
5.6	Configurazione PDO durante l'avvio della comunicazione . . . . .	94
5.7	Campo Data nel frame di scrittura . . . . .	96
5.8	Campo Data nel frame di lettura . . . . .	97
5.9	PDO da master verso slave . . . . .	97
5.10	PDO da slave verso master . . . . .	98
5.11	Cattura frame destinato al PLC . . . . .	101

## Sommario

Il settore dell'automazione industriale sta avendo una profonda trasformazione determinata dalla possibilità per i robot, veicoli e sistemi di controllo industriale di collegarsi in rete e scambiarsi informazioni.

In uno scenario in cui tutti i dispositivi di un impianto sono sempre collegati con persone e processi, la macchina diventa l'attore principale del sistema produttivo che si occupa di comunicare agli operatori il proprio stato di funzionamento, riducendo i fermi macchina e i tempi di inattività. Per questo motivo, le applicazioni industriali richiedono una soluzione di rete eterogenea, che possa integrare i dati di processo e di diagnostica con la rete direzionale dello stabilimento.

Questo processo di integrazione è stato reso possibile adottando la tecnologia Ethernet, già consolidata nelle infrastrutture IT, che applicata in un ambiente critico come quello dell'industria, prende il nome di Industrial Ethernet.

Questa tecnologia, utilizzata a livello di campo, richiede tempi di risposta molto rapidi e altamente deterministici, diversamente dalla tradizionale tecnologia Ethernet da ufficio. Industrial Ethernet, sfruttando la suite degli standard Ethernet e IP, offre un sistema reattivo e flessibile che permette una comunicazione diretta tra i dispositivi in campo e i più alti livelli aziendali.

Utilizzando questa soluzione, le organizzazioni possono costruire un'infrastruttura di produzione che offre una elasticità e una sicurezza maggiore rispetto ai tradizionali bus di campo, oltre che ad un miglioramento della larghezza di banda e ad una maggiore connettività verso i nuovi dispositivi della rete.

Questa tesi si pone l'obiettivo di analizzare e descrivere le reti di comunicazione utilizzate in ambito industriale che, sfruttando la tecnologia Ethernet, permettono lo scambio delle informazioni di processo in tempo reale.

Dopo una descrizione dettagliata dei requisiti richiesti da questa tipologia di reti, verranno presi in esame due protocolli differenti, analizzandone le caratteristiche in termini di funzionamento e implementazione.



# Capitolo 1

## Reti di comunicazione

Le aziende manifatturiere stanno espandendo sempre di più le loro operazioni a livello globale per cercare ed affrontare nuove opportunità. Continuamente, cercano di migliorare l'efficienza della loro produzione, riducendo i costi per le strutture e i processi esistenti.

Per raggiungere una globalizzazione ed un'eccellenza produttiva, è necessario per prima cosa, migliorare la connettività tra i vari impianti per poter condividere il flusso delle informazioni di produzione con i vari livelli aziendali.

Le caratteristiche di comunicazione richieste a livello di campo però sono differenti da quelle che vengono richieste ad un livello gestionale, dove non vengono richiesti vincoli temporali. Al contrario, il livello di campo deve disporre di protocolli di comunicazione che permettano di determinare con precisione il momento in cui un valore viene trasmesso da un nodo della rete ad un altro.

L'utilizzo di una rete eterogenea aiuta a garantire qualità e prestazioni coerenti in tutte le operazioni globali e a ridurre i costi di progettazione, implementazione e supporto dei sistemi di produzione.

Poiché le aziende cercano di raggiungere questi obiettivi cercando di integrarsi con il resto della rete già presente, molte portano la tecnologia Ethernet a livello di stabilimento. E' proprio su questa tecnologia che sono stati sviluppati molti dei protocolli utilizzati per far dialogare i vari componenti di un impianto industriale automatizzato secondo le specifiche richieste dai vincoli temporali.

Le reti industriali che utilizzano la tecnologia Ethernet possono offrire una varietà di vantaggi rispetto a reti tradizionali basate sui bus di campo. Poiché la tecnologia è basata su standard tradizionali, Industrial Ethernet consente alle organizzazioni di abbandonare le vecchie reti di fabbrica chiuse e costose.

Fornendo una piattaforma scalabile che può ospitare più applicazioni, i sistemi di automazione basati su questa tecnologia possono aumentare la flessibilità e accelerare l'implementazione di future applicazioni. Allo stesso tempo, Ethernet fornisce alla rete sicurezza, prestazioni e disponibilità richieste per supportare applicazioni critiche.

## 1.1 Il modello OSI

Il modello OSI (Open Systems Interconnection) è uno standard per la progettazione delle reti, sviluppato nel 1984 dall'International Organization for Standardization (da qui il termine ISO/OSI).

Il modello definisce l'architettura logica di una rete, definendo una struttura a strati composta da una pila di protocolli di comunicazione su 7 livelli distinti, che insieme eseguono tutte le funzionalità della rete seguendo un modello logico-gerarchico.

Ogni livello sfrutta le funzionalità messe a disposizione dal livello sottostante e fornisce nuove funzionalità al livello superiore.

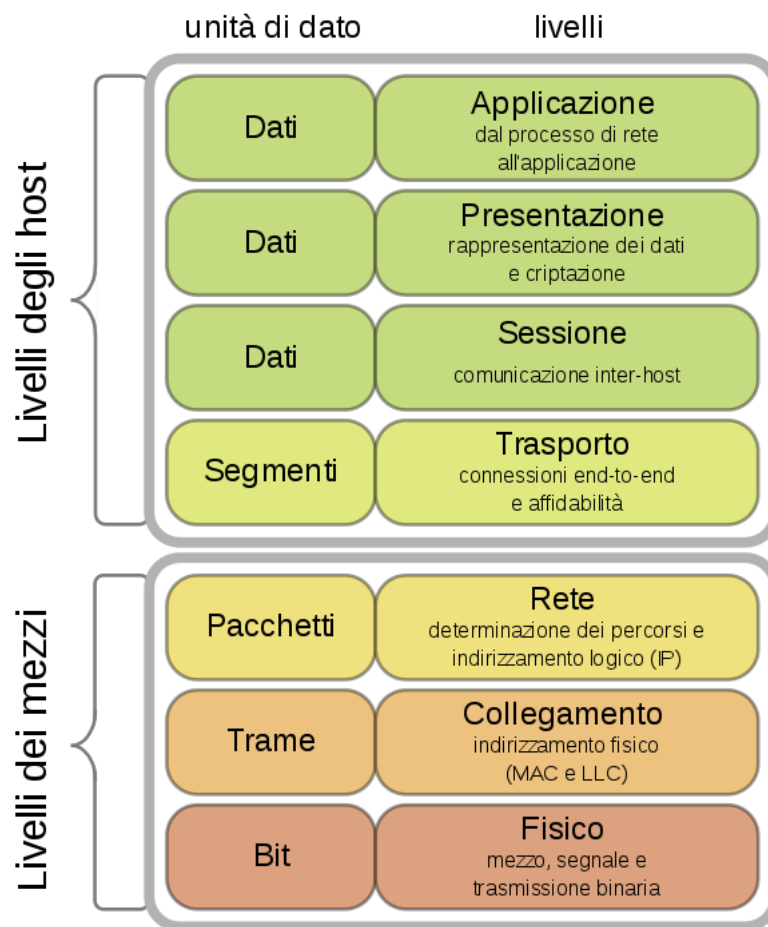


Immagine 1.1: Livelli del modello OSI

Di seguito vengono elencati tutti gli obiettivi principali di ogni livello della pila.

- **Livello fisico:** trasmette un flusso di dati non strutturati attraverso un collegamento fisico, definendo gli aspetti elettrici e di temporizzazione con cui i bit, sotto forma di segnali elettromagnetici, vengono spediti sui canali di comunicazione.
- **Livello di collegamento:** definisce come i dispositivi comunicano sulla rete in modo affidabile attraverso il livello fisico. Invia frame di dati con la necessaria sincronizzazione ed effettua un controllo sugli errori.
- **Livello di rete:** si occupa del trasporto dei pacchetti lungo tutto il percorso dall'origine alla destinazione finale, che per essere raggiunta può richiedere molti salti attraverso router intermedi lungo tutto il percorso.
- **Livello di trasporto:** si basa sul livello di rete per fornire il trasporto dei dati da un processo su una macchina sorgente ad un processo su una macchina destinazione con un livello di affidabilità desiderato e indipendente dalle reti fisiche in uso.
- **Livello sessione:** controlla la comunicazione tra applicazioni. Instaura, mantiene e abbate connessioni (sessioni) tra applicazioni cooperanti. Si occupa della sincronizzazione di invio/ricezione dei messaggi.
- **Livello presentazione:** trasforma i dati forniti dalle applicazioni in un formato standardizzato e offre servizi di comunicazione comuni, come la crittografia, la compressione del testo e la riformattazione.
- **Livello applicazione:** interfaccia e fornisce servizi per i processi delle applicazioni utente.

## 1.2 Ethernet

Ethernet è un insieme di tecnologie standardizzate sviluppate per la connessione di dispositivi su reti locali.

Le sue radici risalgono al 1967 quando Norman Abramson dell'University of Hawaii sviluppò una rete di connessione PC via radio.

Il progetto chiamato ALOHA portò alla nascita del protocollo ad accesso multiplo CSMA/CD (Carrier Sense Multiple Access with Collision Detection) utilizzato per evitare le collisioni dovute alla trasmissione simultanea di più dispositivi sulla stessa rete contemporaneamente.

In seguito Robert Metcalfe e David Boggs, suo assistente allo Xerox PARC (Palo Alto Research Center) formarono un gruppo di lavoro con la collaborazione delle aziende Xerox Corporation, Intel Corporation e Digital Equipment Corporation, dando origine alle specifiche del protocollo di comunicazione 802.3 e pubblicando la versione 1.0 dello standard Ethernet. Nel 1985 la IEEE Computer Society (Institute of Electrical and Electronics Engineers) iniziò un progetto con l'obiettivo di definire uno standard per l'interconnessione tra dispositivi di produttori differenti con il nome di IEEE 802.3

### 1.2.1 Livello fisico

Al livello fisico del modello OSI, lo standard 802.3 prevede esclusivamente trasmissioni via cavo su cavi coassiali, doppini intrecciati (con o senza schermatura) e fibre ottiche.

- **Cavo coassiale:** supporta gli standard 10BASE-5 (Thick Ethernet) e 10BASE-2 (Thin Ethernet). Utilizzato in passato per le diramazioni dipartimentali delle reti locali con velocità massima di 10 Mbps con comunicazioni di tipo half-duplex e topologia a bus seriale.
- **Doppino intrecciato:** Utilizzato sia nella versione schermata STP (Shielded Twisted Pair) e non schermata UTP (Unshielded Twisted Pair), permette di raggiungere una velocità di 10 Gbps. La comunicazione è sia di tipo half-duplex che full-duplex. Il connettore dominante è di tipo RJ45 a 8 pin, usato a due o a quattro terminali in base alla velocità e al tipo di comunicazione.
- **Fibra ottica:** soluzione più costosa rispetto alle precedenti ma con un maggior grado di efficienza. Immune ai disturbi elettrici, è in grado di coprire una distanza fino a 2 km. Supporta gli standard FOIRL, 10BASE-F, Fast Ethernet e Gigabit Ethernet.

### 1.2.2 Il sottolivello MAC

Nella pila dei protocolli del modello OSI, Ethernet occupa la parte inferiore del livello di collegamento. IEEE ha ritenuto opportuno suddividere questo livello in due parti distinte.

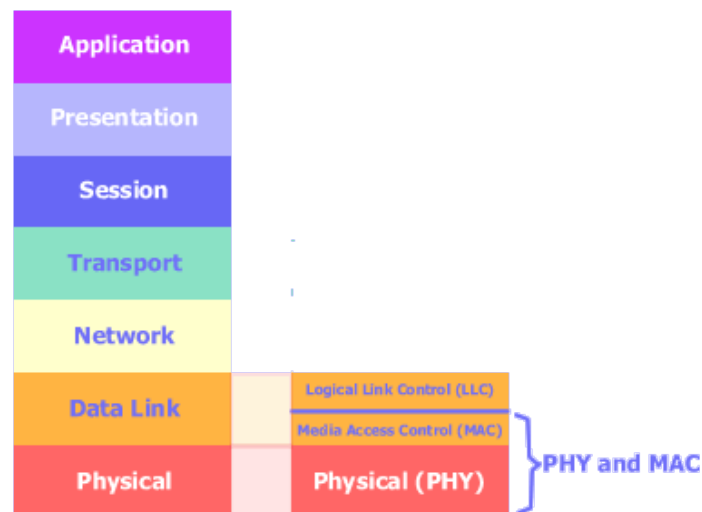


Immagine 1.2: Suddivisione del livello di collegamento

Il sottolivello LLC è comune a tutti gli standard della famiglia IEEE 802 (ad esempio: 802.3 Ethernet, 802.11 wireless LAN), mentre il sottolivello MAC è più strettamente legato al livello fisico, e le sue diverse implementazioni hanno il compito di fornire un'interfaccia comune al livello LLC.

Di seguito il principale compito dei due sottolivelli.

- **Media Access Control (MAC):** contiene le funzionalità di controllo dell'accesso al mezzo fisico per canali broadcast.
- **Logical Link Control (LLC):** fornisce servizi di controllo del flusso, conferma e rilevazione (o correzione) degli errori.

#### 1.2.2.1 Indirizzo

Tutte le stazioni che fanno parte di una rete Ethernet sono dotate di una Network Interface Card (NIC) comunemente chiamata scheda di rete. La scheda viene installata nella stazione e fornisce un suo indirizzo a livello di collegamento. Gli indirizzi Ethernet sono composti da 6 byte (48 bit) e vengono normalmente scritti in notazione esadecimale, con i due punti per dividere i vari byte.

Una caratteristica interessante è che questi indirizzi sono unici al mondo, assegnati da IEEE per garantire che non esistano al mondo due stazioni con lo stesso indirizzo.

#### 1.2.2.2 Trasmissione e ricezione

La trasmissione è sempre broadcast indipendentemente dal fatto che si tratti di frame con destinazione unicast, multicast o broadcast, ovvero tutte le stazioni della rete ricevono i frame inviati.

Il compito di accettare o scartare il frame in arrivo dalla rete è affidato al ricevente, il quale può trattare i frame in arrivo in diversi modi.

- **Unicast:** il destinatario accetta il frame, mentre tutte le altre stazioni lo scartano.
- **Multicast:** le stazioni che appartengono al gruppo multicast lo memorizzano e lo gestiscono, mentre le altre lo scartano.
- **Broadcast:** tutte le stazioni accettano e gestiscono il frame.

#### 1.2.2.3 Frame

Il pacchetto dati ricevuto a livello di collegamento nella pila di protocolli viene chiamato *frame*.

Nonostante l'evoluzione di Ethernet nel corso degli anni abbia subito molti cambiamenti nei meccanismi utilizzati, questo è rimasto fedele alla struttura originale.

Il frame contiene sette campi che vengono descritti di seguito.

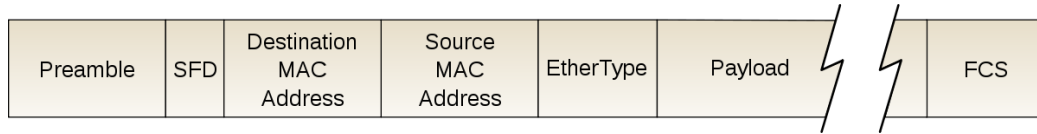


Immagine 1.3: Frame Ethernet

- **Preambolo:** il suo compito è quello di allertare il sistema ricevente sull'arrivo del frame e permettere una sincronizzazione del suo orologio con la trasmissione. Questo campo viene aggiunto al livello fisico e ha una dimensione di 7 byte.
- **SFD:** ha il compito di segnalare alla stazione ricevente l'inizio del frame. Questo campo viene aggiunto a livello fisico e ha una dimensione di 1 byte.
- **Indirizzo di destinazione:** il campo contiene l'indirizzo di collegamento della stazione (o delle stazioni) destinatarie del frame. Quando la stazione destinataria trova in questo campo il suo stesso indirizzo fisico, estrae i dati dal frame e li passa al protocollo di livello superiore. Questo campo ha una lunghezza di 6 byte.
- **Indirizzo sorgente:** contiene l'indirizzo fisico del mittente del frame. Anche questo campo ha una lunghezza di 6 byte.
- **Tipo:** definisce il tipo di protocollo di livello superiore del pacchetto incapsulato nel frame (ad esempio: protocollo IP). Questo campo ha una lunghezza di 2 byte.
- **Dati e padding:** trasporta i dati incapsulati nel frame dai protocolli di livello superiore. Ha una dimensione minima di 46 byte e un massimo di 1500 byte. La rimozione o l'aggiunta del padding è di responsabilità del livello superiore.
- **CRC:** contiene informazioni per il rilevamento degli errori. Il codice di ridondanza contenuto in questo campo viene calcolato sui campi indirizzo, tipo e dati. Se il ricevente, calcolandolo, verifica che non ha valore 0 allora il frame viene scartato dal ricevente in quanto risulta "alterato".

#### 1.2.2.4 Gestione delle collisioni

Dato che il protocollo Ethernet utilizza un approccio broadcast, è stato necessario implementare un metodo di accesso condiviso al mezzo trasmissivo.

Eventuali collisioni dovute all'invio simultaneo di messaggi da parte di stazioni differenti, vengono gestite utilizzando il protocollo *CSMA/CD* (Carrier Sense Multiple Access with Collision Detection).

Quando le stazioni hanno un frame da spedire, controllano il canale e non appena esso risulta libero lo spediscono. Contemporaneamente il canale viene monitorato per trovare collisioni mentre il messaggio viene spedito. Se avviene una collisione, interrompono la

trasmissione con un messaggio caotico per poi ritrasmettere dopo un periodo di tempo casuale.

Un algoritmo utilizzato per determinare l'intervallo di tempo casuale per la ritrasmissione è il *backoff esponenziale binario* (binary exponential backoff), il quale si adatta dinamicamente al numero di stazioni che tentano di trasmettere.

### 1.3 Topologia di rete

Con la continua crescita delle reti Ethernet, la classica architettura lineare della rete è scomparsa, lasciando il posto a nuove tecniche di implementazione. I problemi associati al reperimento di guasti e connettori hanno fatto sì che le stazioni avessero un unico cavo collegato ad un hub centrale.

Un *hub* è un dispositivo che connette elettricamente tutti i cavi collegati, come se fossero saldati insieme. Svolge la funzione di semplice ripetitore (non aumenta quindi la capacità della rete).

Successivamente con la continua aggiunta di stazioni e l'aumento del traffico generato dalle applicazioni, si è assistito all'introduzione di Ethernet Commutata (Switched Ethernet).

Questa architettura di rete utilizza un dispositivo chiamato *switch* (commutatore) che permette di instradare sulle giuste porte di destinazione i frame inviati sulla rete.

Lo switch migliora anche la gestione delle collisioni, riuscendo anche ad eliminarle definitivamente nel caso vengano utilizzati cavi full-duplex per collegare i vari host.

L'utilizzo di questi dispositivi inoltre, aggiunge la possibilità di spedire diversi frame contemporaneamente (da stazioni differenti). Nel caso due frame spediti da due stazioni differenti arrivino contemporaneamente sulla stessa porta, vengono messi in coda all'interno di un buffer, per poi essere trasmessi sulla porta di uscita quando il rispettivo canale risulta libero.

### 1.4 IP: Internet Protocol

Il protocollo IP (Internet Protocol) si posiziona al livello di rete del modello OSI e si occupa di gestire le informazioni necessarie all'instradamento dei pacchetti dati in rete, specificando mittente e destinatario e consentendo ai dispositivi intermedi di scegliere la strada più opportuna per il trasferimento.

I suoi principali compiti sono :

- Definire il formato dei dati (datagramma) da inviare sul mezzo fisico.
- Fornire un metodo di indirizzamento (ad esempio: IPv4).
- Scegliere un percorso di instradamento opportuno, ovvero il percorso che un datagramma deve seguire per raggiungere una determinata destinazione.

- Controllare l'errore sull'intestazione IP.

IP è un protocollo a pacchetti di tipo best effort, cioè che non garantisce nessuna affidabilità della comunicazione sul controllo degli errori, controllo del flusso e controllo della congestione, che come vedremo verrà demandata al livello superiore (livello di trasporto).

#### 1.4.1 Indirizzo IP

All'interno di una rete IP, ad ogni interfaccia connessa alla rete fisica viene assegnato un indirizzo univoco.

L'indirizzo IP è assegnato propriamente all'interfaccia (ad esempio una scheda di rete) e non all'host, perché è questa ad essere connessa alla rete. Un router, ad esempio, ha diverse interfacce e per ognuna occorre un indirizzo IP.

#### 1.4.2 Pacchetto IPv4

Nel pacchetto IP, ovvero nella sua PDU (Protocol Data Unit) , chiamato anche datagramma, i campi presenti sono i seguenti:

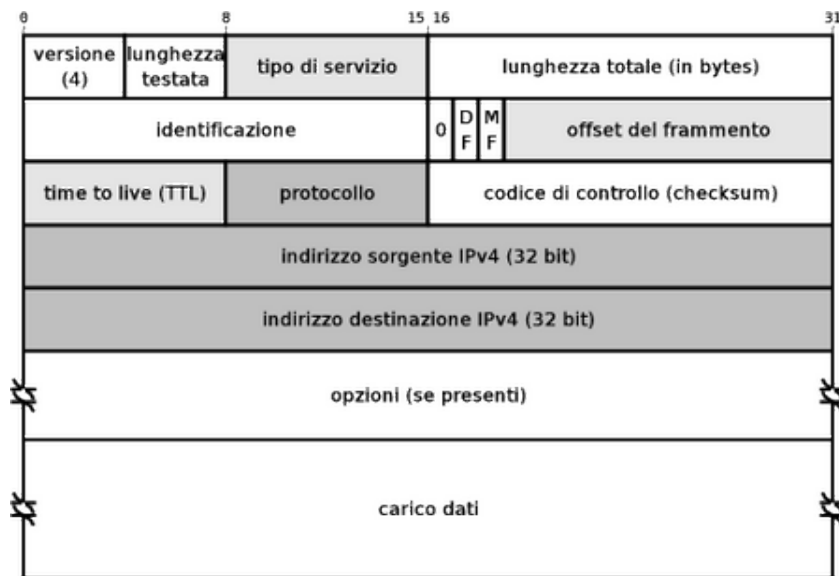


Immagine 1.4: Pacchetto IPv4

- **Versione:** identifica il protocollo utilizzato (ad esempio: IPv4, IPv6). Questo campo ha una dimensione di 4 bit.
- **Lunghezza intestazione:** lunghezza totale del pacchetto IP. La dimensione è di 4 byte.



- **Tipo di servizio:** permette di specificare (quando supportato) un livello di qualità di servizio richiesto dall'utente (ad esempio: affidabilità o velocità di trasferimento). La sua dimensione è di 8 bit.
- **Lunghezza totale:** specifica la lunghezza totale (compresa l'intestazione) del datagramma. La sua dimensione è di 16 bit.
- **Identificazione:** contiene il numero rappresentativo del datagramma. La sua dimensione è di 16 bit.
- **Flags:**
  - **0:** non utilizzato e posto a zero.
  - **DF:** indica se il datagramma può essere frammentato.
  - **MF:** indica se vi sono ulteriori frammenti.
- **Offset del frammento:** indica il numero di byte di dati presenti nel pacchetto ad esso precedente. Se il frammento è il primo o univoco è uguale a 0. La sua dimensione è di 13 bit.
- **TTL :** indica il numero massimo di salti residui che il datagramma può effettuare in rete. La sua dimensione è di 8 bit.
- **Protocollo:** indica a quale protocollo dello stato superiore deve essere trasferito il contenuto informativo del datagramma (ad esempio: TCP=6, UDP=17, ICMP=1). La sua dimensione è di 8 bit.
- **Checksum:** controllo errori sull'intestazione, calcolato come somma in complemento a 1. Se viene rivelato un errore, il datagramma viene scartato. La dimensione è di 16 bit.
- **Indirizzo sorgente:** indirizzo del nodo sorgente. La dimensione è di 32 bit.
- **Indirizzo destinazione:** indirizzo del nodo destinazione. La dimensione è di 32 bit.
- **Opzioni:** campo opzionale che può essere omesso. La sua dimensione dipende dalle opzioni implementate.
  - **Source Route Option:** permette al mittente di specificare i nodi attraverso i quali vuole che il datagramma transiti.
  - **Record Route Option:** permette al mittente di creare una lista vuota di indirizzi IP in modo che ogni nodo attraversato vi inserisca il proprio indirizzo;
  - **Timestamp Option:** inserisce l'istante temporale in cui il datagramma attraversa i nodi.
- **Carico dati:** carico utile dei dati da inviare.

## 1.5 TCP: Trasmission Control Protocol

Il livello di trasporto fornisce servizi al livello applicazione e sfrutta quelli messi a disposizione dal livello di rete.

Il suo principale obiettivo è quello di rendere affidabile la comunicazione tra mittente e destinatario sfruttando i servizi offerti dal protocollo di rete al livello inferiore, il quale definisce il modo di trasferimento dei dati sul canale di comunicazione, ma non offre alcuna garanzia di affidabilità sulla consegna in termini di ritardo, perdita, errori sui pacchetti trasmessi e sul controllo della congestione tra i dispositivi comunicanti.

Di seguito vengono elencate le principali caratteristiche del protocollo.

- TCP è un protocollo orientato alla connessione. Prima di trasmettere i dati, i due host comunicanti stabiliscono una comunicazione, che rimane attiva anche in assenza di scambio di dati e viene chiusa quando non è più necessaria. Mette a disposizione delle funzionalità per creare, chiudere e mantenere attiva una connessione.
- TCP introduce il concetto di affidabilità. La consegna dei segmenti a destinazione è garantita grazie al meccanismo degli *acknowledgements*.
- Permette il trasporto bidirezionale di un flusso di byte tra due applicazioni in esecuzione su host differenti.
- Il flusso di byte generato dall'applicazione mittente viene preso in carico da TCP, frazionato in blocchi (segmenti) e consegnato all'host destinatario che lo passerà all'applicativo indicato dal numero di porta nell'intestazione del pacchetto (ad esempio: porta 502 per il protocollo Modbus).
- Garantisce che i dati trasmessi arrivino a destinazione in ordine ed una sola volta.
- Viene offerta una funzionalità di controllo dell'errore sui pacchetti grazie al campo checksum.
- Fornisce funzionalità sul controllo del flusso tra gli host in comunicazione e controllo della congestione.
- Fornisce un meccanismo di multiplexing delle connessioni sopra un host, tramite il meccanismo delle porte.

### 1.5.1 Il segmento TCP

L'intestazione TCP viene utilizzata dal software di rete del dispositivo ricevente per verificare se tutti i pacchetti sono stati ricevuti correttamente.

L'intestazione TCP è formata dai seguenti campi:

Porta Sorgente(16)			Porta destinazione(16)		
Numero di Sequenza(32)					
Numero di Acknowledgement(32)					
HLEN(4)	Riservati(6)	Flag(6)		Window(16)	
Checksum(16)			Urgent Pointer(16)		
Opzioni				Padding	
Dati					

Immagine 1.5: Segmento TCP

- **Porta sorgente:** identifica il numero di porta sull'host mittente associato alla connessione TCP.
- **Porta destinazione:** identifica il numero di porta sull'host destinatario associato alla connessione TCP.
- **Numero di sequenza:** indica la distanza in byte dell'inizio del segmento all'interno del flusso completo, a partire dall' Initial Sequence Number (ISN).
- **Numero di Acknowledgement:** numero di riscontro. Se il flag ACK ha valore 1 la ricezione del segmento viene confermata.
- **HLEN:** indica la lunghezza dell'intestazione del segmento TCP.
- **Riservati:** sono predisposti per futuri sviluppi del protocollo e non vengono utilizzati.
- **Flag:** bit utilizzati per il controllo del protocollo.
- **Window:** indica la dimensione della finestra di ricezione dell'host mittente.
- **Checksum:** campo di controllo utilizzato per la verifica della validità del segmento.
- **Urgent pointer:** puntatore ad un dato urgente.
- **Opzioni:** opzioni (facoltative) per usi del protocollo avanzati.
- **Dati:** rappresenta il carico utile da trasmettere.

### 1.5.2 Connessione

Come elencato nel paragrafo precedente, i processi applicativi dei terminali in comunicazione, prima di iniziare a trasferire i dati devono stabilire una comunicazione attraverso la definizione del socket (<indirizzo IP, porta>). Al termine della comunicazione, la connessione viene abbattuta.

La procedura di apertura di una connessione prende il nome di *Three-way handshake* (stretta di mano in 3 passaggi) e viene suddivisa in tre fasi distinte:

- Richiesta connessione.
- Accettazione della connessione.
- Conferma.

Al termine della connessione, il protocollo effettua la chiusura, utilizzando la procedura chiamata *Four-way handshake*.

### 1.5.3 Affidabilità della comunicazione

Per garantire l'affidabilità sulla comunicazione e lo scambio di dati tra due host, TCP effettua diversi tipi di controlli. I principali vengono elencati di seguito.

- **Controllo del flusso:** il flusso dei dati in trasmissione viene gestito, facendo in modo che questo non superi la capacità di ricezione del ricevente, evitando così la perdita di dati e ritardi di trasmissione.  
Questo meccanismo viene implementato attraverso la gestione della finestra di ricezione del destinatario, la quale specifica il numero massimo di segmenti che può ricevere.
- **Controllo della congestione:** TCP cerca di evitare i fenomeni di congestione all'interno della rete, che portano ad un ritardo di consegna e alla perdita di pacchetti. Questo controllo non viene effettuato dai dispositivi di commutazione in rete, ma agli estremi, ovvero dai due host comunicanti. Il meccanismo viene gestito mediante la finestra di congestione, la quale assegna, dinamicamente nel tempo, il numero massimo di segmenti da trasmettere al destinatario.
- **Consegna ordinata con eliminazione dei duplicati:** grazie al numero di sequenza sul pacchetto, il destinatario è in grado di consegnare al processo del livello applicativo, i segmenti in ordine corretto, scartando i segmenti in arrivo duplicati e segnalando alla sorgente la perdita di pacchetti sulla rete tramite i meccanismi di acknowledgement.

## 1.6 UDP: User Datagram Protocol

User Data Protocol (UDP) è un protocollo definito al livello di trasporto del modello OSI, utilizzato solitamente in combinazione con il livello di rete IP. Contrariamente a

TCP, il protocollo UDP è senza connessione (Connectionless), ovvero lo scambio dei pacchetti tra mittente e destinatario (o destinatari) non richiede nessuna operazione preliminare prima del trasferimento.

Oltre a questo, UDP introduce altre caratteristiche elencate di seguito:

- Non garantisce l'affidabilità della connessione, il controllo del flusso dei dati e la correzione degli errori.
- E' semplice e presenta un ridotto overhead dei segmenti (più veloce e con basse latenze).
- Il soggetto della comunicazione è il datagramma.

Per queste caratteristiche questo protocollo viene utilizzato da quelle applicazioni che non richiedono un'affidabilità o dove sono richiesti dei vincoli di velocità elevati.

Molto spesso le applicazioni in tempo reale richiedono un bit-rate minimo di trasmissione, non devono ritardare eccessivamente la trasmissione dei pacchetti e possono tollerare qualche perdita di dati (TCP non è adatto a questo).

### 1.6.1 Datagramma UDP

Un datagramma (o pacchetto) UDP è strutturato nel seguente modo:

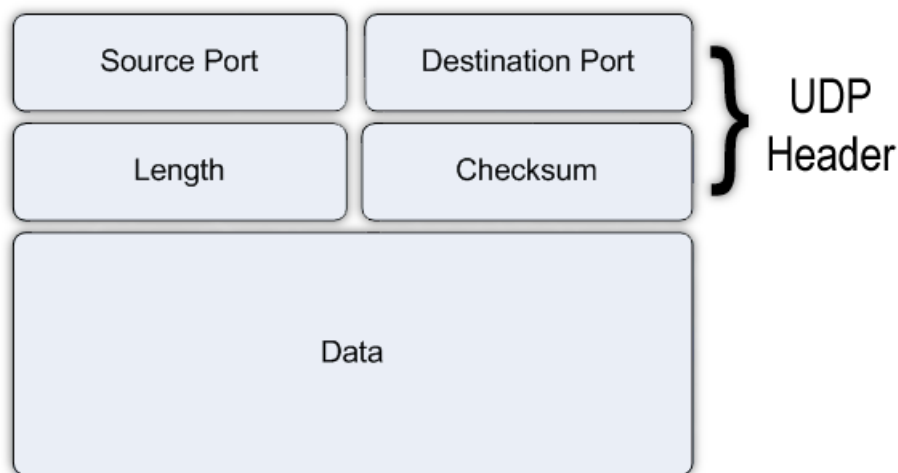


Immagine 1.6: Intestazione UDP

- **Source Port:** identifica il numero di porta sull'host del mittente del datagramma (16 bit).
- **Destination Port:** identifica il numero di porta sull'host del destinatario del telegramma (16 bit).
- **Length:** contiene la lunghezza totale in bytes del datagramma UDP (16 bit).
- **Checksum:** contiene il codice di controllo del datagramma (16 bit).
- **Data:** contiene i dati del messaggio.

## Capitolo 2

# Comunicazione industriale

Tradizionalmente, le reti utilizzate nelle aziende manifatturiere sono state ottimizzate per fornire elevate prestazioni su specifiche applicazioni come il controllo, lo scambio di informazioni e la gestione della sicurezza.

Benché fossero adatte alle funzionalità per cui sono state progettate, queste tipologie di reti non sono state sviluppate con un architettura unificata.

Differenti produttori di dispositivi sono stati costretti a implementare diverse reti di comunicazione, nessuna delle quali riusciva a dialogare in modo rapido ed efficiente con le altre. Di conseguenza, nel corso del tempo, la maggior parte degli ambienti manifatturieri sono stati popolati da numerose reti specializzate, e generalmente incompatibili, che lavoravano in modo indipendente.

Oggi aziende di tutto il mondo cercano di interconnettere l'intera linea aziendale. Non viene richiesto solo il controllo dei processi di produzione ma si vuole permettere a tutti gli utenti dell'intera filiera produttiva di poter accedere ai dati di produzione e integrare queste informazioni con i sistemi informativi aziendali.

Per far fronte a queste richieste, diversi consorzi hanno cercato di standardizzare protocolli di comunicazione che soddisfino tutte queste esigenze.

Ogni costruttore ha scelto il protocollo che meglio si adatta alle proprie esigenze di comunicazione.

## 2.1 Il flusso dei dati

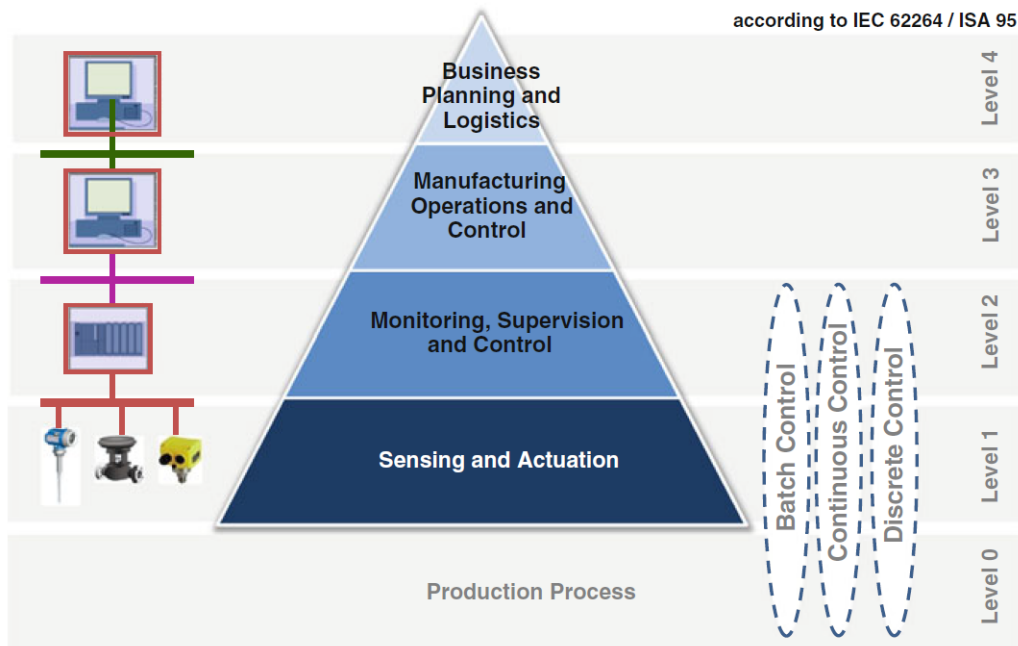


Immagine 2.1: Il flusso dei dati in un sistema produttivo industriale

Le informazioni che transitano sulla rete durante un processo produttivo industriale variano in base al livello gerarchico del sistema sul quale operano. Questi diversi livelli di applicazione formano la piramide *CIM* (Computer Integrated Manufacturing). La piramide è suddivisa in cinque differenti livelli in cui sono presenti sistemi di controllo modulari che si occupano dei diversi processi presenti al livello considerato. Ogni livello è caratterizzato da:

- acquisizione, manipolazione e trasferimento delle informazioni.
- elaborazione delle strategie.
- attuazione delle strategie elaborate.

All'interno di questa struttura esiste uno scambio di informazioni orizzontale (tra i processi automatizzati dello stesso livello) e verticale (tra i processi automatizzati dei vari livelli).

Le elaborazioni effettuate ai livelli più elevati vengono eseguite su dati complessi e strutturati (ad esempio: basi di dati a supporto dei sistemi gestionali).

Scendendo di livello le tempistiche con le quali si memorizzano le informazioni, si elaborano le decisioni e si attuano i comandi diventano sempre più stringenti. I livelli più bassi della piramide possiedono caratteristiche di correttezza temporale oltre che di logica. Non è solo importante che il risultato elaborato sia corretto, ma è fondamentale che



tale risultato arrivi entro un certo istante temporale. Per queste caratteristiche, questo livello utilizza sistemi che si scambiano informazioni utilizzando una o più reti *real-time*. Entriamo ora nel dettaglio descrivendo i cinque livelli che compongono la piramide CIM.

- **Livello di campo:** questo livello è il più basso della piramide e comprende i vari componenti hardware che fisicamente eseguono le elaborazioni e le trasformazioni necessarie per la produzione ed il controllo. In questo livello vengono posizionati i sensori, attuatori e i componenti dell'impianto (meccanici, elettromeccanici, pneumatici, ecc.). A questo livello l'intelligenza dei dispositivi è limitata, in quanto devono solamente trasdurre grandezze fisiche in arrivo dal campo.  
Il livello di campo è il punto di ingresso/uscita al processo per il livello superiore e la sua funzione è quella di riportare al livello sovrastante le misure di processo e attuare i comandi che gli vengono inoltrati da esso. Un sistema di controllo di campo viene visto dal livello superiore come un attuatore virtuale. In tal modo il livello superiore può inoltrare una richiesta (ad esempio: impostare la velocità di movimento di un motore) senza entrare nel dettaglio di come effettivamente venga attuata. Il controllo di campo viene implementato utilizzando sistemi digitali con processore sviluppati ad hoc (controllori embedded).
- **Livello di macchina:** gli elementi del controllo di campo vengono raggruppati al livello superiore per formare gruppi di componenti in grado di fornire una determinata funzionalità. Questi componenti, a livello di macchina sono organizzati in sistemi di controllo con funzioni per il controllo di variabili analogiche per la regolazione del processo e per realizzare operazioni sequenziali. Il controllo a livello macchina viene visto come attuatore virtuale dal livello superiore.  
Considerando il controllo di un robot industriale, a livello macchina viene pianificato il movimento del robot nello spazio operativo e la sequenza delle operazioni che deve effettuare, mentre al livello di campo vengono controllate e attuate le posizioni dei singoli giunti.  
Le apparecchiature utilizzate a questo livello sono sistemi digitali come il controllore logico programmabile (PLC) o sistemi di controllo embedded.
- **Livello di cella:** una cella di produzione è un insieme di macchine fisicamente interconnesse da un sistema di trasporto e controllate in maniera coordinata, in modo da portare a termine un processo produttivo. I sistemi di controllo posizionati a questo livello regolano e supervisionano il funzionamento coordinato di tutte le macchine che costituiscono la cella di lavoro. Le operazioni che vengono effettuate a questo livello sono analoghe a quelle del livello macchina (sono soltanto più complesse perchè coinvolgono più elementi da controllare). Le apparecchiature utilizzate sono le stesse del livello macchina inferiore (PLC e dispositivi embedded).
- **Livello di stabilimento:** questo livello racchiude tutte le celle o le linee produttive di un impianto industriale. Attua sotto forma di piano operativo per la produzione le istruzioni ricevute dal livello gestionale (planning, gestione ordini). Il componente essenziale di questo livello è il sistema di supervisione, controllo e acquisizione dati

(SCADA, Supervisory Control And Data Acquisition). Il sistema di supervisione normalmente viene implementato su workstation con struttura client/server. Da questo livello in su i requisiti per l'elaborazione Real-Time sono fortemente ridotte se non inesistenti.

- **Livello di azienda:** questo è il livello più alto della gerarchia dove avvengono i processi gestionali che sono di supporto a tutti i livelli inferiori. E' un livello decisionale dove le infrastrutture software vengono implementate su workstation connesse al mainframe aziendale. A questo livello non esistono vincoli di tipo temporale.

La struttura gerarchica appena descritta è stata definita nello standard ANSI/ISA-S88.01-1995. Questo standard classifica le funzioni di controllo in tre differenti livelli.

- **Controllo di campo:** è il livello di controllo più basso che si colloca al livello di campo e comprende i sistemi di controllo dei singoli componenti di campo. E' di tipo continuo e viene implementato su dispositivi dedicati come controllori embedded o schede dedicate al controllo dei motori elettrici.
- **Controllo di procedure:** riguarda il controllo di gruppi strutturati di componenti di campo e si colloca al livello di macchina e cella. Può essere di tipo logico o continuo.

Il controllo continuo si trova soprattutto a livello di macchina e riguarda il controllo di gruppi di riferimento di variabili continue o funzioni più avanzate come la determinazione dei segnali di riferimento per il controllo di un motore.

Il controllo logico di procedure riguarda il coordinamento dei sistemi in base alla sequenza delle operazioni che compongono il processo di lavorazione. Questo tipo di controllo svolge anche funzioni per il monitoraggio delle prestazioni del sistema e una diagnostica con relativa gestione automatica dei malfunzionamenti.

Il controllo di procedure può essere implementato su schede dedicate, PC industriali e PLC.

- **Controllo di coordinamento:** è il livello di controllo più elevato che si pone a livello di stabilimento della piramide CIM. Riguarda principalmente il coordinamento e la gestione delle varie celle di produzione: manda in esecuzione, dirige o interrompe le varie procedure in esecuzione sui sistemi di controllo sulla base di algoritmi orientati all'intelligenza artificiale e ai sistemi esperti. Questo tipo di coordinamento non è soggetto a vincoli real time e viene implementato su calcolatori standard.

I vari livelli della piramide CIM si scambiano le informazioni processate attraverso un'infrastruttura di comunicazione informatica che interconnette tutta l'architettura descritta. Tale rete deve consentire una comunicazione orizzontale tra i vari componenti dello stesso livello e verticale tra i vari livelli.

Le esigenze di scambio di informazioni sono differenti tra i vari livelli: i livelli inferiori scambiano informazioni semplici e poco strutturate ma con un'elevata

frequenza e con necessità di determinismo per assicurare che la comunicazione avvenga in modo affidabile entro un istante di tempo certo. Ai livelli più elevati invece, le informazioni sono costituite da dati complessi e strutturati di elevate dimensioni. La frequenza di comunicazione è molto più bassa e non ci sono vincoli temporali.

Per questi motivi la rete di comunicazione è costituita da più sottoreti ognuna delle quali è caratterizzata dai requisiti dei componenti del livello che mette in comunicazione.

All'interno dell'infrastruttura possiamo identificare tre tipologie di reti di comunicazione:

- **Rete di campo:** questa rete mette in comunicazione i dispositivi di campo e i vari sistemi di controllo di macchina. Deve quindi assicurare lo scambio di piccoli pacchetti di dati con elevata velocità e determinismo temporale. L'estensione di questa rete è geograficamente locale.
- **Rete per il controllo:** permette la comunicazione tra i sistemi di controllo dei livelli di cella e macchina. I dati scambiati sono strutturati, di piccola dimensione e con bassa frequenza di trasferimento. I vincoli real time sono meno stringenti. Questa rete ha un'estensione limitata.
- **Rete enterprise (rete per le informazioni):** è la rete informativa dell'azienda che collega tutti i livelli di stabilimento e azienda. I dati sono di grande dimensione e strutturati con una frequenza di comunicazione bassa. Non esistono vincoli real-time. L'estensione della rete è più ampia rispetto alle precedenti in quanto può interconnettere anche più stabilimenti.

## 2.2 Algoritmi di scheduling per reti Real-Time

Come descritto in precedenza, le reti real-time sono orientate alla connessione, ovvero la trasmissione di un messaggio implica la creazione di un canale di comunicazione fisico e logico punto-punto tra trasmettitore e ricevitore. Per quanto riguarda gli algoritmi di scheduling, il problema da affrontare riguarda la gestione dell'ordine di trasmissione tra i nodi della rete e l'accesso al canale condiviso.

In questa sezione vengono analizzati i metodi di scheduling per la gestione dell'ordinamento di trasmissione degli host nelle reti switched real-time. In queste reti i vari nodi possono essere messi in comunicazione tramite diversi percorsi fisici formati da rami (link) di connessione punto-punto collegati tra loro attraverso degli switch utilizzati per instradare i pacchetti.

Questi dispositivi hanno un buffer di ingresso condiviso dai rami in ingresso e da tanti buffer di uscita quanti sono i rami in uscita. Due host che devono comunicare con diversi ricevitori possono instradare i messaggi attraverso il medesimo switch. In questo caso è importante schedare opportunamente la trasmissione dei vari pacchetti dallo switch per soddisfare le specifiche real-time di entrambi gli host. Gli algoritmi di sche-

during utilizzati per soddisfare questi requisiti sono algoritmi di gestione delle code. In letteratura esistono diverse classi di algoritmi per la gestione di questi buffer:

- **FIFO queueing:** vengono utilizzate le politiche di *First In, First Out* in cui la trasmissione è gestita in base all'ordine di arrivo dei messaggi da servire nei vari buffer.
- **Priority queueing:** vengono utilizzate politiche che gestiscono un ordinamento di priorità dei messaggi nei buffer.
- **Fair queueing:** sono algoritmi in cui le code sono servite secondo politiche *Round Robin* per evitare che alcuni messaggi a priorità inferiore non vengano mai presi in considerazione.

Lo scheduling per servizi orientati alla connessione deve essere senza prelazione (*non-preemptive*): una volta iniziata una trasmissione di un frame la connessione tra i dispositivi non viene chiusa fino a quando non è terminata.

Di seguito viene analizzato un algoritmo molto utilizzato nelle reti RT, denominato *Non Preemptive Weighted Fair Queueing Algorithm*, che combina le caratteristiche degli algoritmi fair queueing a quelle degli algoritmi priority queueing. Alla politica round robin viene associato un metodo di ordinamento di priorità dei messaggi nei buffer in modo che tutte le code vengano servite e allo stesso tempo quelle con maggior priorità abbiano maggior banda garantita. Con questa tecnica è possibile ottenere un buon soddisfacimento per quanto riguarda i vincoli temporali, di predicibilità e di determinismo sui tempi di trasmissione con un ritardo di jitter contenuto.

Si consideri uno switch con diversi link in ingresso sui quali arrivano i messaggi che devono essere distribuiti sui vari nodi della rete. A monte di questi link, un nodo  $i$  che vuole trasmettere utilizzerà un proprio buffer di uscita per accodare i messaggi in uscita, implementando una coda FIFO (quando il messaggio arriva in testa alla coda viene inviato al buffer in ingresso del nodo a valle). Sia  $n$  il numero di connessioni attive in ingresso ad un determinato switch, ad ogni connessione viene assegnata una frazione  $u_i$  della banda disponibile tale che

$$U = \sum_{i=1}^n u_i \leq 1 \quad (2.1)$$

In questo modo si suppone che le connessioni in ingresso non saturino completamente la banda disponibile dello switch.

Lo switch per gestire correttamente i messaggi che arrivano dalle  $n$  code a monte, costruisce una coda a priorità definendo un indice per ogni link  $i$  in ingresso chiamato *finishing number*  $f_{n_i}$ . Questo indice è associato al pacchetto in testa alla coda a monte, ovvero al pacchetto pronto ad essere spedito dal nodo  $i$ .

Lo switch instrada i pacchetti dal proprio buffer di uscita e quindi verso la destinazione successiva secondo un ordine stabilito dal finishing number: più questo indice è piccolo più il corrispettivo pacchetto nella coda di ingresso dello switch sarà prioritario. Dal

momento che questo metodo non ammette prelazione, il pacchetto in trasmissione continuerà a essere gestito fino a che la trasmissione non viene completata anche se nel frattempo arrivano pacchetti con priorità maggiore.

Quando il primo pacchetto di una connessione  $i$  arriva sullo switch lo scheduler calcola il finishing number della connessione e inserisce l'identificativo di connessione e il finishing number nel suo buffer  $(i, f_{n_i})$ . Se il canale è libero la trasmissione inizia immediatamente. Se nel frattempo arrivano altre connessioni da altre connessioni  $j$  che vogliono trasmettere, lo scheduler ne calcola il finishing number e lo inserisce nel suo buffer. Quando la trasmissione del pacchetto dal nodo  $i$  è terminata, lo scheduler elimina l'elemento corrispondente nella sua coda di priorità. Se il nodo  $i$  che ha appena utilizzato la connessione dello switch vuole ancora trasmettere, lo scheduler calcola il finishing number del nuovo pacchetto ed inserisce la nuova coppia  $(i, f_{n_i})$  nella sua lista di priorità. A questo punto lo scheduler avvia la trasmissione dal nodo che nella sua lista ha il valore di finishing number più basso.

Il finishing number di un pacchetto che un generico nodo vuole trasferire può essere calcolato utilizzando la tecnica descritta di seguito.

Inizialmente per ogni connessione il finishing number è nullo.

$$f_{n_i}[0] = 0 \quad (2.2)$$

Sia  $d_k$  la dimensione di un pacchetto  $k$ . Il finishing number della connessione  $i$  viene aggiornato all'arrivo di un pacchetto  $k$  pronto per la trasmissione nel seguente modo:

$$f_{n_i}[k] = f_{n_i}[k-1] + \frac{d_k}{u_i} \quad (2.3)$$

Utilizzando questa tecnica l'ordinamento dei pacchetti pronti per la trasmissione da diversi nodi avviene secondo un ordinamento decrescente definito dal finishing number che rappresenta una sorta di tempo virtuale mancante alla fine del task di trasmissione dei nodi.

Sia  $d_i$  la dimensione dei pacchetti in coda dal nodo  $i$ . Nel caso peggiore, la latenza di un pacchetto pronto alla trasmissione dal nodo  $i$  è:

$$L_i = 1 + \frac{d_i}{u_i} \quad (2.4)$$

Utilizzando questo algoritmo per la gestione delle trasmissioni all'interno degli switch, la trasmissione offre garanzie di predicibilità e quindi di determinismo essendo noto il comportamento nel caso peggiore.[4]

## 2.3 I bus di campo

Il *bus di campo* (Fieldbus) è la parte del sistema di comunicazione attraverso il quale vengono trasmesse le informazioni necessarie a controllare i processi industriali distribuiti.

E' un sistema di comunicazione digitale di tipo seriale tra apparati installati nei reparti di produzione, ovvero dispositivi di campo (sensori, attuatori, variatori di velocità, strumenti di misura e regolatori, a livello 1 del modello CIM), oppure tra sistemi di controllo automatico (a livello 1 e 2 del modello CIM). I fieldbus operano a livello di gruppi di byte e implementano i livelli ISO/OSI 1, 2, 7, oltre ad un ulteriore livello "utente". I bus di campo sono connessioni "intelligenti" in grado di ridurre drasticamente il numero di cavi utilizzati, senza diminuire la funzionalità del sistema.

Se in passato i dati di un impianto si basavano sulla misurazione continua delle variabili di processo e sulla generazione di segnali elettrici ad esse proporzionali, oggi il bus di campo è la sorgente di tutte le informazioni necessarie per la definizione delle strategie di controllo, produttività, manutenzione, diagnostica, calibrazione e asset management. Il fieldbus consente l'utilizzo del mezzo fisico (generalmente un doppino) per trasmettere misure, comandi e informazioni digitalizzate in senso bidirezionale. Può essere realizzato con un protocollo proprietario, ma nei sistemi complessi e aperti è preferibile optare per uno standard di mercato.

I bus di campo sono stati standardizzati dalla norma IEC 61158, che definisce servizi e protocolli di comunicazione previsti, raggruppati in tre livelli principali, ciascuno dei quali è definito da uno o più documenti.

- **IEC61158-1:** Introduzione.
- **IEC61158-2 PhL:** specifica del livello fisico e definizione del servizio.
- **IEC61158-3 DLL:** definizione del servizio Data Link.
- **IEC61158-4 DLL:** specifica del protocollo Data Link.
- **IEC61158-5 AL:** definizione del servizio Application Layer.
- **IEC61158-6 AL:** specifica del protocollo Application Layer.
- **IEC61158-7:** Gestione della rete.

## 2.4 Determinismo

Come noto, l'impiego di Ethernet come bus campo è sempre stato penalizzato dalla mancanza di determinismo.

Una rete è deterministica se è possibile calcolare un tempo massimo entro il quale l'informazione inviata da un nodo arriva a destinazione. Per fare questo, è fondamentale l'analisi del caso peggiore per ottenere il valore del tempo di trasmissione massimo, che dipende da:

- Numero di nodi nella rete.
- Collisioni e loro gestione per reti di tipo broadcast.

- Gestione degli errori di trasmissione.

Vediamo come sono state superate queste problematiche per portare la tecnologia Ethernet, tradizionalmente non deterministica, ad un livello deterministico per applicazioni industriali.

### 2.4.1 Fast Ethernet

Introdotta nel 1995, Fast Ethernet è un termine collettivo che descrive un insieme di standard Ethernet che trasportano i dati ad una velocità di 100 Mbps rispetto alla velocità originale di Ethernet di 10 Mbps. Sebbene non sia lo standard più veloce è quello che viene maggiormente utilizzato.

In questa versione di Ethernet non vengono apportate modifiche alla struttura del frame ed al protocollo CSMA/CD.

Molte applicazioni industriali non risentono una sostanziale differenza tra l'utilizzo di Fast Ethernet e Ethernet tradizionale. Molti dispositivi, come sensori, consumano e producono una piccola quantità di dati che riescono ad incapsulare in un frame di 64 byte, ovvero di dimensione minima supportata da Ethernet.

Un aspetto che porta un notevole miglioramento è dato dal tempo di back-off di Fast Ethernet. In questa versione, questo tempo è un decimo di quello di una rete tradizionale a 10 Mbps. In reti sovraccariche, questo miglioramento porta ad una notevole riduzione delle collisioni.

L'aumento del data rate introdotto da Fast Ethernet non risolve il problema del determinismo, in quanto la condizione da soddisfare è quella di riuscire a risolvere le collisioni in modo deterministico, o meglio ancora eliminandole completamente.

#### 2.4.1.1 Trasmissione Full-Duplex

Nella prima standardizzazione di Ethernet, nel 1985, era prevista una sola modalità di comunicazione, quella half-duplex. Un nodo della rete trasmetteva o riceveva nello stesso istante temporale. Non era possibile trasmettere e ricevere contemporaneamente. Questa modalità, posizionava i nodi che condividevano il collegamento sullo stesso dominio di collisione; i nodi sono in competizione per poter trasmettere i dati sul bus ed i frame trasmessi sulla rete possono collidere.

Con la comunicazione full-duplex sono presenti due percorsi separati sulla rete per trasmettere e ricevere i dati. In questo modo sono possibili contemporaneamente trasmissione e ricezione ed ogni nodo ha un unico dominio di collisione, eliminando il pericolo di collisioni e il tradizionale protocollo per la loro gestione, il CSMA/CD.

#### 2.4.1.2 Priorità dei messaggi

L'introduzione della specifica 802.1p introdotta dall'IEEE, ha permesso di aggiungere un tassello in più per arrivare a soddisfare i requisiti del determinismo.

Questa specifica definisce le priorità sui messaggi che transitano sulla rete (CoS, Class Of Service). Il frame Ethernet viene marcato con un valore compreso tra 0 e 7 che

ne identifica la priorità, utilizzato dagli switch per stabilire l'importanza dei messaggi memorizzati nei buffer durante la fase di smistamento, riducendo l'incidenza del traffico non critico sui dati di importanza maggiore.

Il settaggio del livello di priorità viene implementata utilizzando tre bit di un estensione di 4 byte prevista per gli standard 802.1p e 802.1q. Questa estensione di intestazione consente un aumento della dimensione complessiva del frame Ethernet di 1522 byte.

### 2.4.2 Switch

Il problema delle collisioni può essere risolto inserendo all'interno della rete gli switch, ovvero dispositivi hardware che operano a livello data link (livello 2 del modello ISO/O-SI) e creano delle connessioni punto punto virtuali, consentendo l'inoltro dei messaggi solo ai dispositivi interessati.

Ogni dispositivo viene collegato ad una porta dello switch. Al suo interno si creano dei circuiti che permettono l'interscambio dei messaggi tra le varie porte solo con i nodi interessati. Per esempio, se la stazione A vuole inviare un messaggio a B e la stazione C vuole dialogare con D, all'interno dello switch si creano dei circuiti di comunicazione che permettono alla informazioni di transitare direttamente tra C e D e da A e B.

All'interno degli switch sono presenti dei buffer che permettono di memorizzare i pacchetti che non possono essere spediti ad una stazione, in quanto occupata. Ad esempio nel caso in cui la stazione A stia parlando con B e anche la stazione D vuole inviare dati a B, non avremmo collisioni, in quanto lo switch memorizza i dati che la stazione D vuole inviare e reinvia il tutto a B non appena ha finito di dialogare con A.

Tramite l'utilizzo di questi dispositivi è possibile isolare il dominio di collisione attraverso una segmentazione della rete poichè ogni porta viene configurata come singolo dominio di collisione. In questo modo si riesce a ridurre o perfino ad eliminare il numero di collisioni sulla rete.

Una corretta segmentazione della rete rimuove le cause primarie della congestione sulle reti LAN.

I moderni switch introducono anche le seguenti funzionalità:

- **VLAN:** insieme di tecnologie che permettono di segmentare la rete in più reti locali logicamente non comunicanti tra loro, ma che condividono globalmente lo stesso mezzo trasmissivo.
- **Flow Control:** meccanismo che consente di limitare il carico della rete e che può essere utilizzato in alcune situazioni per evitare il degrado delle prestazioni real-time. Questa funzionalità elimina la possibilità di avere pacchetti persi in porte full-duplex congestionate avvisando le stazioni che stanno sovraccaricando la rete.
- **Autonegotiation:** funzionalità che permette il riconoscimento automatico e la negoziazione della velocità di trasferimento da utilizzare ed il tipo di funzionamento (full-duplex o half-duplex).



### 2.4.2.1 Switched Ethernet

Tramite la conoscenza del tipo di switch installati è possibile stimare a priori i parametri che caratterizzano la rete ed i limiti raggiunti.

I tradizionali switch utilizzati sono divisi in tre categorie:

- **Store & Forward:** esegue un controllo sul pacchetto ricevuto prima di inoltrarlo al nodo destinatario. Eventuali pacchetti incompleti o danneggiati vengono eliminati dal buffer. Questo controllo, da applicare ad ogni pacchetto ricevuto, aggiunge un leggero ritardo durante l'inoltro del pacchetto al nodo destinatario.
- **Cut-Throught:** riceve e memorizza solo i byte iniziali di un pacchetto necessari a capire la destinazione del pacchetto, minimizzando il ritardo di trasmissione. Eventuali errori di CRC o pacchetti corrotti possono essere inoltrati senza nessuna verifica di validità.
- **Fragment-Free:** sono una versione più sicura degli switch Cut-Throught in quanto controllano anche che il pacchetto in arrivo sia più lungo di 64 byte (minima lunghezza di un pacchetto Ethernet), impedendo l'inoltro di frame falsi.

Il tempo impiegato da un pacchetto per transitare da un nodo ad un altro dipende fortemente dal numero di volte che questi passano all'interno degli switch.

Anche con l'utilizzo di questi dispositivi, si possono verificare delle situazioni in cui il jitter aumenti in modo non deterministico. I dati inviati su un canale già in uso possono essere accodati in attesa del proprio turno di trasmissione.

Utilizzando dispositivi che sfruttino la priorità dei messaggi CoS, questo problema non viene risolto. L'utilizzo di priorità sui pacchetti, garantisce che un frame con maggiore priorità sorpassi un frame con priorità più bassa all'interno dello switch, ma non permette di interrompere un frame che è già stato spedito sulla rete per inviarne uno con priorità maggiore.

Da queste analisi si può concludere che l'utilizzo di Ethernet tradizionale non garantisce il determinismo in caso di rete trafficate. Per soddisfare i requisiti di *Hard Real-Time* richieste soprattutto in applicazione di *Motion Control*, è necessario portare miglioramenti a livello hardware per ridurre questi ritardi di trasmissione introdotti dai tradizionali switch.

### 2.4.3 Sincronizzazione: IEEE 1588

In architetture di rete distribuite, ovvero dove CPU, sistema di attuazione delle uscite e acquisizione degli ingressi sono nodi distinti della rete, è possibile che i riferimenti temporali siano differenti. In questi contesti è necessaria un'accurata sincronizzazione tra i vari nodi distinti della rete.

Lo standard IEEE 1588 definisce un protocollo per scambiare informazioni relative al tempo, il *Precision Time Protocol* (PTP). Questo protocollo nasce come evoluzione dello standard NTP (Network Time Protocol), utilizzato nel settore delle telecomunicazioni,

migliorandone l'accuratezza di sincronizzazione ma riducendone l'utilizzo solamente a poche sottoreti.

Il protocollo prevede una negoziazione della migliore sorgente di clock (master clock) presente sulla rete e si propone di stimare oltre allo sfasamento (O) tra i riferimenti temporali del master e dello slave, anche il ritardo di trasmissione (D) tra master e slave, assunto identico in entrambe le direzioni.

Per prima cosa l'algoritmo *Best Master Clock* (BMC) individua il miglior clock in termini di affidabilità e accuratezza di tutto il sistema e per ogni sottorete. Il dispositivo con il miglior clock viene identificato come *Grand Master Clock* (GMC), mentre quello di ogni sottorete viene chiamato *Master Clock*. In tutta la rete può essere presente un solo GMC ed un solo Master Clock per ogni sottorete. Lo standard definisce due tipi di clock. Un clock utilizzato nei normali dispositivi detto *Ordinary Clock* ed un *Boundary Clock*, utilizzato in dispositivi destinati alla segmentazione della rete, come switch e hub. Il processo di sincronizzazione avviene in due fasi distinte. In una prima fase vengono corretti gli orologi dei vari dispositivi correggendo l'offset tra master e slave, mentre nella seconda parte vengono determinati i ritardi di trasmissione.

Il master invia un messaggio di sincronizzazione, *SYNC message*, contenente una stima del tempo in cui il messaggio lascerà fisicamente l'host. In parallelo, a livello hardware viene misurato il tempo in cui il messaggio lascia veramente il master. Successivamente, il master invia agli slave un secondo messaggio, *following-up message*, che contiene il tempo esatto di invio del messaggio SYNC. Gli slave a questo punto misurano il tempo di ricezione dei messaggi, riuscendo quindi a calcolare ed a correggere il loro offset.

Nella seconda fase della comunicazione, lo slave invia un messaggio di *Delay-Request* al master, ricevendo in risposta un messaggio di *Delay-Response*, riuscendo così a calcolare i ritardi di trasmissione presenti sulla rete e a regolare i clock sui vari dispositivi. Il ritardo calcolato è un ritardo simmetrico tra master e slave, ed è di fondamentale importanza per la sincronizzazione. Se sulla rete fossero presenti switch, queste misure sarebbero falsate. Per questo sono stati definiti i Boundary Clock che vengono sincronizzati da un master ed a loro volta si comportano da master in ogni porta a bordo per i vari dispositivi slave collegati.

Grazie all'utilizzo di questo algoritmo è possibile raggiungere sincronizzazioni tra dispositivi inferiori a  $1\ \mu\text{s}$ . [15]

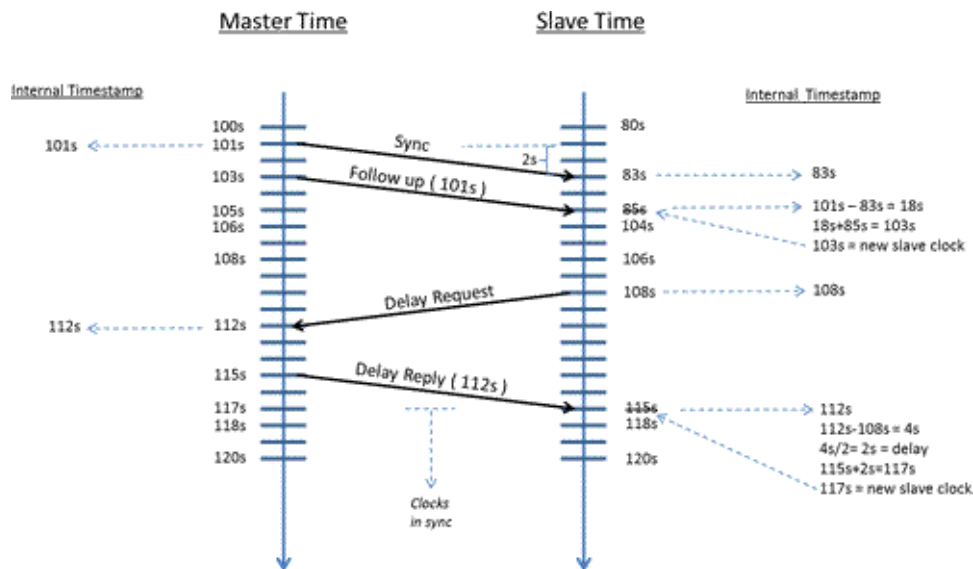


Immagine 2.2: Esempio di sincronizzazione dei clock tra due dispositivi

## 2.5 Industrial Ethernet

Come visto nella sezione precedente, nei sistemi di automazione industriale esistono differenti necessità di comunicazione per i differenti livelli applicativi. La comunicazione deve avere requisiti differenti a secondo del livello in cui viene utilizzata. I livelli di gestione e supervisione devono soddisfare esigenze diverse dai livelli di controllo. La comunicazione dei livelli gestionali gestisce dati strutturati con frequenza non elevata e senza vincoli temporali critici. Non sono presenti rilevanti problemi relativi ai disturbi e interferenze dato che i dispositivi vengono installati in un ambiente non ostile.

La comunicazione ai livelli più bassi, come quella di campo e di cella ha requisiti più restrittivi. I dati che vengono scambiati sono semplici e non strutturati, e vengono scambiati con una frequenza elevata. Dato che la comunicazione avviene tra task real-time, anche lo scambio di dati deve rispettare questi requisiti. La robustezza della rete a questi livelli deve soddisfare particolari specifiche relative a disturbi e interferenze in quanto i nodi della rete vengono installati in ambienti ostili.

La tradizionale tecnologia Ethernet utilizzata nei classici uffici non può essere utilizzata per comunicazioni industriali, ma necessita di qualche modifica alla tecnologia.

Industrial Ethernet permette di applicare le tecnologie di comunicazione fornite dai bus di campo su tecnologia Ethernet per soddisfare le specifiche richieste dal settore.

L'adozione della tecnologia Ethernet per comunicazioni industriali deve per prima cosa soddisfare le seguenti caratteristiche:

- affidabilità.
- robustezza (EMC, robustezza meccanica).

- comunicazione deterministica.
- azioni sincronizzate tra periferiche di campo decentralizzate.
- scambio di dati efficiente.

### 2.5.1 Real-Time Ethernet

Le reti real-time devono garantire determinate prestazioni per quanto riguarda lo scheduling e la verifica dei vincoli temporali, controllo del flusso e sincronizzazione della comunicazione.[12, 13]

Il sistema deve garantire che una elaborazione (*task*) termini entro un dato vincolo temporale (*deadline*). Un corretto sistema RT deve quindi garantire sia un funzionamento corretto del sistema ed una esecuzione puntuale e deterministica.

I sistemi Real-Time vengono suddivisi in due sottocategorie:

- **Soft Real-Time (SRT)**: sistemi in grado di definire una priorità tra eventi. Nel caso si superi la deadline viene provocato un danno non irreparabile.
- **Hard Real-Time (HRT)**: sistemi in grado di completare un'operazione critica in un tempo definito. Nel caso si superi temporalmente la deadline, potrebbero verificarsi situazioni catastrofiche nell'ambiente in cui il sistema opera (ad esempio: morte o danni ambientali).

I sistemi real-time sono costituiti da blocchi, detti *jobs*, dove ognuno di questi ha quantità temporali da rispettare:

- **Release Time**: tempo in cui un job viene reso disponibile al sistema per essere elaborato.
- **Ready Time**: tempo entro il quale il job viene completamente processato.
- **Execution Time**: intervallo tra Ready Time e il completamento dell'esecuzione.
- **Response Time**: tempo più breve in cui un job può iniziare l'esecuzione. Non può mai essere inferiore al Release Time.
- **Deadline**: tempo entro il quale l'esecuzione deve essere terminata, ovvero oltre il quale il job è in ritardo. La deadline può essere Hard o Soft, a seconda delle specifiche temporali del job.

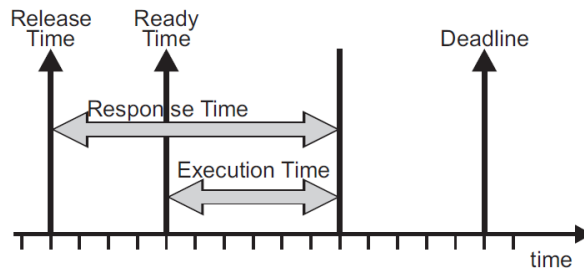


Immagine 2.3: Requisiti dei jobs

La definizione di una rete Real-Time che soddisfi pienamente tutte le specifiche, soprattutto per quelle HRT (Hard Real-Time), richiederebbe una grossa quantità di risorse fisiche, con inevitabile conseguenza di un elevato costo implementativo. Queste specifiche, vengono spesso rilassate definendo delle soglie di tollerabilità che devono essere rispettate dalla rete.

Gli algoritmi di gestione e trasmissione sono progettati e implementati su una soglia di ammissibilità di tipo probabilistico. Nel caso peggiore (per esempio negli istanti di massima congestione della rete), è tollerabile che una piccola percentuale di messaggi impliciti ed espliciti (messaggi periodici e sporadici) possa non arrivare in tempo. Questa specifica viene definita *Missed Packet Rate*.

Un'ulteriore specifica della rete è che non vi siano perdite di pacchetti durante la loro trasmissione. Un classico esempio di perdita di un pacchetto potrebbe verificarsi quando un buffer di un dispositivo ha raggiunto la sua capacità massima. Per evitare che questo accada, occorrerebbe sovradimensionare tali buffer, aumentando i costi senza risultati certi (solo buffer infiniti riuscirebbero a soddisfare questo vincolo). A tal proposito viene definita un'ulteriore ipotesi sulla comunicazione. Nessun task RT pronto per essere eseguito deve essere bloccato perché il buffer di un nodo della rete è esaurito. In pratica, questa specifica permette di inviare messaggi solo a task che sono pronti per essere eseguiti. I buffer scarteranno dei messaggi destinati a task non ancora pronti per l'esecuzione (i messaggi scartati vengono persi). L'algoritmo di gestione della rete deve garantire che la percentuale dei pacchetti persi, *Lost Packet Rate*, sia inferiore ad una soglia ammissibile.

Un ulteriore vincolo, è quello relativo al determinismo della rete. Viene richiesto che la comunicazione RT sia un processo predicibile e quindi deterministico. Viene inserito un indice di valutazione sulla varianza del ritardo di trasmissione, noto come delay jitter. Non è solo importante che il messaggio arrivi entro certi limiti, ma è necessario conoscere l'istante di arrivo del messaggio con determinismo.

Gli algoritmi di trasmissione e scheduling di queste reti sono progettati per limitare il delay jitter.

### 2.5.1.1 Normative

Prima di analizzare le specifiche in termini di prestazione che una rete RTE deve garantire, viene descritto il panorama normativo di riferimento.[9]

La norma IEC 61784-2: Addition profiles for ISO/IEC 8802-3 based communication networks in real-time, specifica i profili RTE. Questa norma aggiunge ulteriori profili di comunicazione (CP: Communication Profile) alle famiglie di profili esistenti (CPF: Communication Profile Families) specificate nella norma IEC 61784-1.

Questi profili soddisfano l'obiettivo del mercato dell'automazione industriale di identificare delle reti di comunicazione RTE che possano coesistere con le specifiche definite nella norma ISO/IEC 8802-3, ovvero lo standard per Ethernet.

I profili definiti permettono di dichiarare correttamente la conformità delle reti RTE con la norma ISO/IEC 8802-3 ed evitano la diffusione di implementazioni differenti.

I profili introdotti possono sfruttare i miglioramenti delle reti Ethernet in termini di larghezza di banda durante la trasmissione ed intervallo di rete.

Un ulteriore requisito essenziale è che le tipiche capacità di comunicazione Ethernet, utilizzate nel mondo degli uffici, vengono completamente mantenute, in modo che il software attualmente utilizzato non debba essere sostituito.

Gli indicatori di prestazione RTE, i quali valori vengono forniti per ogni dispositivo basato sui profili di comunicazione CP definiti, permettono all'utente di abbinare i dispositivi di rete con i requisiti richiesti dalla classe dell'applicazione.

La norma IEC 61784- è formata dalle seguenti parti:

- Parte 1: insieme dei profili per la produzione continua e discreta relativa all'utilizzo dei bus di campo nel controllo industriale.
- Parte 2: profili aggiuntivi per la norma ISO/IEC 8802-3 in applicazioni real-time.
- Parte 3: profili per funzionalità di sicurezza funzionale nelle reti industriali.
- Parte 4: profili per comunicazioni sicure in reti industriali.
- Parte 5: profili per la comunicazione di dati digitali derivati da misure e controllo e da profili di installazione.

Le norme IEC 61784-1 e IEC 61784-2 contengono diverse famiglie di profili di comunicazione (CPF) che specificano uno o più profili di comunicazione (CP). Questi profili identificano i sottoinsiemi definiti nella IEC 61158.

Un profilo di comunicazione può implementare o supportare uno o livello fisici (PhL).

I profili descritti nella IEC 61784-1 e IEC IEC 61784-2 sono stati sviluppati facendo riferimento alle norme IEC 61158-2 e IEC 61158-3-tt.

La norma IEC 61784-2 specifica i profili aggiuntivi per le reti di comunicazione in tempo reale per il settore dell'automazione industriale coesistenti con ISO/IEC 8802-3.

Queste reti di comunicazione RTE utilizzano direttive definite nello standard Ethernet tradizionale per gli strati inferiori dello stack di comunicazione fornendo un trasferimento dei dati in tempo reale più affidabile e prevedibile.

### 2.5.1.2 Indicatori di prestazione

I requisiti che una rete Ethernet RT deve soddisfare sono diversi a seconda dell'ambiente in cui opera. Questi requisiti sono stati specificati nella IEC 61784, e vengono comunemente definiti Performance Indicators (PI).[9]

Di seguito vengono brevemente riassunti.

- **Delivery time:** tempo necessario per trasferire un *Service Data Unit* (SDU) da un nodo sorgente ad un nodo destinatario. Questo tempo viene misurato a livello applicazione. Il calcolo deve essere fatto tenendo conto del tempo di trasmissione e di tutti i tempi di attesa. Il valore massimo dipende da due casi, uno in assenza di errori di trasmissione ed un secondo caso con perdita e recupero di un frame. Per calcolare il delivery time bisogna tenere in considerazione il tempo di trasmissione e tutti i tempi di attesa.
- **Numero di stazioni RTE:** ogni profilo di comunicazione (CP) impone un numero massimo di stazioni collegate alla rete. Un elevato numero di stazioni collegate in rete, porterebbe a tempi di attesa elevati, con conseguente perdita di prestazioni real-time.
- **Topologia della rete:** le differenti topologie implementabili influiscono sul traffico dei messaggi scambiati sulla rete. Le topologie tradizionali sono: stella, ad anello e lineare.
- **Numero di switch tra stazioni:** ogni CP impone un numero massimo di switch tra due nodi RTE della rete. Con questo parametro si può definire, in fase di progettazione, una possibile architettura della rete.
- **Throughput RTE:** quantità totale di dati APDU scambiati dall'applicazione in un secondo.
- **Banda non RTE:** percentuale di banda che può essere utilizzata per una comunicazione non real-time.
- **Accuratezza della sincronizzazione:** indica lo sfasamento temporale massimo che possono avere i clock di due diversi nodi nella stessa rete.
- **Precisione di sincronizzazione non basata sul tempo:** indica il ritardo massimo sulla comunicazione ciclica tra due nodi qualsiasi della rete, utilizzando l'attivazione di eventi periodici per avviare la trasmissione.
- **Tempo di recupero ridondante:** indica il tempo massimo che trascorre tra il fallimento di un singolo evento ed il ripristino completo del sistema, tenendo in considerazione che il fallimento sia permanente.

### 2.5.2 Aspetti implementativi

Come descritto nei paragrafi precedenti, ci sono diversi ambienti di utilizzo di una rete Ethernet Industriale. Per questo motivo, ci sono stati differenti approcci legati all'implementazione di Ethernet come rete di campo, che variano a seconda del tipo di prestazioni richieste.

Possiamo dividere i differenti approcci in tre distinte categorie.

#### 2.5.2.1 Categoria A: Soluzione basata su TCP/IP

Questa soluzione si basa sulla tecnica dell'incapsulamento. I pacchetti dati ad alto livello vengono inglobati in un pacchetto standard TCP o UDP.

In questo caso, Ethernet ed i livelli TCP/IP dello stack del protocollo di comunicazione (livelli ISO/OSI dal livello 1 al 4) vengono lasciati funzionare nel modo standard e la funzionalità viene aggiunta nel livello 7. I dati sono trasportati da TCP/IP utilizzando l'hardware Ethernet standard. Questo è il modo più semplice di utilizzare Ethernet come bus di campo. Per fornire un livello di determinismo accettabile, il controllo sul processo di comunicazione viene effettuato dalle funzioni di alto livello. Questo viene raggiunto sviluppando nuovi protocolli di livello 7 o aggiungendo funzionalità di ancora più alto livello al di fuori del modello ISO/OSI. Esempi di questa implementazione RTE sono Modbus TCP/IP, EtherNet/IP, P-NET, Vnet/IP e Foundation Fieldbus HSE.

#### 2.5.2.2 Categoria B: Ethernet MAC

Questa soluzione utilizza l'infrastruttura Ethernet standard al livello fisico e a livello MAC di collegamento, ma incapsula i dati direttamente, senza utilizzare la suite di protocolli TCP/IP. Questo abbassa il sovraccarico del pacchetto Ethernet e ne migliora ulteriormente il determinismo. In pratica, questa opzione si trova in genere insieme all'utilizzo di TCP/IP. Le applicazioni SRT possono utilizzare l'incapsulamento diretto al livello 2, ma lo stack TCP/IP consente funzionalità aggiuntive per la gestione della rete utilizzando strumenti standardizzati.

Questa opzione è parallela all'uso di TCP/IP. Applicazioni SRT possono utilizzare l'incapsulamento diretto, lasciando disponibile lo stack TCP/IP per funzionalità aggiuntive. Ethernet fornisce uno schema di prioritizzazione in base al quale il traffico SRT può essere incapsulato con una priorità più alta rispetto a quello TCP/IP.

Questo tipo di implementazione non alterano l'hardware Ethernet, ma vengono realizzate implementando uno speciale tipo di protocollo (Ethertype) nel frame Ethernet.

Alcuni esempi di queste implementazioni RTE sono Powerlink e Profinet RT (Class 1).

#### 2.5.2.3 Categoria C: Ethernet modificata

L'unico modo per garantire funzionalità IRT (Isochronous Real-Time, ovvero funzionalità Real-Time sincronizzate) con tempi di risposta e jitter inferiori al millisecondo è quello di cambiare il modo in cui funziona la stessa Ethernet. Ovviamente è possibile sviluppare una tecnologia completamente nuova, ma sono già disponibili tecnologie di



rete di controllo dei dispositivi in tempo reale.

In questa implementazione, viene mantenuta la connettività fisica di Ethernet (stessi cavi e infrastruttura di base), ma la funzionalità del protocollo Ethernet viene modificata direttamente nel livello 2 per fornire funzionalità di comunicazione deterministica. Questo richiede l'utilizzo di speciali interfacce hardware ASIC o FPGA. La compatibilità con protocolli di livello superiore come HTTP e SNMP può essere ottenuta incapsulando questi dati di livello superiore in fasce temporali definite.

Il determinismo viene implementato con una procedura ad istanti fissi di tempo che fornisce cicli di trasmissione prestabiliti ed attraverso una sincronizzazione ciclica tra tutti i nodi (ad esempio tramite lo standard IEEE 1588 PTP).

Esempi di questa implementazioni sono EtherCAT, Sercos III e Profinet IRT (Class 2 e 3).

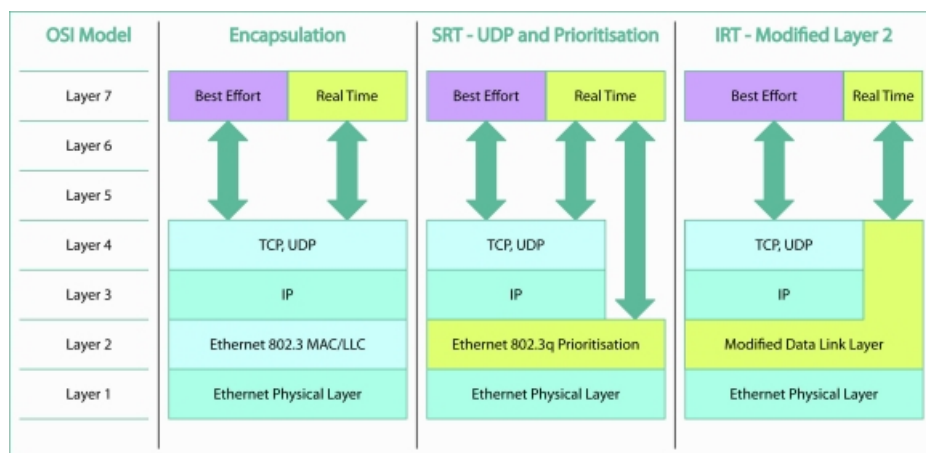


Immagine 2.4: Tre approcci per l'implementazione di protocolli industriali su Ethernet

## Capitolo 3

# EtherNet/IP

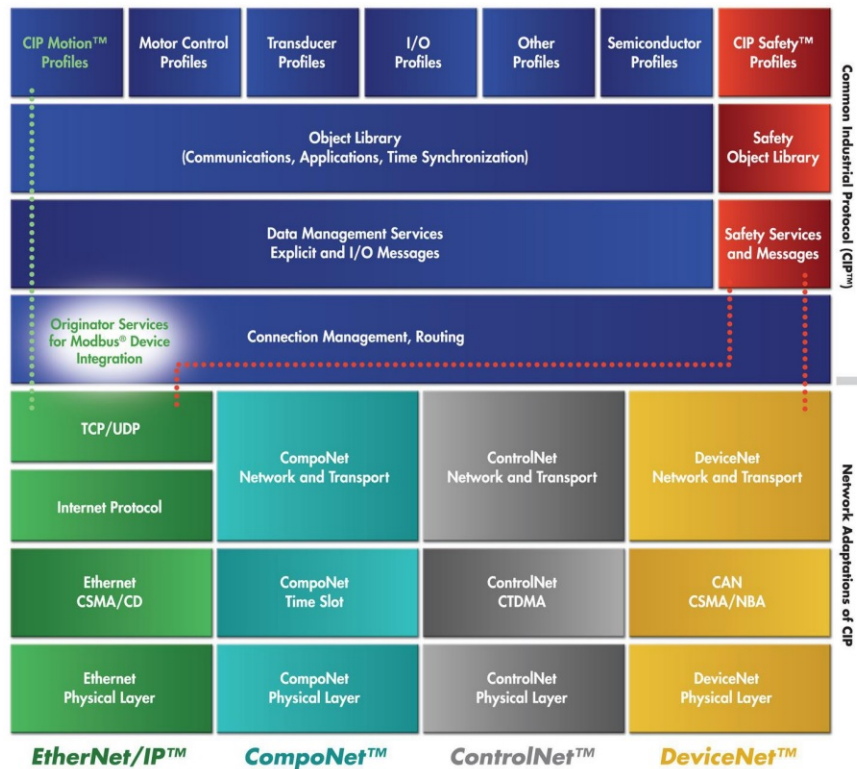


Immagine 3.1: Common Industrial Protocol

EtherNet/IP è un protocollo di rete industriale che utilizza il *Common Industrial Protocol* (CIP), un protocollo di livello applicazione indipendente dal livello di trasporto sottostante. Introdotto per la prima volta nel 1994 con la rete DeviceNet, è stato successivamente adottato dalle reti ControlNet nel 1997, EtherNet/IP nel 2001 e dalla rete CompoNet nel 2006.

In questa sezione ci concentreremo sull'analisi della versione del protocollo CIP adattato alla tecnologia Ethernet, che prende il nome di EtherNet/IP.

La standardizzazione e lo sviluppo del protocollo è supportata da ODVA (Open Device-Net Vendors Association), associazione fondata nel 1995, formata dai principali leader mondiali di aziende del settore. ODVA ha pubblicato le specifiche di queste tecnologie in una raccolta di sei volumi dal titolo *CIP Networks Library*.

Le specifiche per le reti CIP sono in continuo sviluppo per soddisfare e migliorare le richieste degli utilizzatori. ODVA utilizza un processo di gestione dello sviluppo per garantire specifiche stabili per tutte le reti CIP. Nuove edizioni per ogni specifica ODVA vengono pubblicate su base periodica.

Introduciamo ora le caratteristiche principali del protocollo CIP. [1, 16, 17, 18, 21]

### 3.1 CIP: Common Industrial Protocol

CIP comprende una suite completa di messaggi e servizi comuni in un applicativo industriale come controllo, sicurezza, sincronizzazione del movimento, configurazione e gestione della rete .

Il protocollo CIP consente agli utenti di integrare queste applicazioni con la rete Ethernet aziendale ed Internet, offrendo un'architettura unificata in tutta l'azienda.

Questo protocollo può offrire diversi vantaggi:

- integrazione di funzionalità di controllo, di I/O, configurazione di dispositivi e di trasferimento dati.
- implementazione di reti multi-layer senza necessità di bridge o proxy complessi.
- riduzione dei costi di investimento per lo studio del sistema, installazione e manutenzione.
- disponibilità da parte di differenti produttori con la conseguente possibilità di selezionare la migliore soluzione in termini di costi e prestazioni.
- realizzazione di una piattaforma unica e indipendente dal mezzo fisico in grado di connettere le diverse implementazioni del protocollo nelle diverse tecnologie senza necessità di modificare lo strato applicativo.

#### 3.1.1 Orientamento ad oggetti

CIP è un protocollo orientato agli oggetti, dove ogni nodo è definito come un insieme di oggetti. Un dispositivo reale può implementare uno o più nodi.

Gli oggetti all'interno di un nodo sono delle rappresentazioni astratte di un particolare componente o funzionalità del dispositivo; due oggetti identici implementati in due distinti dispositivi presentano lo stesso comportamento in modo da favorire l'interoperabilità sulla rete.

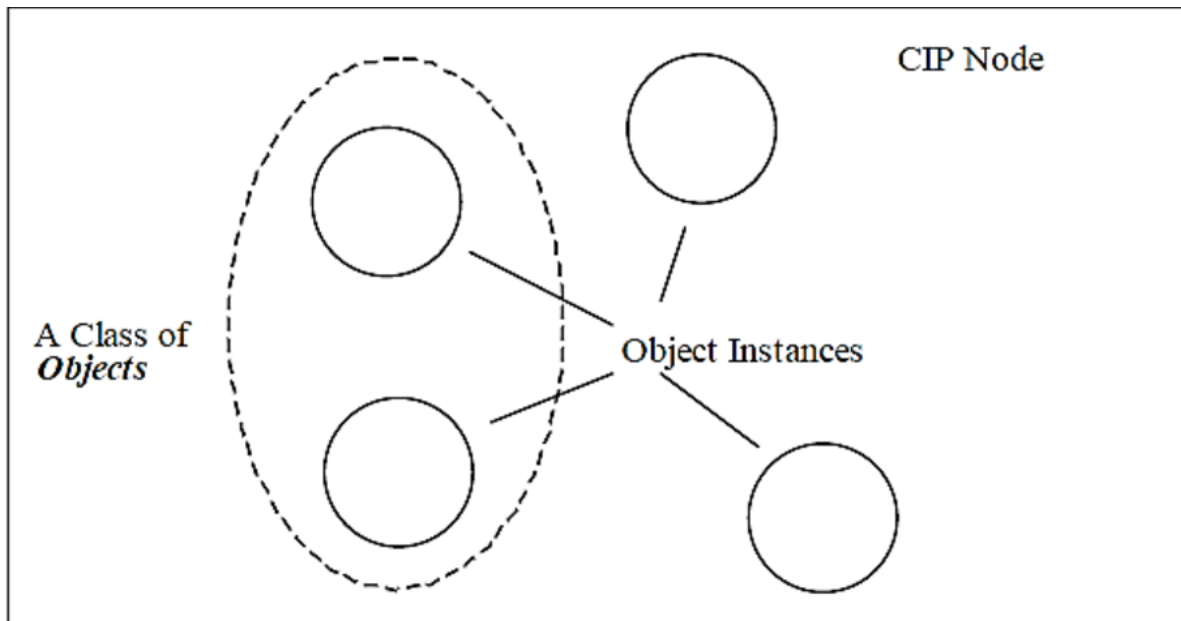


Immagine 3.2: Classe di oggetti

Gli oggetti sono strutturati in:

- **Classe:** è un insieme di oggetti che rappresentano tutti lo stesso tipo di componente. Sono previste classi di tipo pubblico o specifiche di un singolo produttore. Ogni specifico oggetto appartiene ad una determinata classe.
- **Istanza:** è la rappresentazione attuale di un particolare oggetto all'interno della classe. Ogni istanza di una determinata classe ha gli stessi attributi, ma contiene anche un insieme di attributi propri.
- **Attributi:** elementi di dati all'interno di un oggetto. Servono a descrivere le caratteristiche visibili esternamente. Gli attributi possono essere di classe o di istanza.

I servizi, invece, definiscono le azioni che un oggetto o una sua parte è in grado di eseguire quando viene indirizzato. Oltre alle operazioni elementari di scrittura e lettura sono previsti dei servizi base di protocollo che ogni nodo deve necessariamente implementare.

## MAC ID #4

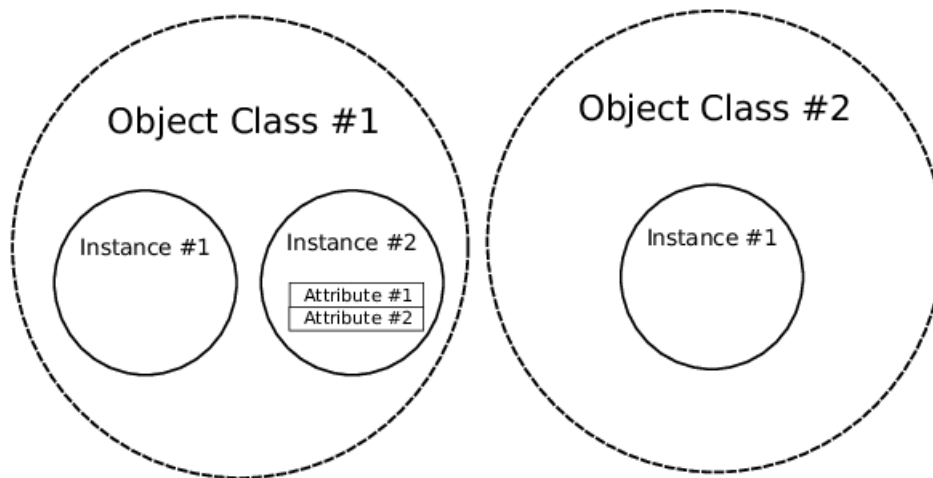


Immagine 3.3: Esempio di nodo nella rete CIP

Gli oggetti e i loro componenti sono indirizzati da uno schema di indirizzamento uniforme composto da:

- **Node Address:** valore numerico intero di identificazione assegnato a ogni nodo della rete. Su EtherNet/IP questo è rappresentato dall'indirizzo IP del nodo.
- **Class Identifier (Class ID):** valore numerico intero di identificazione assegnato a ciascuna classe di un oggetto accessibile dalla rete.
- **Instance Identifier (Instance ID):** valore numerico intero di identificazione assegnato ad un'istanza dell'oggetto che lo identifica tra tutte le istanze della stessa classe.
- **Attribute Identifier (Attribute ID):** valore numerico intero di identificazione assegnato ad un attributo di classe o ad un attributo di istanza.
- **Service Code:** valore numerico intero di identificazione che denota un'azione richiesta che può essere diretta ad una particolare istanza di un oggetto oppure ad un attributo contenuto in un oggetto.

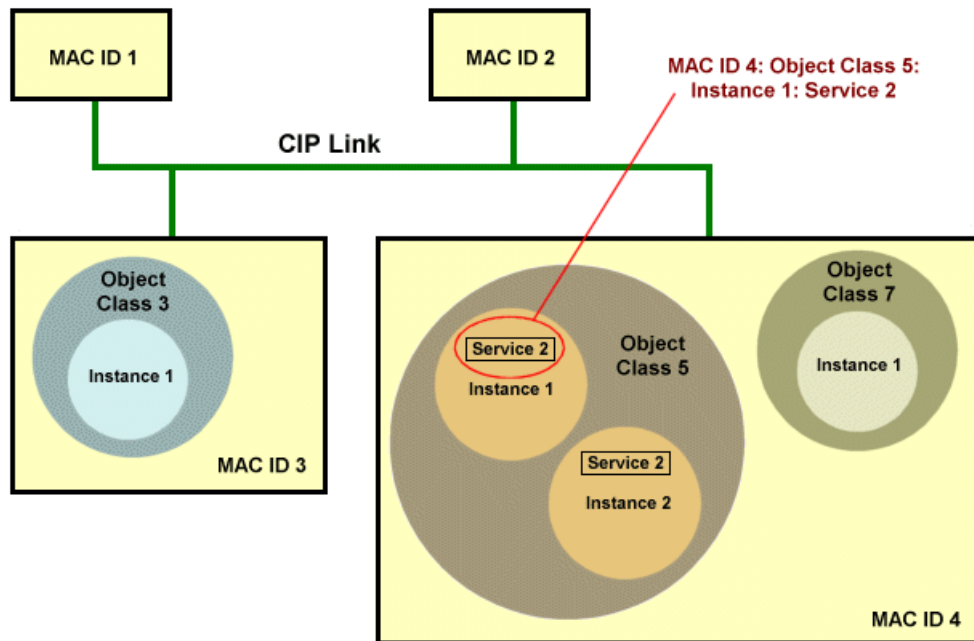


Immagine 3.4: Esempio di indirizzamento all'interno di un nodo

### 3.1.1.1 Messaggistica

Il protocollo di comunicazione previsto dallo standard CIP è orientato alla connessione e basato su uno schema *produttore/consumatore* intrinsecamente multi-cast, piuttosto che sorgente/destinatario, così da consentire un migliore utilizzo della banda disponibile sulla rete.

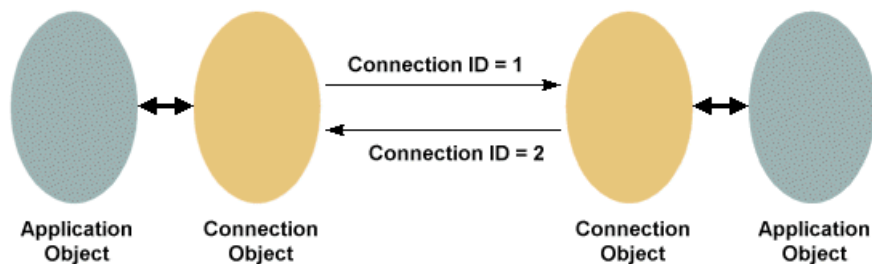


Immagine 3.5: Connessione CIP bidirezionale

Ad ogni connessione attiva sulla rete è associato un identificativo (*CID*) univoco in base al quale i nodi destinatari verificano la necessità di processare il messaggio. Se la connessione stabilita è bidirezionale, vengono assegnati due identificativi *CID*. La connessione viene stabilita da parte del nodo origine mediante invio di un messaggio di tipo *UCMM Forward\_Open*.

Questo messaggio include tutte le informazioni inerenti alla connessione richiesta tra nodo origine e destinatario.

La richiesta di Forward\_Open contiene le seguenti informazioni:

- Informazioni di time-out per la connessione.
- Identificativo CID della connessione tra nodo origine e destinatario.
- Identificativo CID della connessione tra nodo destinatario e origine.
- Informazioni sull'identità del mittente (costruttore e numero di serie).
- Dimensione massima dei messaggi trasferibili sulla connessione.
- Tipo di connessione: unicast o multicast.
- Meccanismi di trigger, ad esempio ciclo o con cambiamento di stato (COS).
- Chiave elettronica (opzionale) per permettere al nodo destinazione di verificare che sia di tipo corretto.
- Percorso di connessione per i dati dell'oggetto nel nodo che saranno prodotti e consumati.
- Segmento dati contenente informazioni per la configurazione del nodo (opzionale).
- Informazioni di routing se la rete si estende su più reti (opzionale).

Tutte le connessioni del protocollo CIP possono essere suddivise in due categorie:

- **Messaggi impliciti (Connessioni I/O):** comunicazioni tra il nodo produttore ed uno o più consumatori per la trasmissione di informazioni particolari, quali, ad esempio, controlli e comandi di attivazione. Questi messaggi possono essere unicast o multicast.  
Non sono contenute informazioni di servizio in quanto si presuppone che il nodo consumatore riconosca implicitamente l'azione da intraprendere in base all'identificativo CID della comunicazione.
- **Messaggi espliciti:** forniscono una comunicazione di tipo richiesta/risposta tra due dispositivi. Nella richiesta sono contenuti i dati che indicano quale servizio dell'oggetto nel nodo destinatario viene richiesto.

## 3.2 Communication Objects

Gli oggetti di comunicazione del protocollo CIP forniscono e gestiscono lo scambio dei messaggi. Questi oggetti sono unici in quanto sono il nucleo principale delle comunicazioni CIP. Ogni oggetto di questo tipo contiene un collegamento ad una parte del produttore, un collegamento ad una parte del consumatore o entrambi.

Le connessioni implicite possono produrre, consumare oppure produrre e consumare, mentre le connessioni esplicite producono e consumano sempre.

I valori degli attributi negli oggetti di connessione definiscono un insieme di attributi che rappresentano i parametri vitali della connessione. In particolare questi attributi specificano se si tratta di una connessione implicita I/O oppure di una connessione di messaggistica esplicita, la dimensione massima dei dati da scambiare, la sorgente e la destinazione dei dati.

Ulteriori attributi descrivono lo stato ed il comportamento della connessione, come i messaggi vengono attivati (ad esempio: l'applicazione attiva i messaggi dopo un cambiamento di stato oppure ciclicamente) e la tempistica delle connessioni (ad esempio: time-out associato alla connessione). Il protocollo CIP permette l'esistenza di più connessioni attive contemporaneamente all'interno di un dispositivo..

Di seguito vengono descritte i due differenti tipi di connessioni possibili.

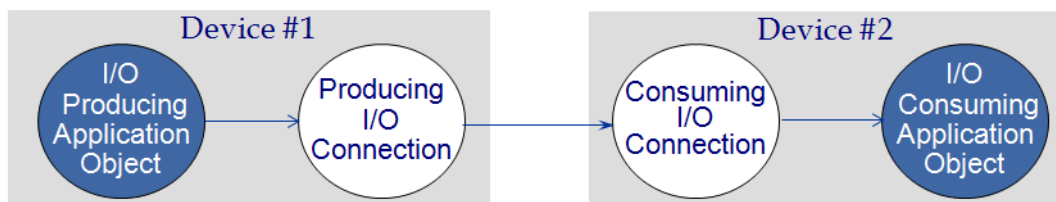


Immagine 3.6: Connessione I/O implicita

Nella connessione implicita la sorgente/destinazione dei dati è l'oggetto Application Object (ad esempio: Assembly object).

Questo trasferimento di dati è molto più efficiente rispetto alla messaggistica esplicita in quanto il significato dei dati è noto in anticipo e non vi è bisogno di interpretazioni.

Il trasferimento può essere avviato su base temporale (ad esempio: trigger ciclico) oppure con un intervallo di richiesta pacchetto (RPI).

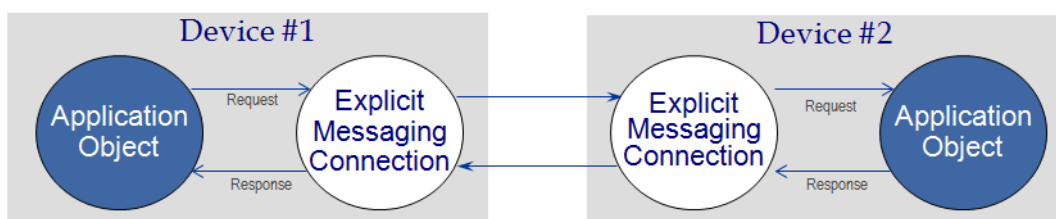


Immagine 3.7: Connessione con messaggistica esplicita

Nella connessione esplicita, l'oggetto connessione lato client riceve una richiesta da parte dell'oggetto applicazione. Il messaggio di richiesta viene inviato all'oggetto connessione lato server, che indirizza la richiesta al suo oggetto *Message Router*.



### 3.3 Libreria degli oggetti

La famiglia dei protocolli CIP contiene una vasta collezione di oggetti comuni suddivisi in tre categorie principali.

Gli oggetti ad uso generico possono essere utilizzati in tutti i tipi di dispositivi. Sono utilizzati per la parametrizzazione, identificazione, comunicazione e la gestione generale del dispositivo.

- |                                   |                                     |
|-----------------------------------|-------------------------------------|
| - Assembly (0x04)                 | - Message Router (0x02)             |
| - Acknowledge Handler (0x2B)      | - Originator Connection List (0x45) |
| - Connection (0x05)               | - Parameter (0x0F)                  |
| - Connection Configuration (0xF3) | - Parameter Group (0x10)            |
| - Connection Manager (0x06)       | - Port (0xF4)                       |
| - File (0x37)                     | - Register (0x07)                   |
| - Identity (0x01)                 | - Selection (0x2E)                  |

Immagine 3.8: Oggetti generici

Gli oggetti specifici dell'applicazione ricoprono gli aspetti fondamentali dei principali dispositivi ad uso industriale. Nuovi oggetti vengono aggiunti dal comitato ODVA.

Ci sono molti oggetti specifici di costruttori definiti dagli sviluppatori per soddisfare i bisogni che non vengono implementati dagli oggetti aperti esistenti contenuti nelle specifiche pubblicate. Gli oggetti specifici per la rete vengono utilizzati dai dispositivi che vengono integrati in reti con architettura particolare.

- |                                  |                                       |
|----------------------------------|---------------------------------------|
| - Base Switch (0x51)             | - Modbus Serial Link (0x46)           |
| - CompoNet Link (0xF7)           | - Parallel Redundancy Protocol (0x56) |
| - CompoNet Repeater (0xF8)       | - Power Management (0x53)             |
| - ControlNet (0xF0)              | - PRP Nodes Table (0x57)              |
| - ControlNet Keeper (0xF1)       | - SERCOS III Link (0x4C)              |
| - ControlNet Scheduling (0xF2)   | - SNMP (0x52)                         |
| - Device Level Ring (DLR) (0x47) | - QoS (0x48)                          |
| - DeviceNet (0x03)               | - RSTP Bridge (0x54)                  |
| - Ethernet Link (0xF6)           | - RSTP Port (0x55)                    |
| - Modbus (0x44)                  | - TCP/IP Interface (0xF5)             |

Immagine 3.9: Oggetti rete

Sebbene lo standard metta a disposizione un gran numero di oggetti, generalmente i dispositivi implementano solo un sottoinsieme di questi oggetti.

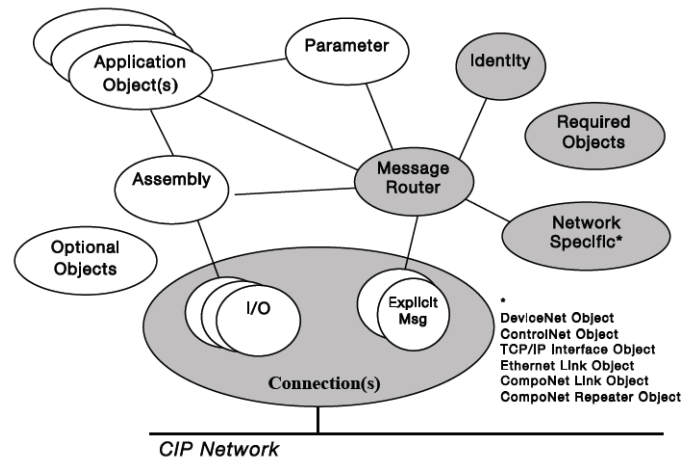


Immagine 3.10: Modello ad oggetti tipico di un dispositivo EtherNet/IP

Gli oggetti richiesti in un tipico dispositivo sono:

- Oggetto connessione (*Connection Object*) oppure oggetto gestione connessione (*Connection Manager Object*).
- Oggetto identità (*Identity Object*).
- Oggetto router messaggi (*Message Router Object*).

In base alle funzionalità del dispositivo da implementare, possono essere aggiunti oggetti specifici. Questa cosa consente a dispositivi semplici di non essere sovraccaricati di inutili oggetti. Gli sviluppatori solitamente utilizzano gli oggetti descritti in precedenza per implementare la composizione del dispositivo ma possono implementare oggetti personalizzati.

### 3.3.0.1 Identity Object (Class ID: 0x01)

Come descritto nella sezione precedente, ogni dispositivo deve avere un oggetto identità.

L'oggetto è formato dai seguenti attributi obbligatori:

- **Vendor ID:** identifica il costruttore del dispositivo (UINT).
- **Device Type:** specifica il profilo utilizzato per il dispositivo (UINT).
- **Product Code:** numero assegnato dal costruttore per identificare il prodotto all'interno del suo catalogo (UINT).
- **Revision:** identifica la revisione del dispositivo (n°.2 USINT).
- **Status:** fornisce informazioni sullo stato del dispositivo.

- **Serial Number:** numero identificativo univoco fornito dal costruttore.
- **Product Name:** fornisce il nome del dispositivo in formato ASCII.

In aggiunta a questi campi, possono essere presenti ulteriori campi non obbligatori come:

- **State:** descrive lo stato del dispositivo in modo meno dettagliato del campo Status (UINT).
- **Configuration Consistency Value:** permette di distinguere un dispositivo configurato correttamente da uno non configurato.
- **Heartbeat Interval:** abilita il messaggio Device Heartbeat con intervallo ciclico impostabile tra 0 e 255 secondi.
- **Supported Language List e International Product Name:** utilizzati per la descrizione del prodotto in diverse lingue.
- **Semaphore:** fornisce un semaforo per l'accesso sincronizzato al dispositivo da parte del client.
- **Assigned Name, Assigned Description e Geographical Location:** attributi impostabili dall'utilizzatore del prodotto per una corretta identificazione.
- **Modbus Identity Info:** fornisce informazioni di identificazione in formato Modbus.
- **Protection Mode:** indica se è presente una protezione sul dispositivo.

I servizi supportati dagli attributi di classe e di istanza su EtherNet/IP sono *Get\_Attributes\_All* e *Reset* che riavvia il dispositivo.

### 3.3.0.2 Parameter Object (Class ID: 0x0F)

L'oggetto parametro fornisce un metodo generale per consentire l'accesso a molti attributi di vari oggetti nel dispositivo senza utilizzare un tool (come un terminale) per scoprire quali oggetti sono implementati nel dispositivo.

Quando utilizzato questo oggetto è disponibile in due versioni: un oggetto completo ed una versione abbreviata. Gli attributi contenuti in questo oggetto sono:

- **Parameter Value:** parametro attuale.
- **Link Path Size:** due parametri che descrivono l'oggetto/istanza/attributo da cui è stato recuperato il valore del parametro.
- **Descriptor:** descrivono le proprietà del parametro (esempio: sola lettura).
- **Data Type:** descrive il tipo di dati utilizzando un meccanismo standard definito dal protocollo.
- **Data Size:** dimensione dati in byte.

### 3.3.0.3 Assembly Object (Class ID: 0x04)

Gli oggetti Assembly permettono di mappare i dati provenienti da attributi di diverse istanze di varie classi in un unico attributo (#3) di un oggetto Assembly.

Questa mappatura è utilizzata nello scambio di messaggi I/O impliciti per ottimizzare il flusso dei dati sulla rete.

Questo modello di mappatura consente di avere a disposizione i dati di processo da scambiare in un unico blocco, senza averli distribuiti tra diversi oggetti del dispositivo. Il protocollo CIP fa distinzione tra oggetti assembly di ingresso e uscita, che vengono considerati dalla prospettiva dell'elemento di controllo (PLC).

Un oggetto assembly di ingresso in un dispositivo colleziona dati in ingresso dal campo (ad esempio: sensori) e li produce sulla rete per essere consumati da un dispositivo di controllo o un interfaccia operatore.

Un oggetto assembly di uscita invece consuma i dati che il dispositivo di controllo invia sulla rete e li scrive nell'applicazione output (ad esempio per il controllo della velocità di un motore).

Possono essere presenti anche degli assembly di configurazione che vengono usati per trasmettere un insieme di parametri di configurazione, senza il bisogno di effettuare un singolo accesso ai vari parametri.

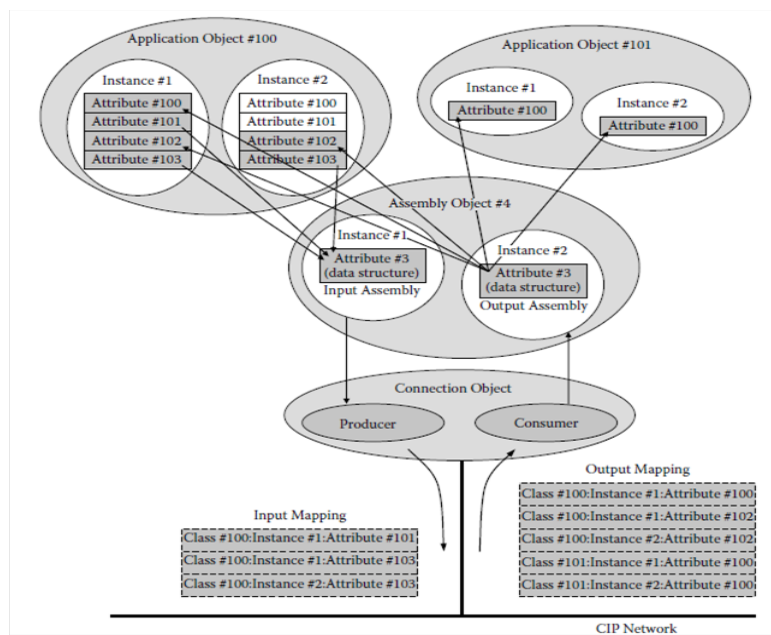


Immagine 3.11: Mappatura oggetti Assembly tipica di un dispositivo

In questa figura viene descritto un possibile esempio di utilizzo degli oggetti Assembly. I dati di processo degli oggetti 100 e 101 vengono mappati in due istanze dell'oggetto assembly. L'istanza 1 è configurata come assembly di ingresso mentre l'istanza 2 come assembly di uscita. Il blocco dati nel dispositivo è sempre accessibile puntando all'attri-

buto 3 della relativa istanza Assembly.

Questo modello di mappatura viene specificata per molti *Device Profiles* che verranno descritti nella sezione successiva. Vengono inoltre definiti oggetti Assembly statici o dinamici.

### 3.4 Device Profiles

Per questioni di riusabilità e semplificazione, dispositivi che implementano funzionalità comuni vengono raggruppati sotto un codice indentificativo ID a cui viene associato un profilo.

Questo profilo contiene una descrizione completa della struttura degli oggetti presenti nel dispositivo e del suo comportamento.

I costruttori del dispositivo devono utilizzare un identificativo univoco da associare al proprio prodotto. Tutti i dispositivi che non rientrano in un profilo elencato, devono implementare il profilo per dispositivi generici (0x2B) oppure un profilo specifico del costruttore.

- |  |   |
|--|---|
| - AC Drives (0x02)                           | - Limit Switch (0x04)                   |
| - CIP Modbus Device (0x28)                   | - Managed Ethernet Switch (0x2C)        |
| - CIP Modbus Translator (0x29)               | - Mass Flow Controller (0x1A)           |
| - CIP Motion Drive (0x25)                    | - Mass Flow Controller, Enhanced (0x27) |
| - CIP Motion Encoder (0x2F)                  | - Motor Overload Device (0x03)          |
| - CIP Motion I/O (0x31)                      | - Motor Starter (0x16)                  |
| - CIP Motion Safety Drive Device (0x2D)      | - Photoelectric Sensor (0x06)           |
| - Communications Adapter (0x0C)              | - Pneumatic Valve(s) (0x1B)             |
| - CompoNet Repeater (0x26)                   | - Position Controller (0x10)            |
| - Contactor (0x15)                           | - Process Control Valve (0x1D)          |
| - ControlNet Physical Layer Component (0x32) | - Programmable Logic Controller (0x0E)  |
| - DC Drives (0x13)                           | - Residual Gas Analyzer (0x1E)          |
| - DC Power Generator (0x1F)                  | - Resolver (0x09)                       |
| - Embedded Component (0xC8)                  | - RF Power Generator (0x20)             |
| - Encoder (0x22)                             | - Safety Analog I/O Device (0x2A)       |
| - Enhanced Mass Flow Controller (0x27)       | - Safety Drive Device (0x2E)            |
| - Fluid Flow Controller (0x24)               | - Safety Discrete I/O Device (0x23)     |
| - General Purpose Discrete I/O (0x07)        | - Softstart Starter (0x17)              |
| - Generic Device, keyable (0x2B)             | - Turbomolecular Vacuum Pump (0x21)     |
| - Human Machine Interface (HMI) (0x18)       | - Vacuum/Pressure Gauge (0x1C)          |
| - Inductive Proximity Switch (0x05)          |   |

Immagine 3.12: Profili dispositivi definiti dallo standard

#### 3.4.1 Descrizione del dispositivo

Il protocollo fornisce diverse modalità di configurazione per i dispositivi.

- Datasheet stampato.
- Oggetti parametro.
- Data Sheet Elettronico (EDS).
- Combinazione tra EDS e oggetti parametro.
- Insieme di configurazione combinato con le modalità descritte sopra.

Di seguito verrà approfondita la configurazione mediante data sheet elettronico essendo la più utilizzata.

Un EDS è file di testo contenente tutte le informazioni della struttura del componente. Il file può essere generato in automatico utilizzando un apposito strumento fornito da ODVA, EZ-EDS.

Le informazioni principali fornite sono le connessioni I/O per lo scambio di messaggi impliciti e quali parametri di configurazione sono presenti all'interno del dispositivo.

Viene riportata la struttura del file con la descrizione di ogni campo presente.

- **File:** descrive il contenuto e la versione del file.
- **Device:** equivale alle informazioni fornite dall'oggetto Identity e viene utilizzato per abbinare un EDS al dispositivo.
- **Device Classification:** descrive a quali reti il dispositivo può essere connesso.
- **ParamClass:** descrive i dettagli di configurazione oltre agli attributi a livello di classe Parameter Object.
- **Params:** identifica tutti i parametri di configurazione nel dispositivo, segue la definizione dell'oggetto Parameter Object.
- **Groups:** identifica tutti i gruppi di parametri nel dispositivo ed elenca il nome del gruppo ed i numeri dei parametri.
- **Assembly:** descrive la struttura degli elementi di dati.
- **Connection Manager:** Descrive le connessioni supportate dal dispositivo.
- **Connection ManagerN:** Descrive le connessioni supportate dal dispositivo che non si applicano alle porte CIP.
- **Port:** descrive le porte di rete che un dispositivo può avere.
- **Capacity:** specifica la capacità di comunicazione dei dispositivi.
- **Connection Configuration:** descrive le caratteristiche delle connessioni degli oggetti Connection Configuration implementati nel dispositivo.

- **Event Numeration:** associa un evento o uno stato del dispositivo ad una stringa.
- **Modbus Mapper:** utilizzato per fornire una descrizione dei singoli elementi Modbus che corrispondono agli attributi CIP.
- **Object class selections:** queste sezioni, una per ogni classe oggetto, possono essere usate per descrivere tutti i dettagli degli oggetti, come istanze, attributi e servizi supportati.

### 3.4.2 Routing

CIP definisce meccanismi che permettono l'instradamento dei messaggi su più reti distinte, sotto la condizione che i dispositivi intermedi (router) tra le reti siano dotati di oggetti e servizi utilizzati nell'architettura CIP. Per i messaggi espliciti non connessi il messaggio da spedire al dispositivo di destinazione viene racchiuso all'interno di un ulteriore servizio di messaggista esplicita chiamato *Unconnected\_Send* (Service code 0x52 dell'oggetto Connection Manager) che contiene tutte le informazioni riguardanti il servizio di trasporto (time-out di connessione, percorso di routing, percorso di richiesta del messaggio).

Il primo router che riceve un messaggio *Unconnected\_Send* estrae il suo contenuto e lo inoltra al prossimo router, come specificato nella sezione Route Path del messaggio.

Il router può sottrarre del tempo dal valore di time-out, riducendone così il valore. Questo processo viene eseguito per ogni router attraversato dal messaggio fino al raggiungimento del router finale.

Una volta che il messaggio *Unconnected\_Send* è arrivato al router di destinazione, il messaggio viene estratto e passato al dispositivo di destinazione, il quale esegue il servizio richiesto e produce una risposta che viene inoltrata al nodo di origine, attraversando in senso contrario il percorso di routing.

Per i messaggi impliciti di I/O, l'instradamento su reti diversi viene abilitato impostando il servizio *Forward\_Open*.

A differenza dei messaggi espliciti, questo tipo di connessione non utilizza nessuna informazione sul percorso di routing, dal momento che questi messaggi sono connessi e utilizzano sempre lo stesso percorso per ogni connessione.

Una caratteristica da notare è che i router che implementano il protocollo CIP non hanno bisogno di nessuna preconfigurazione.

### 3.4.3 Gestione dei dati

L'indirizzamento dei dati viene effettuato tramite segmenti, utilizzati per puntare ad oggetti e attributi nel dispositivo.

La struttura tipica di un segmento è: *[classID][instanceID][attributeID]*.

Ogni elemento di questa struttura può essere di 1 byte, 2 byte oppure 4 byte.

Oltre ai segmenti, il protocollo permette l'utilizzo di tipi di dato che possono essere

strutturati come matrici.

Di seguito viene elencata una lista delle strutture dati più utilizzate.

- 1 bit (trasformato in 1 byte)
  - Boolean, BOOL, Type Code 0xC1;
- 1 byte
  - Bit string, 8 bits, BYTE, Type Code 0xD1;
  - Unsigned 8-bit integer, USINT, Type Code 0xC6;
  - Signed 8-bit integer, SINT, Type Code 0xC2;
- 2 byte
  - Bit string, 16-bits, WORD, Type Code 0xD2;
  - Unsigned 16-bit integer, UINT, Type Code 0xC7;
  - Signed 16-bit integer, INT, Type Code 0xC3;
- 4 byte
  - Bit string, 32 bits, DWORD, Type Code 0xD3;
  - Unsigned 32-bit integer, UDINT, Type Code 0xC8;
  - Signed 32-bit integer, DINT, Type Code 0xC4.

### 3.5 EtherNet/IP: protocollo CIP su Industrial Ethernet

Il protocollo EtherNet/IP, seguendo il modello ISO/OSI, estende l'applicazione di TCP/IP dell'impianto, potendo coesistere con qualsiasi altro protocollo del livello di trasporto TCP/UDP standard.

Utilizzando il protocollo CIP visto nella sezione precedente con lo standard Ethernet, è possibile ottenere una soluzione Ethernet non modificata che consente di mettere in comunicazione la rete a livello macchina con la rete aziendale.

Viene definito un meccanismo di incapsulamento dei messaggi impliciti I/O ed espliciti all'interno di un frame Ethernet.

I messaggi espliciti vengono spediti appoggiandosi sul protocollo TCP/IP mentre i messaggi impliciti sfruttano il protocollo UDP/IP.

Per poter utilizzare Ethernet nelle comunicazioni industriali, bisogna soddisfare i requisiti di determinismo, richiesti dalle applicazioni.

Utilizzando i corretti dispositivi con scambio dati ad elevata velocità su comunicazioni full-duplex, questo requisito viene soddisfatto, annientando le collisioni e le perdite di pacchetti.

Di seguito vengono descritte alcune delle principali caratteristiche del protocollo:



- Indipendente dalla velocità di trasmissione 10, 100, 1000 Mbps.
- I sistemi possono essere costruiti con un'infrastruttura standard.
- Numero virtualmente illimitato di nodi in una rete.
- Le reti possono essere strutturate in sottoreti con router IP.
- Pieno supporto della comunicazione tra sottoreti poiché utilizza l'indirizzamento IP per tutte le comunicazioni.
- Comunicazioni real-time e non possono coesistere sulla stessa sottorete.
- Supporto per Device Level Ring (DLR) che fornisce una tolleranza di errore singola attraverso la ridondanza dei supporti.
- Coesistenza con altri protocolli di livello superiore come HTTP, FTP, VOIP.

### 3.5.1 Livello fisico

Per poter utilizzare la tecnologia di Ethernet in un applicativo industriale devono essere soddisfatti diversi requisiti richiesti da questo ambiente per quanto riguarda l'isolamento, la messa a terra e la costruzione dei cavi.

Per questo motivo vengono definiti due livelli di prestazione per quanto riguarda l'applicazione del protocollo, che verranno elencati di seguito:

- **COTS:** questo livello di applicazione (Commercial Off The Shelf) fornisce una connessione base, permettendo l'utilizzo di connettori RJ45 standard, con una lunghezza dei cavi per un massimo di 100 metri.
- **Livello industriale:** livello applicativo che prevede l'utilizzo di componenti più performanti come connettori RJ45 ad alte prestazioni.

Per quando riguarda la conformazione del cavo viene richiesta una categoria Cat 5E o Cat 6 con o senza schermatura.

Ulteriori approfondimenti riguardo al livello fisico vanno oltre lo scopo di questa tesi. Il consorzio ODVA ha pubblicato una serie di linee guida per approfondire lo strato fisico del protocollo.

### 3.5.2 Struttura dei frame

Come descritto all'inizio del capitolo, EtherNet/IP si basa sulle tecnologie esistenti TCP/IP e UDP/IP, sfruttandole senza apportare modifiche.

Il protocollo TCP viene utilizzato per lo scambio di messaggi espliciti mentre UDP per lo scambio dei dati di processo.

Le connessioni per la messaggistica esplicita su TCP/IP vengono stabilite sulla porta 44818 (0xAF12). Questa porta è utilizzata sia per le comunicazioni che necessitano di

una connessione che per quelle non connesse. Ogni dispositivo supporta almeno due connessioni TCP su questa porta.

Le connessioni di messaggistica implicite di I/O che sfruttano il protocollo UDP avvengono sulla porta 2222 (0x08AE). Questi messaggi possono essere spediti in modalità unicast o multicast garantendo un flusso dei dati più efficiente.

Tutti i messaggi che transitano su questa rete sono incapsulati in un frame Ethernet con intestazione IP e trasporto (TCP o UDP). Il pacchetto EtherNet/IP è formato da un intestazione ed una parte di dati che vengono incapsulate nel frame Ethernet spedito.

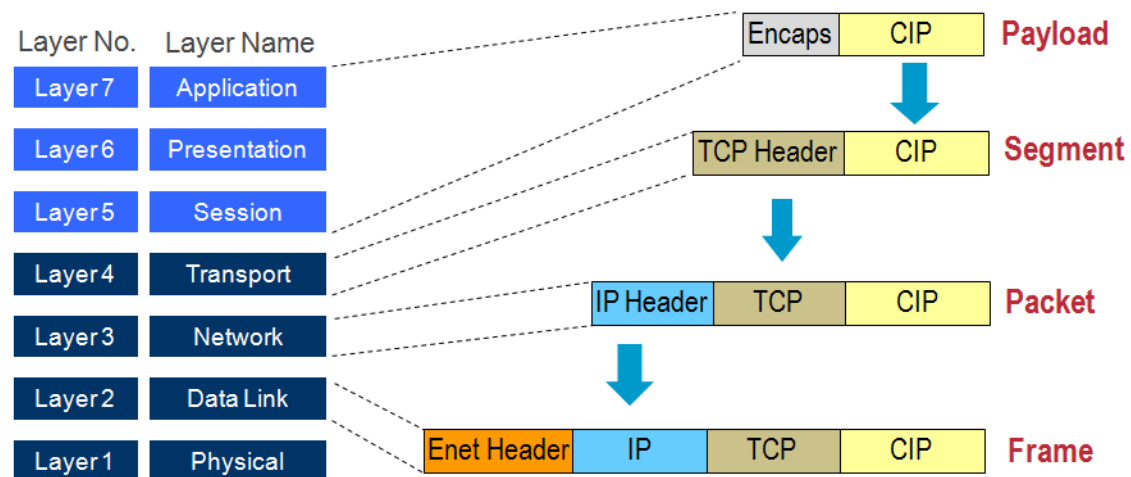


Immagine 3.13: Relazione tra frame CIP e Ethernet

Di seguito ci soffermiamo sull'analisi dei campi che compongono la parte di intestazione e di dati del pacchetto EtherNet/IP.

La parte di intestazione contiene i comandi che determinano cosa deve essere fatto con la parte contenuta nel campo dati.

Molti comandi specificano l'uso del *Common Packet Format*. I messaggi impliciti di I/O spediti con un frame UDP non utilizzano il campo di intestazione, ma seguono il Common Packet Format.

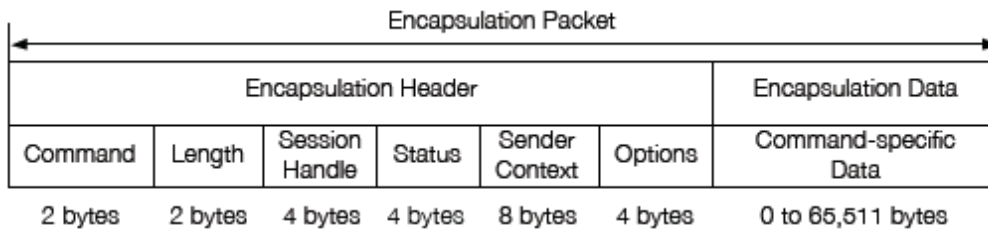


Immagine 3.14: Pacchetto CIP incapsulato nel frame Ethernet

Il campo *Command* può contenere uno dei seguenti comandi:

- **ListIdentity:** questo comando viene inviato in broadcast in UDP per comunicare a tutti i dispositivi EtherNet/IP connessi in rete di inviare un set di dati con informazioni di identità. Tipicamente è utilizzato da strumenti software che analizzano la rete.
- **RegisterSession/UnRegisterSession:** questi due comandi vengono utilizzati per aprire e chiudere una sessione di incapsulamento tra due dispositivi. Una volta che la sessione è stata realizzata, lo scambio di messaggi può iniziare. Tra due dispositivi può esistere solamente una sessione. Il dispositivo che riceve la richiesta di RegisterSession crea un *Session Handle* che ritorna al dispositivo richiedente in risposta alla RegisterSession. Questo valore è utilizzato per identificare i messaggi inviati tra i due dispositivi all'interno di questa sessione.
- **SendRRData/SendUnitData:** questi comandi vengono utilizzati nell'invio di messaggi espliciti. SendRRData è utilizzato per l'invio di messaggi espliciti senza connessione, mentre SendUnitData per quelli connessi. Il dispositivo che trasmette la richiesta SendRRData crea un valore nel campo *Sender Context* che viene restituito con la risposta. SendUnitData non utilizza invece questo campo.
- **Common Packet Format:** costruito che fornisce un meccanismo per strutturare il campo *Encapsulation Data* per quei comandi che specificano dei dati di incapsulamento. Definisce le informazioni da scambiare tra i dispositivi.

Nell'ultimo capitolo di questo elaborato verrà effettuata un'analisi dettagliata dei campi che compongono i pacchetti che transitano in rete utilizzando Wireshark.

### 3.5.2.1 Incapsulazione dati

Come descritto precedentemente, i messaggi espliciti, possono essere connessi o non connessi. Le connessioni EtherNet/IP tra nodi vengono stabilite utilizzando un messaggio *UCMM Forward\_OpenMessage*.

Avere dei messaggi espliciti connessi è importante quando le richieste al nodo vengono inviate in maniera periodica. Tramite una connessione è possibile ottenere risposte più

tempestive. Quando le richieste avvengono raramente ed in maniera irregolare è preferibile utilizzare la messaggistica senza connessione, che permette di non sovraccaricare la rete inutilmente.

I messaggi espliciti vengono inviati con un intestazione TCP/IP utilizzando un identificativo per la parte di comando differente in base all'utilizzo o no della connessione. In particolare:

- **SendRRData Encapsulation:** comando per messaggio esplicito senza connessione.
- **SendUnitData Encapsulation:** comando per messaggio esplicito con connessione.

Per i messaggi impliciti di I/O inviati con un intestazione UDP/IP non è richiesta un'intestazione per l'incapsulazione nel frame Ethernet ma viene utilizzato il Common Packet Format.

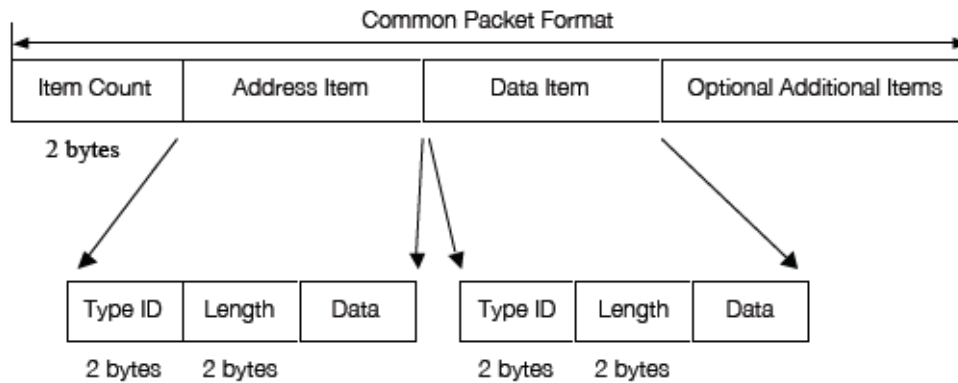


Immagine 3.15: Esempio di Common Packet Format

Di seguito viene riportata una rappresentazione della composizione del Common Packet Format per lo scambio di messaggi impliciti.

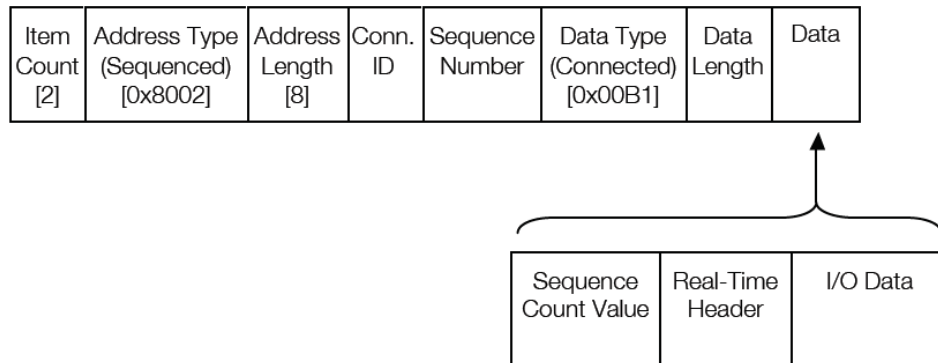


Immagine 3.16: Incapsulazione messaggio I/O

Il campo *Data* contiene un valore di conteggio sequenza per la trasmissione dei pacchetti. Il campo *Real-Time Header* può essere utilizzato per indicare il Run/Idle di un dispositivo. Infine il campo *I/O Data* contiene i dati di processo da scambiare. I messaggi impliciti vengono inviati dalla sorgente alla destinazione in UDP con modalità unicast. I messaggi dal destinatario alla sorgente possono essere inviati in unicast o multicast. I frame multicast permettono agli altri dispositivi EtherNet/IP connessi sulla rete di ascoltare i dati scambiati.

### 3.5.3 Connessione rapida

Mentre la maggior parte dei dispositivi posso attendere per diversi secondi prima di stabilire una connessione, esistono dei campi applicativi in cui è richiesto che il dispositivo entri immediatamente in funzione una volta alimentato elettricamente.

Per soddisfare questi requisiti, sono stati sviluppati diversi meccanismi per ottenere una connessione rapida.

Per una descrizione dettagliata di queste tecniche si rimanda a documentazione specifica descritta dallo standard ODVA.

## 3.6 Classi di dispositivi

All'interno del protocollo CIP sono state definite 4 classi di dispositivi, ovvero 4 principali funzionalità che un dispositivo può offrire.

- Dispositivo elementare con funzionalità di server per messaggi espliciti, utilizzato per la messaggistica esplicita senza collegamento (opzionalmente con collegamento) per upload/download della configurazione, monitoraggio dello stato e raccolta dei dati di processo.

- Dispositivo I/O Server, che aggiunge al server per la messaggistica esplicita, il supporto alla messaggistica implicita. Questi dispositivi vengono chiamati *I/O Adapters*. (Variatori di frequenza, dispositivi di I/O).
- Dispositivo client per i messaggi espliciti, che dialoga con un server di messaggistica esplicita, e si pone verso la rete come origine e destinazione per messaggi impliciti. (IPC, HMI, interfacce di rete).
- Dispositivo completo di tutte le funzionalità, che si pone verso la rete come produttore di messaggi impliciti e che funge da origine e destinazione per i messaggi espliciti. Questi dispositivi vengono chiamati *I/O Scanner* (il classico esempio è il PLC).

### 3.7 Requisiti per il supporto TCP/IP

Oltre ai requisiti richiesti dalle specifiche di EtherNet/IP tutti i dispositivi devono essere dotati delle funzionalità della suite dei protocolli TCP/IP.

Tutti i dispositivi devono supportare:

- Internet Protocol (IPV4 RFC791).
- User Datagram Protocol (UDP RFC 768).
- Transmission Control Protocol (TCP RFC 793).
- Address Resolution Protocol (ARP RFC826).
- Internet Control Message Protocol (ICMP RFC792).
- Internet Group Management Protocol (IGMP RFC 1112).
- IEEE 802.3 (Ethernet RFC 894).

I dispositivi possono supportare altri protocolli basati su Ethernet come HTTP, Telnet, FTP, ecc.

Essendo basato su TCP e UDP questo protocollo può essere integrato senza problemi con gli altri protocolli e servizi offerti dalla rete.

Il protocollo CIP può essere inserito in rete, fornendo un servizio aggiuntivo al sistema esistente senza indebolirlo.

L'immagine successiva illustra le relazioni tra CIP e i tipici protocolli Ethernet.

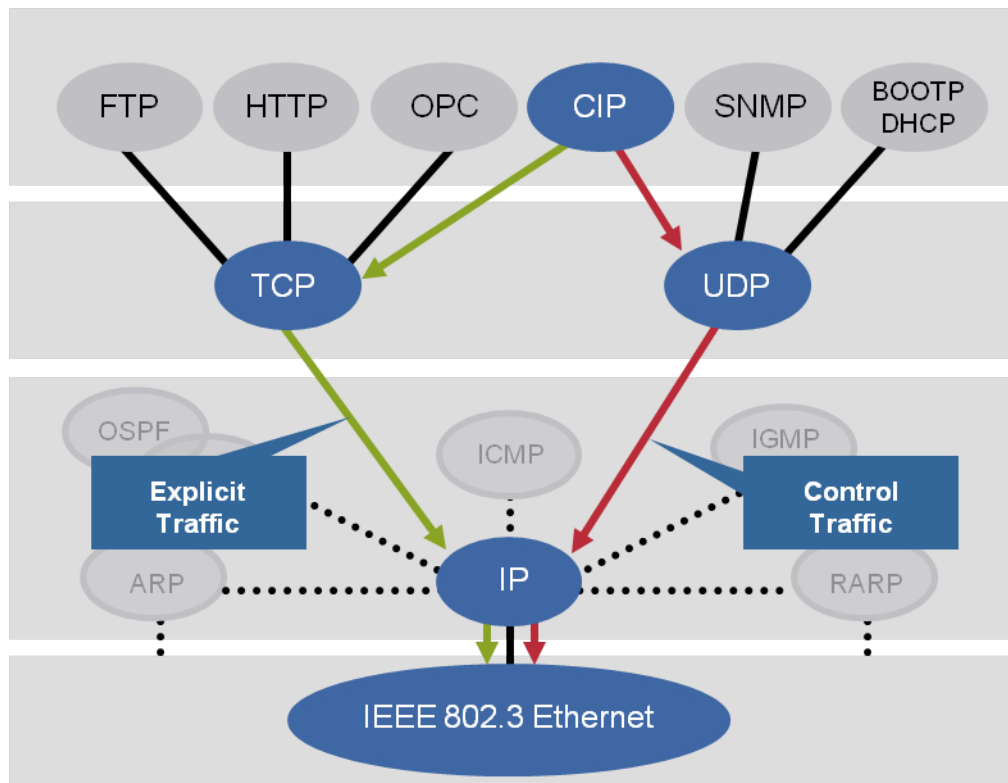


Immagine 3.17: Relazione tra protocollo CIP e i tipici protocolli Ethernet

### 3.8 Architettura Ethernet

Per applicare con successo EtherNet/IP al mondo dell'automazione, è necessario prendere in considerazione il problema del determinismo. Questo vincolo è richiesto in quanto lo scambio di pacchetti in rete deve consegnare i pacchetti in tempo reale, ovvero non sono ammessi ritardi e ritrasmissioni.

Una mancata violazione del vincolo di determinismo potrebbe portare, considerando come esempio due bracci sincronizzati che eseguono movimenti interpolati su due assi X e Y, ad un errore di traiettoria, con pericolo di danni a cose e/o a persone.

Per questo il normale funzionamento della gestione delle collisioni rilevate dai nodi e della ritrasmissione dei pacchetti utilizzata da Ethernet deve essere modificato, perché tale metodologia non garantisce il rispetto di questi vincoli.

Per prima cosa, non possono essere utilizzati normali switch da ufficio, ma devono essere sostituiti con dispositivi più intelligenti che inoltrano solo i frame Ethernet destinati ai nodi connessi sulle varie porte.

Inoltre utilizzando switch con tecnologia Wire-Speed Switching Fabric e switch full-duplex, le collisioni vengono eliminate.

Se più messaggi vengono inviati sullo stesso nodo nello stesso momento, invece di collidere

vengono accodati all'interno dello switch e consegnati in sequenza.

Come descritto nelle sezioni precedenti, è consigliato l'utilizzo di switch che supportano la funzionalità di snooping IGMP.

Molto spesso capita di collegare la rete di controllo con i sistemi gestionali aziendali per il monitoraggio della produzione e dei dispositivi. In questo caso, la rete EtherNet/IP e quella aziendale generale, devono essere separate da un router. Questo impedisce il congestionamento della rete di controllo EtherNet/IP dal traffico broadcast e multicast in arrivo dalla rete ufficio e permette ai pacchetti UDP in transito sulla rete di controllo di arrivare alla rete ufficio.

Inoltre il router può essere configurato per poter far passare i pacchetti TCP per l'invio di messaggi espliciti da un sistema gestionale aziendale verso i dispositivi EtherNet/IP (esempio: sistema SCADA) in modo da consentire ad un PC posizionato in un ufficio di poter monitorare e configurare i dispositivi.

### 3.8.1 Topologia di rete

In questa sezione viene fornita una descrizione delle topologie di rete che possono essere implementate su EtherNet/IP.

Tipicamente i dispositivi delle rete hanno due porte Ethernet, e questo permette di implementare topologie lineari oppure ad anello.

Con la topologia lineare un guasto ad un nodo o ad un collegamento comporta una perdita di tutti i nodi collegati dopo il punto di guasto.

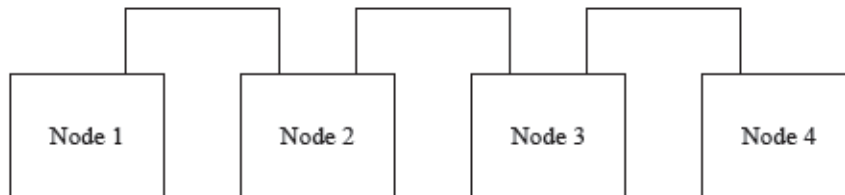


Immagine 3.18: Topologia di rete lineare

Questa cosa può essere evitata utilizzando un protocollo (DLR) implementato nei dispositivi terminali della rete con una topologia ad anello, permettendo di configurare la rete in modo che un guasto in un singolo punto non interrompa la comunicazione con il resto dei dispositivi funzionanti.

Il protocollo *Device Level Ring* (DLR) consente di identificare un guasto sulla rete con topologia ad anello e riconfigurarla in modo da isolare il nodo interessato, consentendo di riprendere con la trasmissione dei pacchetti. Il protocollo DLR opera al livello 2 dello stack ISO/OSI. La configurazione del DLR viene implementata nel protocollo CIP utilizzando un determinato oggetto Device Level Ring (0xF6). Esistono diverse implementazioni del DLR, descritte di seguito.



- **Ring Supervisor:** questa classe di dispositivi fa da nodo supervisore dell'anello. Questi dispositivi devono implementare i requisiti di comportamento richiesti per un supervisore come l'invio ed analisi di frame *Beacon* e frame *Announce*.
- **Ring Node, Beacon-based:** questa classe di dispositivi implementa il protocollo DLR senza la capacità di essere supervisore dell'anello. Il dispositivo deve essere in grado di elaborare e rispondere ai frame *Beacon* inviati dal nodo supervisore.
- **Ring Node, Announce-based:** questa classe di dispositivi implementa il protocollo DLR senza la capacità di essere supervisore dell'anello. Questi dispositivi non hanno la capacità di elaborare i frame *Beacon* ma una volta ricevuti li inoltrano sull'altra porta. Utilizzano i frame *Announce* per indicare lo stato dell'anello.

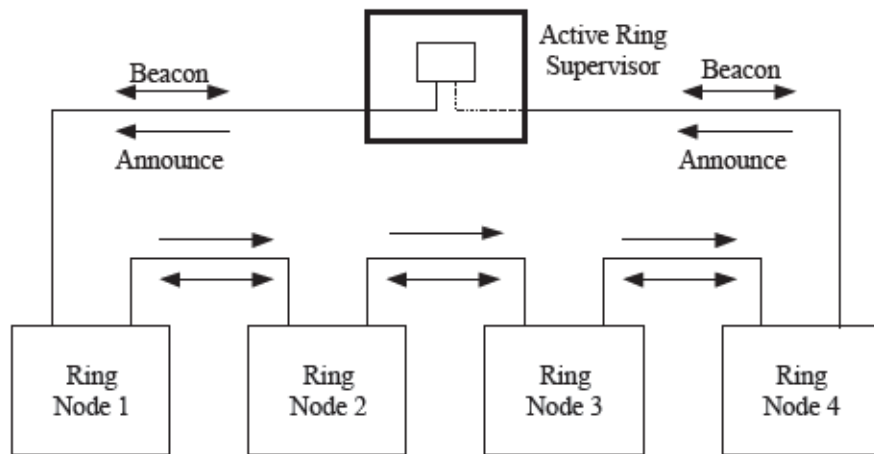


Immagine 3.19: Topologia di rete ad anello con DLR

Durante il normale funzionamento, il nodo DLR supervisore invia i frame speciali Beacon e Announce per identificare un guasto sui nodi dell'anello e ripristinarlo.

Ogni nodo della rete ha 2 porte con integrato uno switch per l'inoltro dei pacchetti ai nodi interessati. Il supervisore dell'anello invece blocca il traffico su una delle due porte, evitando così un loop dei pacchetti sulla rete e definendo un unico percorso che i pacchetti seguono sulla rete. Questo supervisore ad intervalli regolari invia un frame Beacon e Announce su entrambe le porte. Questi due frame permettono di identificare la presenza di guasti (ad esempio: interruzione dell'alimentazione o del collegamento). Quando viene rilevato un guasto, il supervisore riconfigura la rete comunicandolo ai dispositivi e aggiornando le tabelle MAC unicast di inoltro dei pacchetti.

L'utilizzo della topologia ad anello insieme al protocollo DLR permette di avere tempi di risposta ai guasti molto brevi, anche con un elevato numero di nodi. Di seguito vengono elencate queste prestazioni.

Number of Ring Nodes	Beacon Interval, $\mu$ s	Round Trip Time, $\mu$ s	Beacon Timeout, $\mu$ s	Physical Layer Faults Recovery Delay, $\mu$ s	Non-physical Layer Faults Recovery Delay for Beacon Frame Based Nodes, $\mu$ s	Non-physical Layer Faults Recovery Delay for Announce Frame Based Nodes, $\mu$ s	Ring Restore Delay for Beacon frame Based Nodes, $\mu$ s	Ring Restore Delay for Announce Based Nodes, $\mu$ s
25	400	905	1380	980	1858	2335	1330	2260
50	400	1810	1960	1885	2890	3820	2235	4070
100	400	3620	3120	3695	4955	6790	4045	7690
150	400	5430	4280	5505	7020	9760	5855	11310
200	400	7240	5440	7315	9085	12730	7665	14930
250	400	9050	6600	9125	11150	15700	9475	18550

Immagine 3.20: Parametri e prestazioni della configurazione DLR

Oltre al protocollo DLR anche il *Rapid Speed Spanning Protocol* (RSTP) può essere implementato su EtherNet/IP per la risposta ai guasti della rete. Questo protocollo è spesso utilizzato in reti con topologia ad albero dove i vari nodi sono

### 3.9 Oggetti EtherNet/IP aggiuntivi

Nella precedente sezione sulla descrizione del protocollo CIP è stata fornita una panoramica sui principali oggetti che compongono la maggior parte dei comuni dispositivi che si possono trovare sul campo. Di seguito vengono elencati gli oggetti che possono essere aggiunti ai tradizionali oggetti CIP, esclusivamente per il protocollo EtherNet/IP.

- **TCP/IP Interface Object (Class ID: 0xF5):** fornisce un meccanismo per la configurazione dell'interfaccia di rete TCP/IP come l'indirizzo di rete, sottorete e gateway. Ogni dispositivo EtherNet/IP deve avere almeno un istanza di questa classe.
- **Ethernet Link Object (Class ID: 0xF6):** contiene i parametri di configurazione, i contatori di errore e lo stato dell'interfaccia Ethernet IEEE 802.3. Ogni dispositivo ha esattamente un istanza di questa classe per ogni interfaccia di rete Ethernet.
- **Device Level Ring (DLR) Object (Class ID: 0x47):** gestisce il comportamento e i dati per la funzionalità DLR del dispositivo.
- **QoS Object (Class ID: 0x48):** gestisce il comportamento e i dati associati al QualityOfService (QoS) del dispositivo.

- **Base Switch Object (Class ID: 0x51)**: fornisce l'interfaccia alle informazioni di stato per un dispositivo switch Ethernet gestito per il livello applicazione CIP.
- **Simple Network Management (SNMP) Object (Class ID: 0x52)**: fornisce i parametri utilizzati per configurare l'agente SNMP nel dispositivo.
- **Power Management Object (Class ID: 0x53)**: gestisce l'alimentazione del dispositivo.
- **RSTP Bridge Object (Class ID: 0x54)**: fornisce l'interfaccia di configurazione e diagnostica per il protocollo RSTP a livello di bridge.
- **RSTP Port Object (Class ID: 0x55)**: fornisce un'interfaccia di configurazione e diagnostica per il protocollo RSTP a livello di porta.
- **Parallel Redundancy Protocol (PRP) Object (Class ID: 0x56)**: fornisce un'interfaccia di configurazione e diagnostica per i parametri PRP.
- **PRP Nodes Table Object (Class ID: 0x57)**: conserva un elenco di tutti i nodi con capacità PRP che sono stati rilevati sulla rete.

### 3.10 Indirizzi di rete

Nel corso del tempo sono nate diverse tecniche per assegnare un indirizzo IP ad un dispositivo collegato in rete.

Non tutti questi meccanismi però sono adatti ad un utilizzo in ambiente industriale. Per esempio, nei classici uffici, è comune che un dispositivo ottenga un indirizzo dinamicamente all'accensione, tramite il protocollo DHCP (Dynamic Host Configuration Protocol), potendo così ottenerne uno diverso ad ogni riavvio. Questa procedura non è consentita in un ambiente industriale dove ogni dispositivo deve avere sempre lo stesso indirizzo ad ogni accensione.

Lo standard EtherNet/IP definisce una serie di possibilità per assegnare un indirizzo IP ad un dispositivo tramite l'oggetto TCP/IP Interface.

Un dispositivo può ottenere un indirizzo tramite il protocollo BOOT (Bootstrap Protocol), protocollo DHCP o tramite i messaggi espliciti *Set\_Attribute\_Single* o *Set\_Attributes\_All*.

Un problema che può verificarsi quando l'indirizzo viene assegnato manualmente dall'uomo è che nella stessa rete vengano assegnati due indirizzi IP uguali a due distinti dispositivi. Questa situazione viene evitata utilizzando il meccanismo ACD per la risoluzione dei conflitti, conforme a IETF RC 5227.

### 3.11 EIPPI: EtherNet/IP Performance Indicators

In questa sezione viene fornita una formulazione teorica per determinare due indicatori di prestazione (PI) descritti nella IEC 617484-2, il Delivery Time (DT) e il

Throughput RTE (TRTE).[14]

In particolare vengono analizzati due differenti scenari applicativi. Il primo ipotizza che sulla trasmissione non siano presenti errori, mentre il secondo ipotizza che un pacchetto venga perso e rispedito.

Come descritto nel capitolo introduttivo sulla comunicazione Real Time, il Delivery Time è il tempo necessario per trasmettere un APDU (payload del messaggio) dal un nodo sorgente ad un nodo destinazione.

L'espressione analitica del DT fornita dallo standard IEC 61784-2 è la seguente:

$$DT = SD_s + T_{xp} + \sum_{i=1}^{n+1} CD_i + \sum_{k=1}^n SL_k + SD_r \quad (3.1)$$

dove:

$SD_s$  = ritardo sullo stack software del nodo trasmettitore.

$SD_r$  = ritardo sullo stack software del nodo ricevitore.

$T_{xp}$  = tempo di trasmissione del pacchetto sulla rete.

$n$  = numero degli switch presenti tra i nodi trasmettitore e ricevitore.

$CD_i$  = ritardo di propagazione sui segmenti del mezzo fisico.

$SL_k$  = latenza dello switch.

Come descritto in precedenza, il protocollo CIP è implementato a livello applicazione della pila ISO/OSI, e per calcolare con precisione il DT abbiamo bisogno di conoscere con precisione l'esatto istante in cui i dati prodotti dall'applicazione che risiede sul PLC attraversano tutto lo stack EtherNet/IP e vengono consegnati sulla rete. Allo stesso modo è necessario conoscere il preciso istante in cui il dato APDU arriva a livello applicazione del destinatario.

Una possibile soluzione potrebbe essere quella di scrivere un programma PLC che attivi contemporaneamente due uscite digitali, una a bordo del controllore principale ed un'altra sopra un modulo remoto collegato tramite rete EtherNet/IP. Assumendo come ipotesi che le uscite vengano attivate simultaneamente, il DT potrebbe essere calcolato sfruttando la differenza di tempo  $D$  tra l'attivazione dell'uscita digitale sul modulo locale e quella sul modulo remoto.

$$D = T_{a,r} - T_{a,l} \quad (3.2)$$

dove:

$T_{a,r}$  = tempo di attivazione uscita sul modulo remoto.

$T_{a,l}$  = tempo di attivazione uscita sul modulo locale.

In altre parole,  $T_{a,r}$  è il tempo richiesto dalla scheda di uscita remota per attivare il valore di set sul l'indirizzo richiesto dall'arrivo del comando di attivazione dall'applicazione implementata sulla CPU.

$$T_{a,r} = T_{p1} + SD_s + T_{xp} + \sum_{i=1}^2 CD_i + SL + SD_r + D_{c,r} \quad (3.3)$$

dove:

$T_{p1}$  = tempo trascorso tra la preparazione dei dati da parte del PLC ed il reale arrivo al modulo EIP, ovvero al rack remoto.

$SD_s$  = ritardo sullo stack software del nodo trasmettitore.

$T_{xp}$  = tempo di trasmissione dei pacchetti sulla rete.

$CD_i$  = ritardo di propagazione sui segmenti del mezzo fisico.

$SL$  = latenza dello switch.

$SD_r$  = ritardo sullo stack software del nodo ricevitore.

$D_{c,r}$  = tempo impiegato dal modulo remoto per commutare l'uscita digitale da ON a OFF.

Il tempo di attivazione relativo all'uscita locale è il seguente:

$$T_{a,l} = T_{p2} + D_{c,l} \quad (3.4)$$

dove

$T_{p2}$  = tempo trascorso tra la preparazione dei dati da parte del PLC ed l'arrivo al modulo di uscita locale.

$D_{c,l}$  = tempo impiegato dal modulo di uscita locale per commutare l'uscita da OFF a ON.

Non essendo specificati nella documentazione tecnica,  $T_{p1}$  e  $T_{p2}$ , possono essere assunti con lo stesso valore.

La stessa ipotesi può essere applicata a  $D_{c,r}$  e  $D_{c,l}$ . Sotto queste ipotesi e trascurando i ritardi di propagazione del cavo sui vari segmenti della rete, il Delivery Time è dato dalla differenza  $D$  misurata tra il PLC ed il modulo remoto.

$$D = DT = SD_s + T_{xp} + SL + SD_r \quad (3.5)$$

Un ulteriore indicatore di performance di cui si può dare una valutazione è il Throughput RTE (TRTE).

Come già descritto, il Throughput è la quantità di dati APDU che transitano sul collegamento in un secondo. Dipende dalla velocità dei dati del collegamento e dalla modalità di trasmissione (half-duplex oppure full-duplex).

Come esempio prendiamo in considerazione una rete EtherNet/IP a stella, dove un collegamento viene definito dalla IEC 61784-2 come una connessione tra una porta dello switch ed un nodo finale. Il TRTE per una direzione di un collegamento in modalità full-duplex può essere determinato nel seguente modo:

$$TRTE = \sum_{i=1}^k APDU_{size_i} \cdot RTEpr_i \leqslant MAXpr \quad (3.6)$$

dove

$k$  = numero di connessioni CIP che sono attive sul nodo in esame.

$APDU_{size_i}$  = dimensione in byte del messaggio scambiato (APDU) su ogni connessione CIP.

$RT Epr_i$  = velocità del pacchetto del nodo su ogni connessione CIP espressa in pacchetti al secondo.

$MAXpr$  = velocità massima del pacchetto per il nodo.

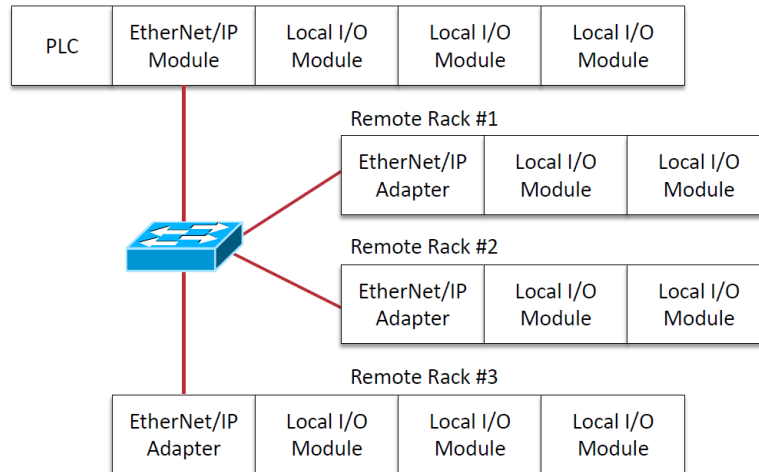


Immagine 3.21: Configurazione della rete in analisi

## Capitolo 4

# EtherCAT

EtherCAT (Ethernet for Control Automation Technology) è una tecnologia Ethernet industriale, basata a livello fisico sullo standard IEEE 802.3 che utilizza frame per lo scambio dei dati. Il protocollo è stato inventato da Beckhoff ed attualmente è sviluppato e standardizzato da ETG (EtherCAT Technology Group).

EtherCAT può rispondere alle specifiche richieste dall'automazione industriale dove sono richieste prestazioni real-time con tempi di risposta deterministici.

Il principio di funzionamento unico del protocollo, risiede nella modalità di processamento dei frame Ethernet. Ogni nodo della rete legge e scrive i propri dati nel frame senza interrompere l'avanzamento di quest'ultimo.

Questo comporta un migliore utilizzo della banda eliminando la necessità di switch e hub che possono introdurre ritardi sulla rete.[1, 6, 7, 22, 23]

### 4.1 Livelli ISO/OSI

Il protocollo EtherCAT viene implementato modificando la funzionalità di Ethernet a livello Data Link per fornire funzionalità di comunicazione deterministiche. In particolare i livelli che vengono implementati sono:

- **Livello fisico e livello di collegamento:** questi due livelli si occupano di tutte le funzioni critiche nel tempo (come il routing dei frame, la valutazione del CRC e l'accesso alla memoria) e vengono implementate tramite un componente dedicato chiamato *ESC* (EtherCAT Slave Controller).
- **Livello applicazione:** è responsabile della gestione della macchina a stati, dello scambio di dati ciclici e aciclici, nonché delle funzioni applicative specifiche del dispositivo. Questo layer viene tipicamente implementato nel firmware di un microcontrollore.

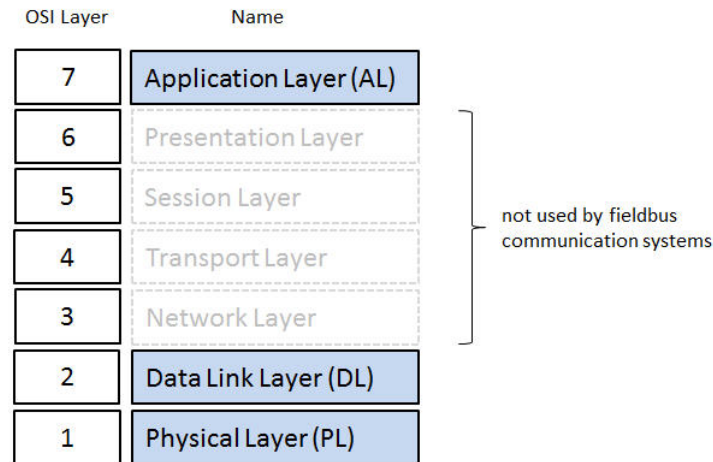


Immagine 4.1: Modello ISO/OSI modificato dal protocollo EtherCAT

#### 4.1.1 Livello fisico

Il livello fisico riceve dal livello di collegamento la sequenza di bit pacchettizzata da trasmettere sul canale e la converte in segnali da trasmettere sul mezzo trasmissivo. Questo livello supporta una vasta gamma di supporti fisici Ethernet tra cui:

- **100 BASE-TX**
  - Strato fisico più popolare per Fast Ethernet.
  - Coppia twistata e schermata (STP) con due coppie di fili.
  - Possono essere utilizzati cavi in categoria CAT 5, 6, 7.
  - E' consentito l'utilizzo di connettori RJ45 standard, connettori M12 con grado di protezione IP67.
- **100 BASE-FX**
  - Permette l'utilizzo di tutte le soluzioni multimediali.

Per trasmissioni su brevi distanze è stato sviluppato un ulteriore strato fisico denominato E-BUS. Si basa sulla trasmissione LVDS (Low Voltage Differential Signaling, standard IEEE P1596.3-1995).

L'unico prerequisito richiesto per questo livello è che il mezzo fisico supporti la trasmissione full-duplex dal momento che le risposte che ritornano al master durante la trasmissione solitamente vengono inviate da parte degli slave ancora prima che il master termini l'invio degli ultimi byte della sua query.

Comunicazioni di tipo half-duplex causerebbero collisioni con conseguente perdita di determinismo da parte della rete.



### 4.1.2 Livello di collegamento

Il livello di collegamento mette in comunicazione il livello fisico con il livello applicazione. Le sue funzioni principali sono:

- Accesso ai transceiver (ricetrasmittitore).
- Indirizzamento.
- Configurazione del dispositivo slave.
- Accesso EEPROM.
- Configurazione e gestione del SyncManager.
- Configurazione e gestione FMMU.
- Configurazione dell'interfaccia per i dati di processo.
- Clock distribuito.

Come descritto nella parte introduttiva, lato slave, l'elaborazione dei frame viene eseguita da un componente dedicato chiamato EtherCAT Slave Controller (ESC). Il routing del frame viene eseguito dalle porte mentre l'unità di elaborazione EtherCAT legge e / o scrive i dati da / verso i datagrammi e la memoria dello slave. Lo spazio di memoria dell'ESC viene utilizzato sia per i dati dell'applicazione (dati di processo, mailbox) che per i registri: in questa area sono definiti diversi registri standard per la diagnostica dell'hardware e del firmware.

All'interno dell' ESC è presente un unità di elaborazione *EPU* che riceve, analizza ed elabora il flusso dati EtherCAT.

Lo scopo principale dell'unità di elaborazione EtherCAT è di abilitare e coordinare l'accesso ai registri interni e all'area di memoria dell'ESC. L'area di memoria dell'ESC può essere indirizzata dal master EtherCAT e dall'applicazione locale tramite l'interfaccia dati di processo (PDI). Lo scambio di dati tra l'applicazione master e l'applicazione slave è paragonabile a una memoria (memoria di processo) con due porte, in cui la memoria è stata estesa con funzioni speciali, ad esempio per il controllo di coerenza (SyncManager) o la mappatura dei dati (FMMU). L'unità di elaborazione EtherCAT contiene i blocchi funzione principali degli slave EtherCAT come le funzioni di auto-forwarding e di loop-back.

La funzione di auto-forwarding riceve i frame Ethernet, li controlla e li inoltra alla funzione loop-back. La funzione di loop-back inoltra il frame alla porta successiva. Quando la porta successiva è chiusa o danneggiata, il frame viene inviato all'indietro e quindi può tornare al precedente dispositivo nella rete.

### 4.1.3 Livello Applicazione

L'ultimo livello implementato dal protocollo è il livello applicazione, responsabile della gestione dei dati di processo e delle comunicazioni mailbox. A questo livello vengono definiti i vari stati che un dispositivo slave può assumere durante il suo funzionamento.

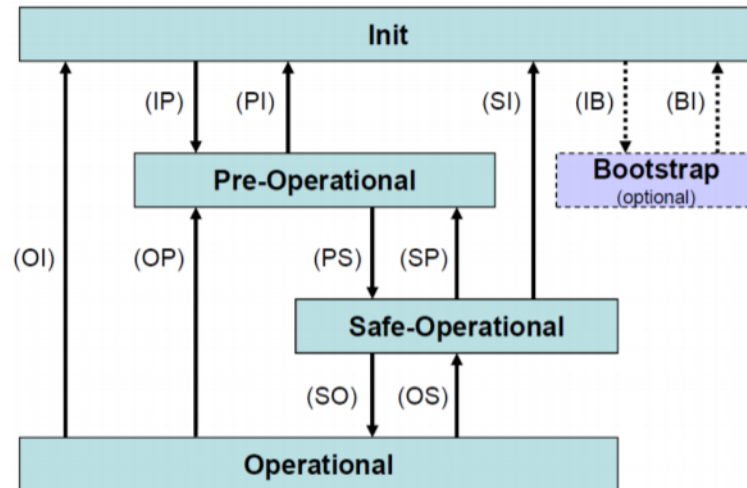


Immagine 4.2: Macchina a stati EtherCAT

- **Bootstrap:** questo stato è utilizzato per l'aggiornamento del firmware e per disabilitare i dati di processo e la mailbox.
- **Init:** è lo stato che si ottiene subito dopo l'accensione del dispositivo. In questo stato non c'è comunicazione diretta tra master e slave. Vengono inizializzati una serie di registri e viene effettuata la configurazione dei Sync-Manager.
- **Pre-Operational:** in questo stato viene attivata la mailbox. Il master e lo slave possono utilizzare la mailbox ed i relativi protocolli per scambiarsi specifici parametri di inizializzazione dell'applicazione. Lo scambio dei dati di processo PDO non è ancora permesso.
- **Safe-Operational:** in questo stato il master e gli slave possono scambiarsi i dati di processo solo per quanto riguarda i dati in ingresso (da slave verso il master). I dati di processo in uscita (da master verso lo slave) sono bloccati.
- **Operational:** master e slave possono scambiarsi i dati di processo, sia in ingresso che in uscita.

Inoltre è possibile inviare richieste al dispositivo e richiedere il suo stato interrogando alcuni registri.

- **AL Control (0x0120):** richiesta da parte del master verso il dispositivo slave ad uno specifico stato.

- **AL Status (0x0130)**: fornisce lo stato attuale del dispositivo.
- **AL Status Code (0x0134)**: fornisce il codice dell'errore del dispositivo.

#### 4.1.3.1 Profili dispositivi

Il profilo del dispositivo descrive i parametri dell'applicazione e il suo comportamento funzionale. L'integrazione con altri protocolli e l'utilizzo di dispositivi differenti viene semplificato utilizzando delle interfacce che aiutano sia gli utenti che i produttori di dispositivi. Uno slave può supportare uno o più profili di comunicazione. Il master viene informato su quali dispositivi sono stati implementati nello slave attraverso il suo file descrittivo.

Di seguito vengono descritti i profili dei dispositivi ed i protocolli che possono essere supportati su rete EtherCAT.

- **CAN Application Layer over EtherCAT (CoE)**: dispositivi CANopen e profili applicativi sono disponibili per un vasto numero di classi di dispositivi come componenti I/O, drive, valvole, encoder. Questo profilo EtherCAT fornisce gli stessi meccanismi di comunicazione implementati da CANopen secondo lo standard EN50325-4 (object Dictionary, PDO - Process Data Objects e SDO - Service Data Objects). Questo permette di implementare EtherCAT in dispositivi precedentemente dotati di interfaccia CANopen.
- **Servodrive Profile over EtherCAT (SoE)**: il protocollo SERCOS è famoso per essere un'interfaccia di comunicazione deterministica in applicazioni che richiedono un controllo assi. Il profilo SERCOS per azionamenti è definito nello standard IEC 61800-7 e contiene anche la mappatura di questo profilo su EtherCAT. L'accesso a tutti i parametri e le funzioni, si basa sulla mailbox EtherCAT, mentre lo scambio dei dati di processo AT ed i dati MDT vengono trasferiti utilizzando il meccanismo di EtherCAT.
- **Ethernet over EtherCAT (EoE)**: utilizzando il layer fisico ed i frame, EtherCAT tramite questo profilo può trasportare all'interno di datagrammi qualunque traffico dati Ethernet. I dispositivi Ethernet sono connessi alla rete EtherCAT attraverso gli Switchport. I frame Ethernet, così come i protocolli internet (TCP/IP, VPN, PPoE, ...) sono veicolati su EtherCAT in modo trasparente tramite tunnelling. Il dispositivo dotato di funzionalità di Switchport si occupa di inserire nel traffico EtherCAT i frammenti TCP/IP ed impedisce che il determinismo della rete venga compromesso. Inoltre, il master comportandosi come uno switch di livello 2, può supportare tutte le tecnologie Internet come il trasferimento FTP, HTTP, web server ed l'invio di e-mail.
- **File Access over EtherCAT (FoE)**: questo protocollo consente l'accesso al file interno di un dispositivo per l'aggiornamento del firmware attraverso la rete.

Il protocollo è stato definito solamente con caratteristiche minimali per poter essere supportato da applicazioni di boot loader senza la necessità di implementare uno stack TCP/IP.

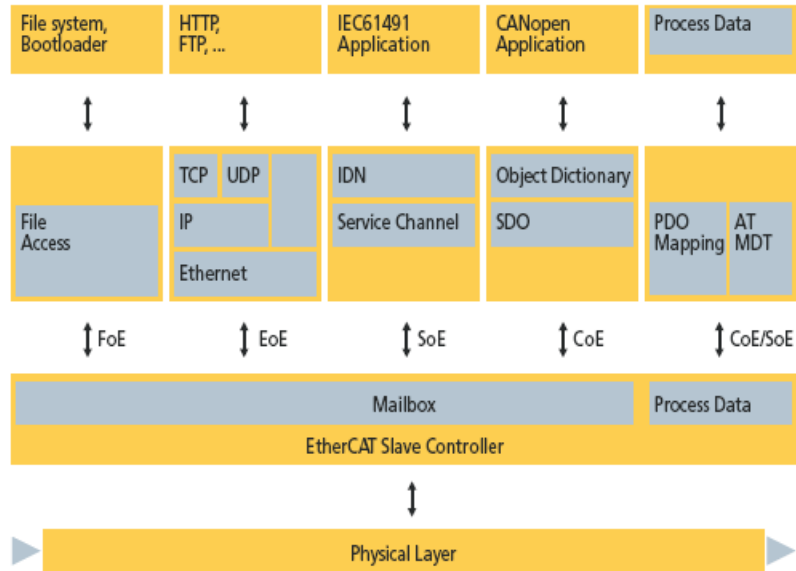


Immagine 4.3: Profili e protocolli supportati da EtherCAT

## 4.2 Il protocollo

La particolarità di questo protocollo, è quella di non scambiare un frame per ogni singolo nodo della rete, come avviene nei sistemi Ethernet tradizionali. EtherCAT supera queste problematiche utilizzando un meccanismo in base al quale un singolo frame è di norma sufficiente a scambiare i dati di processo in ingresso e in uscita di tutti i nodi. Tale meccanismo si ispira ad un treno ad alta velocità in movimento sopra un binario. Il treno (*frame*) è sempre in movimento e non si ferma ad ogni stazione (nodo della rete). Osservando il treno da una sottile finestra passare sul binario è possibile osservarlo per intero. I vagoni (*datagrammi*) hanno una lunghezza variabile e da loro possono essere estratte singole persone (bit) o gruppi di persone (byte) senza necessità di fermare il treno. Questa analogia con questo tipo di caso reale ha portato allo sviluppo di questo protocollo di comunicazione utilizzato nelle reti RT.

### 4.2.1 Dati di processo "on the fly"

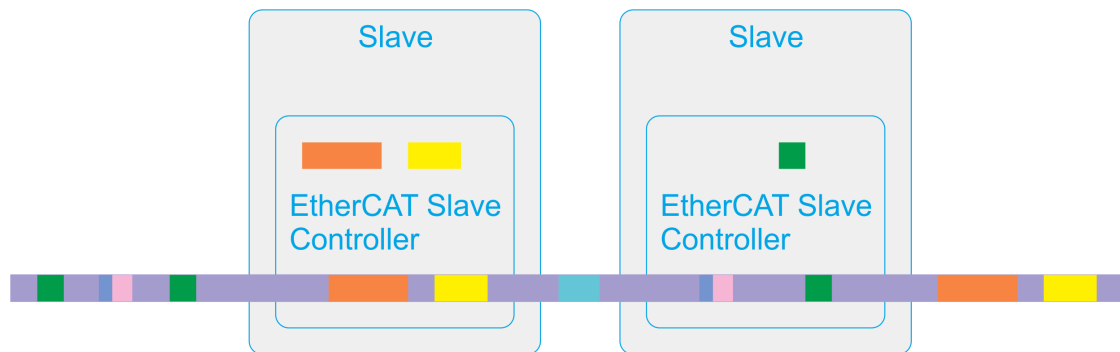


Immagine 4.4: Telegramma in transito sulla rete

Un master EtherCAT invia un telegramma che attraversa tutti i nodi. Ogni slave presente sulla rete legge i dati ad esso destinati e scrive i dati da lui prodotti da consegnare al master. Tutte queste operazioni vengono eseguite e scritte "al volo" ("on the fly") nel frame che transita sulla rete attraversando i vari nodi.

Il ritardo subito dal frame è riconducibile solo al tempo richiesto al frame di attraversare fisicamente lo Slave, non superiore a 60 ns .

L'ultimo nodo della rete reinvia il messaggio al master avvalendosi della comunicazione full-duplex di Ethernet.

Il master EtherCAT è l'unico nodo della rete in grado di inviare attivamente i frame mentre tutti gli altri nodi non fanno altro che inoltrarli.

Il fatto che il pacchetto venga trasmesso continuamente da uno slave all'altro, ne garantisce la continua presenza su tutti i dispositivi della rete, garantendo un certo grado di determinismo, ottimizzando la velocità di trasferimento, la larghezza di banda e prevenendo i ritardi.

Il master utilizza un *Media Access Controller* (MAC) standard, senza necessità di un processore dedicato alla comunicazione. Questo consente di implementare un dispositivo master su qualunque dispositivo dotato di una porta di rete, indipendentemente da hardware e sistema operativo utilizzato.

I dispositivi Slave integrano un *ESC* (EtherCAT Slave Controller) in grado di processare i frame al volo a livello hardware, il che rende le prestazioni della rete predicibili e indipendenti dalla particolare implementazione dei dispositivi slave.

All'interno di ogni slave è presente una *FMMU* (Fieldbus Memory Management Unit) che possiede le stesse funzionalità della MMU (Memory Management Unit) presente sui comuni processori con il compito di verificare e trasferire i dati contenuti nei datagrammi indirizzati al dispositivo.

Come verrà descritto successivamente, per lo scambio dei dati di processo viene utilizzato l'indirizzamento logico.

Il protocollo può contenere più datagrammi e possono essere impiegati frame per la gestione di tutti i telegrammi durante un ciclo di controllo.

Ciascun datagramma indirizza uno specifico sottoinsieme dell'immagine di processo della rete. Durante la configurazione iniziale ad ogni slave viene assegnata una specifica locazione in tale spazio di indirizzamento. Slave allocati nello stesso intervallo possono essere indirizzati dallo stesso datagramma.

Sono consentite comunicazioni broadcast, multicast e comunicazioni tra master e comunicazioni tra slave. Per comunicazioni tra slave vengono utilizzati due meccanismi differenti. Se i due slave sono disposti in serie, la comunicazione tra i due può essere effettuata nello stesso ciclo in modo molto veloce. Diversamente, lo scambio di informazioni può avvenire col master per poi essere passato allo slave, con almeno due cicli di aggiornamento.

Processando i dati a livello hardware è possibile eliminare i ritardi introdotti dagli stack dalle tecnologie Ethernet tradizionali. Di seguito viene riportato un confronto sui ritardi introdotti dagli stack di alcune tecnologie Industrial Ethernet.



		EtherNet/IP <sup>®</sup>	EtherCAT <sup>®</sup> 
Stack Time	Profinet IO	Ethernet/IP	EtherCAT
Average	0.58 ms	1.89 ms	0.11 ms
Max.	0.74 ms	2.96 ms	0.18 ms
Min.	0.54 ms	1.23 ms	0.05 ms

Immagine 4.5: Ritardi introdotti dagli stack

#### 4.2.2 Comunicazione aciclica

Oltre alla comunicazione ciclica utilizzata per scambiare i dati di processo tra i dispositivi, il protocollo predispone un ulteriore tipo di comunicazione ciclica, utilizzata prevalentemente per la configurazione dei dati di processo e la diagnosi dei dispositivi della rete. Per questa funzione viene utilizzata la *Mailbox*, un'affidabile protocollo con funzionalità di recupero automatico in caso di perdita o corruzione dei messaggi.

I dispositivi allocano uno spazio di memoria riservato a questo tipo di comunicazioni, che possono essere ricevute e inviate al master tramite un handshake monitorato. L'accesso a questa area di memoria non viene gestito dalla FMMU, ma tramite un accesso diretto sull'indirizzo del dispositivo.

### 4.2.3 Trasferimento dei dati

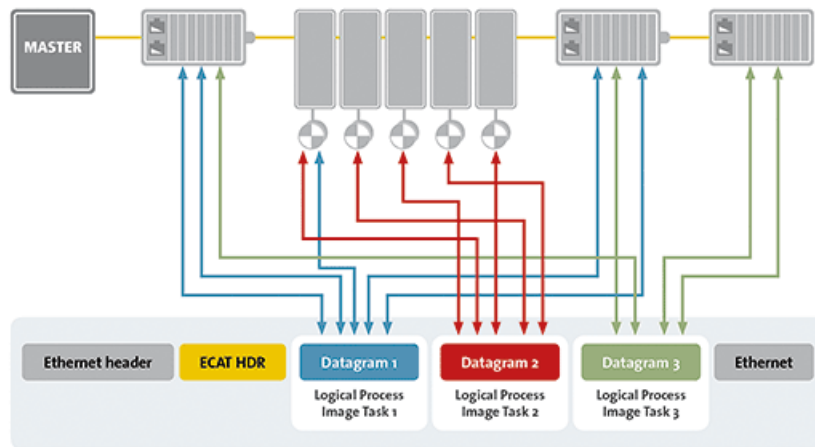


Immagine 4.6: Dati di processo "on-the-fly"

Il protocollo EtherCAT utilizza un architettura master - slave ed è ottimizzato per trasportare i dati di processo utilizzati dagli slave utilizzando un tradizionale frame Ethernet. I frame EtherCAT sono identificati da un particolare EtherType 0x88A4.

Essendo un protocollo ottimizzato per pochi dati di processo ciclici, è possibile evitare l'utilizzo degli stack software TCP/IP e UDP.

Il frame EtherCAT è costituito da diversi sotto-telegrammi, *Datagram* (datagram 1..n), ognuno dei quali serve una particolare area di memoria dell'immagine di processo logico che può raggiungere una dimensione di 4GB. Ciascun Datagram ha un indirizzo verso uno o più dispositivi slave.

L'intestazione di ogni datagramma indica quale tipo di accesso il dispositivo master richiede:

- Lettura, scrittura, lettura e scrittura.
- Accesso ad uno specifico slave tramite indirizzamento diretto, o accesso a più slave attraverso indirizzamento logico (indirizzamento implicito).

La sequenza dei dati è indipendente dall'ordine fisico dei terminali Ethernet presenti sulla rete e l'inserimento dei dati da parte degli slave avviene nel primo datagramma vuoto. L'indirizzamento può avvenire in qualsiasi ordine.

Esiste una variante del frame, EtherCAT UDP che confeziona il protocollo EtherCAT nei datagrammi UDP/IP (porta UDP 88A4), dove vengono aggiunti al frame intestazione IP e UDP. Questo consente ad un qualsiasi controllo che implementa lo stack Ethernet di indirizzare i sistemi EtherCAT. È possibile anche la comunicazione tra i router in altre sottoreti. In questa variante, le prestazioni del sistema sono influenzate dalle operazioni di codifica e dalla presenza di più sottoreti, con conseguente aumento dei tempi

di consegna dei frame.

Entrambe le implementazioni prevedono la rilevazione dei bit errati durante il trasferimento attraverso la valutazione del checksum CRC. La comunicazione Ethernet tradizionale tra i nodi è possibile trasferendo telegrammi TCP/IP attraverso un canale aciclico sullo stesso mezzo trasmissivo senza impattare sullo scambio dati deterministico. Infatti, il protocollo, rende disponibili tutte le tecnologie Internet (HTTP, FTP, Webserver, ...) senza restrizioni sulle prestazioni real-time del sistema.

#### 4.2.3.1 Struttura del frame

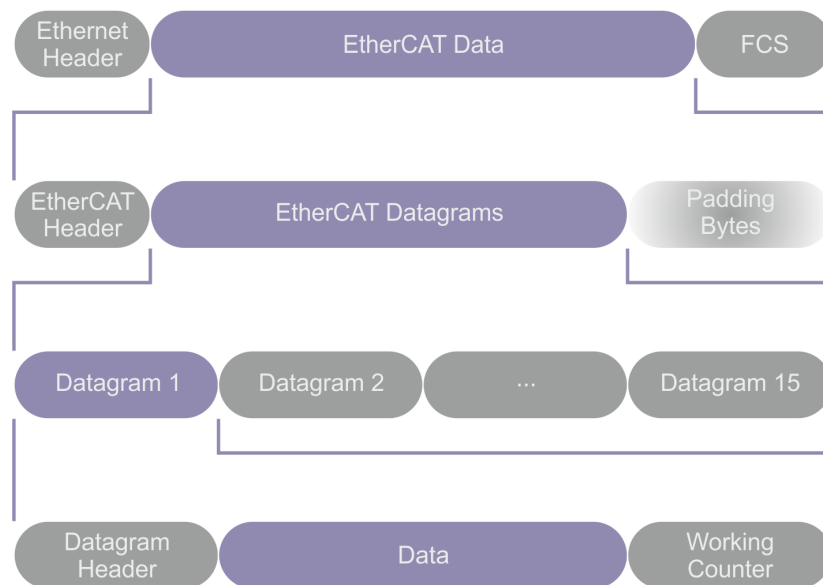


Immagine 4.7: Telegramma EtherCAT

EtherCAT utilizza Ethernet IEEE 802.3 standard. Il payload dei frame EtherCAT trasporta un numero variabile di datagrammi. Ogni datagramma specifica quale operazione viene eseguita sulla memoria dello slave tramite un codice di comando.

Il telegramma EtherCAT inizia con un'intestazione Ethernet, seguita dai dati EtherCAT. Il telegramma è terminato da una sequenza di controllo frame (FCS) costituita da un polinomio CRC a 32 bit con distanza di hamming pari a 4.

Se l'intero frame Ethernet è inferiore a 64 byte, alla fine dei dati vengono inseriti tra 1 e 32 byte di riempimento. Questi dati possono contenere fino a 15 datagrammi.

Un datagramma è formato da un'intestazione, dai dati da leggere o scrivere e da un contatore di lavoro.





Immagine 4.8: Intestazione EtherCAT

L'intestazione EtherCAT è suddivisa in 3 campi:

- **Length:** lunghezza del datagramma EtherCAT (senza FCS).
- **Res:** riservato
- **Type:** tipo di protocollo. I controller EtherCAT slave (ESC) supportano solo i comandi EtherCAT (tipo = 0x1).

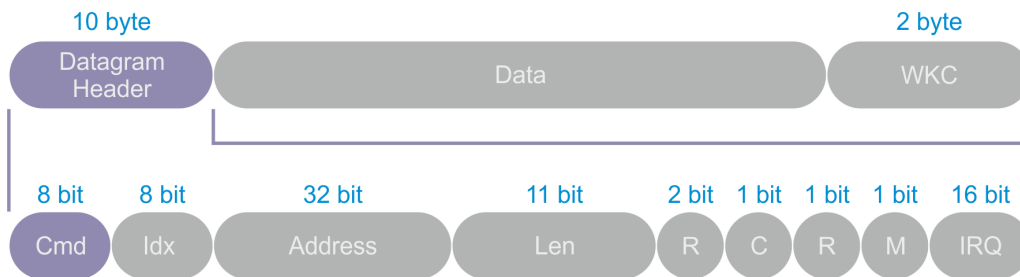


Immagine 4.9: Datagramma EtherCAT

L'intestazione del datagramma contiene informazioni sul tipo di comando da eseguire, un identificatore numerico utilizzato dal master per l'identificazione dei duplicati o dei datagrammi persi e una specifica dell'indirizzo. Questo è seguito da una specifica di lunghezza che indica la lunghezza dei dati successivi all'interno del datagramma, due bit riservati, un bit per valutare la circolazione dei frame, un altro bit riservato, un bit per indicare la presenza di un altro datagramma e infine un registro di richiesta evento.

- **Cmd:** tipo di comando che deve essere eseguito.
- **Idx:** è un identificatore numerico utilizzato dal master per identificare duplicati o datagrammi persi. Gli slave della rete non dovrebbero cambiare l'indice.
- **Address:** indirizzo stazione configurato o indirizzo logico.
- **Len:** lunghezza dei dati che seguono all'interno di questo datagramma.
- **R:** questi 3 bit sono riservati.
- **C:** indica la circolazione del frame. Se ha valore 0 significa che il frame non circola, mentre un valore impostato su 1 identifica che il frame è circolato una volta.

- **M:** indica se sono presenti più datagrammi. Se ha valore 0 significa che il datagramma è l'ultimo, mentre un valore 1 identifica che segue un ulteriore datagramma.
- **IRQ:** registro di richiesta eventi di tutti i dispositivi slave combinati con un OR logico.

I campi successivi all'intestazione sono:

- **Data:** contiene i dati che devono essere letti o scritti.
- **WKC:** Working Counter.

#### 4.2.3.2 Indirizzamento

Il protocollo prevede diverse tecniche di indirizzamento per i dispositivi della rete. In base al valore contenuto nel parametro *Address* dell'intestazione di ogni telegramma viene identificato il tipo di indirizzamento. Di seguito vengono elencate le differenti modalità di indirizzamento utilizzate.

- **Indirizzamento di posizione:** viene utilizzato solamente all'avvio del sistema per la scansione del bus di campo, durante il quale il master assegna un indirizzo ad ogni nodo in rete. In seguito questo tipo di indirizzamento è utilizzato solo per rilevare i dispositivi slave che vengono aggiunti in rete. Il datagramma contiene l'indirizzo di posizione del dispositivo slave indirizzato come valore negativo. Ogni slave incrementa questo indirizzo. Lo slave che legge questo indirizzo come zero viene indirizzato ed esegue il comando corrispondente non appena lo riceve. Il campo *Offset* identifica l'indirizzo del registro locale o l'indirizzo di memoria locale dell'ESC.
- **Indirizzamento dei nodi:** viene utilizzato per l'accesso al registro di singoli dispositivi che sono già stati identificati sulla rete. L'indirizzo di ogni nodo viene assegnato dal master all'avvio e non può essere modificato dallo slave EtherCAT. Lo slave viene indirizzato se il suo indirizzo corrisponde all'indirizzo della stazione configurata o all'alias della stazione configurata. Il campo *Offset* identifica l'indirizzo del registro locale o l'indirizzo di memoria locale dell'ESC.
- **Indirizzamento Broadcast:** utilizzato per inizializzare i dispositivi slave della rete.
- **Indirizzamento logico:** questo tipo di indirizzamento riduce il contenuto di comunicazione non necessario nella trasmissione dei dati di processo. Tutti i dispositivi leggono e scrivono nello stesso intervallo di indirizzi del telegramma EtherCAT. Ogni slave utilizza la FMMU per mappare i dati dall'immagine dei dati di processo logici al relativo indirizzo locale nella sua area di memoria. Il master configura le FMMU di ogni slave durante l'avvio. Utilizzando le informazioni di configurazione delle sue FMMU, uno slave riconosce quali parti dell'immagine dei dati di processo devono essere estratti o scritti dal datagramma. Lo slave viene indirizzato se la configurazione FMMU corrisponde al campo *Address*.

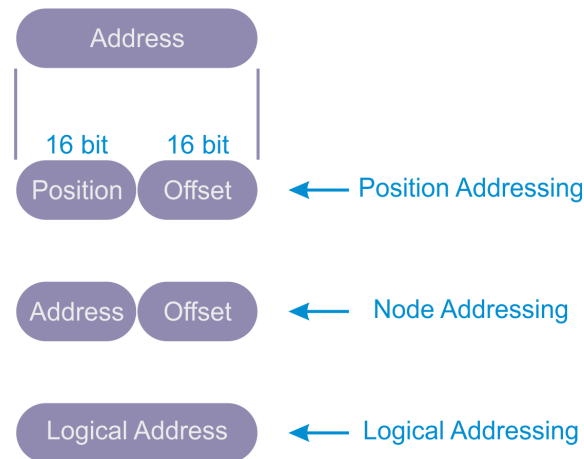


Immagine 4.10: Indirizzamento EtherCAT

#### 4.2.3.3 Comandi

Di seguito vengono elencati i vari comandi supportati dal protocollo che possono essere inseriti nel campo *Cmd* presente nell'intestazione del datagramma. Per le operazioni combinate di lettura e scrittura, l'operazione di lettura viene eseguita prima dell'operazione di scrittura.

- **NOP (0) - No Operation:** lo slave ignora il comando.
- **APRD (1) - Auto Increment Read:** uno slave incrementa l'indirizzo. Uno slave scrive i dati letti nel datagramma EtherCAT se l'indirizzo ricevuto è zero.
- **APWR (2) - Auto Increment Write:** uno slave incrementa l'indirizzo. Uno slave scrive i dati in un'area di memoria se l'indirizzo ricevuto è zero.
- **APRW (3) - Auto Increment Read Write:** uno slave incrementa l'indirizzo. Uno slave inserisce i dati letti sul datagramma EtherCAT e scrive i dati appena acquisiti nella stessa area di memoria se l'indirizzo ricevuto è zero.
- **FPRD (4) - Configured Address Read:** uno slave inserisce i dati che ha letto nel datagramma EtherCAT se il suo indirizzo slave corrisponde a uno degli indirizzi configurati nel datagramma.
- **FPWR (5) - Configured Address Write:** uno slave scrive i dati in un'area di memoria se il suo indirizzo slave corrisponde a uno degli indirizzi configurati nel datagramma.
- **FPRW (6) - Configured Address Read Write:** uno slave inserisce i dati letti nel datagramma EtherCAT e scrive i dati appena acquisiti nella stessa area di memoria se il suo indirizzo slave corrisponde a uno degli indirizzi configurati nel datagramma.

- **BRD (7) - Broadcast Read:** tutti gli slave inseriscono nel datagramma un OR logico tra i dati della loro memoria e i dati del datagramma Ethercat. Tutti gli slave incrementano il campo *Position*.
- **BWR (8) - Broadcast Write:** tutti gli slave scrivono dati in un'area di memoria. Tutti gli slave incrementano il campo *Position*.
- **BRW (9) - Broadcast Read Write:** tutti gli slave inseriscono nel datagramma un OR logico tra i dati della loro memoria e i dati del datagramma EtherCAT e li scrivono nell'area di memoria. In genere, BRW non viene utilizzato. Tutti gli slave incrementano il campo *Position*.
- **LRD (10) - Logical Memory Read:** uno slave inserisce i dati che ha letto sul datagramma EtherCAT se l'indirizzo ricevuto corrisponde a una delle aree FMMU configurate per la lettura.
- **LWR (11) - Logical Memory Write:** gli slave scrivono i dati nell'area di memoria se l'indirizzo ricevuto corrisponde a una delle aree FMMU configurate per la scrittura.
- **LRW (12) - Logical Memory Read Write:** uno slave inserisce i dati che ha letto sul datagramma EtherCAT se l'indirizzo ricevuto corrisponde a una delle aree FMMU configurate per la lettura. Gli slave scrivono i dati nell'area di memoria se l'indirizzo ricevuto corrisponde a una delle aree FMMU configurate per la scrittura.
- **ARMW (13) - Auto Increment Read Multiple Write:** uno slave incrementa il campo *Address*. Uno slave inserisce i dati che ha letto sul datagramma EtherCAT quando l'indirizzo ricevuto è zero, altrimenti scrive i dati nell'area di memoria.
- **FRMW (14) - Configured Read Multiple Write:** uno slave inserisce i dati letti nel datagramma EtherCAT se l'indirizzo corrisponde a uno dei suoi indirizzi configurati, altrimenti lo slave scrive i dati nella posizione di memoria.

#### 4.2.3.4 Working Counter

Il datagramma EtherCAT termina con un campo relativo al *Working Counter*. Viene utilizzato dal dispositivo master per identificare se i dispositivi slave hanno effettuato correttamente le operazioni richieste durante la ricezione dei telegrammi. Il master invia un campo nel frame impostato su zero. Questo campo è utilizzato come contatore. Gli slave devono aumentare il valore del campo ogni volta che leggono o scrivono le loro informazioni. Ad ogni datagramma può essere assegnato un valore previsto per il contatore dopo che il telegramma ha attraversato tutti i dispositivi. Il master può verificare se un datagramma è stato elaborato correttamente confrontando il valore previsto con il valore effettivo del contatore dopo che questo ha attraversato tutti i dispositivi. Il valore del contatore viene gestito e incrementato nel seguente modo:

- **Comandi di lettura:** i comandi di lettura incrementano il contatore di 1 se il comando viene eseguito correttamente, altrimenti il valore attuale rimane invariato.
- **Comandi di scrittura:** i comandi di scrittura incrementano il contatore di 1 se il comando viene eseguito correttamente, altrimenti il valore attuale rimane invariato.
- **Comandi di lettura/scrittura:** i comandi in lettura incrementano il contatore di 1, i comandi in scrittura di 2 mentre i comandi di lettura e scrittura di 3. Se il comando non viene eseguito correttamente, il valore attuale del contatore rimane invariato.

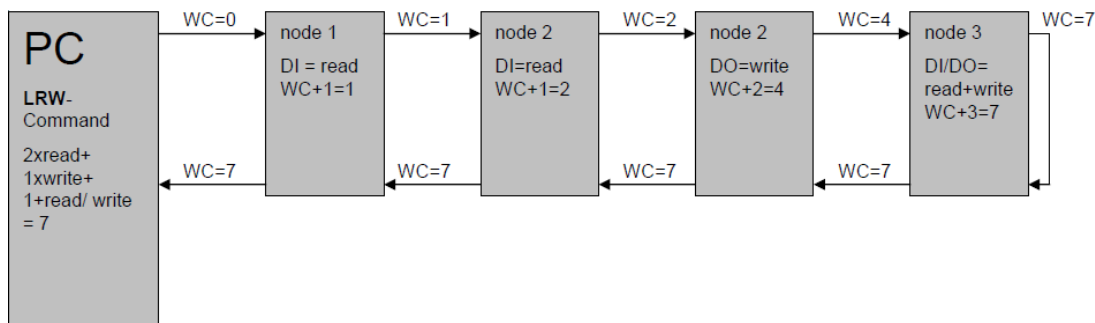


Immagine 4.11: Gestione del Working Counter

#### 4.2.3.5 SyncManager

Per quanto riguarda la memoria interna, i dispositivi slave EtherCAT sono suddivisi in due porzioni principali, con una dimensione massima fino a 64 kbyte. I primi 4 byte sono destinati ai registri interni del dispositivo, mentre i successivi 60 byte (*DPRAM*) sono destinati ai dati di processo.

Per proteggere le zone di memoria dello slave da accessi simultanei da parte del master, viene utilizzato un meccanismo di sincronizzazione sullo scambio di messaggi tra il master ed i dispositivi slave. Il protocollo prevede due modalità di sincronizzazione:

- **Modalità buffer:** in questa modalità viene garantito l'accesso ai nuovi dati in qualsiasi momento per entrambe le parti, master e slave, senza nessuna restrizione. Per questa modalità sono necessarie tre aree di memoria consecutive, chiamate *Buffers*, di cui una è sempre disponibile per la scrittura dallo slave ed un'altra contiene sempre i dati aggiornati per la lettura da parte del master. Questa modalità è generalmente utilizzata per la comunicazione dei dati di processo.
- **Modalità mailbox:** in questa modalità viene utilizzato un meccanismo di handshake tra master e slave, in quanto è disponibile un'unica area di memoria di lettura e scrittura. La scrittura da parte del master o dello slave può avvenire solo quando il buffer è vuoto, ovvero quando la controparte, slave o master, ha completamente

letto i dati in esso contenuti. Analogamente la lettura può avvenire solo quando il buffer è stato scritto completamente dalla controparte.

#### 4.2.3.6 Clock distribuito

Nel caso di applicazioni distribuite che richiedono azioni simultanee è di particolare importanza una corretta sincronizzazione di tutti i dispositivi (ad esempio: azionamenti che eseguono movimenti coordinati).

Per soddisfare questo requisito, EtherCAT utilizza i clock distribuiti (DC) come descritto nello standard IEEE 1588 (PTP- Precision Time Protocol).

Rispetto ad una comunicazione completamente sincrona, i clock distribuiti sincronizzati possiedono un'elevata tolleranza nei confronti del jitter di comunicazione.

La regolazione dei clock nei singoli nodi avviene a livello hardware. Il riferimento temporale del primo slave è distribuito ciclicamente a tutti gli altri dispositivi del sistema. Poiché il riferimento temporale inviato dal clock di riferimento del master giungerà con un certo ritardo agli altri slave della rete, ogni nodo della rete deve misurare la differenza temporale tra il frame di andata e quello di ritorno. Il ritardo di propagazione di ogni singolo slave può essere determinato tramite queste misurazioni dal clock principale. Tutti i vari clock della rete vengono corretti in base a questo valore di ritardo. Questo meccanismo permette di avere un jitter nella sincronizzazione dei clock inferiore ad 1  $\mu$ s. Dato che tutti i nodi della rete condividono lo stesso riferimento temporale, possono attivare simultaneamente le proprie uscite ed acquisire gli ingressi con un riferimento temporale molto preciso.

Il meccanismo dei distribuiti clock DC non viene solo sfruttato per le sincronizzazioni tra i dispositivi, ma riveste un ruolo fondamentale nelle applicazioni di controllo assi.

In questo tipo di applicazioni, la velocità di movimento viene derivata dalla posizione misurata, per cui è fondamentale che le misure di posizione avvengano ad intervalli regolari di tempo con elevata precisione. Errori di misura sull'istante campionato possono portare ad un errore maggiore sul calcolo della velocità.

Per quanto riguarda la sincronizzazione tra più reti distinte accoppiate, questa può essere fatta senza perdita di prestazione mediante l'utilizzo di dispositivi *Bridge*. Ogni slave della rete ha all'interno un modulo per la gestione della sincronizzazione.

### 4.3 Topologia di rete

EtherCAT supporta praticamente tutte le topologie di rete possibili. Linea, albero, stella o cascata possono essere realizzate tramite le porte di connessione Ethernet disponibili sui dispositivi della rete.

E' possibile implementare una topologia di rete *daisy-chain* con centinaia di nodi senza le limitazioni che si possono presentare durante l'introduzione di switch e hub.

Di particolare utilità è la possibilità di avere una combinazione di linee e di rami. Ogni segmento della rete può utilizzare il mezzo fisico più adatto alla situazione applicativa. Per la connessione di nodi con una distanza massima di 100 m possono essere utilizzati

cavi Ethernet standard 100BASE-TX. Fibre ottiche 100BASE-FX o Plastic Optical Fibres possono essere utilizzate per integrare il sistema con applicazioni speciali con una distanza massima di 20 km.

Una rete EtherCAT può includere fino a 65.535 dispositivi e l'espandibilità della rete è illimitata. E' possibile ottenere una ridondanza sui cavi di comunicazione connettendo l'ultimo nodo della rete ad una porta aggiuntiva del master. Un evento di ridondanza come l'interruzione di un cavo o un malfunzionamento di un nodo viene individuato dal master con tempi di reazione inferiore a 15  $\mu$ s, in modo da perdere al massimo un frame ciclico.

Oltre all'individualizzazione degli errori sulla rete, è importante anche la loro localizzazione. Ogni slave della rete effettua un controllo di ridondanza ciclico su ogni frame. I dati contenuti nel frame vengono utilizzati solo se il frame ricevuto è valido. Se viene individuato un errore, un corrispondente contatore viene incrementato ed i nodi successivi vengono informati sul fatto che il frame è corrotto.

Il dispositivo master scarta il frame danneggiato e localizza la posizione di danneggiamento analizzando i contatori di errore degli slave.

Come descritto nella sezione precedente, all'interno dei frame è presente un Working Counter che permette di monitorare la consistenza dei dati in ogni datagramma. Ogni nodo indirizzato da un datagramma incrementa automaticamente il Working Counter, consentendo al master di verificare ciclicamente se tutti i nodi stanno lavorando con dati consistenti. Se il valore del Working Counter è diverso da quello atteso, il master non inoltra i dati del datagramma all'applicazione di controllo. La causa del comportamento inatteso può essere identificata con l'aiuto delle informazioni di stato e di errore provenienti dai nodi.

Nella sezione successiva analizzeremo il traffico di rete tramite Wireshark che include uno specifico plugin per questo protocollo.

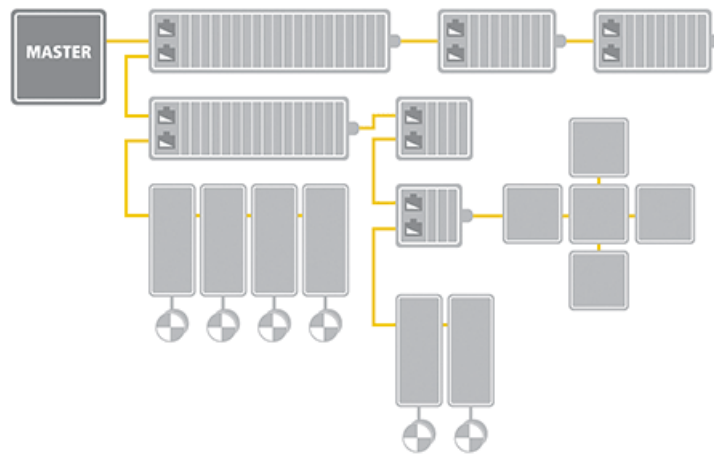


Immagine 4.12: Topologia flessibile: linea, albero o stella

## 4.4 Prestazioni

EtherCAT è stato progettato per raggiungere elevate prestazioni con un elevato numero di nodi nella rete. E' possibile ottenere tempi di ciclo brevi in quanto l'elaborazione del protocollo avviene esclusivamente a livello hardware ed è indipendente dalle prestazioni della CPU e da ogni componente software. Tramite l'accesso diretto alla memoria DMA, è possibile trasferire i dati tra la scheda di rete e il processore master oppure gli I/O dello slave con coinvolgimento minimo della CPU. Gli slave sono incaricati di mappare i telegrammi appropriati riducendo in questo modo il lavoro del master. Il sistema è ottimizzato per gestire I/O distribuiti e l'utilizzo di frame Ethernet fa aumentare il datarate utilizzabile di oltre il 90 %. Tramite un unico frame possono essere scambiati fino a 1486 byte. Di seguito vengono elencate alcune caratteristiche sui tempi di aggiornamento fornite dal produttore:

- 256 I/O in 11  $\mu$ s.
- 1000 I/O digitali distribuiti su 100 nodi in 30  $\mu$ s. (trasferimento di 125 byte su Ethernet a 100 Mbps.).
- 200 I/O analogici (16 bit) in 50  $\mu$ s, corrispondenti a 20 kHz di sampling rate.
- 100 Servo-Assi (ognuno con 8 Byte In + Out) in 100  $\mu$ s.
- 12000 I/O digitali in 350  $\mu$ s.

L'utilizzo dell'ampiezza di banda è massimizzato, poiché ogni nodo e ogni dato non richiedono un frame separato. Per questo, è possibile ottenere tempi di ciclo estremamente brevi ( $\leq 100 \mu$ s). Utilizzando le funzionalità full-duplex, è possibile ottenere velocità di trasmissione dati superiori a 100 Mbps.

### 4.4.1 Metodi di analisi per le prestazioni

Il tempo ciclico di EtherCAT è costituito da una somma pesata tra i seguenti componenti:

- Tempo di inoltro del pacchetto principale.
- Ritardo del livello fisico del master.
- Ritardo del livello fisico dello slave.
- Ritardo di inoltro degli slave.
- Ritardo di propagazione lungo i cavi.
- Periodo di inattività della rete.



Le prestazioni vengono misurate utilizzando il tempo minimo di ciclo raggiungibile in funzione dei numeri dei nodi presenti sulla rete (slave).

Prima di proseguire con l'analisi, dobbiamo fornire alcune definizioni.[8]

$T_m^f$  = tempo di inoltro del pacchetto sul master.

$T_p$  = ritardo massimo introdotto dal livello fisico (Rx + Tx).

$T_c^p$  = ritardo di propagazione lungo i cavi.

$T_s^f$  = ritardo di inoltro del pacchetto all'interno dello slave.

$T_{max}$  = tempo di inoltro del pacchetto di dimensioni massime.

$T_s^{tot}$  = ritardo totale introdotto da un dispositivo slave.

$n$  = numero di slave della rete.

$k$  = numero di pacchetti utilizzati per ciclo EtherCAT.

$p$  = payload per slave in byte.

$bw$  = larghezza di banda in bit al secondo.

$T^{ETC}$  = tempo minimo raggiungibile dalla rete EtherCAT.

Di seguito viene descritto il procedimento per il calcolo del tempo ciclico su una topologia di rete lineare. Per il calcolo del tempo minimo di ciclo raggiungibile il peririodo di inattività viene considerato come nullo.

La seguente formula permette di calcolare il ritardo totale introdotto da un dispositivo slave.

$$T_s^{tot} = T_p + T_c^p + T_s^f \quad (4.1)$$

Supponendo che sia necessario un solo pacchetto per gestire tutte le comunicazioni EtherCAT in un solo ciclo, il tempo minimo di ciclo raggiungibile può essere calcolato nel seguente modo:

$$T_l^{ETC} = T_m^f + T_p + n \cdot T_s^{tot} \quad (4.2)$$

dove per semplicità il ritardo di inoltro del pacchetto sullo slave  $T_s^f$  ed il ritardo di propagazione associato al mezzo trasmissivo  $T_p$  e  $T_c^p$ , possono essere impostati a  $1 \mu s$ , ottenendo quindi:

$$T_l^{ETC} = T_m^f + T_p + n \cdot 1\mu s \quad (4.3)$$

Se sono richiesti più di un pacchetto, il calcolo del tempo minimo è calcolabile con la seguente formula:

$$T_l^{ETC} = (k - 1) \cdot T_{max} + T_{rest} + T_p + n \cdot 1\mu s \quad (4.4)$$

Il numero dei pacchetti con la dimensione massima che la variabile  $k$  richiede è ottenuto dal quoziente della quantità di dati da inviare ed il carico utile (payload) massimo in ogni pacchetto, 1488 byte.

$$k - 1 = \left\lfloor \frac{np}{1488} \right\rfloor \quad (4.5)$$

$T_{rest}$  è il tempo di inoltro di un pacchetto che non è di dimensione massima, calcolabile nel seguente modo:

$$T_{rest} = \frac{np - 1488 \left\lfloor \frac{np}{1488} \right\rfloor}{1488} \quad (4.6)$$

Infine, il tempo di inoltro per il pacchetto di dimensione massima è ottenuto da:

$$T_{max} = \frac{1538}{bw} \quad (4.7)$$

E' possibile costruire topologie di collegamento molto complesse che logicamente possono essere ricondotte ad una topologia lineare. I tempi di ciclo ottenuti in questo caso sono quasi uguali a quelli ottenuti per la topologia lineare vista in precedenza con lo stesso numero di dispositivi slave introducendo leggeri ritardi che possono essere ignorati. Come descritto nella sezione delle varie topologie di rete ammesse da EtherCAT, è possibile realizzare una topologia a stella con switch store and forward tradizionali. I calcoli visti per la topologia lineare vanno leggermente modificati.

Il risultato dipende dal tempo di inoltro di tutti i pacchetti dal master, il tempo di inoltro di tutti i dispositivi switch ed il ritardo di inoltro degli slave.

Il calcolo del tempo minimo di ciclo per una topologia a stella composta da diverse topologie lineari collegate ad uno switch, il quale è collegato al master, può essere ottenuto dalla seguente formula:

$$T_s^{ETC} = \sum_k T_m^f + 2T_{switch}^f + T_l^{ETC} + \sum T_n \quad (4.8)$$

dove  $\sum T_n$  è la somma di tutti i ritardi sul mezzo fisico e i ritardi di propagazione sui cavi della rete mentre  $T_{switch}^f$  è il tempo di inoltro di un pacchetto sullo switch.

## 4.5 Sicurezza funzionale

Oltre allo scambio deterministico dei dati di controllo, è possibile scambiare sullo stesso mezzo trasmissivo le informazioni per la sicurezza funzionale.

Per questo tipo di trasferimento viene implementato il protocollo Safety over EtherCAT

(FSOE : Fail Safe over EtherCAT).

Il protocollo è sviluppato seguendo la normativa IEC 61508. Un'analisi dettagliata di questa normativa va oltre gli scopi di questa tesi. Di seguito si vuole comunque dare qualche informazione sul contenuto della norma.

IEC 61508 è uno standard ingegneristico internazionale che descrive delle procedure su come applicare, progettare, implementare e mantenere sistemi di protezione automatici su dispositivi elettronici come sistemi elettromeccanici, sistemi elettronici allo stato solido e sistemi basati su PC, utilizzati in ambienti con problematiche di sicurezza.

Come noto, lo sviluppo di sistemi critici, ovvero sistemi che in caso di fallimento sul funzionamento possono portare a morte, danni fisici a persone e/o materiali ed a danni ambientali, non richiede solo un processo di testing ma anche un rigoroso processo sullo sviluppo, basato sul concetto di *Safety Integrity Level* (SIL).

Ogni sistema viene valutato in termini di probabilità di un suo fallimento e sulle conseguenze che ne derivano.

Nella figura di seguito, vengono presentati i vari livelli di sicurezza definiti dallo standard.

Tolerable Hazard Rate (THR) per hour	Failure mode	SIL	Hazard (If system is uncontrollable)
$\geq 10^{-9}$ to $< 10^{-8}$	The system cannot be recovered by the operator.	4	Catastrophic and fatal outcome
$\geq 10^{-8}$ to $< 10^{-7}$	Only an experienced operator can recover the system.	3	Critical and fatal outcome.
$\geq 10^{-7}$ to $< 10^{-6}$	An operator can recover the system.	2	Marginal, Injuries may occur.
$\geq 10^{-6}$ to $< 10^{-5}$	Minor availability issues. Always recoverable.	1	Negligible, Minor injuries may occur.
	There is no safety requirement on the system.	0	Nuisance, Dissatisfying to the user.

Immagine 4.13: Safety Integrity Level

Da questa norma derivano altre specifiche per diversi settori di applicazione o sottosistemi, come:

- **ISO 26262**: è un adattamento dello standard IEC 61508 per la sicurezza funzionale nei dispositivi elettrici ed elettronici nelle automobili.
- **IEC 62279**: fornisce un'interpretazione specifica per le applicazioni ferroviarie.
- **IEC 61511**: definisce le pratiche di sviluppo della sicurezza funzionale per sistemi industriali del settore chimico, petrolchimico e farmaceutico.

- **IEC 61513**: fornisce requisiti e raccomandazioni per la strumentazione ed il controllo della sicurezza nelle centrali nucleari.
- **IEC 62061**: fornisce requisiti per la progettazione dei sistemi di sicurezza su macchinari industriali.

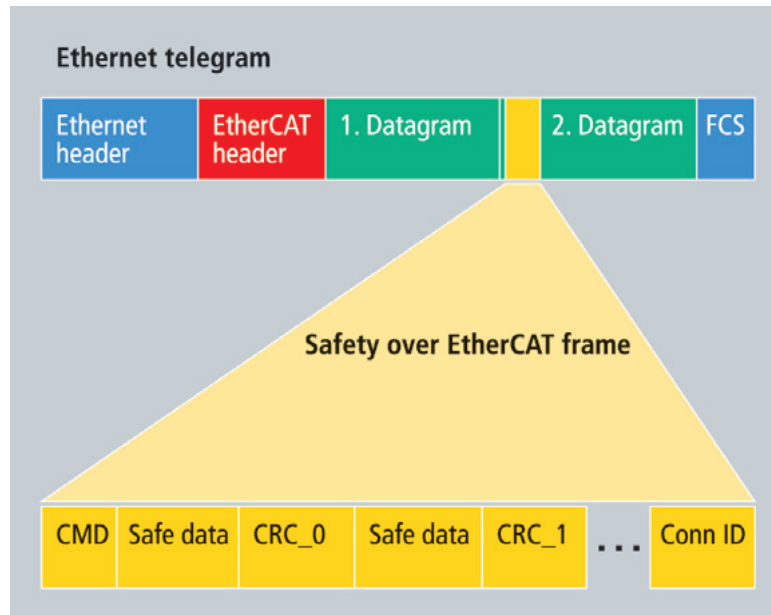


Immagine 4.14: Fail Safe over EtherCAT

L'applicazione safety sviluppata all'interno del master EtherCAT, incapsula il frame contenente i dati di sicurezza all'interno dello stesso frame che contiene i dati di processo del dispositivo. I frame di sicurezza (Safety Container) contengono i dati di processo critici. Ogni 2 byte di dati safety, vengono inseriti 2 byte per il controllo CRC.

## 4.6 Implementazione master

Come visto in precedenza, i dati scambiati si appoggiano su frame Ethernet standard IEEE 802.3.

Per l'implementazione di un dispositivo master non devono essere utilizzate schede speciali, ma il requisito minimo è un'interfaccia Ethernet o un'interfaccia di rete standard (NIC). Questo permette un accesso diretto alla memoria del controllore (DMA) senza coinvolgere l'unità di controllo.

Un'implementazione puramente software di un master EtherCAT può essere implementata su una vasta gamma di sistemi operativi real time (RTOS) come: eCOS, INtime, MICROWARE OS-9, MQX, On Time RTOS-32, Proconos OS, Real-Time Java, RT Kernel, TR-Linux, RTX, RTXC, RTAI Linux, PikeOS, Linux con RT-Preempt, QNX,

L'immagine di processo non viene mappata nel master EtherCAT ma negli slave gravando ulteriormente meno la CPU. Inoltre sono disponibili numerose implementazioni open source.

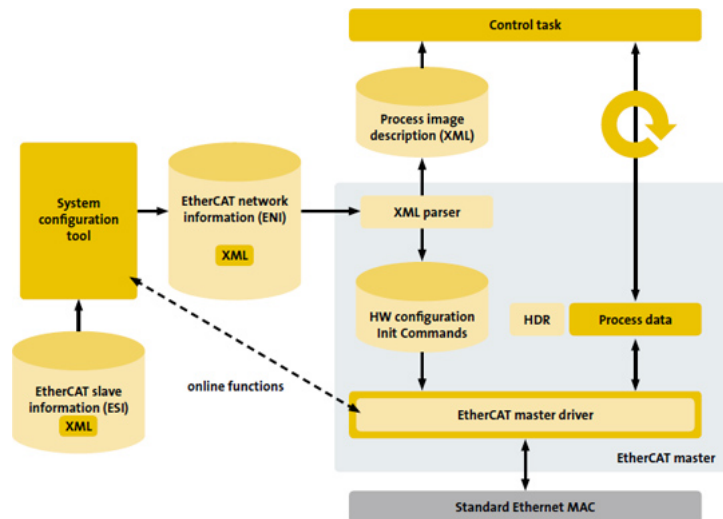


Immagine 4.15: Implementazione di un dispositivo master

## 4.7 Implementazione slave

I dispositivi EtherCAT slave utilizzano un EtherCAT Slave Controller (ESC) di basso costo sotto forma di ASIC, FPGA, o integrati in microcontrollori standard. Slave semplici non richiedono neppure un microcontrollore, in quanto ingressi e uscite digitali possono essere connessi direttamente all'ESC. Per dispositivi slave più complessi, le prestazioni della comunicazione dipendono solo in minima parte dalle prestazioni del microcontrollore, e nella maggior parte dei casi è sufficiente un controllore a 8-bit. La configurazione hardware è salvata in una memoria non volatile (ad esempio: EE-

PROM) chiamata *Slave Information Interface* (SII), la quale contiene informazioni relative alle funzionalità elementari del dispositivo che il master può leggere durante la fase di avvio per poter gestire lo slave anche in assenza del file descrittivo del dispositivo. Il file EtherCAT Slave Information (ESI) fornito insieme allo slave e basato su formato XML contiene la descrizione completa delle proprietà dello slave, come i dati ciclici di processo e le loro opzioni di mappatura, oppure le modalità di sincronizzazione supportate. Il software di configurazione della rete utilizza queste informazioni per la definizione della struttura della rete stessa.

## 4.8 Conformità e certificazione

Per ottenere uno standard di comunicazione di successo è necessario rispondere ai requisiti di conformità e interoperabilità.

Il consorzio *EtherCAT Technology Group* offre diversi servizi per mantenere ed ottenere questi requisiti. Di seguito viene fornita una breve illustrazione.

- **Plug Fest:** ogni anno vengono organizzati degli incontri dove sviluppatori di master e slave si incontrano per verificare come i loro dispositivi operino assieme, provandoli a collegare tra loro.
- **EtherCAT Conformance Test Tool:** applicativo software che permette di testare automaticamente un dispositivo slave. Viene inviato un frame al dispositivo e il software verifica che la risposta corrisponda con quella attesa.  
I test vengono definiti sotto forma di file XML che vengono rilasciati con costanti aggiornamenti e revisioni dal *TWG* (Technical Working Group).
- **Technical Working Group Conformance:** la politica di test EtherCAT richiede che i costruttori di dispositivi slave verifichino ogni propria implementazione tramite una versione valida dell'EtherCAT Conformance Test Tool prima che il prodotto venga immesso sul mercato. Il costruttore poi esegue il test internamente.  
Il *Technical Committee* (TC) ha creato un *Technical Working Group* (TWG) *Conformance* che stabilisce le procedure, contenuto ed implementazione del Conformance Test Tool. I test ed il loro livello di dettaglio sono in continua estensione.
- **EtherCAT Test Center:** strutture che permettono di eseguire l'EtherCAT Conformance Test ufficiale. Questo include tutti i test automatizzati effettuati con il software CTT e il test di interoperabilità all'interno di una rete. Il superamento di queste verifiche rilascia un certificato di *EtherCAT Conformance Tested*.

## Capitolo 5

# Analisi dei protocolli

### 5.1 Diagnostica con EtherCAT

EtherCAT comprende una vasta gamma di funzionalità diagnostiche inerenti al sistema che aiutano a rilevare e localizzare con precisione gli errori sulla rete. Fondamentalmente, le caratteristiche diagnostiche di un bus di campo dovrebbero soddisfare due principali requisiti:

- **Risposta veloce**

Il sistema di controllo dovrebbe essere in grado di rilevare immediatamente una condizione anomala nella rete, al fine di reagire il più rapidamente possibile e prevenire danni meccanici, perdita di materiali o sincronismo all'interno dell'applicazione.

- **Analisi precisa**

La rete dovrebbe fornire al controller informazioni dettagliate sullo stato, consentendo all'applicazione master di determinare la causa di errore presente.

Grazie alle sue proprietà di protocollo e alla sua struttura, EtherCAT soddisfa entrambi i requisiti allo stesso modo.

In questa sezione vengono analizzati i frame che viaggiano su una semplice rete in cui un dispositivo master invia i comandi necessari a pilotare con precisione un servozionamento il quale traducendo i segnali digitali in segnali elettrici, comanda un motore con estrema precisione (ad esempio il motore potrebbe essere l'attuatore di un braccio robotizzato).

In una rete EtherCAT ogni dispositivo slave possiede un file di descrizione per permettere al master di conoscere quali dati deve inviare e ricevere sulla rete per poterlo pilotare. Questo file è chiamato ESI (EtherCAT Slave Identification) ed è composto da diversi campi:

- **Vendor:** questo campo fornisce le informazioni generali sul costruttore.
- **Groups:** vengono utilizzati dal tool di configurazione per ordinare i parametri del dispositivo.

- **Devices:** in questa sezione sono contenuti tutti i parametri del dispositivo.

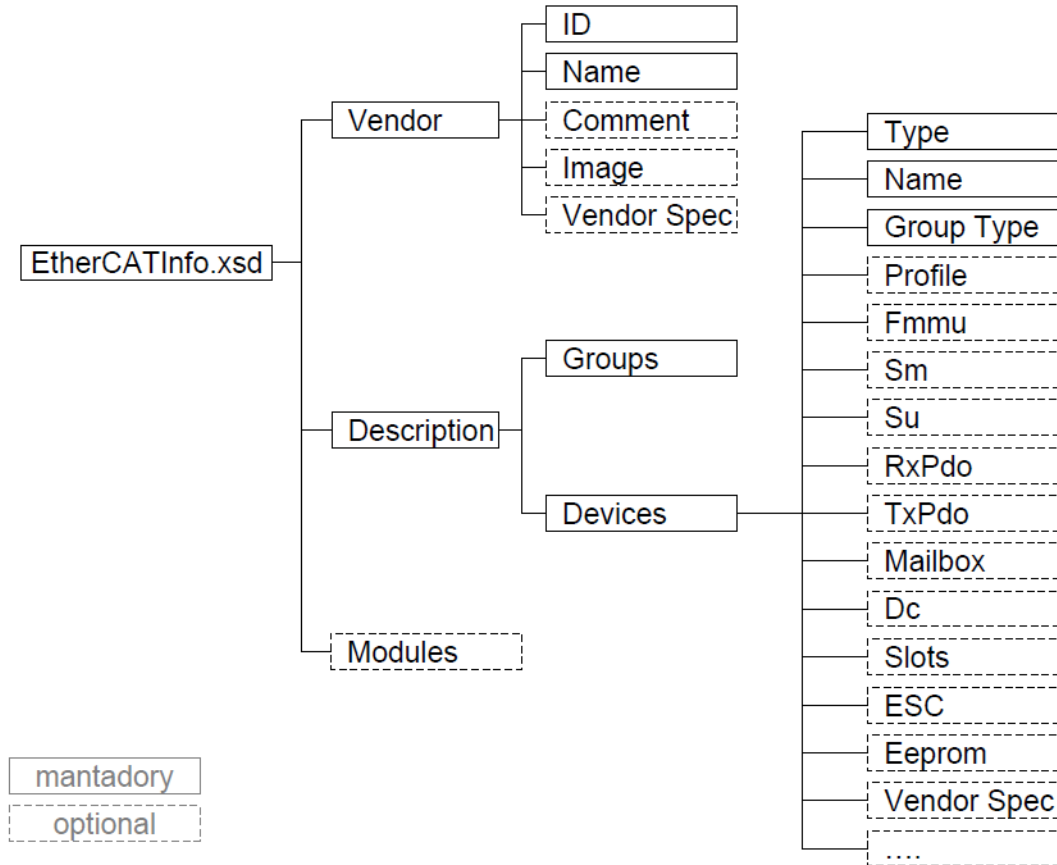


Immagine 5.1: File di descrizione del dispositivo slave in formato XML

### 5.1.1 EtherCAT con Wireshark

Utilizzando Wireshark è possibile analizzare i frame EtherCAT che viaggiano sulla rete tra i vari dispositivi. E' giusto ricordare che inserire uno switch per catturare i pacchetti compromette il funzionamento real-time della rete.

Wireshark mette a disposizione specifici comandi per l'analisi della rete.

I filtri per la visualizzazione disponibili sono :

- **ecat:** mostra solo il traffico basato su EtherCAT.
- **ecat.ado == 0x130:** filtro su uno specifico offset ad un determinato indirizzo (ado).
- **ecat.adp == 0x3e9:** filtro su uno slave specifico in base alla sua posizione o indirizzo fisso (adp).



- `ecat.cmd == 0x1`: filtro su uno specifico comando (in questo esempio APRD).
- `ecat.sub1.idx < 0x80`: filtro su uno specifico comando di un campo secondario.
- `ecat_mailbox`: visualizza solo il traffico di tipo mailbox.

I filtri di cattura impostabili sono:

- `etere proto 0x88a4`: cattura solo il traffico Ethercat-over-Ethernet.
- `udp port 0x88a4`: cattura solo il traffico Ethercat-over-UDP.

### 5.1.2 Struttura dello slave

Come descritto nelle sezioni precedenti, EtherCAT può trasmettere protocolli differenti. Il servoazionamento utilizzato (OMRON serie 1S) utilizza il profilo CiA 402 (IEC 61800-7 CAN Application Protocol over EtherCAT). Il dizionario degli oggetti (Object Dictionary) implementato a livello applicazione contiene dati ed i parametri del dispositivo, nonché le informazioni sulla mappatura tra l'interfaccia dei dati di processo e la parte applicativa del dispositivo. E' formato da tutti gli oggetti che formano il dispositivo, ovvero delle rappresentazioni astratte di particolari componenti (Communication Object, Application Object) formati da dati, parametri e metodi. L'oggetto PDO è formato dagli oggetti presenti nel dizionario degli oggetti che possono essere utilizzati per lo scambio dei dati di processo. La comunicazione ciclica legge e scrive l'oggetto PDO. Le comunicazioni di tipo mailbox (SDO) utilizzano dei messaggi asincroni in cui è possibile leggere e scrivere tutti gli elementi del dizionario degli oggetti.

L'oggetto per la mappatura dei PDO (PDO mapping) permette di interfacciare gli oggetti per la comunicazione ciclica con il dizionario degli oggetti utilizzati dal dispositivo.

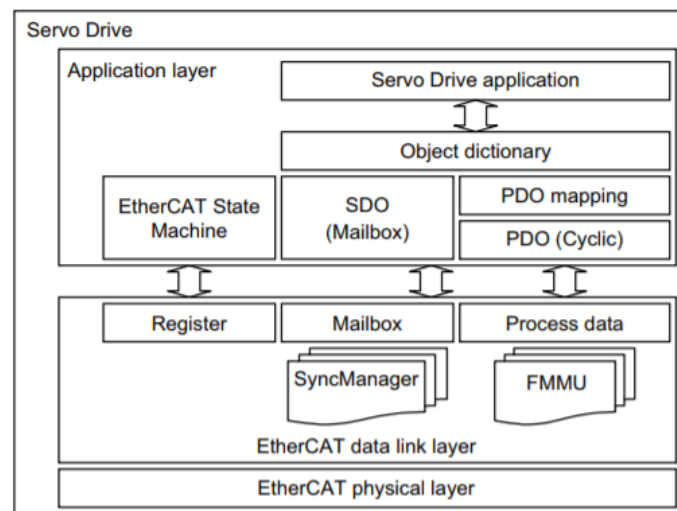


Immagine 5.2: Struttura del dispositivo slave in esame

### 5.1.3 Analisi frame EtherCAT

In questa configurazione di rete, oltre ai campi del file di descrizione del dispositivo, possiamo notare la mappatura dei dati PDO di processo in ingresso ed in uscita dal dispositivo.

Alcuni dei principali valori in ingresso sono:

- 0x6040: word di controllo.
- 0x6060: modalità richiesta dal dispositivo.
- 0x607A: posizione richiesta.
- 0x60FF: velocità richiesta.
- 0x6071: coppia richiesta.

Tra i principali parametri in uscita invece possiamo notare:

- 0x6041: word di stato.
- 0x6061: modalità assunta dal dispositivo.
- 0x6064: valore attuale della posizione.
- 0x6077: valore attuale della coppia.

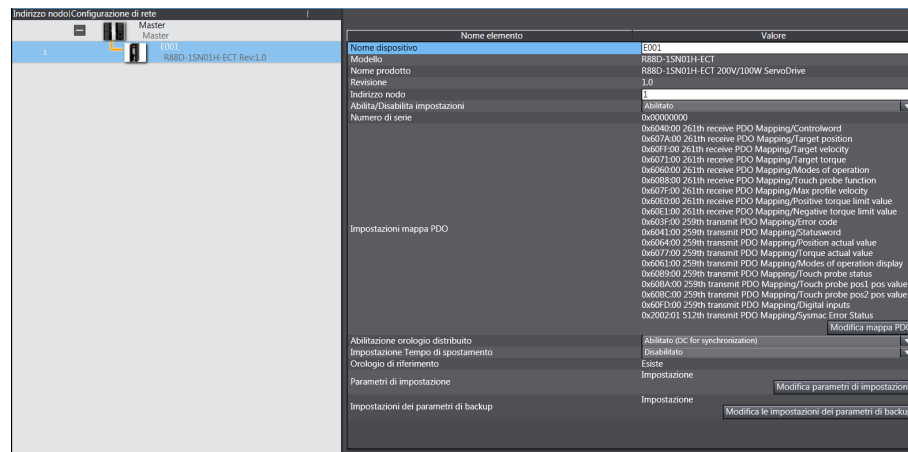


Immagine 5.3: Configurazione rete in esame

In questa semplice applicazione viene analizzato lo scambio di messaggi durante un tipico scenario presente in un controllo automatico. Inizialmente il master non invia nessun comando allo slave. Dopo che un particolare comando (*MC\_Power*) viene inviato dal master verso lo slave per consentirne l'abilitazione, vengono spediti i successivi comandi

di start con un riferimento di posizione (*MC\_MoveVelocity*) per consentire all'attuatore collegato al motore di portarsi ad una determinata posizione.

```

1 // Makes a Servo Drive ready to operate
2 PowerAxis(Axis:=Axis_1, Enable:=Power_Slave);
3
4 // Performs velocity control with the Position Control Mode of the Servo Drive.
5 MoveVel(Axis:=Axis_1, Execute:=Start, Velocity:=VEI, Direction:=_eMC_DIRECTION#_mcPositiveDirection);
6
7 // Decelerates an axis to a stop.
8 StpAxis(Axis:=Axis_1, Execute:=Start, Deceleration:=Acc);
9

```

Immagine 5.4: Comandi di movimento inviati allo slave

A livello di frame Ethernet, tutte le comunicazioni EtherCAT sono di tipo broadcast. L'indirizzamento dei nodi avviene a livello sub-frame. Per questo motivo solo il master possiede un MAC address, mentre gli slave ne sono privi. Analizziamo il seguente frame inviato dal master verso gli slave.

#### Intestazione Ethernet

- Destination: broadcast, il frame è indirizzato a tutti i dispositivi slave presenti in rete.
- Source: MAC address del dispositivo master.
- Type: indica che all'interno del frame è presente il protocollo EtherCAT 0x88a4.

#### Intestazione frame EtherCAT

- Length: lunghezza del telegramma EtherCAT senza considerare il campo di controllo FCS, in questo caso 71 byte.
- Res: campo riservato.
- Type: sono supportati esclusivamente i comandi EtherCAT, 0x1.

All'interno di questo telegramma EtherCAT, sono presenti 3 datagrammi.

#### Datagramma 1

- Header datagramma
  - Cmd: comando 12 (Logical Read Write).
  - Index: non viene modificato dallo slave.
  - Log Addr: indirizzo logico 0.
  - Length: la lunghezza dei dati che seguono all'interno di questo datagramma è di 26 byte.

- C: il valore 0 indica che il frame non circola.
- M: il valore 1 segue un ulteriore datagramma.
- Data: contiene i dati che devono essere letto o scritti.
- WC: valore del contatore impostato su 0. Ogni slave lo incrementa in base alle operazioni che effettua.

#### Datagramma 2

- Header datagramma
  - Cmd: comando 10 (Logical Read).
  - Index: non viene modificato dallo slave.
  - Log Addr: indirizzo logico 2048.
  - Length: la lunghezza dei dati che seguono all'interno di questo datagramma è di 1 byte.
  - C: il valore 0 indica che il frame non circola.
  - M: il valore 1 segue un ulteriore datagramma.
- Data: contiene i dati che devono essere letti o scritti.
- WC: valore del contatore impostato su 0. Ogni slave lo incrementa in base alle operazioni che effettua.

#### Datagramma 3

- Header datagramma
  - Cmd: comando 13 (Auto Increment Read Multiple).
  - Index: non viene modificato dallo slave.
  - Slave Addr: indirizzo slave 0. Il primo slave della rete ha indirizzo 0 (adp = 0).
  - Offset Addr: indirizzo registro 2320 (ado = 0x910).
  - Length: la lunghezza dei dati che seguono all'interno di questo datagramma è di 8 byte.
  - C: il valore 0 indica che il frame non circola.
  - M: il valore 0 indica che non sono presenti ulteriori datagrammi.
- DC SysTime(0x910): viene richiesta la lettura del clock sullo slave.
- DC SysTime L (0x910): identifica i 4 byte inferiori del clock.
- DC SysTime H (0x914): identifica i 4 byte superiori del clock.

- WC: valore del contatore impostato su 0. Ogni slave lo incrementa in base alle operazioni che effettua.

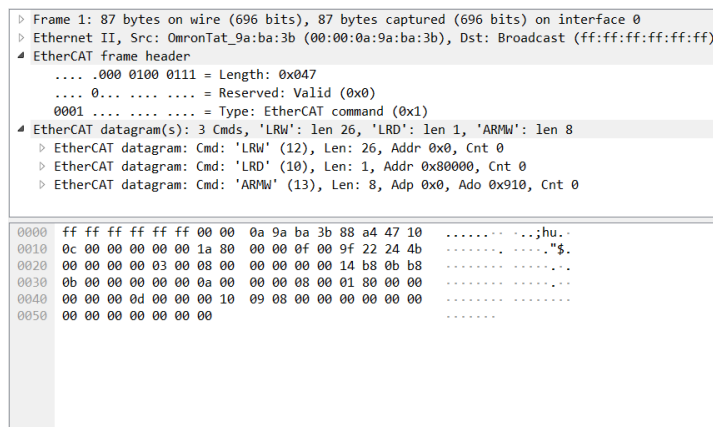


Immagine 5.5: Cattura frame inviato dal master verso la rete

La configurazione dei PDO impostati per lo scambio dati, avviene durante la fase di avvio della comunicazione, in cui lo slave si trova nello stato di Pre-Op. Per questa operazione viene sfruttata la mailbox per l'invio di comandi CANopen (CoE).

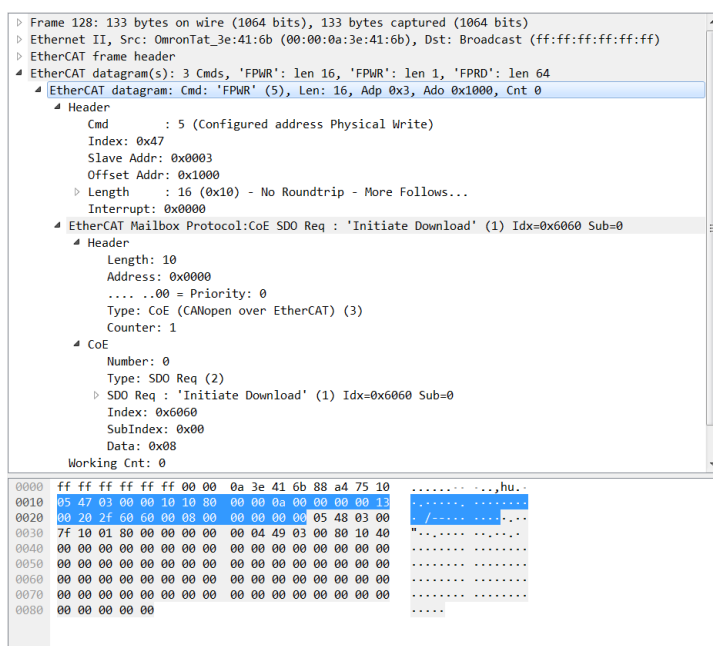


Immagine 5.6: Configurazione PDO durante l'avvio della comunicazione

Analizziamo in dettaglio il campo Data del Datagramma 1, in quanto all'interno di questo sono contenuti i dati di processo PDO tra il dispositivo master e tutti gli slave della rete.

I PDO sono trasmessi nello stesso ordine in cui vengono configurati. Quando un comando richiede la scrittura e la lettura di dati il frame viene dimensionato in base pacchetto PDO più grande. Nel caso in esame, i dati inviati in scrittura dal master verso lo slave hanno una dimensione di 23 byte, mentre i dati inviati dallo slave verso il master, hanno una dimensione di 26 byte. Il frame ha quindi una dimensione di 26 byte. Nel frame di scrittura vengono inseriti 3 byte di padding per raggiungere la stessa dimensione del frame di lettura.

Il campo Data del frame di scrittura è strutturato nel seguente modo:

- 0f 00: Control word (0x6040).
- fc b1 e6 26: Target position (0x607A).
- 00 00 00 00: Target speed (0x60FF). Non viene considerata in modalità CSP (modalità predefinita di controllo assi sulle CPU in esame). Questa modalità prevede l'invio di una nuova target position ad ogni ciclo EtherCAT.
- f4 f7: Target torque (0x6071).
- 08: Modes of operation (0x6060).
- 00 00: Touch probe function (0x60B8). Questa è la word di controllo per le operazioni di latch della posizione.
- 00 00 00 14: Max profile velocity (0x607F).
- b8 0b: Positive torque limit (0x60E0).
- b8 0b: Negative torque limit (0x60E1).
- 00 00 00: byte di padding.

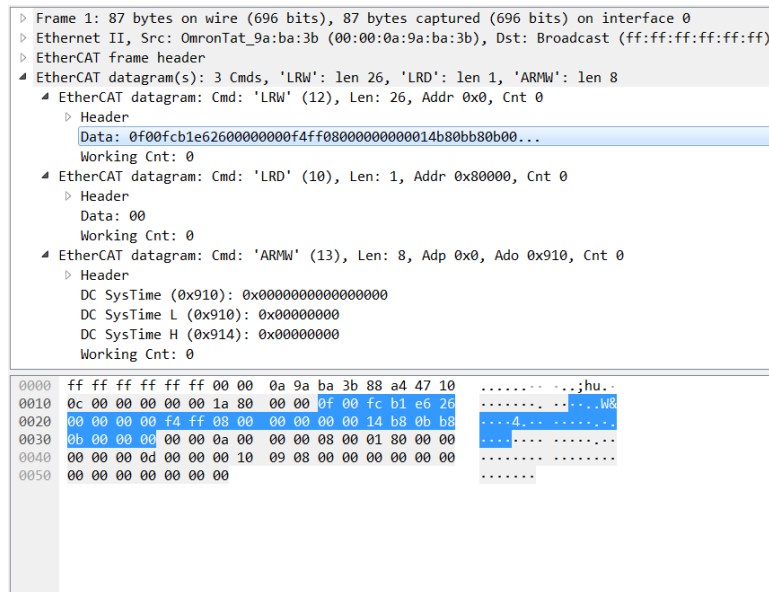


Immagine 5.7: Campo Data nel frame di scrittura

Il campo Data del frame in lettura è strutturato nel seguente modo:

- 00 00: Error code (0x603F).
- 37 12: Status word (0x6041).
- 3d a6 e6 26: Position actual value (0x6064).
- f4 ff: Torque actual value (0x6077).
- 08: Modes of operation display (0x6061).
- 00 00: Touch probe status (0x60B9).
- 00 00 00 00: Touch probe pos1 pos value (0x60BA).
- 00 00 00 00: Touch probe pos2 pos value (0x60BC).
- 00 00 00 1c: Digital inputs (0x60FD).
- 01: Sysmac error status. (0x2002).

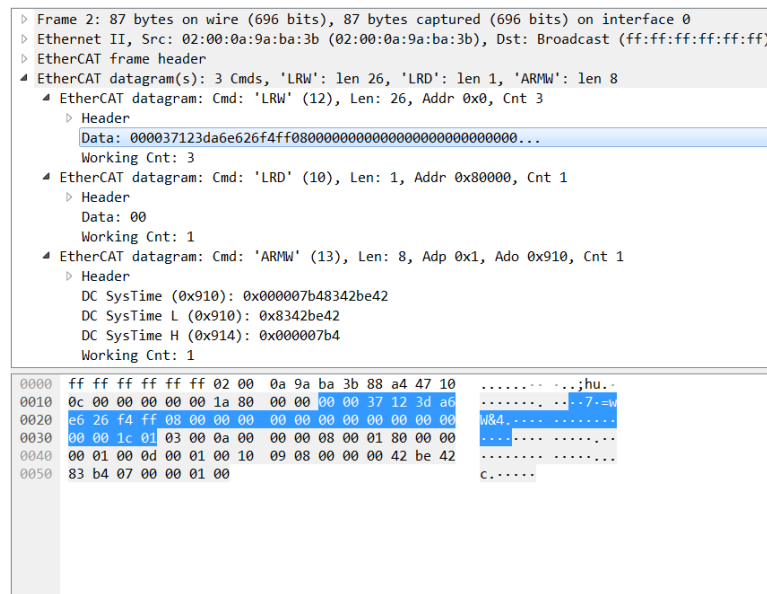


Immagine 5.8: Campo Data nel frame di lettura

Di seguito viene riportata la mappatura dei PDO configurati in questo caso applicativo.

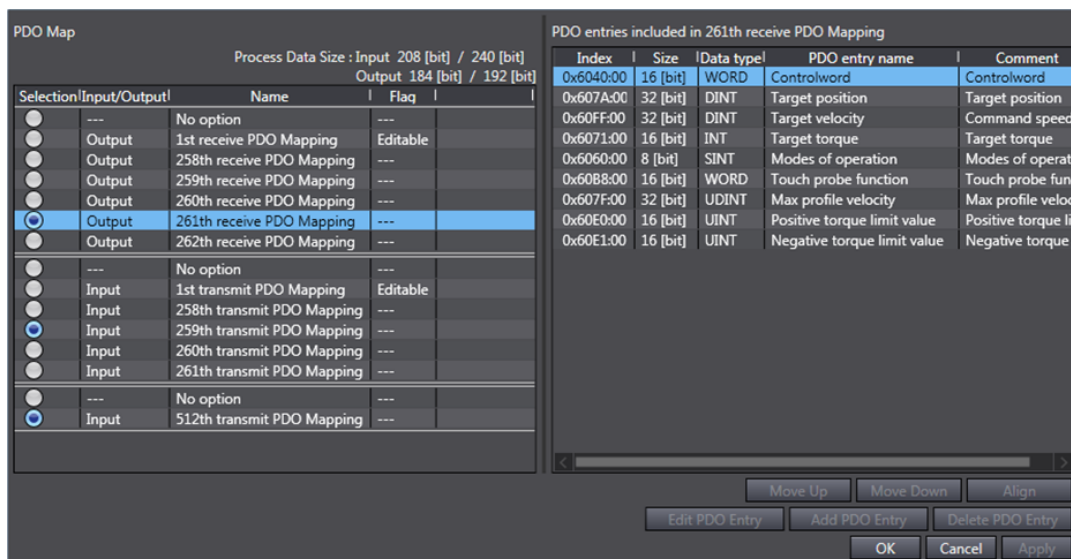


Immagine 5.9: PDO da master verso slave



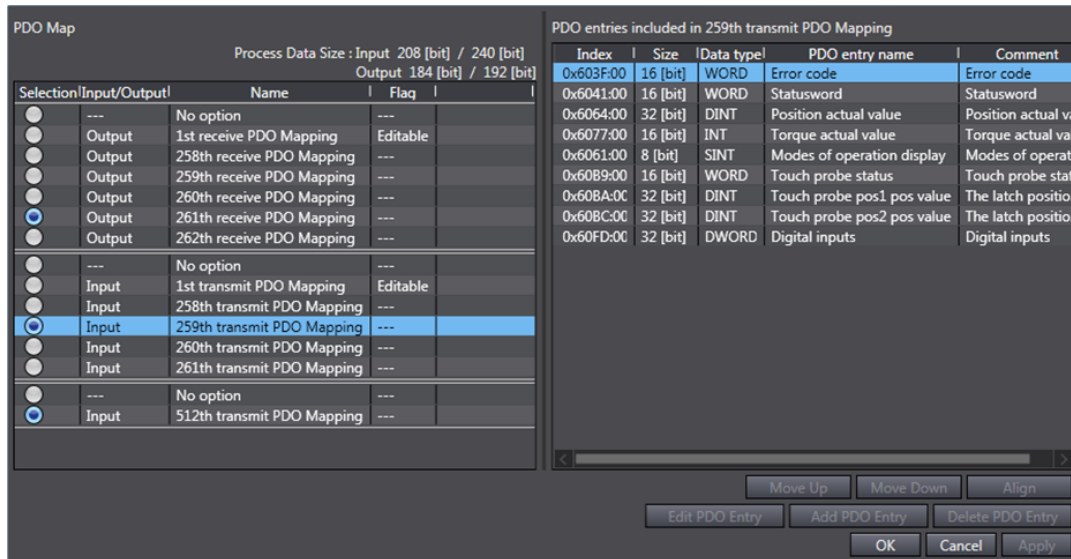


Immagine 5.10: PDO da slave verso master

## 5.2 Diagnostica con EtherNet/IP

### 5.2.1 EtherNet/IP con Wireshark

Come descritto nella sezione relativa al protocollo EtherNet/IP, i pacchetti utilizzati per lo scambio di dati real-time della rete EtherNet/IP sono di tipo UDP, mentre la comunicazione esplicita è basata su TCP.

In questa analisi, viene analizzato il flusso dei dati tra due dispositivi, un PLC ed un variatore di frequenza per motori asincroni (OMRON MX2).

Per prima cosa viene descritta la struttura del variatore di frequenza ed il campo *Data* del pacchetto UDP contenente il protocollo CIP. Successivamente viene analizzato lo scambio dei messaggi impliciti durante l'invio di un comando per un movimento ad una determinata velocità.

### 5.2.2 Profilo AC Drives 0x02

Il variatore di frequenza utilizzato implementa le funzionalità definite dal profilo *AC Drives 0x02* definite dallo standard EtherNet/IP. Questo tipo di dispositivo mette a disposizione diversi oggetti assembly come blocchi di indirizzamento, tutti con funzionalità differenti.

In questa applicazione viene utilizzato l'oggetto assembly *Extended Speed Control* (Assembly 21) per la parte di comando, mentre un oggetto *Extended Speed Status* (Assembly 71) per la parte di controllo.

Di seguito segue la descrizione dei vari campi che compongono questi oggetti.

### 5.2.2.1 Assembly Object ID 21: Extended Speed Control Output

Questo oggetto raccoglie i parametri relativi al comando del variatore indirizzati dal PLC. E' composto da 4 byte. I primi 2 byte corrispondono alla word di controllo mentre gli ultimi 2 sono relativi alla velocità di comando.

Ogni bit della word di comando si riferisce ad uno specifico parametro del dispositivo:

- 0: RunFwd. Imposta la direzione di movimento avanti (parametro 1000).
- 1: RunRev. Imposta la direzione di movimento all'indietro (parametro 1003).
- 2: FaultRst. Permette di cancellare gli allarmi presenti sul dispositivo. (parametro 1001).
- 5: NetCtrl. Imposta il riferimento per la sorgente di avvio (parametro 1004).
- 6: NetRef. Imposta la modalità di selezione del riferimento di velocità (parametro 1005).

I 2 byte rimanenti fanno riferimento alla velocità di controllo, SpeedRef (parametro 1002).

### 5.2.2.2 Assembly Object ID 71: Extended Speed Control Input

Questo oggetto raccoglie i parametri relativi allo stato del variatore di frequenza. E' composto da 6 byte. I primi 2 byte corrispondono alla word di stato, i successivi identificano lo stato del dispositivo mentre gli ultimi 2 sono relativi all'effettiva velocità di movimento.

Ogni bit della word di stato si riferisce dal uno specifico parametro del dispositivo:

- 0: Faulted. Indica la presenza di un errore sul dispositivo (parametro 1006).
- 1: Warning. Indica la presenza di un avviso sul dispositivo (parametro 1009).
- 2: Running 1. Indica che il dispositivo si sta muovendo in avanti (parametro 1007).
- 3: Running 2. Indica che il dispositivo si sta muovendo all'indietro (parametro 1010).
- 4: Ready. Il dispositivo è pronto per essere avviato. (parametro 1011).
- 5: CtrlFromNet. Indica il tipo di riferimento per la sorgente di comando impostato (parametro 1012).
- 6: RefFromNet. Indica il tipo di riferimento per il comando di avvio impostato (parametro 1013).
- 7: AtReference. Indica che il dispositivo ha raggiunto la velocità impostata (parametro 1014).

Gli ultimi 2 byte si riferiscono alla velocità attuale di controllo, SpeedActual (parametro 1008).

### 5.2.3 Analisi frame EtherNet/IP

Entrambi i dispositivi (PLC e variatore) instaurano una connessione. Questo fa sì che entrambi si comportino sia da produttore che da consumatore di dati sulla rete. Tralasciamo la descrizione dei campi della parte UDP, concentrandosi invece sulla parte CIP.

#### Intestazione del protocollo CIP trasportato nel pacchetto UDP

- Item Count: questi 2 byte rappresentano il numero di Common Packet Format a seguire. Nella messaggistica implicita questo campo ha sempre valore 2 (2 byte).
- Type ID: rappresenta l'identificativo dei dati trasportati (2 byte).
- CIP header Length: indica la lunghezza dell'intestazione CIP, incluso il campo Connection Identifier e Sequence Number (2 byte).
- Connection Identifier: contiene l'identificativo della comunicazione (4 byte).
- Encapsulation Sequence Number: indica la lunghezza dei pacchetti per la connessione attuale (4 byte).
- Type ID: indica il tipo di dati utilizzato (4 byte).
- Data Length: lunghezza del campo dati (4 byte).
- Data: contiene i dati dell'applicazione. La grandezza varia a seconda dell'applicazione e dal dispositivo.

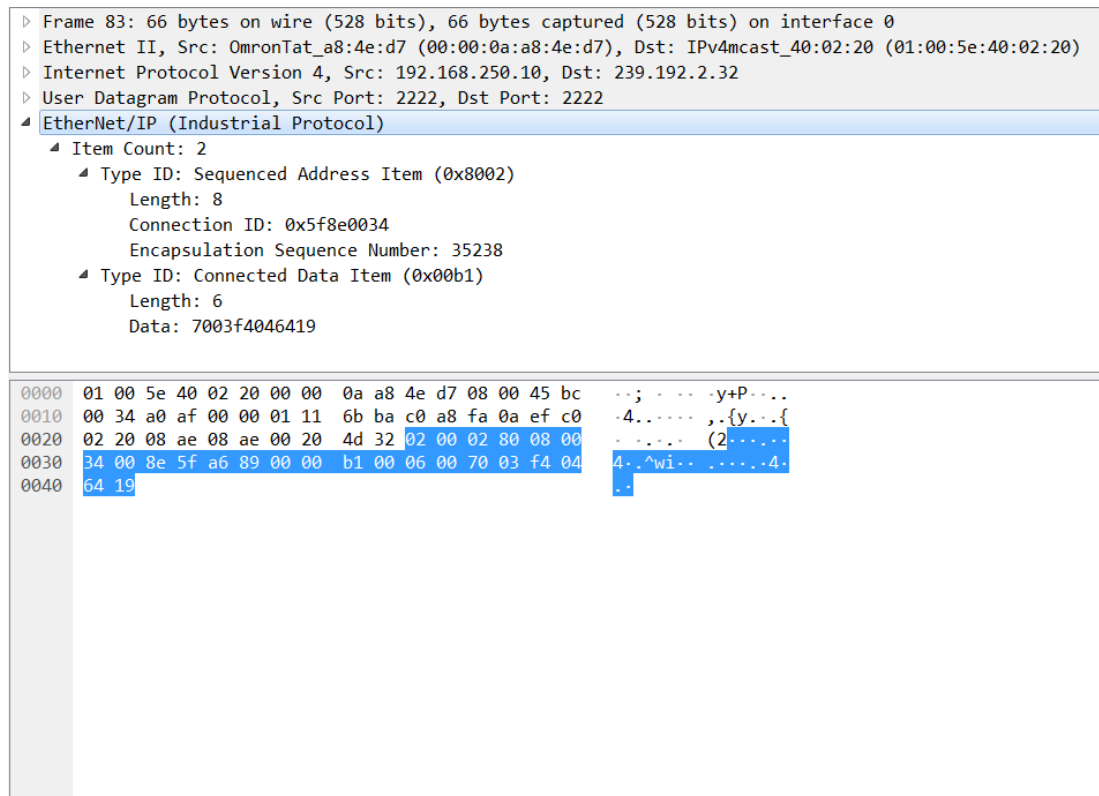


Immagine 5.11: Cattura frame destinato al PLC

Di seguito viene descritto il contenuto di ogni campo che compone l'intestazione ed il campo dati CIP.

- 02 00: come sempre nel protocollo UDP CIP questo campo ha valore 2.
- 02 80: identificativo dei dati trasportati.
- 08 00: identifica una lunghezza dell'intestazione CIP di 8 byte.
- 34 00 8e 5f: identifica la connessione 5f 8e 00 34.
- a6 89: indentifica il numero sequenziale 89 a6 (35238).
- b1 00: indica una comunicazione con connessione.
- 06 00: lunghezza dei campo dati CIP di 38 byte.
- 70 03 f4 04 64 19: campo dati.

In particolare il campo dati CIP è composto dalle seguenti parti:

- 70 03: numero di sequenza. Consente al master di riconoscere eventuali pacchetti duplicati.

- f4: word di stato.
- 04: stato del dispositivo (4 = abilitato).
- 64 19: velocità di movimento.

# Conclusioni

In questa tesi sono stati analizzati alcuni concetti chiave del settore dell'automazione industriale: le reti ed i protocolli di comunicazione utilizzati in ambienti real-time.

Partendo dai requisiti che devono essere soddisfatti da una rete implementata in una macchina automatica industriale, sono stati analizzati due dei tanti protocolli che si sono affermati nel settore: EtherNet/IP e EtherCAT.

In particolare, è stato analizzato come è possibile modificare lo standard Ethernet affermatosi nei sistemi IT (Information Technology) tradizionali per adattarlo al livello operativo OT (Operational Technology) delle linee produttive.

L'analisi del protocollo EtherNet/IP ha permesso di analizzare il primo approccio implementativo di Ethernet come bus di campo. In particolare, è stata analizzata la sua implementazione a livello applicazione (livello 7) della pila ISO/OSI.

L'analisi del protocollo EtherCAT invece, ha permesso di analizzare il terzo approccio implementativo dove le funzionalità richieste vengono aggiunte a livello di collegamento (livello 2).

Sucessivamente è stato analizzato uno scambio di messaggi in rete tra dispositivi per il controllo industriale (PLC e controllori per motori elettrici) tramite il software Wireshark, allo scopo di comprendere nel dettaglio la struttura e la composizione dei dati trasmessi.

## 5.3 Sviluppi futuri

Il presente elaborato si focalizza solamente su due dei diversi protocolli di comunicazione sviluppati per il settore dell'automazione.

Un possibile sviluppo potrebbe essere l'estensione dell'analisi dei restanti protocolli per valutarne l'implementazione e le caratteristiche.

In aggiunta, le prestazioni potrebbero essere testate analizzando il flusso dei dati su architetture di rete identiche basate su protocolli differenti.

# Ringraziamenti

Prima di tutto vorrei ringraziare il prof. Roberto Alfieri, relatore di questa tesi di laurea, per la disponibilità e l'aiuto che mi ha fornito durante tutto il periodo di stesura.

Un grande ringraziamento alla mia famiglia che mi ha sempre sostenuto durante questo percorso universitario.

Ringrazio tutti i miei colleghi di lavoro, ed in particolare Andrea e Francesco di OM-  
RON per il materiale ed il supporto fornito.

Per ultimi, non meno importanti, i miei amici.

Nicolò Toscani

# Bibliografia

- [1] Frithjof Klasen, Volker Oestreich, Micheal Volz (Eds.). *Industrial Communication with Fieldbus and Ethernet*. VDE VERLAG GMBH, Berlin, Germany, 2011.
- [2] Andrew S. Tanenbaum, David J. Wetherall. *Reti di calcolatori*. PEARSON, Milano-Torino, Italia, 2011.
- [3] Behrouz A. Forouzan, Firouz Mosharraf. *Reti di calcolatori: un approccio top-down*. McGraw-Hill, Milano, Italia, 2013.
- [4] Claudio Bonivento, Luca Gentili, Andrea Paoli. *Sistemi di automazione industriale*. McGraw-Hill, Milano, Italia, 2011.
- [5] Pasquale Chiacchio, Francesco Basile. *Tecnologie informatiche per l'automazione*. McGraw-Hill, Milano, Italia, 2004.
- [6] EtherCAT Technology Group. *EtherCAT - il Fieldbus Ethernet*. ETG Headquarters, Norimberga, Germania, 2018.
- [7] EtherCAT Technology Group. *EtherCAT Communication: Communication Principles*. ETG Headquarters, Norimberga, Germania, 2018.
- [8] Gunnar Prytz. *A performance analysis of EtherCAT and PROFINET IRT*. ABB AS Corporate Research Center, Billingstad, Norway, 2008.
- [9] Ludwig Winkel. *Real-Time Ethernet in IEC 61784-2 and IEC 61158 series*. Siemens AG, Karlsruhe, Germany, 2006.
- [10] George Thomas. *Introduction to Switch Technology*. The Extension, A Technical Supplement to Control Network, 2000.
- [11] George Thomas. *Introduction to Fast Ethernet*. The Extension, A Technical Supplement to Control Network, 2000.
- [12] Paula Doyle. *Introduction to Real-Time Ethernet I*. The Extension, A Technical Supplement to Control Network, 2004.
- [13] Paula Doyle. *Introduction to Real-Time Ethernet II*. The Extension, A Technical Supplement to Control Network, 2004.



- [14] E. Alessandria, L. Seno, S. Vitturi. *Performance analysis of EtherNet/IP networks*. Italian National Council of Research, Department of Information Engineering, University of Padova, 2007.
- [15] P. Ferrari, A. Flammini, D. Marioli, A. Taroni. *Sincronizzazione in reti RTE tramite Ieee 1588*. Fieldbus & Networks, Gennaio 2006.
- [16] Viktor Schiffer. *Common Industrial Protocol (CIP) and the family of CIP networks*. ODVA Inc, Ann Arbor, Michigan, USA, 2016.
- [17] ODVA, Open DeviceNet Vendor Association. *Network Infrastructure for EtherNet/IP: introduction and consideration*. ODVA Inc, Ann Arbor, Michigan, USA, 2007.
- [18] ODVA, Open DeviceNet Vendor Association. *Quick Start for Vendors Handbook. A guide for EtherNet/IP developers*. ODVA Inc, Ann Arbor, Michigan, USA, 2008.
- [19] OMRON Industrial Automation, [www.industrial.omron.it](http://www.industrial.omron.it).
- [20] Sysmac Studio, [www.industrial.omron.it/it/products/sysmac-studio](http://www.industrial.omron.it/it/products/sysmac-studio).
- [21] ODVA, [www.odva.org](http://www.odva.org).
- [22] EtherCAT Technology Group, [www.ethercat.org](http://www.ethercat.org).
- [23] BECKHOFF Information System, [www.infosys.beckhoff.com](http://www.infosys.beckhoff.com).
- [24] Wirshark, [www.wireshark.org](http://www.wireshark.org).