UNIVERSITÀ DEGLI STUDI DI ROMA

"LA SAPIENZA"

LA SAPIENZA

FACOLTÀ DI INGEGNERIA

TESI DI LAUREA

IN

INGEGNERIA ELETTRONICA

"Studio per la realizzazione del software per l'interfaccia di rete su una radio ricetrasmittente HF e validazione del relativo hardware."

Materia: Comunicazioni Elettriche

RELATORE

Ch.ma Prof.ssa M.G. Di Benedetto

LAUREANDA

Sara Giorni

CORRELATORE

Ch.mo Ing. Gianluca Vizza

ANNO ACCADEMICO 2000/2001

INDICE

Introduzione

Di anno in anno, le reti di calcolatori sono diventate sempre più importanti, e l'esigenza di poter usufruire di questo mezzo di comunicazione, oramai standardizzato, ha favorito l'introduzione di una interfaccia di rete su numerosi apparati elettronici.

Il lavoro presentato, eseguito presso la **Marconi Mobile**, si basa sulla validazione hardware e la realizzazione software di una interfaccia di rete su una Stazione radio ricetrasmittente HF.

Quest'ultima, in dotazione all'Esercito Italiano, è stata progettata con lo scopo di rispondere all'esigenza di soddisfare nuovi requisiti d'impiego dei mezzi di comunicazione.

Questi nuovi requisiti hanno origine dalla necessità di adeguare le comunicazioni HF campali ad una richiesta di maggiori capacità di traffico per applicazioni di tipo tattico sul territorio nazionale e all'estero, di controllo del territorio e di protezione civile.

A questo fine si è provato ad introdurre un nuovo servizio che, accanto a quelli tradizionali di comunicazioni in fonia e telegrafia, permetta alla

Stazione radio di interfacciarsi con il mondo esterno (Internet), attraverso l'uso della rete locale.

La realizzazione necessita da una parte dell'introduzione di alcune modifiche circuitali e tecnologiche negli apparati, in modo tale da poter permettere l'accesso fisico alla rete locale, dall'altra dell'adozione di nuove metodologie operative e di impiego, come, ad esempio, la gestione da remoto della Stazione radio, utilizzando un qualsiasi computer collegato alla stessa rete.

La Stazione ammodernata deve mantenere sia le vecchie caratteristiche fisiche di ingombro sia quelle di installazione, ed inoltre deve disporre della capacità di interoperare con quella attualmente in servizio.

Per quanto riguarda la realizzazione del software, si è implementato un processo per gestire la struttura protocollare **TCP/IP**, utilizzando un sistema operativo multi-tasking embedded real-time, il **pSOSystem**, che, attraverso un suo componente, il **pNA+**, fornisce le primitive per la gestione dell'interfaccia di rete.

Questo ha permesso l'implementazione di alcuni applicativi di tipo Client-Server, che hanno consentito di inserire l'apparato nella rete aziendale.

I. La rete LAN

Una rete locale, generalmente chiamata **LAN** (Local Area Network), è un mezzo di trasporto limitato ad un ambito locale (senza attraversamento di suolo pubblico), ad alta velocità e basso tasso di errore, equamente condiviso tra tutte le stazioni che vi si collegano.

Il mercato delle medie prestazioni è ormai dominato dallo standard **IEEE 802.3** (evoluzione di Ethernet), mentre quello delle alte prestazioni è in grande fermento.

Queste reti adottano come mezzo trasmissivo preferenziale il doppino di rame e la fibra ottica, ottenendo velocità trasmissive comprese nell'intervallo tra 4 Mb/s - 100 Mb/s.

Gli attributi che deve possedere una LAN sono quelli classici delle reti di calcolatori e cioè:

- *affidabilità*: la tecnologia delle LAN è assolutamente consolidata e consente di ottenere affidabilità elevatissime, tali da permettere a molti costruttori di produrre schede di rete locale con garanzia illimitata;
- *flessibilità*: oggi le LAN sono utilizzate per applicazioni molto disparate, da quelle costituite da soli PC a quelle con l'integrazione PC-mainframe, fungendo da supporto unificato per più architetture di rete, tra loro incompatibili ai livelli superiori del modello OSI;
- *modularità*: possono essere realizzate utilizzando componenti di molti costruttori diversi, perfettamente intercambiabili;
- *espandibilità*: le LAN sono strutture appositamente concepite per fornire una crescita graduale nel tempo, secondo le esigenze dell'utente;
- *gestibilità:* la maggior parte dei componenti delle reti locali prodotti negli ultimi anni, sono concepiti per essere gestiti mediante accessi remoti utilizzando il protocollo SNMP (*Simple Network Management Protocol*), che è basato su UDP/IP.

Affinché queste ed altre proprietà vengano soddisfatte è comunque indispensabile un accurato progetto a priori che tenga conto delle esigenze attuali dell'utilizzatore e delle possibili evoluzioni.
Le proprietà precedentemente elencate, unite all'*economicità*, sono state elemento chiave per la diffusione delle LAN.

II. Architettura di rete

La figura II.1 mostra l'architettura dell'Internet Protocol Suite e la paragona con il modello di riferimento **ISO/OSI**.
Questa architettura permette l'esistenza di più pile di protocolli alternative tra loro ed ottimizzate per determinate applicazioni.

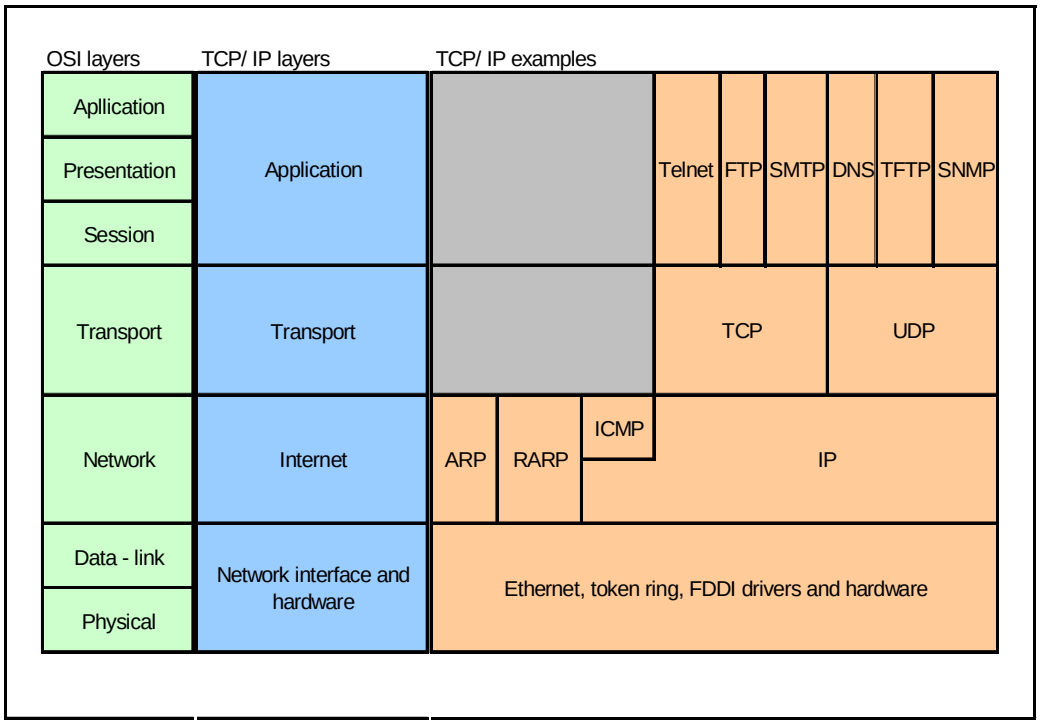


Fig. II.1

III. Internet Protocol Suite

Nella prima metà degli anni '70, la Defence Advanced Research Project Agency (DARPA) dimostrò interesse per lo sviluppo di una rete a commutazione di pacchetto per l'interconnessione di calcolatori eterogenei, da utilizzarsi come mezzo di comunicazione tra le istituzioni di ricerca degli Stati Uniti. DARPA finanziò a tal scopo l'Università di Stanford e la BBN (Bolt, Beranek and Newman) affinché sviluppassero un insieme di protocolli di comunicazione.

Verso la fine degli anni '70, tale sforzo portò al completamento dell'*Internet Protocol Suite*, di cui i due protocolli più noti sono il *TCP* e l'*IP*.

Il nome più accurato per l'architettura di rete rimane quello di Internet Protocol Suite, anche se comunemente si fa riferimento ad essa con la sigla **TCP/IP** (*Transmission Control Protocol/Internet Protocol*).

TCP/IP è l'architettura adottata dalla rete Internet che, con le sue decine di milioni di calcolatori, è la più grande rete di calcolatori al mondo.

I protocolli appartenenti a questa architettura sono specificati tramite standard che si chiamano RFC (*Request For Comments*) facilmente reperibili sulla rete Internet.

Famoso è lo RFC 791 Internet Protocol, datato 1981, che specifica appunto il protocollo IP.

IV. Il Protocollo IP

Il protocollo IP (*Internet Protocol*) è il protocollo principale dello strato di rete (livello 3) dell'architettura TCP/IP. Si tratta di un protocollo semplice, di tipo datagram, cioè non connesso (*connectionless* o CLNS), specificato in RFC 791.

Insieme a TCP costituisce il nucleo originale e principale dell'Internet Protocol Suite.

IP si occupa di instradare i messaggi sulla rete, ma ha anche funzioni di frammentazione e riassemblaggio dei messaggi e di rilevazione degli errori (non correzione).

Il formato dell'header del pacchetto IP è mostrato in figura IV.1.

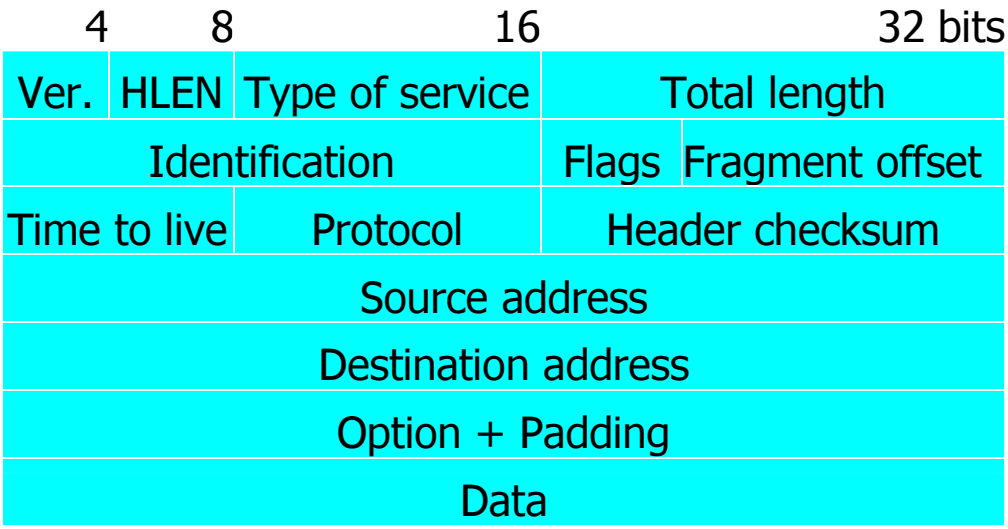


Fig. IV.1

Il significato dei campi del pacchetto IP è il seguente:

- *version*: è il numero di versione del protocollo IP che ha generato il pacchetto; attualmente questo campo vale sempre 4;

- ***HLEN*** *(Header LENgth)*: è la lunghezza dell'header IP, variabile in funzione del campo option, espressa come numero di parole da 32 bit;
- ***service type***: specifica come un protocollo di livello superiore vuole che il pacchetto sia trattato; è possibile assegnare vari livelli di priorità utilizzando questo campo;
- ***total length***: è la lunghezza del pacchetto IP (header più dati) in byte;
- ***identification***: questo campo contiene un numero intero che identifica il pacchetto; è usato per permettere il riassemblaggio di un pacchetto frammentato;
- ***flags***: specificano se un pacchetto può essere frammentato e se si tratta dell'ultimo frammento di un pacchetto;
- ***fragment offset***: è l'offset del frammento in multipli di 8 byte;
- ***time to live***: è un contatore che viene decrementato con il passaggio del tempo; quando il contatore arriva a zero il pacchetto viene scartato. Esso permette di eliminare i pacchetti che, a causa di un malfunzionamento, sono entrati in loop;
- ***protocol***: identifica il protocollo di livello superiore contenuto nel campo dati del pacchetto.
- ***header checksum***: è un campo utilizzato per controllare che l'header IP sia corretto;
- ***source e destination address***: sono gli indirizzi IP di mittente e destinatario, entrambi su 32 bit;
- ***option***: è un campo usato dall'IP per fornire varie opzioni, quali la sicurezza e il source routing.

L'header IP è seguito dal campo dati che contiene la PDU (Protocol Data Unit) del protocollo di livello superiore.

V. Indirizzamento IP

L'indirizzamento IP è parte integrante del processo di instradamento dei messaggi sulla rete.

Gli indirizzi IP, che devono essere univoci sulla rete, sono lunghi 32 bit (quattro byte) e sono espressi scrivendo i valori decimali di ciascun byte separati dal carattere punto. Agli indirizzi IP si associano per comodità uno o più nomi che possono essere definiti localmente in un file "hosts".

Questo approccio diviene impraticabile quando la rete IP cresce di dimensione e allora si preferisce utilizzare una base di dati distribuita per la gestione dei nomi.

Gli indirizzi IP comprendono due o tre parti.

La prima parte indica l'indirizzo della rete (*network*), la seconda, se presente, quello della sottorete (*subnet*) e la terza quello dell'host.

Occorre subito puntualizzare che non sono i nodi ad avere un indirizzo IP, bensì le interfacce. Quindi se un nodo ha tre interfacce, esso ha tre indirizzi IP; poiché la maggior parte dei nodi ha una sola interfaccia, è comune parlare dell'indirizzo IP di un nodo.

Questo tuttavia è senza dubbio sbagliato nel caso dei router, che hanno, per definizione, più di una interfaccia. Gli indirizzi IP sono assegnati da un'unica autorità e quindi sono garantiti univoci a livello mondiale.

Essi sono suddivisi in cinque classi, come schematizzato in figura V.1.

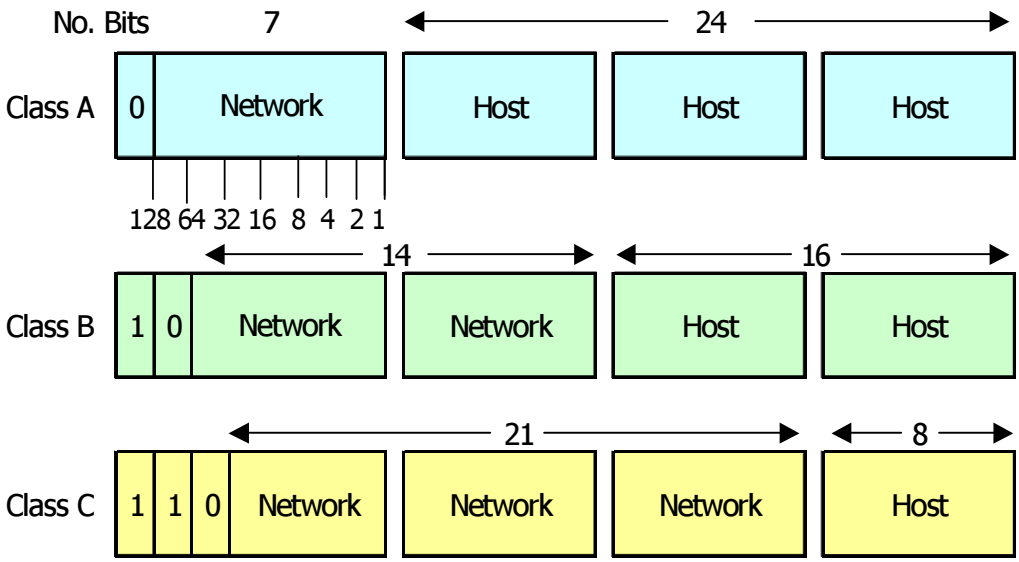


Fig. V.1

- *Classe A*. Sono concepiti per poche reti di dimensioni molto grandi. I bit che indicano la rete sono 7 e quelli che indicano l'host 24. Quindi si possono avere al massimo 128 reti di classe A, ciascuna con una dimensione massima di circa 16 milioni di indirizzi. Gli indirizzi di classe A sono riconoscibili in quanto il primo campo dell'indirizzo è compreso tra 0 e 127.
- *Classe B*. Sono concepiti per un numero medio di reti di dimensioni medio-grandi. I bit che indicano la rete sono 14 e quelli che indicano l'host 16.

  Quindi si possono avere al massimo circa 16000 reti di classe B, ciascuna con una dimensione massima di circa 64000 indirizzi. Gli indirizzi di classe B sono riconoscibili in quanto il primo campo dell'indirizzo è compreso tra 128 e 191.
- *Classe C*. Sono concepiti per moltissime reti di dimensioni piccole. I bit che indicano la rete sono 21 e quelli che indicano l'host 8. Quindi si possono avere al massimo 2 milioni di reti di classe C, ciascuna con una dimensione massima di 256 indirizzi. Gli indirizzi di classe C sono riconoscibili in quanto il primo campo dell'indirizzo è compreso tra 192 e 223.

- *Classe D*. Sono riservati ad applicazioni di multicast secondo quanto descritto nel RFC 1112. Gli indirizzi di classe D sono riconoscibili in quanto il primo campo dell'indirizzo è compreso tra 224 e 239.
- *Classe E*. Questi indirizzi sono riservati per usi futuri. Gli indirizzi di classe E sono riconoscibili in quanto il primo campo dell'indirizzo è compreso tra 240 e 255.

Possiamo trovare anche degli indirizzi IP con significato particolare, come ad esempio un indirizzo di loopback è 127.0.0.1.

La parte host di un indirizzo di classe A, B o C può essere ulteriormente divisa in due parti: la subnet e l'host. Il subnetting è discusso nel RFC 950.

L'ampiezza dei campi subnet e host può essere definita in modo molto flessibile tramite un parametro detto *netmask*.

La netmask contiene bit a uno in corrispondenza dei campi network e subnet, e a zero in corrispondenza del campo host.

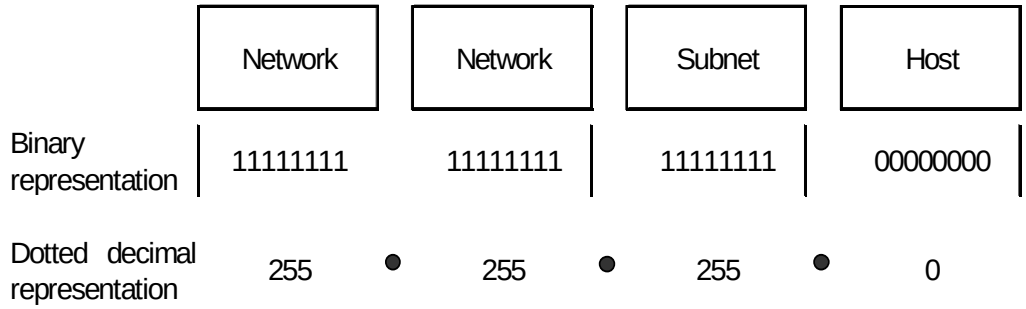


Fig. V.2

Ad esempio una netmask 11111111·11111111·11111111·00000000, più comunemente scritta come indirizzo IP 255.255.255.0 o in esadecimale ffffff00, indica che il campo host coincide con l'ultimo byte dell'indirizzo. (Figura V.2)

All'interno di una rete IP la netmask deve essere univoca, in quanto il partizionamento della rete in sottoreti deve essere univoco. La netmask

viene messa in AND bit a bit con gli indirizzi IP per estrarre la parte network e subnet. Tramite questo procedimento è possibile verificare se due indirizzi appartengono alla stessa sottorete.

L'importanza di comprendere se due indirizzi appartengono o no alla stessa sottorete è fondamentale in quanto il primo livello di routing è implicito nella corrispondenza fissata in TCP/IP tra reti fisiche e sottoreti: *una rete fisica deve coincidere con una sottorete IP*.

Il concetto di sottorete introduce un livello di gerarchia nelle reti TCP/IP. Il routing diventa un routing all'interno della sottorete e tra sottoreti.

Il routing all'interno della sottorete è banale in quanto essa coincide con una rete fisica che garantisce la raggiungibilità diretta delle stazioni ad essa collegate.

L'unico problema che si può incontrare è quello di mappare gli indirizzi IP nei corrispondenti indirizzi MAC di livello 2 .

Il formato dell'indirizzo fisico o MAC è di 48 bit, di cui i primi 24 (6 cifre esadecimali) rappresentano l'*OUI*, *Organization Unique Identifier*, che viene assegnato dalla IEEE Standards Office.

Il mappaggio tra indirizzo IP e indirizzo MAC è oggi quasi sempre gestito in modo automatico, tramite i protocolli *ARP* (*Address Resolution Protocol*) e *RARP* (*Reverse Address Resolution Protocol*) .

VI. Il Protocollo TCP

Il TCP è un protocollo di trasporto di tipo connection-oriented che fornisce un servizio di tipo full-duplex (bidirezionale-contemporaneo), con acknowledge (conferma) e controllo di flusso.

Il TCP è utilizzato dalle applicazioni di rete che richiedono una trasmissione affidabile dell'informazione.

Le applicazioni si connettono alle porte TCP e ad alcune applicazioni principali sono associate delle porte che hanno lo stesso numero su tutti i calcolatori (l'elenco delle porte standard è contenuto in RFC 1700).

Il TCP è un protocollo a *sliding window* (finestre) con meccanismi di time-out e ritrasmissione. La ricezione dei dati deve essere confermata dall'applicazione remota. La conferma può essere inserita in una PDU in transito nella direzione opposta, con una tecnica di piggybacking.

Come tutti i protocolli di tipo *sliding window*, TCP ha un massimo numero di dati in attesa di acknowledge. Tale dimensione massima è specificata come numero di byte (window) e non come numero di segmenti TCP.

Il formato dell'header del pacchetto TCP è mostrato in figura VI.1.

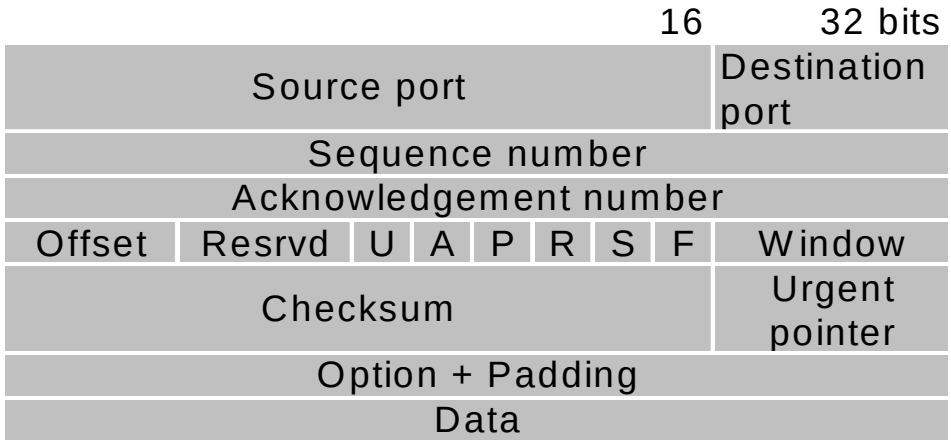


Fig. VI.1

I significati dei campi del pacchetto sono i seguenti:

- La *source port* e la *destination port* sono i numeri delle porte cui sono associati gli applicativi che usano la connessione TCP.
- Il *sequence number* è il numero di sequenza del primo byte del campo dati del messaggio. È utilizzato anche come identificatore della sliding window.
- Lo *acknowledge number* è il campo di acknowledge con tecnica di piggybacking della trasmissione nella direzione opposta. Contiene il numero di sequenza del primo byte che il mittente si aspetta di ricevere.
- Il campo *data offset* indica il numero di parole da 32 bit che compongono l'header TCP, variabile in funzione del campo option.
- Il campo *flag* contiene informazioni varie.
- Il campo *window* contiene la dimensione della receiving window del TCP mittente e quindi lo spazio disponibile nei buffer per il traffico entrante.
- Il campo *urgent pointer* punta al primo byte urgente nel pacchetto.

VII. Introduzione al progetto

Per collegare la Stazione radio alla rete locale via cavo, dopo aver validato la realizzazione hardware dell'interfaccia di rete installata, si dovrà, per prima cosa, ottenere l'indirizzo fisico (MAC), e quindi l'OUI, da associare a tale interfaccia.

Successivamente si assegnerà l'indirizzo IP in modo da permettere la mappatura tra indirizzo fisico e indirizzo IP relativo alla Stazione radio in questione. Questo consentirà l'instradamento dei pacchetti sia verso la rete che verso la radio.

Il passo successivo sarà quello di inserire nel sistema operativo che gestisce il processore della radio (**pSOSystem**), il componente software relativo alla gestione del protocollo TCP/IP, chiamato **pNA+**.

Dopo aver risolto i vari conflitti, insorti a causa dell'aggiunta del nuovo modulo, la prima prova che verrà eseguita sarà quella di far girare un semplice applicativo del tipo **Client-Server**, installando sia la parte client che la parte server sulla radio, in modo tale da trasferire i dati in un loopback interno senza uscire in rete.

A questo punto, dopo aver verificato il funzionamento del nuovo componente software, si dovrà analizzare la riuscita della connessione con la rete locale dell'azienda.

Per far questo si è implementato il comando **ping,** in modo tale da verificare la raggiungibilità della Stazione Radio da un altro terminale connesso alla stessa rete locale.

Una volta verificata l'esistenza in rete della Stazione Radio si potrà realizzare su di essa la componente Server, per poter rispondere a qualsiasi richiesta di qualsiasi terminale collegato alla rete aziendale.

Capitolo 1

Configurazione Hardware

1.1 Introduzione

La stazione Radio è costituita da un sistema a microprocessore (Motorola 68360) che interfaccia:

- l'apparato radio vero e proprio tramite delle linee I/O;
- un'adattatore d'antenna interna, ATU, tramite due linee seriali (una sincrona e una asincrona);
- un'ATU esterna, alternativa alla precedente, tramite la stessa linea seriale
- asincrona impiegata dall'ATU interna;
- un telecomando, tramite una linea seriale asincrona;
- la Testa di Controllo (TDC), tramite una linea seriale asincrona che costituisce l'interfaccia utente del sistema, ed è anch'essa realizzata tramite un altro sistema a microprocessore;

- un Modem, che realizza la parte di livello fisico della trasmissione dati secondo modalità standardizzate (STANAG, FSK, ARQ);
- il modulo IF Audio ed altre parti che, tramite un bus parallelo definito Host Bus, riguardano la trattazione del segnale IF attraverso processi digitali, realizzando tutte le modalità di emissione e ricezione.

Come sistema operativo verrà impiegato il ***Real-Time Multitasking Kernel pSOS+***.

Per quanto riguarda il software il sistema è costituito in generale da un modulo di controllo, che interagirà con il modulo Atu Manager per la gestione dell'Atu interna del Manpack HF o di una eventuale Atu esterna. Tramite il modulo Update Software, parte integrante del modulo Controllore, si potrà avviare le procedure di aggiornamento software previste.

1.2 Componenti hardware del modulo di rete

I principali componenti del modulo della stazione radio che realizzano l'interfaccia di rete, sono i seguenti:

- Processore Master **Motorola MC68360** @ 25MHz con indirizzi da 32 o 16 bit;
- 1 Mbyte Dynamic Ram, organizzata in 256Kx36 bit per i dati e per i segnali di parità con un tempo di accesso di 80 nsec.
- 128 Kbyte boot EPROM, usata per immagazzinare programmi di debug. Il tempo di accesso è pari a 200 nsec.
- 512 Kbyte Flash Memory, suddivisa in 4 flash da 128 Kbyte, con un tempo di accesso di 120 nsec.
- 256 byte serial EPROM.

- Processore **LXT908**, che implementa le operazioni relative allo strato fisico (*Physical Layer Signaling, PLS*), interfacciato da una parte con il processore testa di controllo (TDC) e dall'altra con un trasformatore (con rapporto n=1) che trasporta in ingresso i segnali di trasmissione e ricezione dei dati.
- Porta seriale con connettore RS-232 collegato al terminale o al computer host.
- Alimentatore da 5V e 12V in dc.

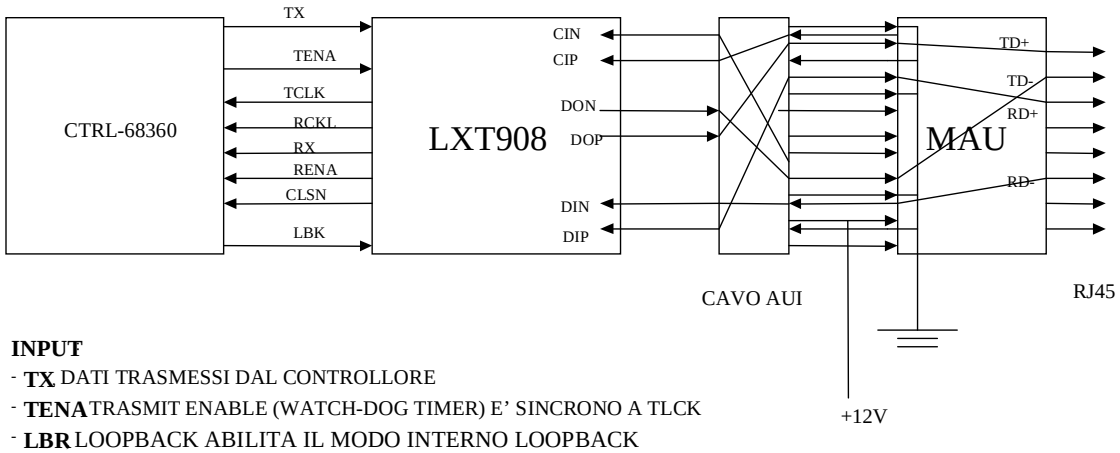
1.3 Componentistica dell'interfaccia di rete

I componenti hardware, relativi all'interfaccia di rete, sono:

- il processore ***MC68EN360***, chiamato Testa di Controllo, che gestisce le funzioni relative allo strato fisico dello standard IEEE 802.3.
  Esso si occupa della gestione dei pacchetti ricevuti e trasmessi dalla\alla rete, comprendendo funzioni di incapsulamento e decapsulamento dei dati, gestione del collegamento, generazione del preambolo del pacchetto, frammentazione e riassemblamento, verifica dell'indirizzo di destinazione e ritrasmissione di trame .
- il *transceiver*, costituito dal processore ***LXT908***, che permette la trasmissione e la ricezione dei pacchetti tra la Stazione Radio ed il mezzo trasmissivo (cavo RJ-45), convertendo il segnale da analogico a digitale e viceversa. Per lo standard Ethernet questo dispositivo prende il nome di **MAU** (*Medium Attachment Unit*) e si collega al modulo relativo all'interfaccia di rete, alloggiata nell'apparato, tramite il connettore a 15 pin **AUI** (*Attachment Unit Interface*).

Nel lavoro realizzato il processore LXT908 viene usato solo in modalità encoder/decoder, con uscita AUI, per poi essere connesso ad un transceiver esterno. Quest'ultimo, implementando le operazioni allo strato fisico, si occupa di segnalare al livello MAC tutte le condizioni del mezzo fisico, tra cui le collisioni avvenute, se il mezzo condiviso è libero o occupato, etc.

Nella figura 1.3.1 vengono descritti sia i collegamenti tra i due processori (MC68EN360 e LXT908), montati sulla scheda di rete della stazione radio, sia i collegamenti con il MAU attraverso il connettore AUI.



INPUT
- **TX** DATI TRASMESSI DAL CONTROLLORE
- **TENA** TRASMIT ENABLE (WATCH-DOG TIMER) E' SINCRONO A TLCK
- **LBR** LOOPBACK ABILITA IL MODO INTERNO LOOPBACK

OUTPUT
- **TCLK** TRASMIT CLOCK (10MHZ), PUO ESSERE CONNESSO DIRETTAMENTE AL CLOCK IN DEL CONTROLLORE
- **RCLK** RECEIVE CLOCK (10MHZ), E' SINCRONO AI DATI RICEVUTI E CONNESSO AL CLOCK DEL CONTROLLORE
- **RX** DATI IN USCITA CONNESSI DIRETTAMENTE AI DATI RX DAL CONTROLLORE
- **RENA** CARRIER DETECT (CD) CHE NOTIFICA AL CONTROLLORE RIGUARDO ALLE ATTIVITA' SULLA RETE
- **CLSN** COLLISION DETECT RIVELA LE COLLISIONI IN INGRESSO AL CONTROLLORE (COL)

Fig. 1.3.1 Descrizione dei collegamenti tra i due processori montati sulla scheda di rete.

1.4 LAN 802.3

La rete locale usata in questo progetto è basata sullo standard IEEE 802.3, che si differenzia dallo standard Ethernet 2.0, in quanto presenta specifiche diverse riguardo ai mezzi trasmissivi usati, alla struttura dei pacchetti trasmessi, e alla struttura del livello data link, ma può mantenere un livello di interoperabilità quando entrambi sono presenti sulla stessa rete.

Riguardo al metodo usato per arbitrare l'utilizzo del canale trasmissivo tra le stazioni della rete (MAC), viene usato il CSMA/CD per entrambe le tipologie di rete.

Lo standard IEEE 802.3 nasce come architettura a bus su cavo coassiale ed evolve successivamente verso le topologie a stella basate sull'utilizzo di cavi UTP e fibre ottiche.

Le velocità trasmissive sono 1 Mb/s e 10 Mb/s (versioni 10Base5, 10Base2, 10BaseT). Nel nostro caso tratteremo soltanto la trasmissione a 10 Mb/s che è quella più usata e conosciuta.

1.4.1 Architettura dello standard LAN 802.3

La relazione tra il livello fisico e il livello Data Link del modello di riferimento OSI e quello IEEE 802.3 è illustrato nella seguente figura 1.4.1.1.

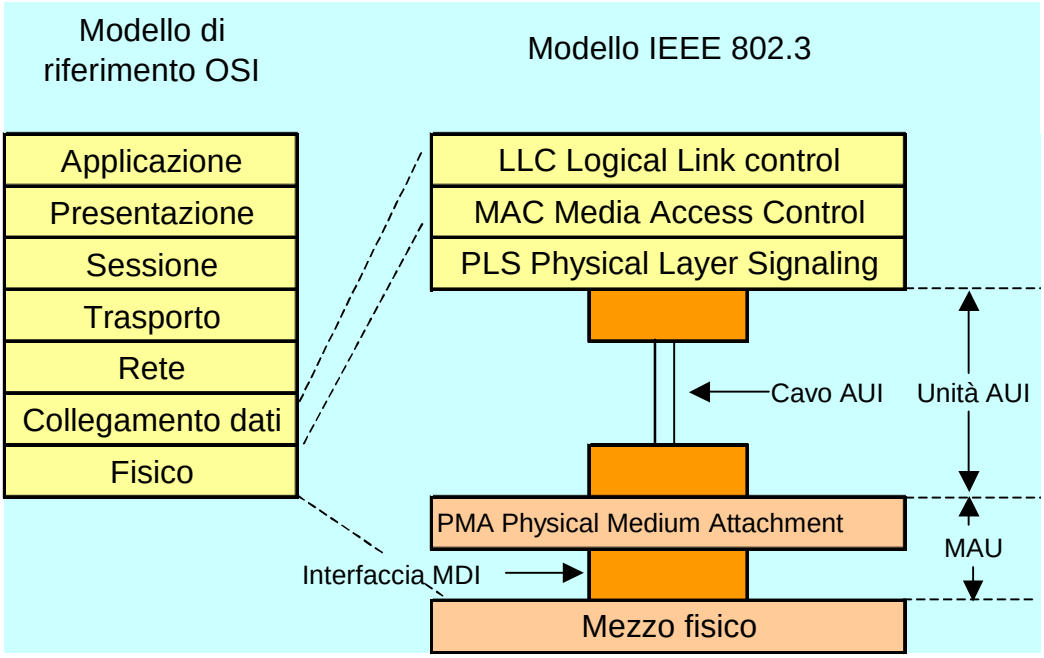


Fig. 1.4.1.1 Relazione tra il modello OSI e IEEE 802.3

Lo standard IEEE 802.3 suddivide il livello fisico del modello OSI in quattro sottolivelli e separa il modello Data Link in altri due.

A livello fisico vengono definite tutte le componenti che intervengono nella trasmissione dei dati tra i dispositivi: caratteristiche elettriche e meccaniche (cavi e connettori), il significato dei contatti dei connettori, il livello del segnale elettrico che rappresenta il livello logico della codifica di linea, la temporizzazione e relativa durata del bit (bit time) e il dispositivo

che si interfaccia con il mezzo fisico, detto *transceiver* (transmitter/receiver).
I quattro sottolivelli dello strato fisico sono:

- Physical Layer Signalling (**PLS**),
- Attachment Unit Interface (**AUI**),
- Physical Medium Attachment (**PMA**),
- Medium Dependent Interface (**MDI**).

PMA e MDI insieme formano il Medium Attachment Unit (**MAU**).

Il *Physical Layer Signalling*, che si interfaccia con il soprastante sottolivello MAC, è il sottolivello che si occupa di segnalare al MAC tutte le condizioni del mezzo fisico (per esempio mezzo occupato o libero, collisione avvenuta). Queste funzioni sono realizzate nel transceiver, conosciuto anche come MAU.

L'*Attachment Unit Interface* definisce invece le specifiche del cavo transceiver (cavo AUI o cavo drop) che è costituito da cinque coppie di conduttori schermati (con l'aggiunta di una schermatura globale) che hanno il seguente significato:

- *Data Out*: trasmette i dati dalla stazione al MAU.
- *Data In*: trasmette i dati dal MAU alla stazione.
- *Control In*: usato per trasmettere segnali di controllo dal MAU alla stazione.
- *Control Out*: segnale opzionale per la trasmissione dei segnali di controllo tra il MAU e la stazione.
- *Voltage*: fornisce alimentazione al MAU (dalla stazione).

Il cavo AUI inoltre presenta un'impedenza di 78 ohm e la sua lunghezza è limitata a 50 metri.

Il connettore terminale del cavo AUI è a 15 contatti di tipo "D". Il significato di questi ultimi sono elencati nella tabella seguente.

Contatto n.	Ethernet v2.0	IEEE 802.3
1	Shield	Control in shield
2	Collision presence +	Control in A
3	Transmit +	Data out A
4	Reserved	Data in shield
5	Receive +	Data in A
6	Power return	Voltage common
7	Reserved	Control out A
8	Reserved	Control out shield
9	Collision presence -	Control in B
10	Transmit -	Data out B
11	Reserved	Data out shield
12	Receive -	Data in B
13	Power	Voltage
14	Reserved	Voltage shield
15	Reserved	Control out B

Tab. 1.4.1.1 Assegnazione dei contatti e loro descrizione per i cavi IEEE 802.3

La definizione dei collegamenti è diversa per i cavi Ethernet V2.0 (che non presentano schermature) e IEEE 802.3, e nonostante si possano ritenere sostanzialmente intercambiabili (il significato dei collegamenti è dato dal transceiver e dalla scheda di rete della stazione), è opportuno usare i cavi 802.3 con schede e transceiver 802.3, in quanto maggiormente immuni ai disturbi elettromagnetici.

L'AUI e il MAU sono entità sempre presenti ma non sempre visibili, dipende dallo standard MDI usato. Nel caso dello standard 10Base-5, il transceiver è un dispositivo esterno alla scheda di rete della stazione. In questo caso la connessione al mezzo fisico è realizzata tramite cavo AUI e transceiver.

Negli altri casi (lo standard MDI verrà descritto in seguito), MAU e AUI sono integrati nella scheda di rete (Network Interface Card o Ethernet Controller).

Le funzioni generali di un MAU sono:

• trasmissione dei segnali elettrici sul mezzo fisico,
• ricezione dei segnali elettrici dal mezzo fisico,
• rilevazione del segnale elettrico (e relativo livello) sul mezzo fisico,
• monitoraggio delle collisioni.

I sottolivelli *PMA* (*Physical Medium Attachment*) e *MDI* (*Medium Dependent Interface*) insieme formano lo standard dei transceiver IEEE 802.3.

Per quanto riguarda lo strato **Data Link**, i parametri principali del sottolivello **MAC** sono i seguenti:

- *Slot time* = 512 bit time (51,2μs), il tempo base di attesa prima di una ritrasmissione;
- *Inter Packet Gap* = 9.6 μs, la distanza minima tra due pacchetti;
- *Attempt limit* = 16, numero massimo di tentativi di ritrasmissione;
- *Max frame size* = 1518 ottetti, lunghezza max del pacchetto;
- *Min frame size* = 64 ottetti (512 bit), lunghezza minima del pacchetto;
- *Address size* = 48 bit, lunghezza indirizzi MAC.

Le funzioni del sottolivello MAC, specifico per ogni LAN, sono:

• In *fase di trasmissione*, l'assemblaggio dei dati all'interno di una trama che contenga anche un campo di indirizzi e un campo di intercettazione degli errori.
• In *fase di ricezione*, il disassemblaggio della trama e l'esecuzione del controllo dell'errore e del riconoscimento degli indirizzi.
• L'*arbitraggio* dell'accesso al mezzo trasmissivo della rete locale.

Il sottolivello **LLC** invece è il livello più elevato del modello di riferimento IEEE 802 ed è collocato immediatamente sopra il sottolivello MAC.

Esso ha lo scopo primario di fornire un'interfaccia unica e omogenea verso il livello superiore (quello di rete), dal momento che esiste un sottolivello MAC per ogni specifica tecnologia.

Il sottolivello LLC non "vede" direttamente il livello fisico come livello sottostante, per cui non deve occuparsi di intercettare ed eventualmente correggere gli errori da esso introdotti. Inoltre, non deve neppure delimitare le trame o gestire la trasparenza del campo dati (usando la tecnica denominata di "bit stuffing").

Fig. 1.4.1.2 Supporto multiprotocollo offerto da LLC

LLC fornisce servizi al livello superiore e riceve servizi dal livello inferiore attraverso i cosiddetti punti di accesso al servizio o ***SAP*** (*Service Access Point*), che sono interfacce di comunicazione tra livelli (Fig. 1.4.1.2). Lo standard specifica tre tipi di servizi:

• *Servizio senza connessione e senza riscontro* (unacknowledged connectionless service), che specifica un servizio nello stile datagram (non viene attivata alcuna connessione logica tra gli utenti LLC), e permette di trasmettere e ricevere le unità dati di protocollo LLC o LLC-PDU, senza alcuna forma di riscontro che ne assicuri la consegna e senza meccanismi di

controllo di flusso e di controllo dell'errore (supporta indirizzamenti di tipo broadcast) .

• *Servizio in modalità connessa* (conection-mode service), che permette a due utenti LLC di comunicare tra loro stabilendo una connessione logica prima di attivare un trasferimento dati. Fornisce anche meccanismi di controllo di flusso, di corretta sequenza, e di recupero da situazioni di errore.

• *Servizio senza connessione con riscontro* (acknowledged connectionless service), che fornisce un meccanismo attraverso cui un utente LLC può trasmettere unità dati e ricevere un riscontro che ne indichi la corretta consegna, senza la necessità di attivare una connessione logica tra due utenti LLC.

Il pacchetto utilizzato dallo standard 802.3 ha una lunghezza variabile compresa tra 64 e 1518 ottetti, con un preambolo di 7 ottetti che serve alla stazione ricevente per sincronizzarsi sul clock di quella trasmittente, seguito da un ottetto di *Start Frame Delimeter* (*SFD*), codificato con la sequenza di bit 11010101, che indica l'inizio del pacchetto.

Nella figura 1.4.1.2 viene raffigurata la struttura del pacchetto.

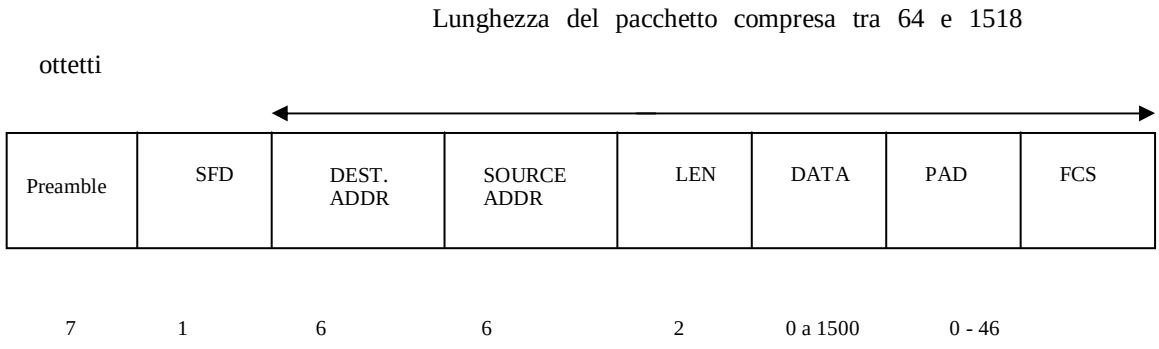


Fig. 1.4.1.3 Struttura del pacchetto per lo standard IEEE 802.3

Nel campo *destination address* è contenuto l'indirizzo della stazione di destinazione, mentre nel *source address* è contenuto l'indirizzo della stazione che ha originato il pacchetto.

Il campo *length* indica il numero di ottetti contenuti nel campo data, il *PAD* viene appeso in coda al precedente campo solo se quest'ultimo è più corto di 46 ottetti e contiene un numero di ottetti calcolato in modo da garantire che venga rispettata la lunghezza minima del pacchetto (64 ottetti).

Il campo *data* contiene le LLC-PDU del sottolivello LLC, mentre il *Frame Check Sequence* (*FCS*) contiene il valore di CRC calcolato sulla base dei campi descritti precedentemente.

Come in Ethernet 2.0 non esiste un segnalatore di fine pacchetto: tale ruolo è assunto dall'*Inter Packet Gap*, la cui durata non può quindi scendere sotto il valore minimo fissato.

1.4.2 Il transceiver

Questo componente contiene l'elettronica che si occupa della rilevazione della portante e delle collisioni (*CSMA\CD*). Nelle reti Lan più recenti, per esempio nello standard 802.3, essi vengono realizzati internamente alla scheda di rete e avvertono se si è verificata una collisione, inviando un segnale speciale che informa gli altri transceiver appartenenti a quella rete di scartare l'ultimo segnale ricevuto.
Il transceiver è costituito principalmente da:

a) Due driver di cui:
- uno trasmette all'interfaccia i dati ricevuti dal mezzo trasmissivo;
- l'altro invia all'interfaccia un segnale di collisione (quando viaggiano i dati sia in trasmissione che in ricezione) nel caso in cui questa sia avvenuta; inoltre il driver di collisione invia all'interfaccia, alla fine di

ogni trasmissione, un segnale chiamato *Collision Presence Test* (**CPT** o *Heartbeat*) il cui scopo è testare il circuito di collisione ed avvisare l'interfaccia del corretto funzionamento di tale circuito;

b) Un receiver che riceve i dati dall'interfaccia e li trasmette, tramite ulteriori circuiti, sul mezzo trasmissivo, usando una codifica e decodifica *Manchester* dei bit ;

c) Un alimentatore da 12V per alimentare i circuiti elettronici interni al transceiver (nel nostro caso abbiamo preso i 12V da un generatore a 24V che serve per alimentare gli altri moduli della radio), senza creare continuità tra le masse elettriche.

1.4.2.1 Codifica Manchester

Per trasmettere il segnale di clock insieme ai dati, ad ogni bit viene applicata una codifica Manchester per garantire almeno una transizione del segnale elettrico in ogni bit. Questo permette ad appositi circuiti del ricevitore di agganciare in fase il loro clock a quello del trasmettitore durante la ricezione del preambolo e quindi di effettuare una ricezione del pacchetto con la corretta temporizzazione.

Così invece di avere una codifica binaria diretta, in cui il bit '0' indica 0V e il bit '1' indica 5V, il ricevente, per non avere ambiguità, deve determinare l'inizio, la metà e la fine di ogni bit senza riferimento a clock esterno.

Ogni periodo di bit viene suddiviso in 2 intervalli uguali:

- il bit ‘1’ viene spedito prima con voltaggio alto e poi basso;
- il bit ‘0’ viene spedito prima con voltaggio basso e poi alto.

Nel seguente schema, figura 1.4.2.1.1, viene evidenziata la differenza tra la codifica binaria e la codifica Manchester.

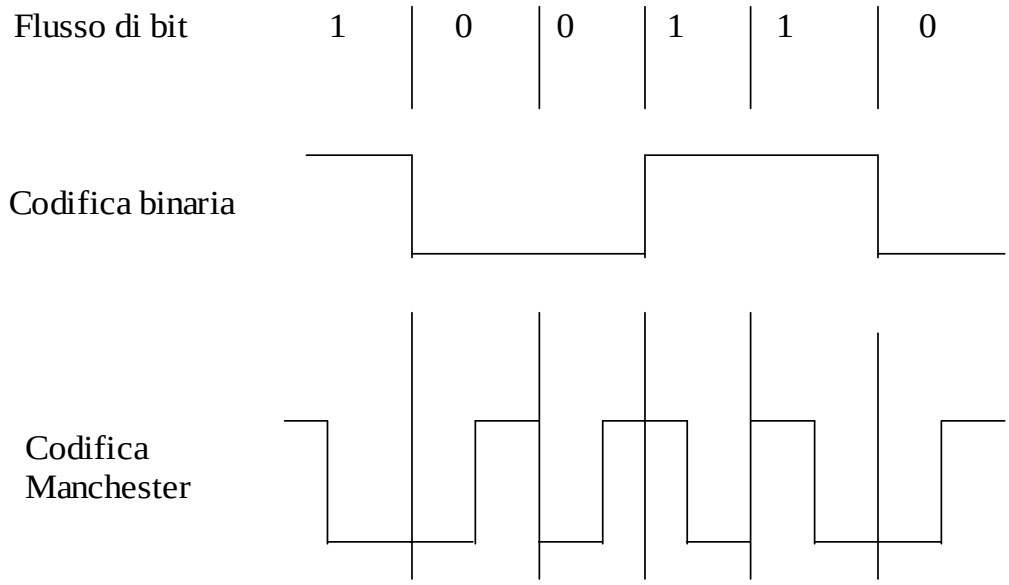


Fig. 1.4.2.1.1 Rappresentazione della codifica Manchester e differenze con quella binaria.

1.4.2.2 Standard 10BASET

Le specifiche di questo standard riguardano le caratteristiche dei MAU e dei mezzi trasmissivi alla velocità di 10 Mb/s (come indicato nel primo

campo del nome 10BaseT) su un segmento di Twisted Pair (doppino) come indicato dalla “T” presente nel secondo campo dello standard, su cui viaggia un segnale in banda base, da qui il nome BASE.
Questo standard ammette la connessione di due sole stazioni in modalità punto-punto.

Una sua particolarità è l’utilizzo di ripetitori multiprova per poter connettere più di due stazioni in rete, facendo diventare la topologia di tipo stellare.

Pertanto, data l’esatta corrispondenza di specifiche con gli standard per i cablaggi strutturati sia in termini di topologia che in termini di caratteristiche elettriche dei mezzi trasmissivi, questo standard è particolarmente adatto per essere utilizzato in tali installazioni.

Una sua funzione molto importante è il test di *Link Integrity*, per proteggere la rete dai problemi causati da un’eventuale interruzione del circuito di ricezione.

Dal momento che la trasmissione dei segnali avviene in modalità "simplex", ossia una coppia per trasmettere da stazione all’hub e l’altra coppia per trasmettere in direzione opposta, deve necessariamente essere garantito che la connessione sia sempre attiva in entrambe le direzioni.
Oltre ai dati, viene trasmesso a intervalli regolari un segnale di Link Integrity (TP_IDL) verso il MAU corrispondente (questo perché la trasmissione dei dati veri e propri avviene in modo discontinuo). La frequenza del segnale di Link Integrity è di 1 MHz.

Al termine della trasmissione di una trama dati si attende un tempo compreso tra 8 e 24 msec prima che il trasmettitore invii il segnale successivo. In mancanza di dati da trasmettere, questo processo avviene ad intervalli regolari (sempre compresi tra 8 e 24 msec).

Se la porta 10Base-T non riceve il segnale di Link Integrity (o dati veri e propri) entro un tempo compreso tra 50 e 150 msec, la connessione entra nello stato di link test fail e viene considerata in errore.

La situazione di errore viene superata e il collegamento riattivato quando vengono ricevuti dal partner corrispondente, tra 2 e 10 segnali di

Link Integrity consecutivi sul collegamento di comunicazione. Questa funzione viene monitorata dall'algoritmo di auto-partition/reconnection definito nello standard 10Base-T.

L'algoritmo consente all'hub (ripetitore) di escludere automaticamente la porta (e quindi la stazione) che fallisce il test di Link Integrity, e di riattivarla quando la condizione di errore viene recuperata. In questo modo la parte restante della rete continua a funzionare.

1.4.2.3 MAU 10BASET

IL MAU (transceiver) 10BASE-T è in grado di trasmettere e ricevere dei segnali elettrici lungo un segmento di doppino di circa 100 m. Esso termina con una presa di tipo RJ45 (jack a 8 contatti con chiave centrale), la cui assegnazione dei contatti è mostrata in figura 1.4.2.3.1.

Contact	MDI Signal
1	TD+
2	TD-
3	RD+
4	Not used
5	RD-
6	Not used
7	Not used
8	Not used

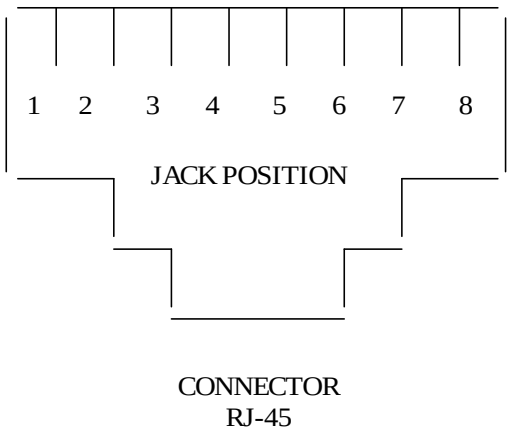


Fig. 1.4.2.3.1 Assegnazione dei contatti su RJ-45

Le funzioni principali del *MAU 10BASE-T* sono:

- funzione di trasmissione: trasferisce i dati codificati secondo la codifica Manchester dal circuito **DO** (*Data Output*) alla coppia di trasmissione **TD** (*Trasmit Data*); se sul circuito **DO** non c'è alcuna trasmissione in corso trasmette sulla coppia **TD** un segnale di idle detto *TP_IDL*;
- funzione di ricezione: trasferisce i dati codificati ricevuti sulla coppia **RD** (*Receive Data*) al circuito **DI** (*Data In*);
- funzione di rilevamento della collisione: quando rileva simultaneamente la presenza di dati sia sulla coppia **RD** che sul circuito **DO**, riporta un segnale di collisione sul circuito **CI** (*Collision In*);
- *SQE test*: invia un segnale di test del circuito di rilevazioni delle collisioni sul circuito **CI** alla fine della trasmissione del pacchetto;
- funzione *jabber*: quando riceve una stringa di dati da **DO** superiore alla lunghezza massima ammessa del pacchetto 802.3 interrompe la funzione di trasmissione;
- funzione di *loopback*: durante il trasferimento dei dati dal circuito **DO** alla coppia **TD** esegue anche lo stesso trasferimento dei dati verso il circuito **DI**;
- funzione di *link integrity test*: protegge la rete dalle conseguenze di un'eventuale rottura del link **RD**; se in un intervallo di tempo compreso tra 50 e 150 ms il MAU non riceve dei dati oppure il segnale *TP_IDL*, entra in uno stato di *link test fail*.

Nella figura 1.4.2.3.2 viene illustrato il collegamento tra una workstation e un ripetitore tramite un cavo twister-pair, in cui vengono evidenziate tutte le parti appena adesso descritte.

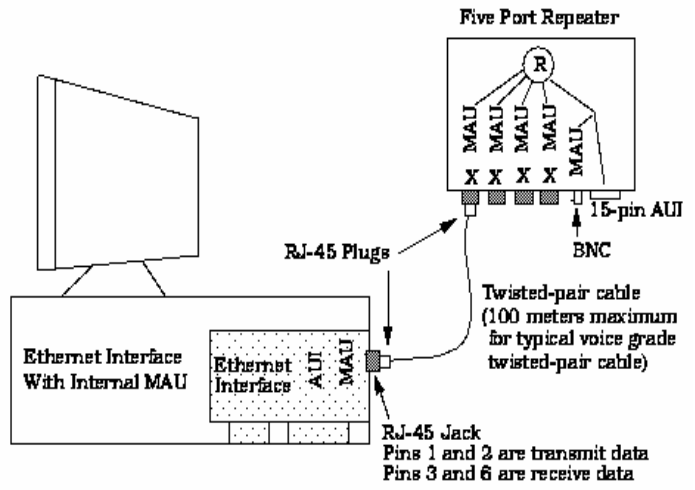


Fig. 1.4.2.3.2. **Descrizione dei collegamenti tra una workstation e un ripetitore tramite cavo twister pair.**

Il MAU 10Base-T è suddiviso, come per tutti gli altri standard, nei due sottolivelli PMA e MDI.

Il PMA gestisce le funzioni elencate in precedenza, mentre il MDI fornisce l'interfaccia fisica verso il mezzo fisico. A questo scopo è stato adottato il connettore RJ-45 a otto contatti (tabella in questa pagina).

Il collegamento tra la stazione e l'hub prevede che la coppia di trasmissione di un MAU diventi la coppia di ricezione del MAU corrispondente, e viceversa. L'incrocio viene normalmente realizzato all'interno dell'hub, in modo che il cablaggio venga eseguito senza inversioni tra il MAU della stazione (anche in questo caso integrato sulla scheda di rete) e il MAU dell'hub.

Capitolo 2

Software per l'interfaccia di rete

2.1 Componenti software della Stazione Radio

I componenti software che si interfacciano all'interno del sistema considerato e dall'esterno con il sistema stesso sono:

- *ALE*, Automatic Link Establishment, che costituisce la modalità di comunicazione adattativa e comprende le funzioni di selezione della migliore tra le frequenze disponibili, la chiamata selettiva dei corrispondenti e l'instaurazione del collegamento;
- un'*ATU* interna tramite due linee seriali (una sincrona e una asincrona) che rappresenta l'adattatore d'antenna, necessario per la funzione ALE;
- un'*ATU* esterna, alternativa alla precedente, tramite la stessa linea seriale asincrona impiegata dall'ATU interna;

- un *Controllore* che realizza le seguenti funzioni per la gestione della Radio: gestione interfaccia ALE, gestione interfaccia esterna (TDC, DTE, Telecomando) e gestione dei moduli interni;
- un telecomando, che, tramite una linea seriale asincrona, gestisce le funzioni inerenti al controllo remoto dell'operatività del sistema inteso sia dal punto di vista della programmazione che dal punto di vista operativo;
- la Testa di Controllo (TDC), che, tramite una linea seriale asincrona, permette il controllo e la visualizzazione dello stato della radio da parte dell'operatore: essa costituisce l'interfaccia utente del sistema ed è realizzata tramite un altro sistema a microprocessore);
- un Modem, che realizza la parte di livello "fisico" della trasmissione dati secondo alcune modalità standardizzate(STANAG, FSK, ARQ);
- il modulo IF Audio che esegue la trattazione del segnale IF attraverso processi digitali, realizzando tutte le modalità di emissione/ricezione.

Come sistema operativo verrà impiegato il Real-Time Multitasking Kernel pSOS+.

2.2 Nozioni sul sistema operativo pSOS+

Il pSOSystem è un'architettura modulare, costruita attorno ad un Kernel real-time multi-tasking e ad una collezione di componenti software, che funzionano come moduli indipendenti, inter-scambiabili tra un'applicazione e un'altra (vedi figura 2.2.1).

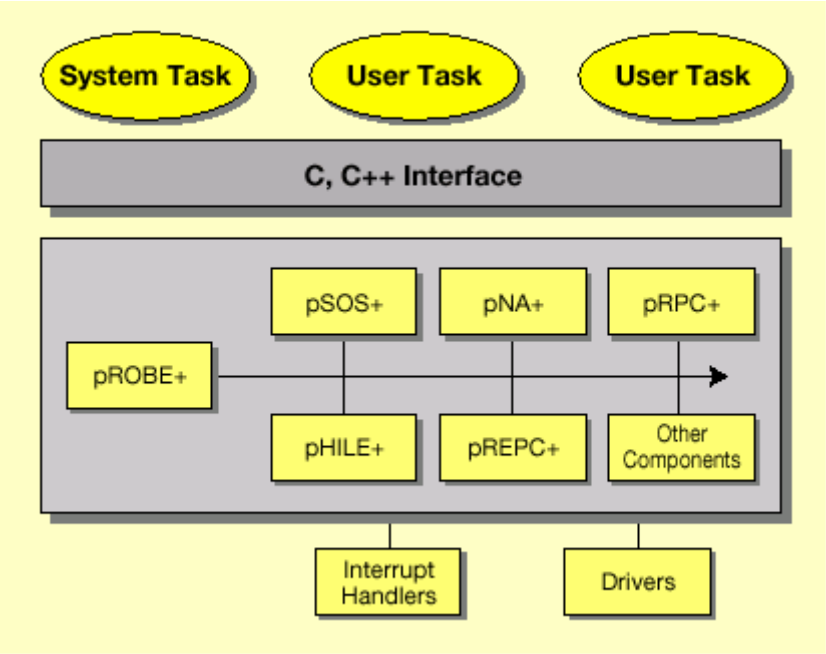


Fig. 2.2.1.1 Architettura del pSOSystem

Ogni componente software utilizza una tabella di configurazione che serve all'utente per configurare i parametri relativi all'hardware durante lo startup.

Ogni modulo implementa un insieme di chiamate, dette ***system calls***, che appaiono come funzioni C e che vengono chiamate da ogni applicazione.

I vari componenti del sistema sono:

- ***pSOS+*** *Real-time Multitasking Kernel*, che provvede a coordinare le attività dell'intero sistema real-time;
- ***pNA+*** *TCP/IP Network Manager*, che rappresenta una completa implementazione del protocollo TCP/IP, UDP, ARP e ICMP; vengono usati, per il trasferimento dati, i socket standard, stream, datagram e raw socket;
- ***pRPC+*** *Remote Procedure Call Library*, che serve per costruire applicazioni usando le procedure familiari del C;

- ***pREPC+*** *ANSI C Standard Library*, che provvede alle funzioni run-time familiari ANSI C, come printf(), scanf(), etc….
- ***pROBE+*** *system-level debugger*, che si occupa della gestione del debugging;

2.2.1 pSOS+ Real-Time Kernel

Il pSOS kernel è il nucleo del sistema operativo che si occupa di gestire e allocare le risorse, e di coordinare le diverse attività asincrone.
La più piccola unità di esecuzione usata dal sistema è il ***task***, che vive in un ambiente virtuale creato dal kernel e che può usufruire delle risorse di sistema (CPU, componenti I/O, regioni di memoria, etc,..).

Concettualmente un task può essere eseguito sia insieme ad altri task che separatamente, in quanto è il pSOS+ kernel che gestisce l'esecuzione di task differenti.
Per esempio, il kernel può interrompere l'esecuzione di un task per iniziarne un'altra di un task differente e per continuare la precedente in un istante successivo. Sebbene il task sia costituito da un set di azioni logicamente separate, esso deve cercare di coordinarsi e sincronizzarsi con gli altri task o con le altre chiamate di sistema (*System Calls*).

Il pSOS+ kernel non impone nessun limite sul numero di task che possono coesistere in un'applicazione. Semplicemente, nella tabella di configurazione del pSOS+, dovrà essere specificato il numero dei task che verranno attivati contemporaneamente e lo spazio di memoria sufficiente per allocare le strutture dati richieste per la gestione dei numerosi task.

Un task non può esistere se prima non viene creato dal kernel pSOS+ e per essere eseguito deve essere anche inizializzato. Un task creato ma non inizializzato rimane in uno stato embrionale.

2.2.2 Stati di transizione

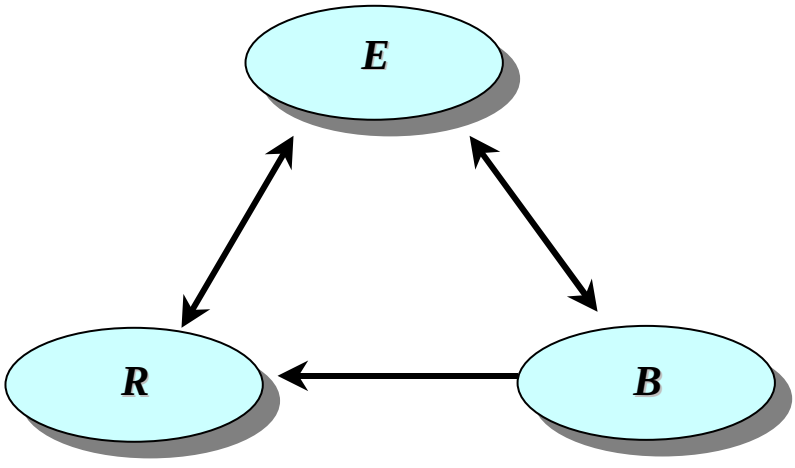
Una volta inizializzato, il task può stare in uno dei tre stati seguenti:

- *Ready*, pronto per essere eseguito;
- *Running*, in esecuzione;
- *Blocked*, in attesa;

Solo un task nello stato ready può essere eseguito, e può stare in attesa solo se ci sono task a priorità più alta che richiedono la CPU.

Un task può stare nello stato running solo se c'è stata una chiamata di sistema da un altro task in running, ricordando che solo un task alla volta può essere eseguito; di solito quest'ultimo è quello a priorità più alta rispetto agli altri task.

Solo un task in running può passare ad uno stato blocked, perchè è l'unico che in quello stato può eseguire delle chiamate di sistema. Tutti i possibili passaggi tra i vari stati vengono mostrati nella figura 2.2.2.1.



E = stato di Running

R = stato di Ready

B = stato di Blocked

Fig. 2.2.2.1 Grafico di tutti I possibili passaggi tra i vari stati dei task.

Ad ogni task creato deve essere assegnata una priorità.
Ci sono 256 livelli, di cui il 255 è il livello di priorità più alto mentre 0 è il più basso.
Quando il task entra nello stato ready, il pSOS kernel lo inserisce in una coda dietro ad altri task di priorità uguale o più alta.

2.2.3 Gestione dei task

Il pSOS Kernel è un sistema che opera ad oggetti, come task, code e semafori.
Ad esempio, ogni task viene creato "runtime" e viene riconosciuto dal sistema attraverso due identificatori: un nome pre-assegnato e un identificatore (ID), entrambi di 32 bit (4 caratteri per il codice ASCII).

La loro assegnazione viene fatta tramite specifiche chiamate di sistema, con cui viene allocata una regione di memoria (***t_create***) e per porre l'oggetto, in questo caso il task, in uno stato di ready (***t_start***).

Un eccezione è fatta per il task principale, detto ***Root*** task, che viene creato e inizializzato dal kernel allo "startup", in cui viene assegnata staticamente o dinamicamente la memoria richiesta dal Kernel.

Quando il task ha terminato le sue operazioni può essere cancellato (***t_delete***) o da altri task o da se stesso, e quindi può essere rilasciata la regione di memoria occupata, targandola come libera per poter essere usata in seguito.

2.3 Il componente pNA+

Le capacità di accesso alla rete sono descritte da un componente del pSOS+, chiamato ***pNA+***, che implementa i protocolli TCP, UDP, IP, ICMP e ARP.

Il pNA+ supporta pienamente 2 livelli IP multicast come specificato in *RFC 1112*, includendo una implementazione del ICMP e supportando anche innumerevoli collegamenti punto-punto come specificato nei requisiti del router IP nel RFC 1716.

Tramite il pNA+ è possibile implementare applicativi Client-Server, in modo tale da trasmettere e ricevere dati su una network TCP-IP.

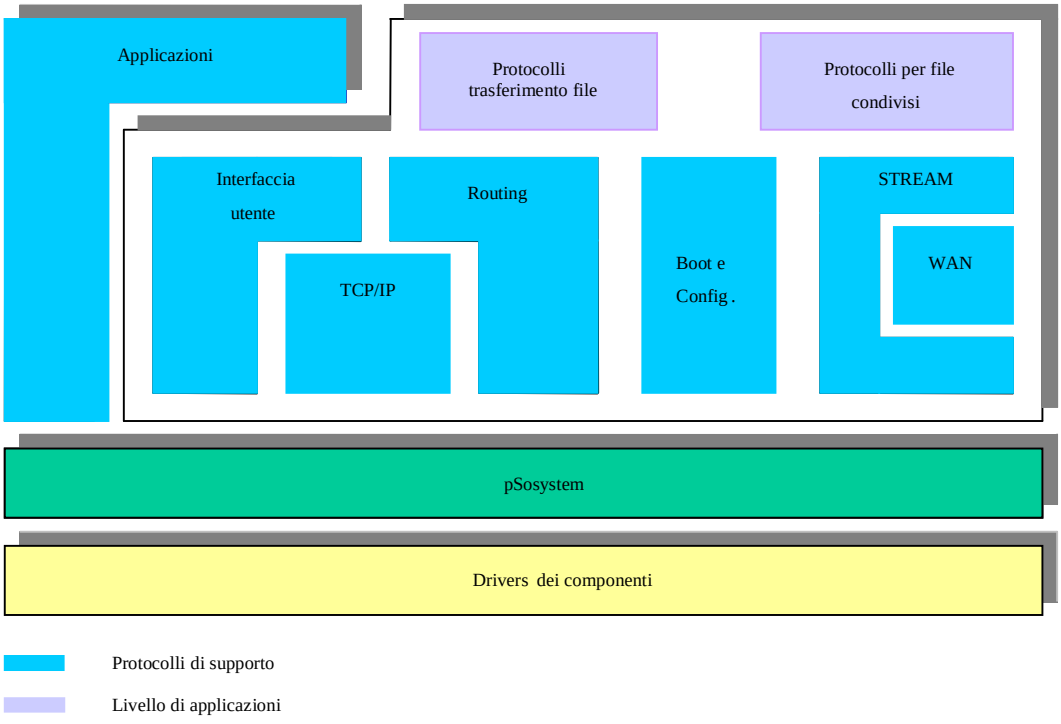


Fig. 2.2.3.1 Architettura protocollare pSOSystem e pNA+

2.3.1 Architettura del pNA+

Il pNa+ è organizzato in 4 strati, come viene mostrato in figura 2.3.1.1:

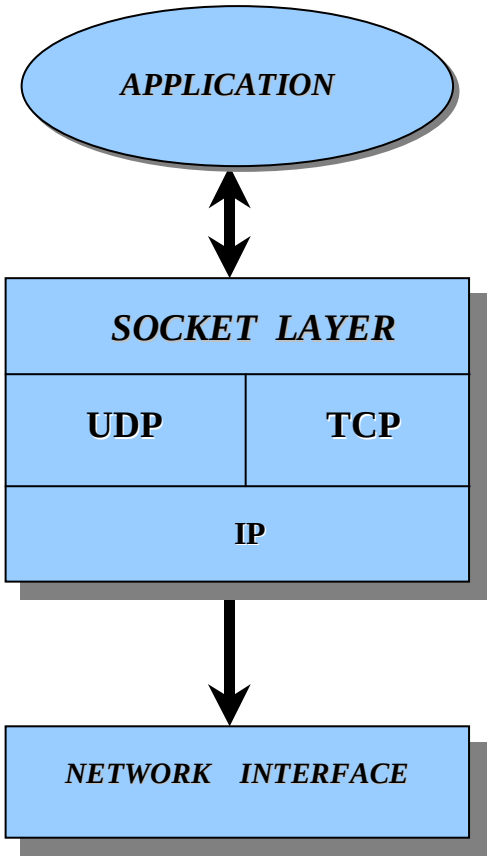


Fig. 2.3.1.1 Architettura del pNA+

1) ***Network Interface*** *(NI)* è lo strato più basso dell'architettura che isola lo strato superiore (IP) dallo strato fisico. Esso dipende dall'hardware e quindi non fa parte propriamente del pNA+ ed è responsabile del trasporto dei pacchetti in una singola rete.
2) ***Strato IP*** è usato per trasmettere blocchi di dati chiamati datagrammi, ed è posto al di sopra della NI. Questo strato permette l'instradamento dei pacchetti, la frammentazione e la ricostruzione della lunghezza degli

interi datagrammi attraverso una rete o internet. Il supporto multicast IP è implementato dallo strato IP.

3) ***Strato di Trasporto*** posto al di sopra dello strato IP , supporta due protocolli di trasporto: TCP (Protocollo di Controllo di Trasmissione) e UDP. Questi protocolli provvedono ai servizi di trasporto della rete. Il protocollo TCP trasferisce un flusso informativo full duplex, task to task e effettua operazioni di multiplazione e de-multiplazione. E' un protocollo con connessione ed esegue:
   - Controllo e recupero d'errore;
   - Controllo di flusso;
   - Ri-ordinamento delle unità informative;
   - Indirizzamento di uno specifico utente all'interno di un host.

   Il protocollo UDP, al contrario del TCP, è senza connessione, perciò il pacchetto deve avere al suo interno un minimo di intestazione. Non è assicurato l 'arrivo del pacchetto.
4) ***Socket layer*** è l'interfaccia con le applicazioni. Questo strato provvede ai servizi, alle procedure con cui accedere ai protocolli di rete. In aggiunta ai protocolli descritti, il pNA+ supporta anche ARP (per la corrispondenza tra indirizzi internet e indirizzi fisici), ICMP, IGMP.

2.3.1.1 Socket Layer

Questo strato serve per capire come sono descritti e usati i socket dai componenti del pNA+.

Un socket identifica una specifica connessione tra due terminali su una determinata porta. Le porte nominate non sono vere e proprie porte fisiche, ma porte logiche, cioè identificativi numerici con i quali TCP-IP è in grado di stabilire più connessioni contemporanee sulla stessa macchina.

I task comunicano inviando e ricevendo dati attraverso i socket. Questi sono distinti a seconda delle caratteristiche della comunicazione che essi supportano. Così il pNA+ distingue tre tipi di socket che supportano tre tipi diversi di servizio:

- ***Stream socket*** usati dal protocollo TCP, provvedono ad un servizio di comunicazione con connessione; di fatti prima di trasmettere dati tra due stream socket viene stabilita una connessione tra loro.
- ***Datagram socket*** usati dal UDP, provvedono ad un servizio di comunicazione senza connessione (ho quindi bisogno di un minimo di intestazione per scambiare dati).

Non è garantito l'arrivo dei pacchetti.

- ***Raw socket*** abilitano 1'implementazione di protocolli di trasporto diversi dal TCP/UDP sopra lo strato IP. Essi provvedono ad un servizio di comunicazione senza connessione.

2.3.2 Gestione dei Socket

I socket vengono gestiti all'interno del task tramite le *System Calls*.
La chiamata di sistema che si occupa della creazione di un socket è detta *socket()*. Il tipo di socket (stream, datagram, raw) è dato come parametro di ingresso nella chiamata. Da questa ci ritornerà il descrittore del socket, che sarà usato dall'utente per accedere al socket.

Un esempio di chiamata usata per creare uno stream socket è il seguente:

s = **socket** (AF_INET, SOCK_STREAM, 0);

Il descrittore del socket s che ci ritorna dalla chiamata può essere usato solamente dal task in cui l'ho creato. Spesso per far usare il descrittore da un altro task si crea un nuovo s.descriptor, ns, tramite la chiamata di sistema *shr_socket*; ad esempio

ns= **shr_socket** (s, tid);

il parametro di ingresso s è il descrittore del socket già esistente nel task (creato dalla chiamata *socket()*). Il parametro tid è l'ID di un altro task che vuole accedere allo stesso socket.

2.3.2.1 Indirizzamento

I socket sono creati senza indirizzi. Fino a che non ho assegnato un indirizzo al socket questo non può essere usato per ricevere dati. L'indirizzo del socket è costituito da un numero di 16 bit che identifica l'utente, e uno da 32 bit che definisce l'indirizzo internet. Questo indirizzo è come un nome che viene usato da altre entità, come i task che risiedono su altri nodi della rete, per riferirsi al socket.

La chiamata di sistema *bind()* è usata per associare un indirizzo ad un socket.

I parametri di ingresso sono il descrittore del socket, l'indirizzo da associare e la sua lunghezza:

bind (s, addr, addrlen).

2.3.2.2 Connessioni

Quando due task vogliono comunicare il primo passo da fare è creare il socket, decidere il tipo di socket da utilizzare (molto spesso sono utilizzati degli stream socket) e stabilire una connessione, usualmente asimmetrica di tipo client-server.

Per stabilire la connessione, il task che opera come server deve associare un indirizzo al socket creato e tramite la chiamata di sistema *listen()* esegue il set-up del socket, così che esso possa accettare la richiesta dal task client.

listen (s, backlog)

dove backlog specifica il numero di connessioni richieste che possono essere accodate per essere accettate dal socket.

A questo punto il task client può iniziare una connessione con il task server usando la chiamata di sistema *connect()*. I parametri di ingresso sono l'indirizzo del socket e il suo descrittore che identificano il socket rappresentante il terminale del client nella connessione client-server che il server sta "ascoltando".

Per concludere il server, con la chiamata di sistema *accept()*, crea un nuovo socket, con uguali proprietà dell'originale, che viene connesso al socket del client, mentre quello iniziale viene lasciato libero per altri client che vogliono usare *connect()* per richiedere una connessione con il server (vedi figura 2.3.2.2.1).

SERVER	**CLIENT**
socket (domain, type, protocol);	*socket* (domain, type, protocol);
bind (s, addr, addrlen);	
listen (s, backlog);	*connect* (s, addr, addrlen);
accept (s, addr, addrlen);	

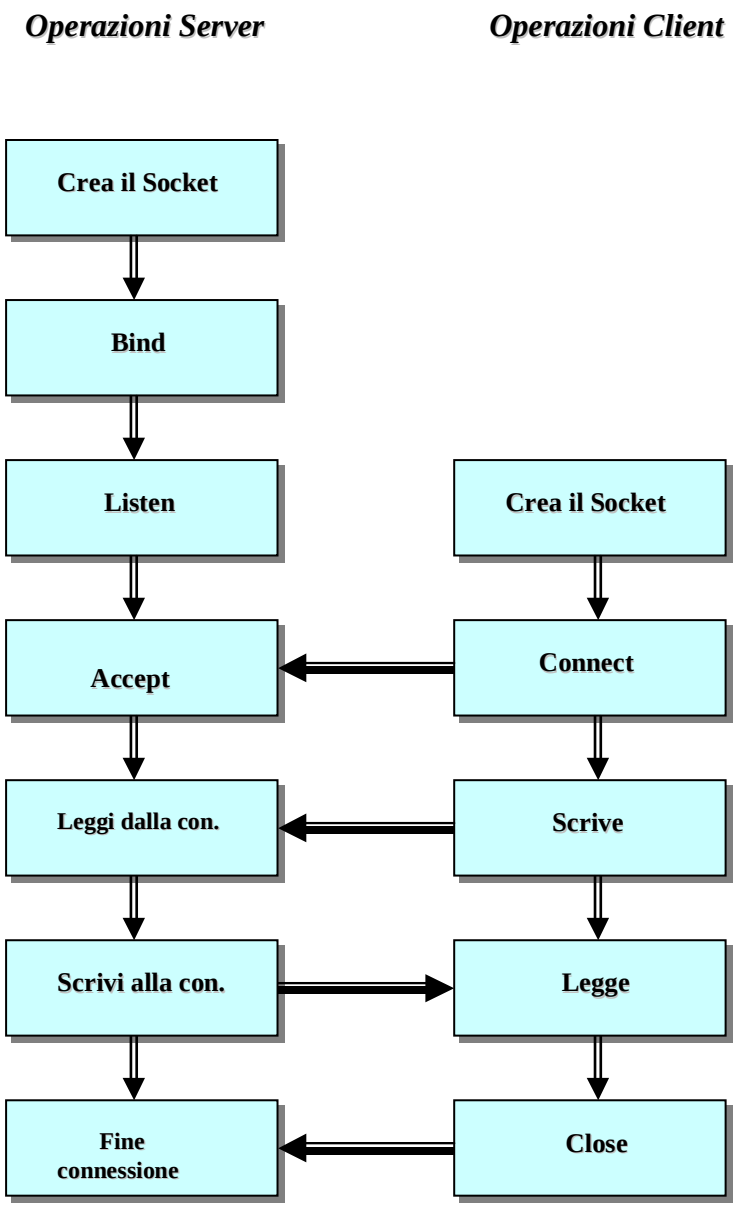


Fig. 2.3.2.2.1 Catena di chiamate tra Client-Server.

2.3.2.3 Trasferimento dati

Dopo aver stabilito una connessione i dati possono essere trasferiti. Vengono usate due chiamate di sistema, *send()* e *recv()* i cui parametri d'ingresso sono rispettivamente il descrittore del socket, il buffer contenente i dati da trasmettere, la lunghezza del buffer e un set di flags:

send (s, buf, buflen, flags)

recv (s, buf, buflen, flags)

Quando un task vuole trasmettere dei dati attraverso una connessione usa la chiamata di sistema *send()*.

Un flag può essere usato per marcare il dato come "fuori banda", cioè ad alta priorità, un altro può essere usato per disabilitare la funzione di instradamento per il dato.

Il socket specificato dal parametro s è conosciuto come ***local socket***, mentre il socket dall'altra parte della connessione è detto ***foreign socket***.

Quando viene chiamato send(), il componente pNA+ copia i dati dal buffer specificato dal task in esecuzione in un send buffer associato al socket e attende per trasmettere i dati al foreign socket.

Se non ci sono dei send buffer disponibili al local socket per mantenere i dati, il send() si blocca.

La misura del send buffer del socket può essere decisa con la chiamata *setsockopt()*.

Un task per ricevere dati usa la chiamata *recv()*, che accetta come input il descrittore del socket che specifica il terminale della comunicazione, l'indirizzo, la lunghezza del buffer per ricevere i dati e un set di flags.

Un flag può essere settato per indicare che il recv() è solo per dati che sono stati marcati come "fuori banda" dal task client.

Se i dati richiesti non sono raggiungibili dal socket e questo non è marcato come bloccato, recv() blocca il task client finchè il dato non è ricevuto.
Alla fine della chiamata recv() il server task troverà i dati copiati in uno speciale buffer.

2.3.2.4 Socket senza connessione

Quando vengono usati, nella trasmissione, dei datagram socket, non è richiesto stabilire una connessione. In questo modo l'indirizzo di destinazione è dato in ogni momento del trasferimento dei dati.
Per inviare dati viene fatta la chiamata *sendto()*:

sendto (s, buf, buflen, flags, to, tolen);

I parametri s, buf, buflen e flags sono gli stessi di send(), mentre to e tolen sono valori usati per indicare l'indirizzo del foreign socket che riceverà i dati.
Per ricevere i dati invece viene usato recvfrom():

recvfrom (s, buf, buflen, flags, to, tolen);

dove to e tolen sono valori usati per indicare l'indirizzo del socket che ha spedito i dati.

Quando cancello un task uso la chiamata *close()* per chiudere tutti i descrittori dei socket che sono stati chiamati dal task in esecuzione.

2.3.3 Comportamento del pNA+

Quando si eseguono le chiamate di sistema ci possono essere tre risposte da parte del pNA+:

A. Il componente pNA+ esegue il servizio richiesto e lo invia direttamente al chiamante;

B. La chiamata di sistema non può essere completata immediatamente, ma non richiede al chiamante di aspettare. In questo caso, il componente pNA+ scheda le operazioni necessarie e invia il controllo al chiamante. Per esempio, la chiamata send() copia i dati dal buffer dell'utente sul buffer interno. I dati non possono essere trasmessi subito.

C. La chiamata non può essere completata immediatamente e il chiamante può aspettare. Per esempio l'utente può aspettare di leggere i dati chenon sono ancora disponibili. In questo caso il componente blocca il task chiamato.

2.3.4 Instradamento dei pacchetti

Il pNA+ include anche le specifiche per l'instradamento completo dei pacchetti.

Quando il componente pNA+ invia pacchetti si basa su cammini (*route*) che definiscono la connessione tra nodi.

I route possono essere classificati in *diretti* o *indiretti*.

Un route diretto definisce un percorso con un nodo connesso direttamente. I pacchetti destinati a quel nodo sono inviati direttamente al nodo di destinazione finale.

Un route indiretto definisce un cammino con un nodo connesso indirettamente. I pacchetti indirizzati ad un nodo connesso indirettamente sono instradati attraverso un altro nodo (*gateway*).

I route possono essere anche classificati in *host* o *network*.

Un *host route* specifica un cammino verso un particolare nodo di destinazione, basato su un indirizzo IP completo del nodo di destinazione.

Un *network route* specifica un cammino verso il nodo di destinazione con solo una parte dell'indirizzo IP relativo alla rete (lo posso instradare solo verso la rete di destinazione e non fino alI'host).

I *cammini diretti* provvedono a creare una mappatura tra un indirizzo di destinazione e una NI (Network Interface). Questi cammini sono aggiunti durante l'inizializzazione della NI. Quando una NI è introdotta nel sistema, il pNA+ introduce un cammino diretto per quella NI. Se il collegamento è punto-punto, un nodo pNA+ è connesso ad un singolo nodo e il route è un host route. In altri casi ho network route.

I *cammini indiretti* provvedono invece ad associare un indirizzo di destinazione con l'indirizzo di un gateway. Al contrario dei cammini diretti, quelli indiretti non sono creati automaticamente dal componente pNA+.

Il componente pNA+ supporta un meccanismo di instradamento finale, rappresentato da un gateway di default, che può essere specificato nella tabella di configurazione del pNA+.

Questo specifica l'indirizzo a cui tutti i pacchetti devono essere inviati quando nessun altro cammino è stato trovato per quei pacchetti.

In conclusione il componente pNA+ usa il seguente algoritmo per determinare il cammino di un pacchetto:

1) Il componente pNA+ cerca, per prima cosa, l'host route usando l'indirizzo IP completo del nodo di destinazione. Se esiste ed è un cammino diretto il pacchetto è inviato direttamente al nodo di destinazione. Se è un cammino indiretto, il pacchetto è inviato al gateway specificato nel route.
2) Se un host route non esiste, il componente pNA+ cerca un network route usando la parte dell'indirizzo del nodo di destinazione relativo alla rete. Se questo esiste ed è un cammino diretto, il pacchetto è spedito direttamente al nodo di destinazione. Se è un cammino indiretto, il pacchetto è inviato al gateway specificato nel route.
3) Se non esiste il network route, il componente pNA+ invia il pacchetto al default gateway.
4) Altrimenti il pacchetto viene perso.

Il cammino viene configurato dal pNA+ durante l'inizializzazione della sua tabella di configurazione.

2.3.5 IP MULTICAST

Il supporto IP Multicast permette ad un host di inserirsi in un host group.

L'host group è definito come un set di zero o più host che sono identificati da un indirizzo IP multicast. Un host può associarsi o lasciare il gruppo quando vuole. Non è detto che un host debba essere necessariamente

un membro del gruppo per spedire pacchetti al gruppo. Ma è necessario associarsi ad un gruppo per ricevere pacchetti indirizzati al gruppo.

L'affidabilità di spedire pacchetti multicast IP è la stessa nel caso di pacchetti IP unicast. Non è garantita la consegna del pacchetto.

2.3.6 Interfaccia di rete

Il componente pNA+ accede alla rete attraverso uno strato software, gestito dall'utente, chiamato **NI**, *Network Interface.*

L'interfaccia tra il componente pNA+ e NI è standard ed è indipendente dalla topologia fisica della rete. Essa infatti isola il componente pNA+ dalle caratteristiche fisiche della rete.

L'interfaccia di rete è essenzialmente un congegno che provvede l'accesso ad una trasmissione media (il termine di Network Interface, NI, e network driver sono usati indistintamente per indicare l'interfaccia di rete).

Deve esistere sempre una NI per ogni nodo pNA+ connesso ad una rete. Spesso un nodo può essere connesso a più reti simultaneamente e quindi dovrà avere più NI.

Ogni indirizzo IP è assegnato ad un unica interfaccia di rete.

Ogni connessione alla rete (NI) ha un certo numero di attributi associati. Essi sono i seguenti:

- L'indirizzo del punto di ingresso alla tabella di configurazione della NI;
- L'indirizzo IP relativo alla NI;
- L'unità di massima trasmissione (**MTU**);
- La lunghezza del suo indirizzo fisico (indirizzo hardware);
- I flags di controllo;
- La maschera di rete relativa all'indirizzo IP;
- L'indirizzo IP di destinazione (nei collegamenti punto-punto).

Il componente pNA+ immagazzina questi attributi per tutte le interfaccie di rete installate nel proprio sistema nella tabella di configurazione della NI. Questi attributi possono essere modificati usando la chiamata di sistema *ioctl()*.

2.3.6.1 Unità di massima trasmissione (MTU)

Molte reti sono limitate nel numero di bytes che possono essere fisicamente trasmessi in una singola connessione.

Ogni NI ha un'associata ***MTU***, Unità di Massima Trasmissione, la quale identifica il numero massimo di pacchetti che possono essere spediti o ricevuti.

Se la misura del pacchetto eccede l'MTU della rete, lo strato IP frammenta il pacchetto per la trasmissione. Di conseguenza lo strato IP del nodo di ricezione riassembla i frammenti nel pacchetto originale.
Il minimo MTU utilizzato dal componente pNA+ è di 64 bytes. Non c'è un limite massimo.

Un MTU più ampio comporta un minore frammentazione dei pacchetti, ma in questo modo aumentano i requisiti di memoria interna della NI. Generalmente è ragionevole una MTU tra 512 bytes e 2k bytes. Ad esempio la MTU per Ethernet è 1500.

2.3.6.2 Indirizzi Hardware

Ogni NI, in aggiunta al proprio indirizzo IP, ha un indirizzo fisico (indirizzo hardware). L'indirizzo internet è usato dallo strato IP, mentre l'indirizzo fisico viene usato dall'interfaccia di rete quando i pacchetti vengono trasferiti fisicamente sulla rete.

Il processo che associa un indirizzo IP ad uno hardware è chiamato *address resolution*. La lunghezza dell'indirizzo hardware, al contrario dell'indirizzo IP che è lungo 4 bytes, varia a seconda del tipo di rete. Per esempio, un indirizzo hardware relativo alla rete Ethernet è lungo 6 bytes mentre l'indirizzo di memoria condivisa è di 4 bytes.

Il componente pNA+ può supportare indirizzi hardware di lunghezza 14 bytes. La lunghezza dell'indirizzo hardware della NI deve essere specificato.

2.3.6.3 Maschera di rete e indirizzo di destinazione

Ogni rete può avere una maschera di rete associata ad essa per supportare l'indirizzamento attraverso quella data rete.

La *maschera di rete* è un valore di 32 bit con gli uno in tutte le posizioni dei bit che sono interpretati come porzione di rete.

In un collegamento punto-punto, i due host sono posti dalle parti opposte di una interfaccia di rete. L'*indirizzo di destinazione* è specificato nella tabella NI come *DSTIPADDR* per reti punto-punto.

2.3.7 La tabella NI

Il componente pNA+ immagazzina i paramentri descritti per ogni NI in una tabella di interfaccia di rete.

La dimensione della tabella è riportata nella tabella di configurazione tramite la costante NC_NNI, che definisce il massimo numero di reti che possono essere connesse al nodo pNA+.

Si può entrare nella tabella NI in due modi:

1. Dalla tabella di configurazione del componente pNA+, tramite la struttura *NC_ INI* che contiene il puntatore alla tabella NI iniziale. I contenuti di quest'ultima saranno copiati su un'attuale tabella NI durante l'inizializzazione del pNA+ (***vedi Appendice***).
2. Tramite la chiamata di sistema *add_ni()* si può entrare nella tabella NI dinamicamente, cioè dopo che il componente pNA+ è stato inizializzato (***vedi Appendice***).

2.3.8 Address Resolution

Ogni NI ha due indirizzi associati ad essa, un indirizzo IP e un indirizzo hardware. L'Address Resolution è il processo che associa un indirizzo fisico ad un indirizzo IP.

In molti sistemi, esso viene applicato durante l'inizializzazione dell'interfaccia di rete, ed è di difficile implementazione.

Quindi per semplificare l'uso di questo processo, il componente pNA+ esegue i seguenti passi:

1. Il componente pNA+ esamina i flags della NI per determinare se può agire internamente al protocollo. Se non può (ad esempio il flag ARP

è disabilitato) il componente pNA+ passa l'indirizzo IP richiesto direttamente all'interfaccia di rete.

2. Se invece il flag ARP è abilitato, il componente pNA+ cerca nella sua tabella ARP, in cui sono memorizzati alcuni indirizzi IP con gli associati indirizzi fisici, l'entrata contenente l'indirizzo IP richiesto. Se viene trovato, il corrispondente indirizzo hardware viene passato all'interfaccia di rete.
3. Se l'indirizzo IP non viene trovato nella tabella ARP, allora il componente pNA+ usa il protocollo ARP per ottenere l'indirizzo hardware dinamicamente.

2.3.8.1 Protocollo ARP

Il protocollo **ARP** (*Address Resolution Protocol*) serve per determinare l'indirizzo hardware di un nodo dinamicamente, dato il suo indirizzo IP.

Il protocollo opera come segue:

A. L'utente, per conoscere l'indirizzo hardware del nodo di destinazione, prepara e invia in broadcast un pacchetto ARP contenente l'indirizzo IP di destinazione.
B. Ogni nodo della rete riceve il pacchetto e compara il proprio indirizzo IP con l'indirizzo specificato nel pacchetto inviato in broadcast.
C. Se un nodo ricevente ha il proprio indirizzo IP uguale a quello richiesto, esso prepara e trasmette un pacchetto ARP di risposta al nodo di partenza, contenente l'indirizzo hardware relativo al proprio indirizzo IP.

Il protocollo ARP può essere usato solo se tutti i nodi della rete lo supportano.

Se la rete considerata contiene solo nodi pNA+ allora è possibile usare il protocollo ARP. Altrimenti dovrò fare in modo che i nodi non pNA+ possono supportarlo.
Il componente pNA+ tratta i pacchetti IP in modo diverso dai pacchetti ARP.

Quando viene inizializzata l'interfaccia di rete deve essere specificato il tipo di pacchetto usato, che può essere IP o ARP. Mentre quando un nodo pNA+ riceve un pacchetto, è l'interfaccia di rete che deve dare il tipo di pacchetto trasmesso.
Per esempio il pacchetto Ethernet contiene il campo type.

Per le interfaccie di rete che non supportano l'ARP, il parametro del tipo di pacchetto può essere ignorato in trasmissione ed inizializzato come IP per i pacchetti entranti.

2.3.8.2 La tabella ARP

Il componente pNA+ costruisce una tabella ARP in cui vengono memorizzati l'indirizzo hardware e il relativo indirizzo IP, come <internet address, hardware address>.
La tabella ARP viene creata durante l'inizializzazione del pNA+.

L'associazione tra l'indirizzo IP e indirizzo hardware viene determinato dinamicamente dal protocollo ARP. Quando il componente pNA+ usa questo metodo esso immagazzina la nuova coppia <internet address, hardware address> nella tabella ARP.
Questo è il normale metodo per aggiornare la tabella ARP.

2.3.9 Gestione della memoria

Nel componente pNA+, i pacchetti, passando da uno strato protocollare ad un altro, sono soggetti a varie manipolazioni, del tipo:

- Aggiunta di intestazioni, per specificare il tipo di protocollo;
- Cancellazione delle intestazioni;
- Frammentazione del pacchetto;
- Riassemblamento del pacchetto;
- Copia dei pacchetti.

L'unità base di dati usata dal componente pNA+ è chiamata *messaggio*. I messaggi sono immagazzinati in strutture.

Una struttura contiene uno o più triple di messaggi collegate tra loro in una singola lista. Ogni tripla di messaggi contiene un blocco contiguo di memoria che definisce una parte del messaggio.
Definiamo i vari blocchi del messaggio:

- ***Messagge block***: contiene le caratteristiche del messaggio definito dalla tripla.
- ***Data block*** : contiene le caratteristiche del buffer a cui punta, e può essere contenuto in diverse triple. Spesso c'è una corrispondenza uno ad uno tra i data block e i buffer.
- ***Data buffer***: è un blocco contiguo di memoria che contiene dati.

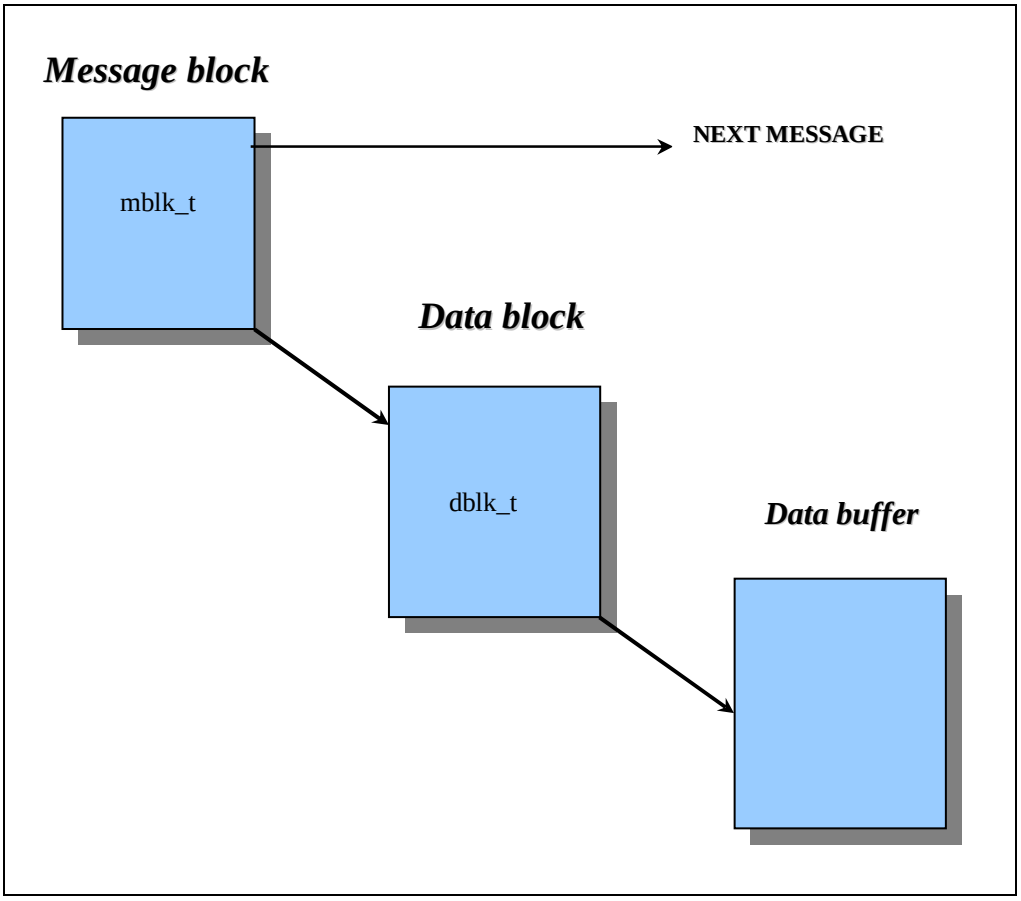


Fig. 2.3.9.1 Struttura di una tripla di messaggi.

Un messaggio completo è formato dai collegamenti delle triple in una singola lista, vedi figura 2.3.9.2.

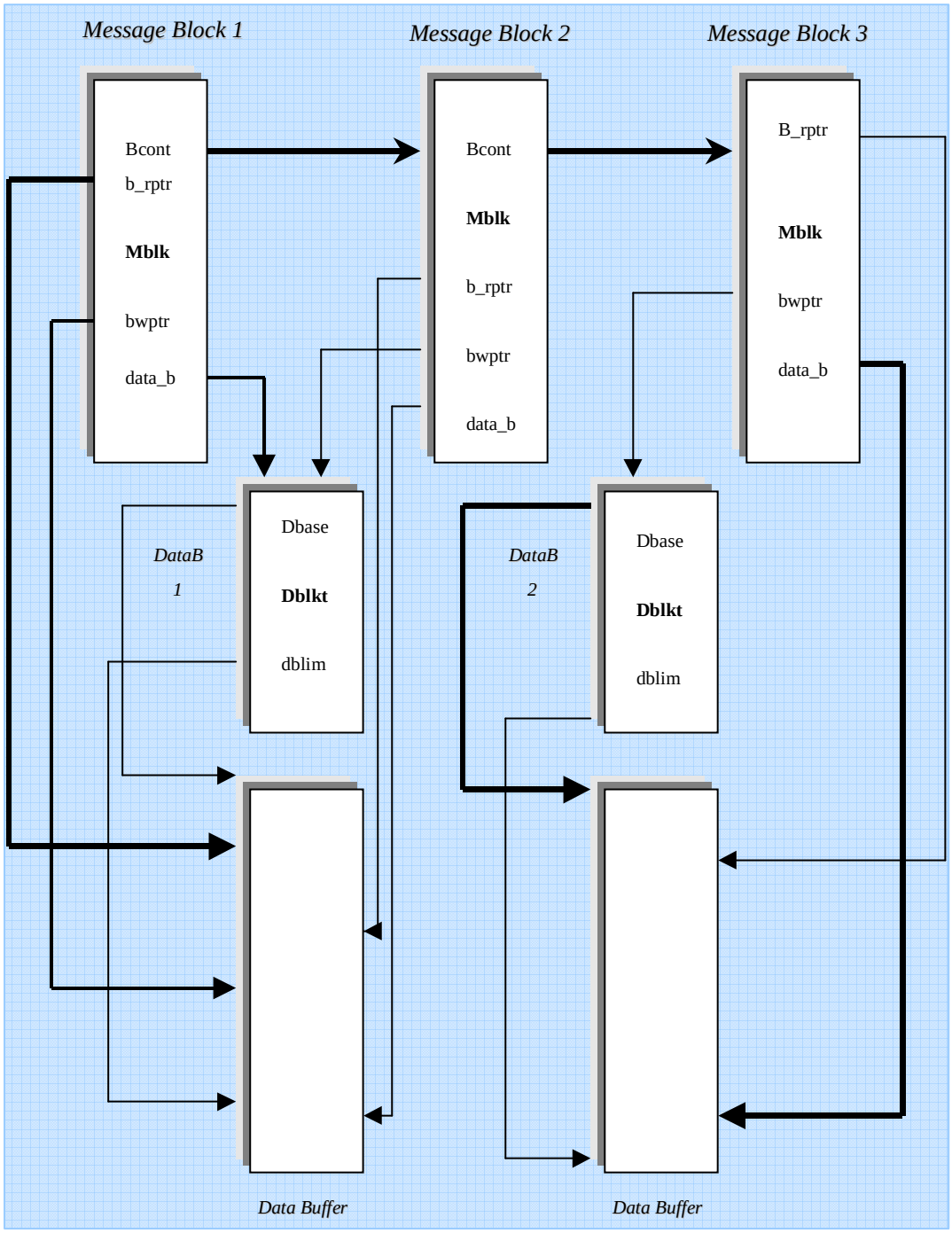


Fig. 2.3.9.2 Struttura di un messaggio completo, costituito dai collegamenti delle varie triple di blocchi di messaggi.

Le definizioni del linguaggio C della struttura dati per i message block e i data block sono nel file di intestazione <**pna.h**> (vedi ***Appendice***).
L'unità base della trasmissione usata dagli strati protocollari nel componente pNA+ è il *pacchetto*.

Il pacchetto contiene sia l'intestazione riferita al protocollo sia i dati incapsulati dal protocollo. Ogni strato targa con la sua intestazione il pacchetto, che poi passerà allo strato sottostante per la trasmissione. Quest'ultimo a sua volta inserirà sia i dati che la propria intestazione per poi passare il pacchetto allo strato inferiore.
I pacchetti sono immagazzinati sotto forma di messaggi.

Il buffer, nel componente pNA+, viene usato per immagazzinare dati, intestazioni dei protocolli, e indirizzi.
I dati passano attraverso due interfaccie:

- All'interfaccia di utente, i dati sono trasmessi tramite le chiamate di sistema send(), sendto() e sendmsg().
- All'interfaccia di rete, i dati sono passati attraverso la chiamata "*Announce Packet*" (vedi ***Appendice***).

Il componente pNA+ alloca una tripla di messaggi e copia i dati dal buffer esterno nel buffer associato alla tripla. Il messaggio viene poi passato allo strato protocollare per maggiore manipolazione.

Quando i dati passano attraverso vari strati, vengono allocate delle triple di messaggi addizionali per immagazzinare le intestazioni e vengono poi collegate al messaggio.

Anche quando viene instaurata una connessione, sia il nodo sorgente che il nodo di destinazione pNA+ allocano temporaneamente triple di messaggi per memorizzare gli indirizzi dei relativi socket creati dalle chiamate di servizio pNA+.

I messaggi che passano attraverso i vari strati della pila protocollare sono soggetti a varie manipolazioni (frammentazione, riassemblamento e copie).

Per esempio, quando si prepara un pacchetto per la trasmissione, lo strato TCP del nodo sorgente fa una copia del pacchetto dal buffer relativo al socket creato, aggiunge l'intestazione TCP e passa il pacchetto allo strato inferiore IP. Così lo strato IP frammenta i pacchetti che riceve dallo strato di trasporto TCP per adattarli all'MTU dell'interfaccia di rete di uscita (NI).

2.3.9.1 Configurazione delle regioni di memoria

Durante l'inizializzazione del componente pNA+, vengono create e inizializzate varie strutture di memoria, come message block, data block e data buffer. Il numero delle strutture presenti viene configurato nella tabella di Configurazione del pNA+.

Essendoci una relazione uno ad uno tra i data block e i data buffer, viene allocato un data block per ogni buffer configurato nel sistema.

Nella inizializzazione dello spazio di memoria a disposizione si deve tener conto del fatto che avere pochi buffer può far peggiorare le prestazioni del sistema, ma anche aumentarne il numero.

Quindi per ottenere delle ottime prestazioni si devono trovare dei compromessi modificando i seguenti parametri:

- Numero di message block
- Configurazione dei buffer
- Dimensione dei buffer
- Dimensione dei buffer di 128 bytes
- Dimensione dei buffer zero

2.3.9.2 Configurazione dei buffer

Per la configurazione dei buffer, vengono determinati due attributi: la dimensione e il numero dei buffer.

Il buffer viene usato per immagazzinare i dati, le intestazioni dei protocolli, e gli indirizzi.

Il componente pNA+ copia i dati che passano attraverso le due interfacce, quella di utente e quella di rete, in dei buffer interni il cui numero e dimensione deve essere configurata dal sistema tramite il seguente algoritmo:

1. Per prima cosa si deve cercare di trovare una dimensione del buffer uguale a quella dei dati da memorizzare.
2. Se non è possibile allora si cerca la più piccola dimensione che possa contenere tutti i dati richiesti da memorizzare.
3. Se non è possibile si seleziona per il buffer la dimensione massima.

Dopo aver selezionato la dimensione, il componente pNA+ cerca nella lista di buffer di quella misura quelli liberi. Se non ci sono buffer disponibili viene restituito zero, oppure se sono disponibili ma la loro dimensione non è sufficiente, i dati vengono copiati in più buffer.

Ci sono alcune dimensioni fisse, tipo zero e 128 bytes, che servono rispettivamente per creare triple di messaggi con buffer esterni e per memorizzare le intestazioni dei protocolli e gli indirizzi.

La gestione della memoria è ottimizzata per copiare i dati e per la frammentazione dei pacchetti. Durante queste operazioni, il componente pnA+ alloca un message block addizionale per poter riusare il data block e il buffer. Il numero di copie e di frammentazioni per buffer dipende dalla misura del buffer e dalla misura del MTU della NI configurata nel sistema.

Il massimo numero di frammentazioni, per buffer di dimensione più piccola del MTU è due, e il massimo numero di frammentazioni per tutti gli altri buffer è la dimensione diviso il MTU.

Il numero di message block configurati dovrebbero essere uguali al numero totale dei frammenti che possono essere formati dai buffer configurati nel sistema.

In molti casi, è sufficiente configurare il numero totale dei message block come il doppio del numero totale dei buffer configurati .

Capitolo 3

Il Protocollo ICMP e le sue applicazioni

3.1 Il protocollo ICMP

Il protocollo IP fornisce un meccanismo di trasferimento dei pacchetti dal mittente al destinatario secondo un approccio *best effort* (letteralmente: sforzo migliore).

Questo vuol dire che lo strato IP non è in grado di garantire la consegna dei pacchetti al destinatario, ma esegue dei tentativi di consegna, al meglio delle possibilità di cui dispone e non si fa remore di ignorare i datagrammi che per qualche motivo non riesce a gestire; di fatti alcuni datagrammi vengono ignorati, cioè "dropped on the floor" (lasciati cadere a terra).

Per questo motivo viene usato un meccanismo che consente un *error-reporting*, delegato ad un altro protocollo, l'**ICMP**, che sta per *Internet Control Message Protocol*, cioè "protocollo per i messaggi di controllo in Internet" (vedi figura 3.1.1).

Oltre a ciò comprende un certo numero di messaggi per richiedere ed ottenere informazioni.

I messaggi ICMP vengono spediti in diverse situazioni: per esempio, quando un datagramma non trova la sua destinazione, quando il gateway non ha la capacità di memorizzare i dati per poi rispedirli, oppure quando il gateway può direzionare l'host per spedire il traffico su un cammino più breve.

Il protocollo IP non garantisce l'arrivo sicuro del pacchetto dati.

Lo scopo di questi messaggi di controllo è di fornire feedback sulle operazioni di scambio dei pacchetti e di migliorare le prestazioni della rete.

La specifica del protocollo ICMP e' contenuta nel documento *RFC 792*.

La maggior parte dei messaggi ICMP vengono trasmessi da un router intermedio di percorso alla stazione trasmittente per indicare l'avvenuto scartamento di un pacchetto per vari motivi.

E' da notare che, in caso di scartamento di un pacchetto ICMP stesso, non viene inviato un altro messaggio ICMP, in modo tale da poter evitare effetti valanga o loop infiniti.

Percio' la presenza di ICMP non garantisce che le stazioni trasmittenti scoprano la perdita di tutti i pacchetti scartati; il controllo di flusso e la ritrasmissione devono essere affidati a protocolli a livelli piu' alti.

ICMP e' implementato direttamente sopra IP e viene considerato un *protocollo di supporto*, al contrario dei *protocolli di servizio* (figura 3.1.1).

I protocolli di supporto svolgono funzioni non direttamente utili per l'utente, cioè non portano dati, ma sono funzionali al corretto funzionamento di altri protocolli, portando solitamente dati "di controllo".

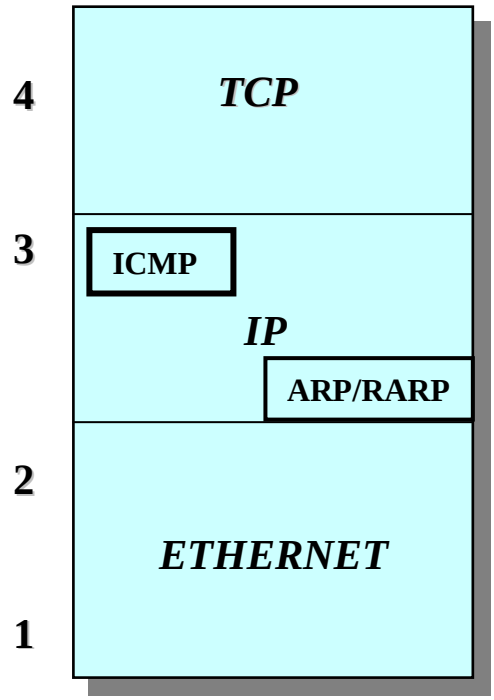


Fig. 3.1.1. Protocollo ICMP

Vengono associati ad uno strato della pila ISO-OSI ma questa associazione ha solo un valore concettuale. Essi viaggiano normalmente imbustati in pdu di protocolli "che offrono servizi".

La scelta del protocollo che serve da veicolo è basata su criteri di convenienza architetturale e non è legata al livello alla pila ISO-OSI a cui concettualmente viene associato il protocollo di supporto (es: alcuni protocolli di routing vengono veicolati da protocolli di livello 4, arp e rarp vengono veicolati dal livello 2).

I protocolli di servizio invece sono quelli che compiono il lavoro utile, cioè portano i dati dell'utente; essi sono associati ad un servizio offerto a livello superiore e non hanno alcun vincolo sul tipo di dato portato.

L'ordine di imbustamento rispetta quello della pila, cioè pdu di protocolli di livello n vengono imbustati in pdu di livello n-1 (esempi: tcp, ip, ftp, ethernet).

Essi vengono associati ad uno strato della pila ISO-OSI, ma questa associazione ha solo un valore concettuale.

Tali protocolli viaggiano normalmente imbustati in PDU di protocolli "che offrono servizi".

Tutti i messaggi ICMP iniziano con una testata comune a 32 bit (vedi figura 3.1.2).

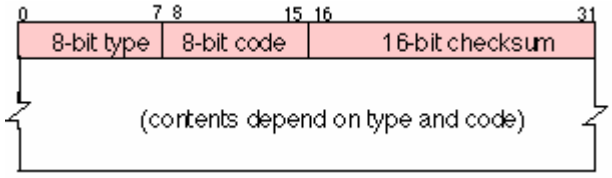


Fig. 3.1.2. Struttura del pacchetto ICMP.

Il campo **Checksum** e' calcolato con lo stesso algoritmo della testata IP.

Il campo **Tipo** esprime il tipo di messaggio, il campo **Codice** esprime una variante che dipende dal tipo.

3.2 Errori rilevabili dal protocollo IP

Lo strato Internet può rilevare un buon numero di errori: errori di checksum (solo header), TTL spirato, mancanza di route per la rete di destinazione e impossibilità di consegnare il pacchetto al destinatario (per esempio nessuno risponde ad una chiamata ARP).

Generalmente lo strato IP scarta tutti i datagrammi che presentano qualcuno di questi problemi.

Alcuni di essi (per esempio l'errore di checksum) non potrebbero in ogni caso essere notificati, perchè non si dispone nemmeno di un indirizzo del mittente che non sia affetto da errore.

Per questo vengono definiti molti tipi di messaggi ICMP (vedi tabella 2).

TAB. 2

Tipi di Messaggi ICMP

Tipo	Descrizione
0	Echo Reply
3	Destinaion Unreachable
4	Source Quench
5	Redirect
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time To Live Exceeded
12	Parameter Problem
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply

Possiamo dividere i messaggi ICMP in due classi: i *messaggi di errore* e i *messaggi di informazione*.

I messaggi di errore possono essere così elencati:

- SOURCE_QUENCH, definito come rallentamento, soffocamento della sorgente.
- TIME_EXCEEDED, tempo scaduto. Viene mandato quando un pacchetto arriva ad un *intermediate system* con un *time-to-live* pari a zero.

- DESTINATION_UNREACHABLE, la macchina destinazione del pacchetto per qualche motivo è irraggiungibile.
- REDIRECT , ridireziona. Un gateway intermedio si accorge che il prossimo gateway cui dovrebbe inoltrare il pacchetto sta sulla stessa sottorete del mittente, e avvisa quest'ultimo dell'esistenza di una via più breve per la destinazione.
- PARAMETER_PROBLEM, problema con i parametri. Il gateway ha ricevuto un pacchetto IP il cui valore del checksum è corretto (non ci sono errori di trasmissione), ma di cui non riesce ad interpretare i valori dei parametri.

I messaggi di informazione invece sono i seguenti:

- ECHO_REQUEST (richiesta di echo). Un host chiede alla macchina destinazione di rimandare indietro lo stesso pacchetto.
- ECHO_REPLY (risposta ad una richiesta di echo). La macchina destinazione di un precedente ECHO_REQUEST rimanda indietro il pacchetto ricevuto (dopo aver scambiato il campo sorgente con quello destinazione e dopo aver modificato il campo che specifica se si tratti du un request o di un reply).
- INFORMATION_REQUEST e INFORMATION_REPLY.
- TIMESTAMP e TIMESTAMP_REPLY. Si tratta di un pacchetto che continene informazioni necessarie per la sincronizzazione degli orologi delle macchine.

Per il lavoro eseguito si sono utilizzati i messaggi di informazione per controllare la funzionalità della connessione instaurata tra la stazione radio e un terminale qualsiasi della rete aziendale.

Per l'esattezza si sono usati i messaggi ***Echo Request*** ed ***Echo Reply***.

Con il primo un host chiede alla macchina di destinazione di rimandare indietro lo stesso pacchetto, mentre con il secondo la macchina di

destinazione di un precedente Echo Request rimanda indietro il pacchetto ricevuto (dopo aver scambiato il campo sorgente con quello di destinazione e dopo aver modificato il campo che specifica se si tratti di un request o di un reply).

3.3 Il comando PING

Un comando che fa uso del protocollo ICMP è il **ping**, che invia una successione di pacchetti ad una stazione per verificarne la raggiungibilità.

Vediamo come funziona: supponiamo che dalla stazione A si voglia controllare l'integrità della connessione fino alla stazione B.
Si esegue il comando ping, passandogli come argomento l'indirizzo della stazione B.

Il programma manda una serie di messaggi ICMP del tipo **Echo_Request** (generalmente uno al secondo) dalla stazione A verso la stazione B.

Quando la stazione B riceve un pacchetto Echo_Request, il suo strato Internet si occupa di rispondere con un nuovo datagramma ICMP del tipo **Echo_Reply**, che viene mandato indietro alla macchina A.

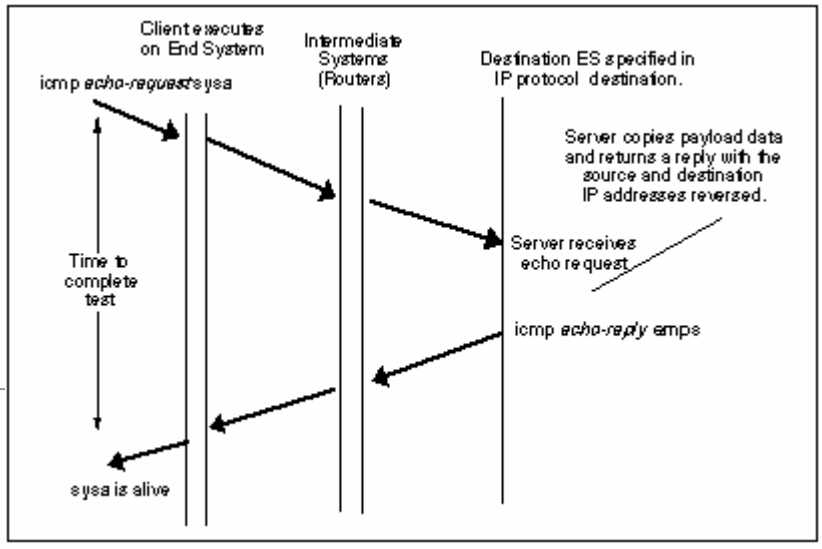


Fig. 3.3.1. Diagramma spazio-tempo del comando Ping.

Il programma ping userà le informazioni così collezionate (esistenza dei pacchetti di ritorno, tempo percorso da ogni pacchetto, etc.) per calcolare dei valori statistici sulla bontà della connessione e presentarli all'utente.

3.4 Vantaggi e svantaggi del PING

Il comando ping è un ottimo strumento di verifica della connessione instaurata tra due terminali, ma oltre a presentare dei vantaggi, purtroppo presenta anche degli svantaggi.

I vantaggi possono essere così elencati:

- Inserisce in ogni pacchetto, che viene trasmesso, una sequenza numerica unica e riporta quale sequenza numerica ha ricevuto. Così si può vedere se il pacchetto è stato duplicato, riordinato o perso.
- Esegue il checksum di ogni pacchetto che scambia. Possiamo così capire alcune forme dei pacchetti danneggiati.
- Inserisce in ogni pacchetto un timestamp, che è di ritorno e può essere facilmente usato per sapere quanto tempo c'è voluto per avere ogni pacchetto (**RTT**).

- Riporta altri messaggi ICMP che possono altrimenti essere non riconosciuti dal sistema.

Gli svantaggi invece sono i seguenti:

- Alcuni router possono silenziosamente buttare dei pacchetti. Il ping non dice le ragioni del perchè un pacchetto non ha risposta (su Ethernet capita di perdere pacchetti perchè non ha un acknowledgments link-layer).
- Non può spiegare perchè un pacchetto viene duplicato, buttato o ritardato.
- Non può descrivere passo dopo passo l'host che maneggia il pacchetto e ogni cosa che succede ad ogni step del cammino compiuto.

Capitolo 4

Configurazione dei componenti software

4.1 Startup del Sistema Operativo

La configurazione del pSOSystem è facilmente modificabile dall'editing del file **sys_conf.h** (vedi ***Appendice***).

In questo file sono presenti definizioni di strutture, costanti e variabili che controllano molti parametri di sistema e, a seconda delle specifiche dell'apparato, definiscono il numero dei task che possono essere attivi concorrentemente.

Il **sys_conf.h** risiede appunto nella directory di applicazione.

Ogni componente software ha la propria configurazione e le proprie caratteristiche di startup, che dipendono dalle capacità richieste.

I componenti software, come il **pSOS+**, **pROBE+**, **pNA+** e **pREPC+**, costituiscono i blocchi di base di tutto il sistema.

Ad ogni componente viene associata una tabella di configurazione, che viene utilizzata dai componenti corrispondenti per ottenere i propri parametri di configurazione.

La figura 4.1.1 mostra le relazioni tra i vari elementi che i componenti usano per trovare i loro parametri, le aree dati e altre informazioni di configurazione.

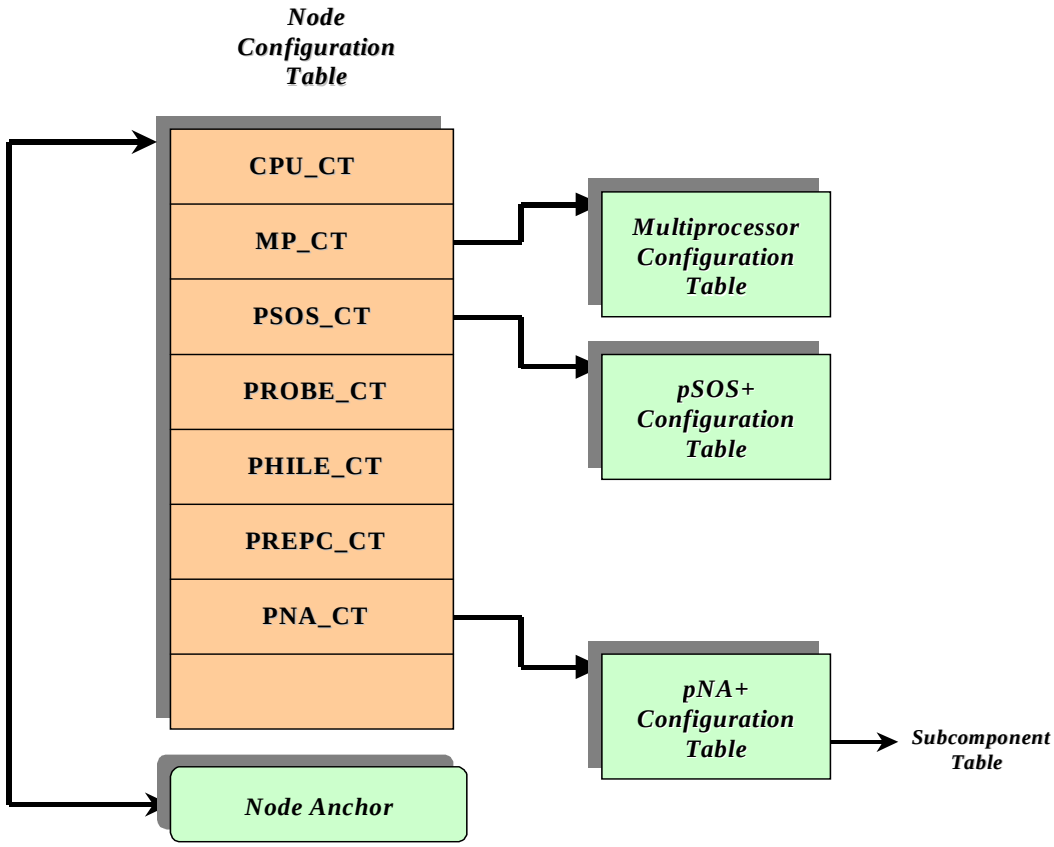


Fig. 4.1.1. Struttura della Tabella di Configurazione.

Il *Node Anchor* è il singolo punto fisso di riferimento per tutti i componenti software installati nel sistema.

Questo nodo è un collegamento critico perchè la localizzazione delle informazioni di configurazione dei vari componenti presenti dipende da esso.

Durante lo starup di sistema, il pSOSystem inizializza tutti i componenti presenti nella tabella di configurazione.

Il codice che inizializza la tabella di configurazione risiede nel file condiviso e di sola lettura **sysinit.c** che risiede nella directory di sistema.

In questo file, il codice sorgente contiene molte linee dove la compilazione dipende dai valori definiti nel file di sistema **sys_conf.h**.

Ad esempio, sono inclusi i parametri relativi al driver di rete LAN e i propri indirizzi IP.

4.2 Configurazione dell'interfaccia di rete

I seguenti parametri controllano la configurazione dell'interfaccia LAN:

- **SD_LAN1**, abilita o disabilita l'interfaccia di rete, a seconda che il suo valore sia YES o NO;
- **SD_LAN1_IP**, rappresenta l'indirizzo IP usato per l'interfaccia LAN. Alternativamente, la costante può essere inizializzata dalla **USE_RARP**, nel qual caso il pSOSystem utilizzi il protocollo *RARP* per ottenere l'indirizzo IP.
- **SD_LAN1_SUBNET_MASK**, è la maschera di sottorete utilizzata per l'interfaccia di rete, 0 per nessuna.

Se si vuole avere una sola interfaccia di rete, si può abilitare inizializzando *SD_LAN1* a YES. Se *SD_LAN1* è NO, il valore degli altri parametri *SD_LAN_** non verranno inizializzati.

Per quanto riguarda l'occupazione di uno spazio di memoria, viene definito il parametro **SC_RAM_SIZE** a zero.

Normalmente il pSOSystem usa l'intero spazio di memoria a disposizione sulla scheda per un'allocazione dinamica, a partire da una *Regione zero* di memoria. Si può sovrascrivere questo spazio inizializzando il parametro *SC_RAM_SIZE* ad un valore diverso da zero. Se viene fatto, il pSOSystem non toccherà nessuno spazio di memoria dopo il primo *SC_RAM_SIZE* bytes.

Questo metodo viene usato quando si costruisce una Boot ROM, in quanto c'è bisogno di RAM disponibile per il downloading del codice.
Il parametro **SD_DEF_GTWY_IP** specifica il gateway di default usato per l'instradamento dei pacchetti.

Quindi abbiamo visto come alcuni valori dei parametri definiti nel file di sistema **sys_conf.h** controllano i valori di quelli definiti nelle tabelle di configurazione corrispondenti ai vari componenti del sistema.

Sebbene la maggior parte di questi componenti sono determinati dai valori nel file **sys_conf.h** , ci sono altri elementi che invece non lo sono, perchè non possono essere specificati, in quanto contengono indirizzi di partenza del pSOS+ Kernel.

4.3 Sequenza dello Startup del Sistema

La seguente figura 4.3.1 illustra i *Vettori di Sistema* per i processori 68K.

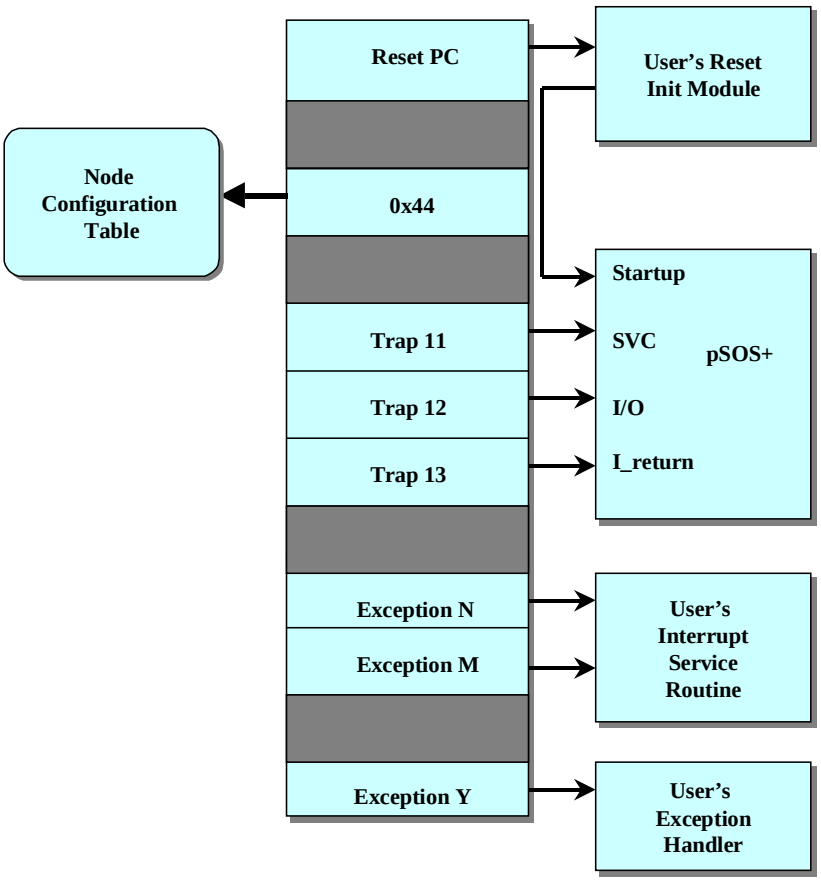


Fig. 4.3.1. Descrizione dei Vettori di Sistema dei processori 68K.

Si possono descrivere i componenti richiesti per un tipico sistema **pSOS+** di base:

- Un modulo di startup per far partire il sistema o per il reset dell'inizializzazione;
- Il pSOS+ e altri componenti richiesti (per esempio, pROBE+ e pHILE+);
- Il *Node Anchor* e il *Node Configuration Table*;
- Le tabelle di configurazione per il pSOS+ e per l'installazione degli altri moduli;

- I task di applicazione, **ISR**, i codici e i dati di inizializzazione, se presenti;

Per un sistema basato su ROM, alcuni di questi componenti risiedono nella ROM. Per un sistema basato su RAM o per un sistema sotto test o integrazione, alcuni di essi possono essere caricati nella RAM, altri dall'utente durante l'inizializzazione o durante il sistema di Debug (**pROBE+**).

La maggior parte dei casi, specialmente in un sistema *memory-mapped*, è possibile caricare i task di applicazione o i driver dinamicamente, cioè *run time*.

La seguente figura 4.3.2 mostra la possibile sequenza di startup del sistema.

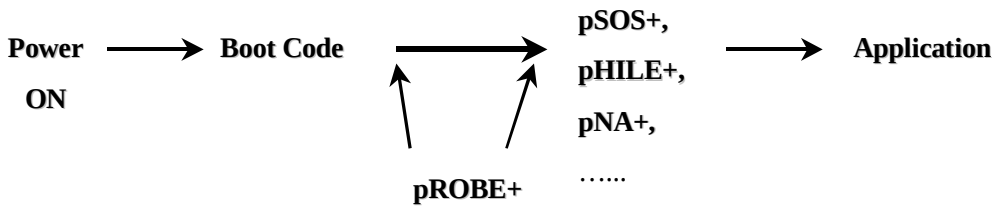


Fig. 4.3.2. Sequenza di Startup del Sistema.

Tipicamente l'accensione o il reset passano il controllo al codice di Boot dell'utente, il quale gestisce le necessarie inizializzazioni.

Esso potrebbe anche inizializzare la configurazione richiesta per il componente software nel sistema, che consiste nel *Node Anchor*, nel *Node configuration table* e in altri componenti presenti nella tabella.

Durante il debug, per esempio, è possibile settare la configurazione per poter far partire il pROBE+ e farlo eseguire. Se il pROBE è presente, il codice di Boot potrebbe passare il controllo ad esso.

Il pROBE+ può essere usato per scaricare altri componenti dalla lista e far partire l'esecuzione delle applicazioni del pSOS+.
Si può far partire il pSOS+ da uno dei seguenti modi:

- Dal passaggio del controllo allo Startup del pSOS+, assumendo che la CPU è nello stato di supervisore;
- Dal pROBE+, con alcuni comandi;
- Specificando un modo di startup "silenzioso" nella configurazione del pROBE, così che esso inizializzi se stesso per passare il controllo allo startup del pSOS+ senza fermarsi.

Dopo lo startup del pSOS+, per prima cosa viene usato il Node Anchor per localizzare la tabella di configurazione del pSOS+. Dopo di chè il pSOS+ prenderà un segmento di memoria dall'inizio della *Regione zero* di memoria per la propria area dati.

Esso userà la parte iniziale di questa area per le sue strutture dati chiave. Il prossimo segmento di memoria andrà inserito nella pila di sistema.
Ora il pSOS+ può verificare la tabella di configurazione.

Se altri componenti runtime sono presenti, il pSOS+ localizza e chiama lo Startup di questi componenti per poterli inizializzare.

L'ultimo step nello Startup del pSOS+ è quello di creare e attivare il task di **IDLE** e il task **ROOT** di utente, che poi sarà eseguito e darà inizio all'esecuzione degli altri task progettati.

4.4 Tabella di configurazione del pNA +

La tabella di configurazione del pNA+ viene gestita dall'utente e provvede sia alle informazioni hardware sia alle specifiche applicazioni richieste dal pNA+.

La tabella può risiedere sia nella RAM che nella ROM. L'indirizzo di partenza della sua tabella di configurazione deve essere specificato dall'entrata **nc_pnact** nel Node Configuration Table (vedi ***Appendice***).

La tabella deve includere i parametri relativi agli indirizzi di inizio del codice del pNA+ e di inizio dell'area dati (che devono essere allocati nella RAM, inclusa la misura dell'area) e al numero di interfaccie che devono essere installate nel proprio sistema (nel nostro caso solo una).

Deve essere presente anche l'indirizzo di dove sono stati immagazzinati i dati relativi alla tabella dell'interfaccia di rete, la quale definisce le caratteristiche dell'interfaccia di rete che è inizialmente installata nel sistema.

Il contenuto dell'iniziale tabella del **NI** (*Network Interface*) sarà copiata sull'attuale tabella del NI durante l'inizializzazione del pNA+.

4.5 Accesso del pNA+ alla Network Interface

Il pNA+ accede alla rete attraverso l'interazione con un livello software, gestito dall'utente, chiamato ***Network Interface*** (**NI**).

L'interfaccia tra il pNA+ e la NI è standard ed è indipendente dallo strato fisico; di fatti il pNA+ viene isolato dalle caratteristiche fisiche della rete.

Essenzialmente l'NI è un componente driver che permette l'accesso ad una trasmissione media.

Devono essere presenti un'interfaccia NI per ogni rete connessa al nodo pNA+. Ad esempio, un nodo è connesso ad una sola rete quindi deve avere una sola NI.

In altri casi un nodo può essere connesso a più reti simultaneamente e può avere quindi più interfaccie di rete.
Ad ogni NI deve essere assegnato un unico indirizzo IP.

4.6 Gestione dei pacchetti al livello della NI

La fondamentale unità di comunicazione usata dal pNA+ è il **pacchetto**.
Per trasmettere i dati, il pNA+ prepara un pacchetto e poi lo passa alla NI per la trasmissione.

Questa è la responsabilità della NI: inviare il pacchetto verso una specifica destinazione.
Il pNA+ supporta due tipi di interfaccie:

(a) pNA+-*Interfaccia Indipendente*;
(b) pNA+-*Interfaccia Dipendente*.

E' il pNA+ che determina quale tipo di interfaccia viene supportato, tramite il settaggio di un **flag** (*IFF_RAWMEM*) nella struttura della tabella di interfaccia (**ni_init**), inclusa nel file **pna.h**.

L' *Interfaccia Indipendente* supporta pacchetti che sono contenuti in blocchi contigui di memoria chiamati buffer packet; quando il pNA+ chiama l'NI per spedire un pacchetto, passa il puntatore al buffer che contiene il pacchetto. Quando viene ricevuto il pacchetto, l'NI lo passa al pNA+, restituendo il puntatore al buffer usato per immagazzinare i dati.
Questo abilita l'NI ad avere la gestione della propria memoria.

Nel nostro caso invece, abbiamo usato il tipo di ***Interfaccia Dipendente***, in cui il pNA+ internamente ottimizza la gestione della memoria per trasferire i pacchetti tra i vari strati protocollari.

Ogni pacchetto è rappresentato da una lista collegata di **triple** costituite dalle seguenti strutture dati: ***Message Block***, ***Data Block*** e ***Data Buffer***.

L'interfaccia supporta il trasferimento dei pacchetti usando liste collegate di message block.

Quando il pNA+ spedisce un pacchetto, passa alla NI un puntatore ad un message block. Similarmente, quando viene ricevuto un pacchetto, viene attaccato un message block al buffer dei dati e viene passato al pNA+ un puntatore al message block attraverso l'entrata **Announce_Packet**.

Questo metodo offre una maggiore performance, eliminando il bisogno di copiare i dati tra l'NI e il pNA+.

Durante la chiamata della funzione di inizializzazione dell'interfaccia di rete, *NI_INIT*, viene passato alla NI un puntatore alle funzioni di gestione della memoria (**pna_allocb**, **pna_esballoc**, **pna_freeb** e **pna_freemsg**) (in ***Appendice***).

La NI userà queste routines per allocare e deallocare le triple di message block.

Per la trasmissione dei pacchetti, il pNA+ chiama le funzioni *NI_SEND* o *NI_BROADCAST* per preparare e spedire il pacchetto verso la destinazione.

Deve anche essere responsabile del rilascio della memoria occupata dalla lista dei message block, dopo averli spediti.

Una volta ricevuto il pacchetto, tipicamente attraverso un *Interrupt Service Routine* (**ISR**), il driver agirà nei seguenti modi:

1. L'interrupt trasferisce il controllo al pacchetto ISR, che fa parte dell'NI, e che riceve il pacchetto nel buffer dei dati.

2. Il driver attaccherà il buffer al message block usando la chiamata di servizio **pna_esballoc**. Poi chiamerà l'**Announce_Packet** e passerà al pNA+ il puntatore al message block. Il pNA+ accoderà il pacchetto e ritornerà al ISR.
3. Il driver ripeterà lo step 2 per ogni pacchetto sospeso.
4. Il pNA+ processerà il pacchetto e poi libererà il buffer .

Quando un pacchetto arriva, l'NI deve informare il pNA+ tramite la chiamata **Announce_Packet**. Il suo indirizzo viene passato all'NI dal pNA+ come parametro di ingresso quando viene chiamata la funzione NI_INIT.

Per chiamare Announce_Packet, l'NI inserisce sei parametri di ingresso nella pila e poi usa una istruzione **JSR** (*Jump Service Routine*) per passare il controllo alla chiamata. Il pNA+ processa il pacchetto e restituisce il comando all'NI usando una istruzione **RTS** (*Return Service Routine*).

Non ci sono parametri di uscita.

L'NI può inizializzare i parametri relativi alla rete, come l'indirizzo IP, la maschera IP, l'indirizzo di destinazione IP per collegamenti punto-punto, l'MTU, l'indirizzo IP di broadcast o alcuni flags dell'interfaccia.

4.6.1 pNAD Daemon Task

I protocolli internet non sono sempre sincroni.

Non tutte le attività del componente pNA+ sono direttamente inizializzate dalla chiamata di un task applicativo. Piuttosto certi processamenti di attività sono gestiti in risposta ad eventi esterni come pacchetti entranti e fine di cicli temporali.

Per gestire queste operazioni asincrone il componente pNA+ crea un task *DAEMON* chiamato ***pNAD***.

Esso è creato durante l'inizializzazione del pNA, con priorità più elevata (255) rispetto agli altri task, per assicurarsi sempre il comando.

Il pNAD è normalmente bloccato in uno stato di attesa, aspettando che si verifichi uno di questi due eventi, codificati nei bit 30 e 31, prima di sbloccarsi e ricominciare a girare:

a) Il primo evento (bit 31) viene inviato dal pNA+ dopo la ricezione di un pacchetto quando viene chiamata la *Announce_Packet*, o quando avviene un interrupt (*ISR*) o un polling sulla seriale (ni_poll). Il pNAD, basandosi sul contenuto del pacchetto, esegue diverse azioni: si sblocca e invia un pacchetto di reply. Se l'esecuzione del pNAD viene inibita o ritardata anche l'instradamento del pacchetto sarà inibito o ritardato.
b) Il secondo evento (bit 30) viene inviato ogni 100 ms come risultato di una TM_EVEVERY system call del pSOS+. Quando il pNAD si sblocca ogni 100 ms, esso elabora le specifiche sui timeouts, ed al termine esegue un sondaggio con la system call ni_poll su ogni interfaccia che ha il suo flag POLL settato.

4.6.2 Struttura del pacchetto

Un pacchetto, nel pNA+, consiste in liste collegate di message block, in cui ognuno rappresenta solo una parte del pacchetto.
La sua struttura è definita nel modo seguente:

```
struct msgb {
        struct msgb *b_next;
        struct msgb *b_prev;
        struct msgb *b_cont;
        unsigned char *b_rptr;
        unsigned char *b_wptr;
        struct datab *b_datap;
        };
typedef struct msgb mblk_t;
```

dove **b_next** contiene il puntatore al prossimo pacchetto in coda, **b_prev** contiene il puntatore al precedente pacchetto in coda, **b_cont** contiene il puntatore al prossimo message block del pacchetto, **b_rptr** punta al primo byte non letto nel data buffer relativo al message block, **b_wptr** punta al primo byte non scritto nel data buffer e **b_datap** punta al data block relativo al message block (vedi ***Appendice***).
Il data block specifica le caratteristiche del data buffer a cui esso è riferito.

La sua struttura (**datab**) comprende un puntatore al primo byte del buffer, all'ultimo byte più uno del buffer, al numero di riferimenti del data buffer e al tipo di pacchetto.

Il data buffer è costituito da un blocco contiguo di memoria usato per immagazzinare i pacchetti.

4.6.3 Requisiti per i buffer e le aree dati

I requisiti per l'area dati del pNA+ dipendono da come vengono inizializzate dall'utente le tabelle di configurazione del pNA+ e del pSOS+.

La dimensione dell'area dati è la somma dei valori generati da alcuni parametri, come le variabili statiche del pNA, la tabella di interfaccia, la tabella di routing, la tabella ARP e la tabella relativa ai parametri dell'host considerato.

Mentre la quantità di memoria totale richiesta dipende dalla configurazione del buffer, insieme al numero dei message block configurati e può essere allocata a partire sia dalla Regione Zero che da una fissata locazione, la quale dipende dal parametro **nc_data** all'interno della Tabella di Configurazione (vedi ***Appendice***).

Il pNA+ viene configurato nel modo supervisore, usando lo stack relativo al supervisore per immagazzinare temporaneamente le variabili.

Nel caso peggiore lo stack usato è 900 bytes più grande dello stack per il driver di rete.

Capitolo 5

Progetto dell'interfaccia di rete

5.1 Introduzione

Come è stato già presentato nei capitoli precedenti, il microprocessore MC68EN360, della famiglia Motorola, lavora con canali seriali e porte con interfaccia parallela per implementare i protocolli scelti dall'utente e per gestire i canali SDMA (*Serial Device Memory Access*) che trasferiscono dati tra le SCC (*Serial Comunication Controller*) e la memoria interna (Dual-Port-Ram).

Ogni SCC ha un canale diverso di ricezione e trasmissione e i dati associati ad ognuna di esse sono immagazzinati nei buffer, di lunghezza massima (64K-1) bytes.

Ad ogni buffer viene associato un *Buffer Descriptor*, **BD**, che il controllore utilizza per riportare le informazioni dei dati sia trasmessi che ricevuti, come mostrato nella figura 5.1.1.

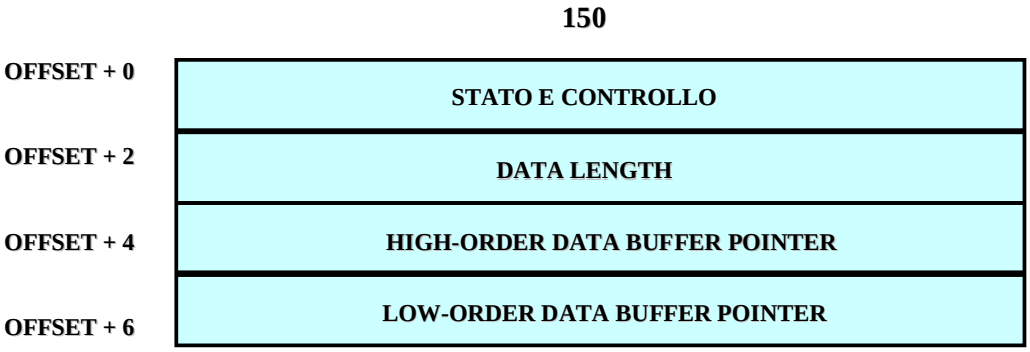


Fig. 5.1.1. Struttura del Buffer Descriptor (BD).

Il primo campo, *Status and Control*, di lunghezza 2 byte, è un registro che rappresenta lo stato del buffer descriptor di trasmissione o di ricezione relativo alla seriale considerata (vedi figura 5.1.2a e 5.1.2b).

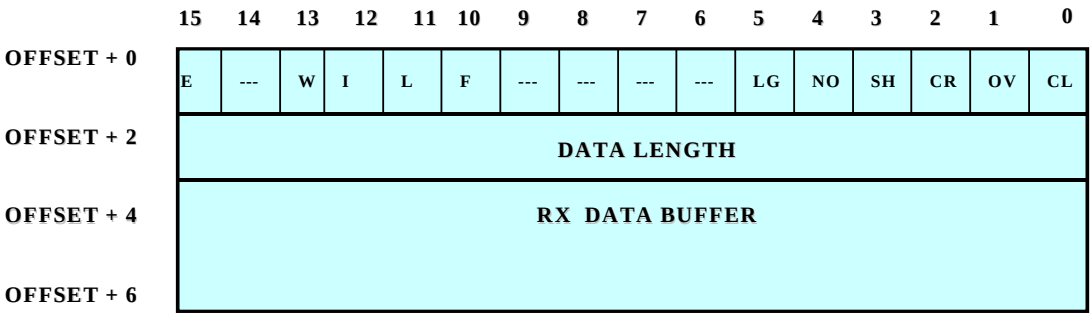


Fig. 5.1.2a

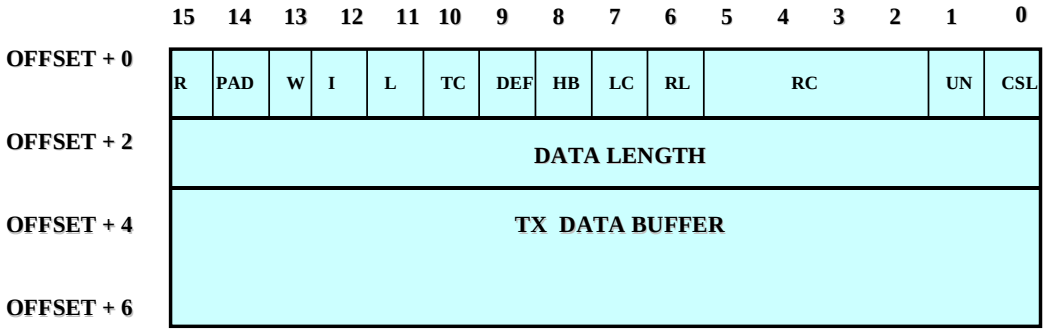


Fig. 5.1.2.b

Lo stato del BD verrà inizializzato in un primo momento dall'utente e poi verrà modificato dal processore, durante le operazioni di trasmissione e ricezione dei pacchetti (vedi ***Appendice***).

Spesso l'allocazione dei BD viene definita dall'utente, il quale può selezionare fino a 100 Buffer Descriptor per la seriale di ricezione e 20 per la seriale di trasmissione, disponendo di 224 BD nella memoria interna del processore da condividere tra le quattro SCC.

La tabella costituita dai BD di trasmissione e ricezione forma una coda circolare di lunghezza programmabile dall'utente, il quale definisce, con un puntatore (rbase, tbase), le entrate della locazione iniziale nella memoria interna (dual-port-RAM) per il set di BD associati ad ogni canale.
In questo modo il corrispondente canale verrà abilitato.

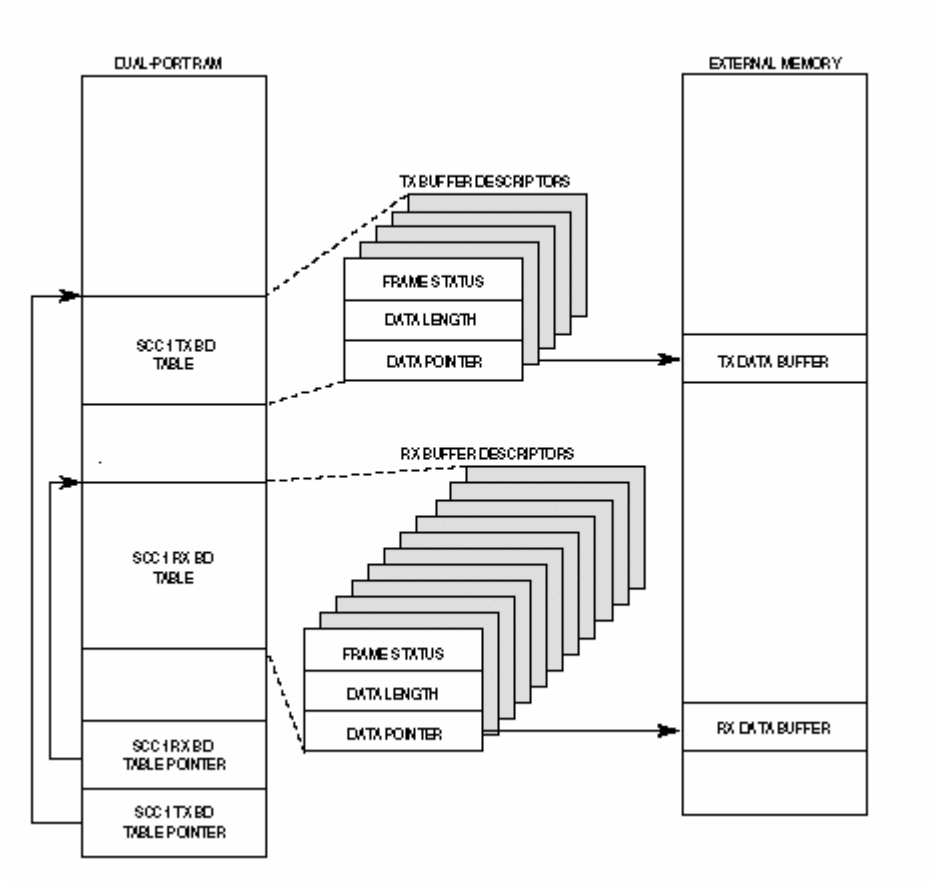


Fig. 5.1.3. Descrizione dell'associazione tra i BD e i buffer di memoria.

5.2 Trasmissione dei pacchetti

In trasmissione, l'utente provvede ad inizializzare i campi della trama IEEE 802.3, inserendo l'indirizzo di destinazione, l'indirizzo sorgente, il tipo di protocollo usato, la lunghezza del pacchetto e i dati da trasmettere.

Il controllore automaticamente aggiunge padding se la trama ha meno di 46 bytes nel campo dati da trasmettere, come richiesto dalle specifiche di trama minima.

Quando viene abilitata la trasmissione (**R**-bit settato ad uno), il controllore userà il primo TxBD della tabella del canale corrispondente. Ogni richiesta del buffer verrà eseguita ogni 128 clock seriali.

Se l'utente però ha la trama da trasmettere pronta, il bit **TOD** (*Trasmit On Demand*) del registro di trasmissione può essere settato per eliminare l'attesa per la prossima richiesta.

Il controllore, dopo essersi assicurato che il canale di trasmissione non sia occupato, inizierà a prelevare i dati dal buffer, asserendo il bit **TENA**, trasmettendo il preambolo, il delimitatore di inizio trama e poi i dati.

Una volta spostati i dati dal Buffer Descriptor alla coda FIFO di trasmissione, il controllore preleverà dalla tabella il prossimo BD, aspettando di inizializzare il bit R di quel BD.

Se durante la trasmissione si verifica una collisione, il controllore immagazzina dai primi 5 agli 8 byte della trama da trasmettere nella RAM interna.

Al termine del corrente BD, viene settato il bit **L** nel Buffer Descriptor e il controllore aggancerà l'FCS alla fine della trama, con la negazione del bit TENA. Esso scriverà lo stato di trama nel BD e cancellerà il bit R.

5.3 Ricezione dei pacchetti

Dopo aver abilitato il lato ricevente della SCC, il controllore prenderà il primo BD presente nella tabella.

Una volta arrivati i dati dal canale trasmissivo, verranno inseriti nel BD relativo alla SCC scelta per Ethernet, e il controllore sposterà i dati, dopo averli processati, nel buffer corrispondente.

Ottenuti i primi bytes della trama, il controllore inizierà la funzione di ricognizione dell'indirizzo, che può essere fisico, di gruppo o di broadcast. Nessuna trama ricevuta viene scritta in memoria finchè l'algoritmo interno di ricognizione dell'indirizzo non viene completato.

Questo migliora l'utilizzazione del bus nel caso di trama non indirizzata a questa stazione.

Se si verifica una collisione durante la ricezione di una trama, i Buffer Descriptor associati ad essa vengono riutilizzati.

Quando il buffer dati è stato riempito, il controllore cancella il bit **E** (*Empty*) nel BD di ricezione e genera un interrupt se il bit **I** è settato.

Se le trame entranti eccedono nella lunghezza del buffer dati, il controllore andrà a prendere il prossimo BD nella tabella e, se è vuoto, continuerà a trasferire il resto della trama nel buffer associato a quel BD.

La lunghezza del BD di ricezione è determinato dal parametro **MRBLR**, di valore pari a 1520, inizializzato durante lo startup del sistema, e collocato nella sezione di memoria interna al processore (parametri RAM) relativa alla SCC scelta.

Una volta completata la ricezione, il controllore ha l'opzione di "campionare" un byte dalla porta parallela B di I/O e di agganciare questo byte alla fine dell'ultimo BD nella trama.

Adesso può essere inizializzato il bit **L** nel campo status del BD di ricezione, cancellato il bit **E** e il controllore può generare un interrupt per indicare che la trama è stata ricevuta ed è in memoria.

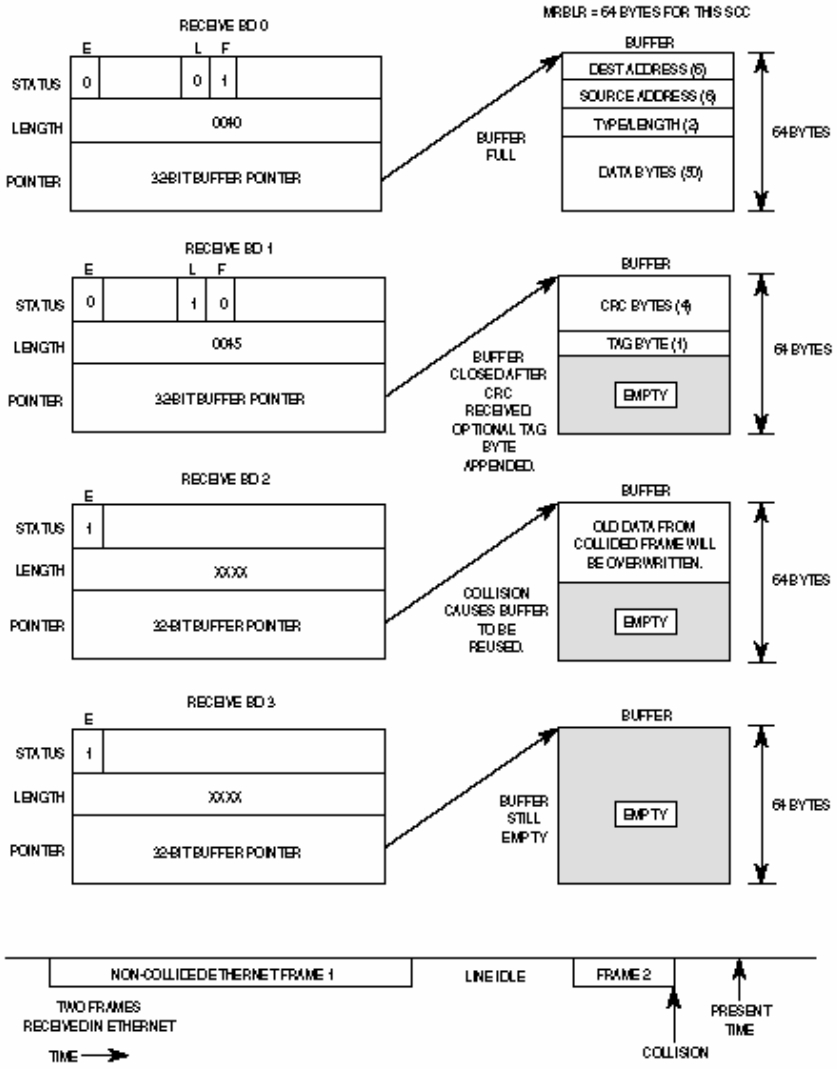


Fig. 5.3.1. Struttura dei BD in ricezione se ho collisione durante la ricezione dei pacchetti.

5.4 Ethernet Address Recognition

Il controllore Ethernet può filtrare le trame ricevute basandosi su differenti tipi di indirizzamenti: fisico, di gruppo, di broadcast o promiscuo.

La differenza tra un indirizzo individuale e uno di gruppo è determinato dal bit **I/G** nel campo di destination address (I/G = 0 implica un indirizzo individuale, mentre I/G = 1 implica un'indirizzo di gruppo). Nel diagramma seguente viene rappresentato l'algoritmo di ricognizione dell'indirizzo di una trama ricevuta.

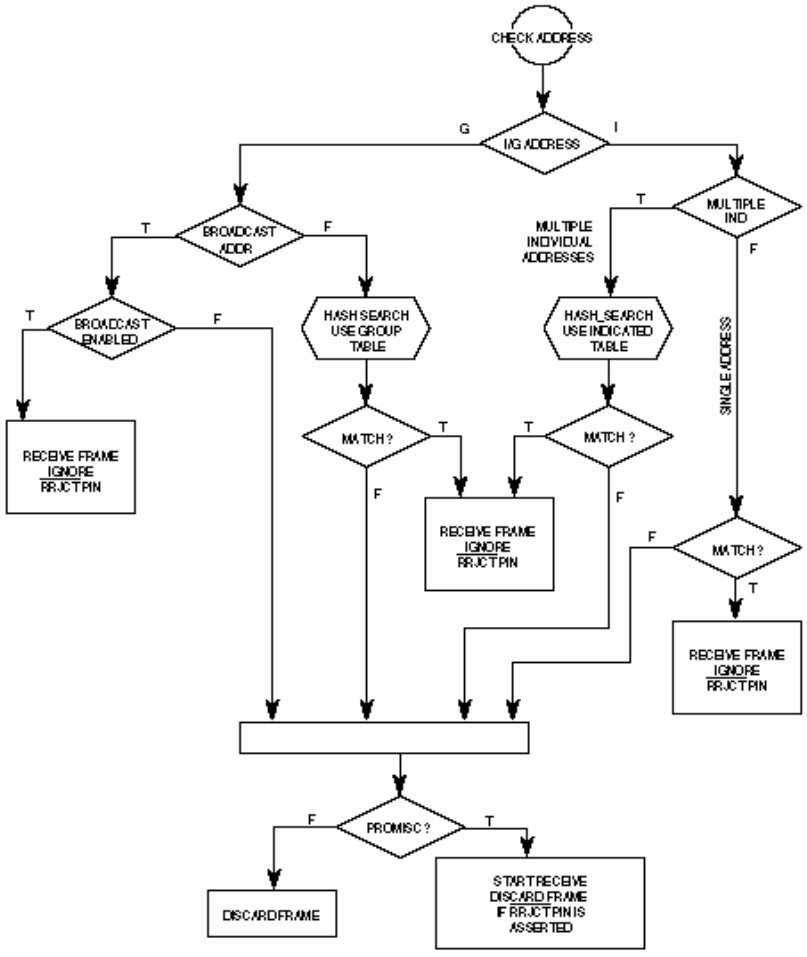


Fig. 5.4.1. Algoritmo di rivelazione dell'indirizzo fisico.

Il controllore, nel caso di indirizzo fisico, confronterà il campo di destination address della trama ricevuta con l'indirizzo fisico che l'utente ha programmato, durante la fase di startup del sistema, nel parametro **PADDR1** della RAM (vedi ***Appendice***).

5.5 Realizzazione del Ping

Per testare l'esistenza nella rete locale aziendale della Stazione radio, si è provato ad implementare il programma Ping in ricezione, inviando ad essa una **Echo_Request** da un qualsiasi pc collegato alla stessa rete.

In particolare, nel nostro caso, la Stazione Radio è stata collegata point-to-point ad un pc attraverso un cavo cross, in modo tale da ridurre la quantità di pacchetti entranti.

Tutto ciò viene attuato solo dopo aver installato l'interfaccia di rete NI sull'apparato.

L'obiettivo è ricevere il pacchetto ICMP contenente la *Echo_Request*, elaborarlo al livello fisico di interfaccia, e passarlo al livello superiore per essere elaborato dal componente pNA+. Al termine si invierà l'**Echo_Reply** di risposta.

Le PDU del protocollo ICMP, essendo quest'ultimo implementato direttamente sopra il protocollo IP, viaggiano normalmente imbustate in PDU di IP, che poi verranno incapsulate dentro le PDU dello strato MAC per essere trasferite sullo strato fisico.

Nel loro cammino verso il basso i dati vengono "rivestiti" di ulteriori informazioni come ad esempio i vari header (intestazioni) ogni volta che attraversano gli strati IP e MAC (vedi figura 5.5.1).

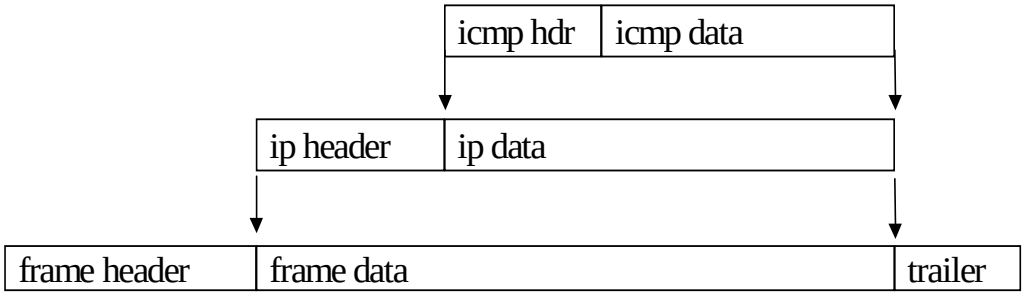


Fig. 5.5.1

Il tempo tra la spedizione dell'Echo_Request e la ricezione dell'Echo_Reply è sempre di 3,7 ms (approssimato ad 1 s), tranne quando l'indirizzo hardware di destinazione non è presente nell'ARP cache della sorgente.

In questo caso l'invio di un **Arp_Request** e l'arrivo della corrispondente **Arp_Reply** può ritardare di qualche millisecondo la spedizione del primo pacchetto ICMP contenente la Echo_Request.

Il Ping misura anche il *Round Trip Time* (**RTT**), informandoci sulla distanza tra l'host di sorgente e l'host di destinazione, in relazione al *Time To Live* (**TTL**), il tempo di vita del pacchetto.

Il TTL viene decrementato ogni volta che il pacchetto viene ricevuto da un host del suo percorso, in modo tale da scartare il pacchetto se questo non riuscisse a raggiungere la destinazione prima che il TTL risulti uguale a zero.

Questo viene fatto per eliminare il traffico di pacchetti sulla rete.

Il pacchetto del Ping è lungo 74 bytes di cui 14 vengono utilizzati per l'Ethernet Header, 20 per l'IP header e gli ultimi 40 bytes per immagazzinare il pacchetto ICMP (può essere visualizzato usando un qualsiasi programma di cattura dei pacchetti che viaggiano in rete).

Ethernet Header:

- Destination Ethernet Address (6 bytes);
- Source Ethernet Address (6 bytes);
- Type (2 bytes), che assumerà il valore di 0x0800 per il protocollo IP:

IP Header:

- Version (4 bits);
- Header length (4 bits);
- Type of service (1 byte);
- Total length (2 bytes), lunghezza del header + dati del pacchetto IP= 60;
- Identification (2 bytes);
- Fragmentation Flags (3 bits);
- Fragment offset (13 bits);
- Time to live (1 byte);
- Protocol (1 byte), codice per identificare il protocollo del livello superiore;
- Header checksum (2 bytes), per controllare che l'header IP sia corretto;
- Source IP Address (4 bytes);
- Destination IP Address (4 bytes);

Pacchetto ICMP:

- Type del messaggio (1 byte), 0 per l'Echo_Reply e 8 per l'Echo_Request;
- Type subcode (1 byte);
- Checksum (2 bytes);
- Identifier (2 bytes);
- Sequence number (2 bytes);
- Optional data (32 bytes);

Tutte queste informazioni devono essere inserite nel buffer di ricezione preparato dalla NI che a sua volta lo deve elaborare per poterlo passare al

livello superiore, rappresentato dal pNA+, trasformandolo nella lista collegata di triple di messaggi.

Questo avviene attraverso la funzione che annuncia l'arrivo dei pacchetti dalla rete, l'**Announce_Packet**, come abbiamo già descritto nel capitolo 4.

Una volta costituita la lista collegata delle triple di message block, è il pNA+ che la elabora per estrapolare tutte le informazioni del pacchetto ed inserirle, dopo averle modificate se necessario, nelle varie strutture di ricezione e trasmissione (TxhfreeHead, RxbfreeHead, etc..).

Per ricevere il ping è stato creato il task **PingRX**, che include le system calls del componente pNA+, con una adeguata priorità rispetto agli altri task, e con un relativo identificatore (vedi ***Appendice***).

5.5.1 Task PingRX

Per prima cosa viene creato il socket, di tipo *raw*, tramite la system call *socket()*, per ricevere i pacchetti ICMP.

Abbiamo scelto un *socket raw*, in quanto il protocollo ICMP, appoggiandosi su IP, è senza connessione, quindi non c'è certezza sull'arrivo del pacchetto a destinazione.

Il pNA+ usa due strutture dati per gestire i socket creati:

1. ***SCB***, *Socket Control Blocks*, servono per immagazzinare le informazioni sullo stato del socket. Durante l'inizializzazione il componente pNA+ crea un numero fisso di SCBs. Un Socket

Control Block viene allocato per ogni socket quando quest'ultimo viene creato attraverso la chiamata *socket()*.

2. ***Open Table***, per ogni task viene costruita una tabella "aperta" che verrà associata al socket creato. Questa viene usata per immagazzinare gli indirizzi degli SCB associati ai socket creati nei task. In questo modo un socket descriptor è un indice di questa tabella.

Quando viene chiamata una system call del pNA+, si possono verificare tre possibili eventi:

I. Il componente pNA+ esegue i servizi richiesti e ritorna al punto in cui è stato chiamato.

II. La system call non può essere completata immediatamente, ma non richiede a chi la richiesta di aspettare. In questo caso il componente pNA+ memorizza le necessarie operazioni e restituisce il controllo al richiedente del servizio. Per esempio la *sendto()* copia i dati dal buffer dell'utente su un buffer interno. I dati però non possono essere attualmente trasmessi anche se il controllo ritorna al task chiamante, il quale continua a girare.

III. La system call non può essere completata immediatamente e il task richiedente deve così aspettare. Per esempio l'utente può attendere di leggere i dati che non sono ancora raggiungibili. In questo caso il componente pNA+ blocca il task chiamante. Il task bloccato è eventualmente rischedato dalla sottosequenza asincrona delle attività.

Per gestire queste operazioni asincrone il componente pNA+ crea un task *DAEMON* chiamato ***pNAD***.

Questo task gestirà le operazioni di polling, le operazioni di ricezione e trasmissione di pacchetti sull'interfaccia inizializzata.

Quindi il pNAD farà da tramite tra l'interfaccia NI e il componente pNA+.

Dopo aver creato il socket viene allocata tanta memoria al message block, tramite la system call *pna_allocb()*, quanta è la dimensione dei dati che vengono ricevuti.

Per attendere i pacchetti ICMP viene usata la *recvfrom()* all'interno di un ciclo while infinito, che viene sbloccato solo quando la system call ha riempito tutta la dimensione del buffer di cui richiede l'indirizzo come parametro di ingresso.

Essendo il socket di tipo raw, la recvfrom() riceve un datagramma alla volta, la cui dimensione viene scelta da chi l'ha inviato. Se la dimensione del parametro è minore di quella del datagramma, allora parte del pacchetto viene buttata. La prossima chiamata della recvfrom() leggerà il successivo datagramma ricevuto e non la parte mancante del precedente pacchetto.

Se non viene usata la system call *ioctl()* per marcare il socket come non-bloccante, la recvfrom() bloccherà il task richiedente fino a quando il datagramma raw non sarà disponibile al socket.

La prima Echo_Request ricevuta verrà passata al pNA+ come message block, verrà processata, verificando se si tratta di un pacchetto di tipo Echo e verranno elaborati i dati in modo tale da costruire il pacchetto ICMP di Echo_Reply, con l'intestazione MAC, quella IP e infine quella ICMP.

Prima di essere inviato al destinatario, il pacchetto ICMP, deve essere passato dal componente pNA+ all'interfaccia NI, tramite la chiamata della funzione *NiLan()*, che, tramite i parametri di ingresso che gli vengono assegnati, gestisce le varie funzioni della NI (polling, broadcast, trasmissione, installazione di nuove NI, etc…). Avendo assegnato come parametro di ingresso a NiLan() la funzione NI_SEND, allora verranno preparati i buffer di trasmissione per l'invio dell'Echo_Reply.

Alla fine il task eseguirà la system call *sendto()* per spedire il messaggio di Echo_Reply al destinatario. Questo avviene per tutti e quattro i pacchetti ricevuti.

Sono riportate nelle figure seguenti, le strutture dei buffer di ricezione e di trasmissione utilizzate per immagazzinare i pacchetti, e i propri valori assegnati, dopo aver ricevuto un interrupt di avviso dell'arrivo di pacchetti, grazie al programma di debug usato, il Vision Ice 7.0. Per la compilazione dei file è stato usato il programma di edit UltraEdit.

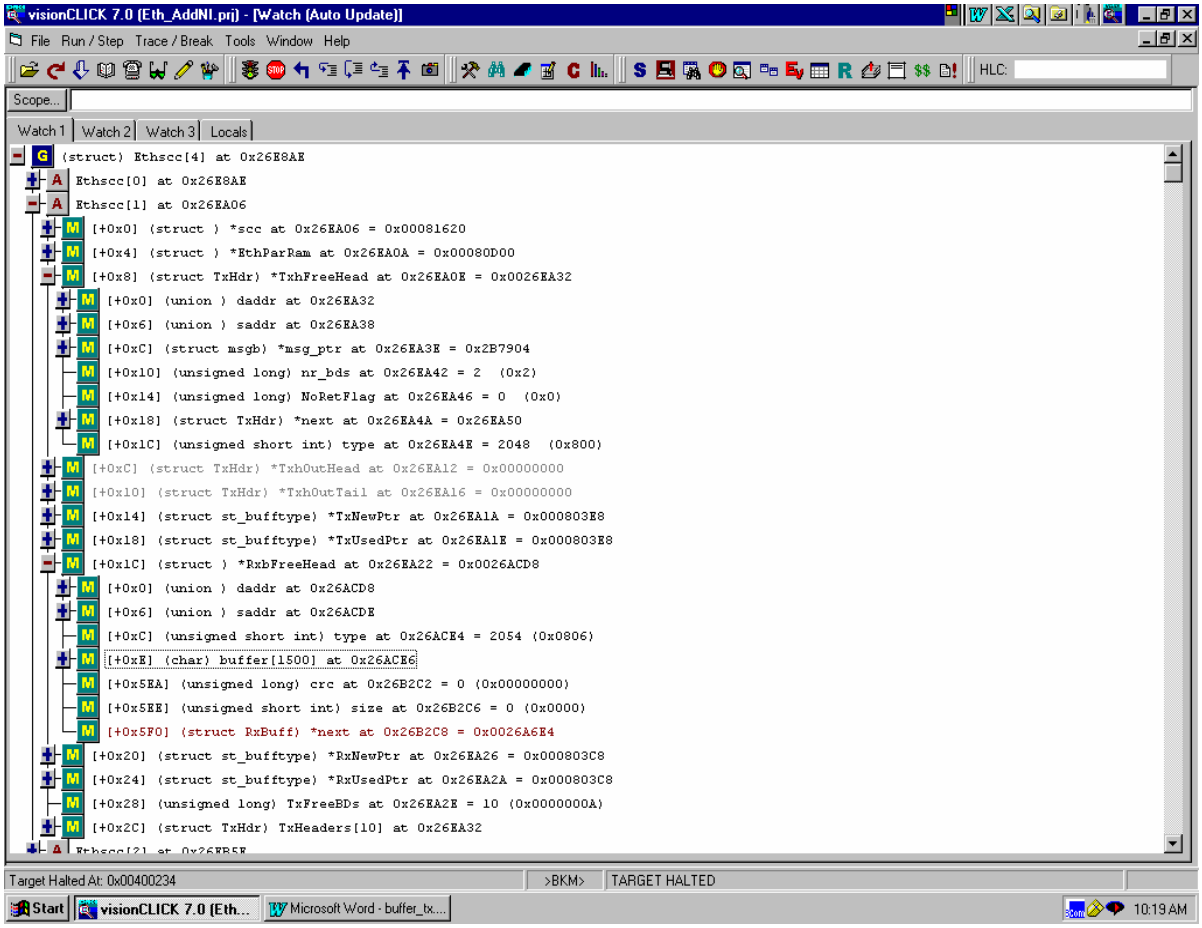


Fig. 5.5.1.2. Strutture dei buffer in ricezione e trasmissione

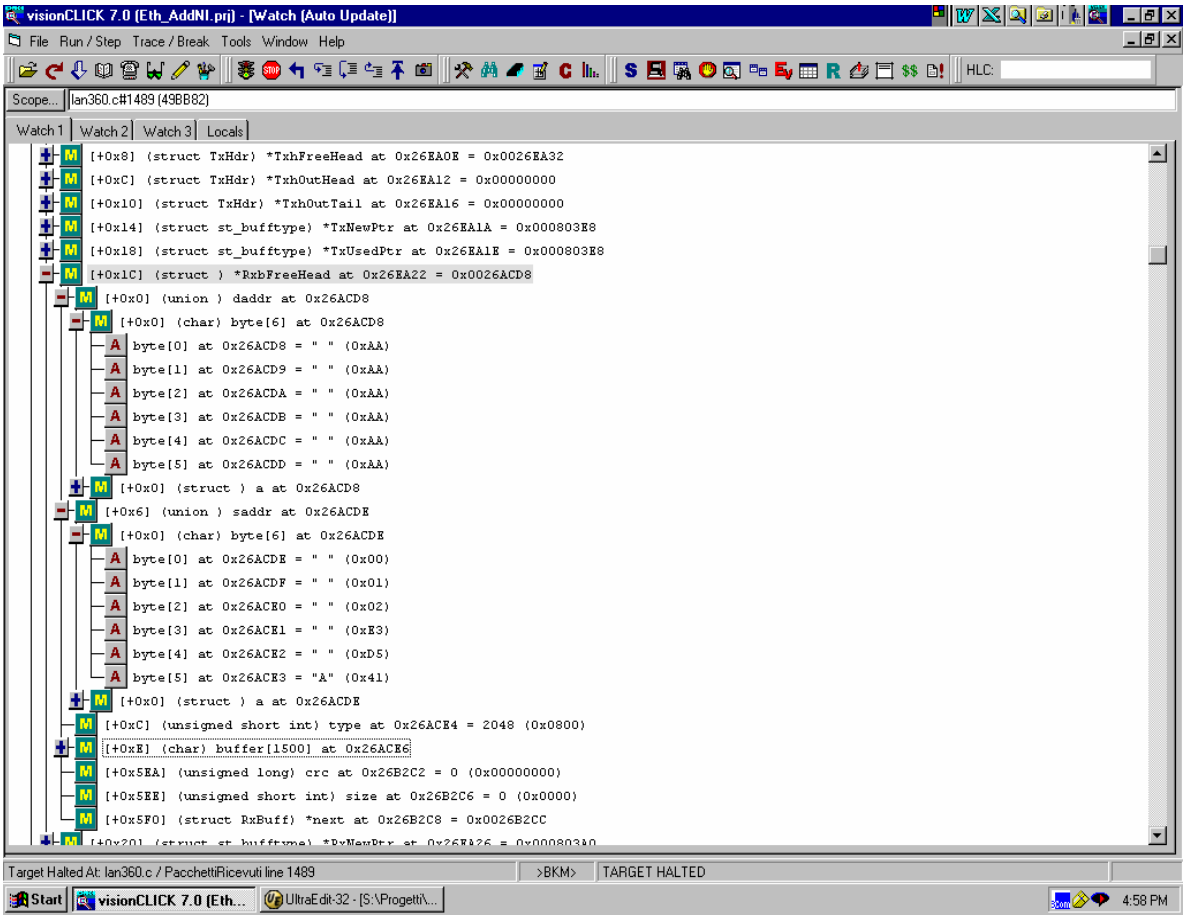


Fig. 5.5.1.3 Struttura e campi del buffer in ricezione della Radio in risposta al Ping

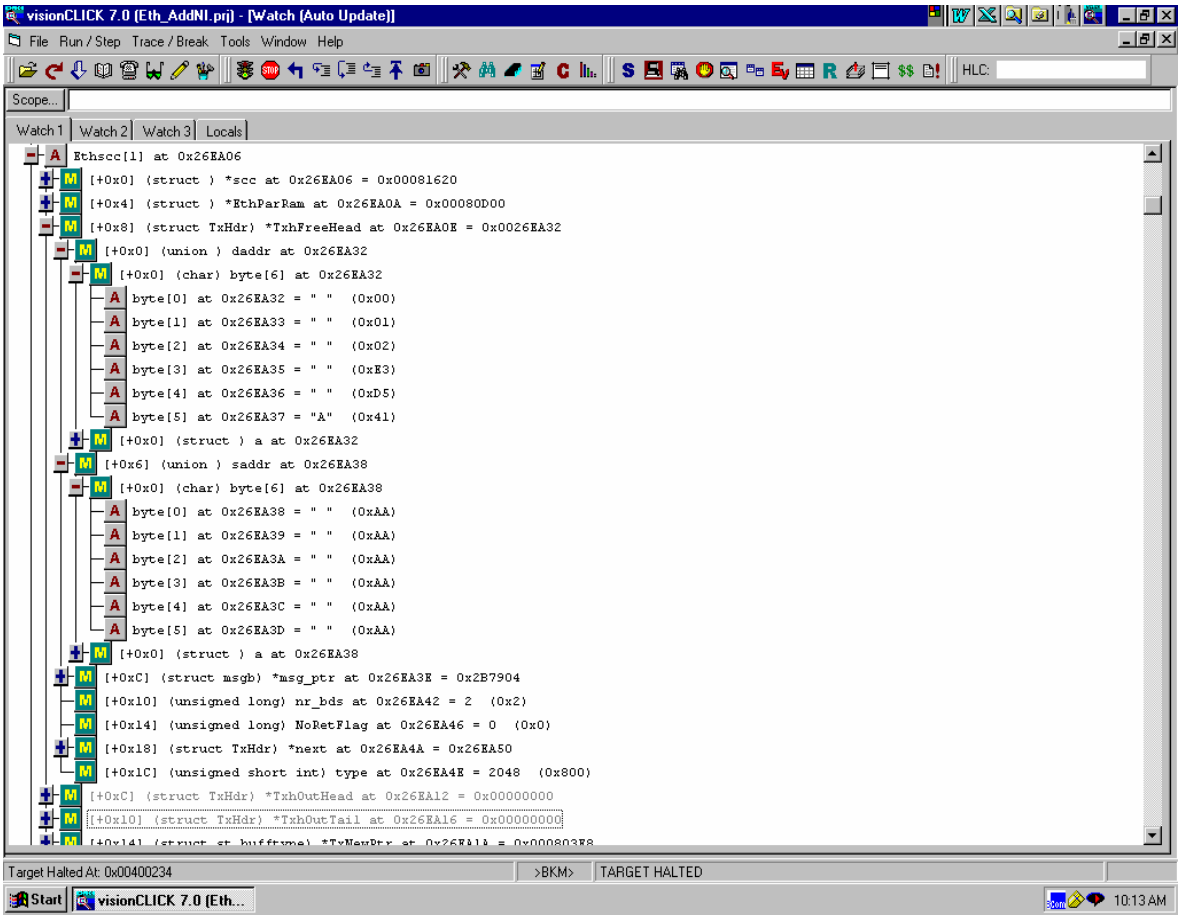


Fig. 5.5.1.4 Struttura e campi del buffer in trasmissione della Radio per l'Echo_Reply

5.6 Conclusioni

Lo studio effettuato, con lo scopo di realizzare un'interfaccia di rete su una Stazione Radio ricetrasmittente HF, ha portato alla validazione della configurazione hardware progettata per l'installazione del driver, comprensiva di un microprocessore Motorola con caratteristiche per reti Ethernet per elaborare a livello fisico i dati ricevuti e di un Transceiver Intel per la codifica e decodifica del segnale.

E' stato poi realizzato il relativo software, aggiunto al sistema operativo, e verificato prima il suo funzionamento all'interno del software già esistente.

Questo è stato eseguito, progettando un task Client e un task Server che, dialogando tra loro in un loopback interno, senza uscire in rete, hanno verificato il funzionamento delle chiamate di sistema (system calls) del componente software aggiunto (pNA+).

Mentre per la realizzazione dell'accesso in rete, è stato implementato il programma Ping, che ha permesso all'apparato di "farsi vedere" in rete dagli altri terminali collegati, ricevendo e trasmettendo pacchetti ICMP e ARP.

BIBLIOGRAFIA

[1] Integrated Systems, Inc., "*pSOSytem System Calls*", Settembre 1996.

[2] Integrated Systems, Inc., "*pSOSytem System Concept*", Settembre 1996.

[3] Integrated Systems, Inc., "*pSOSytem Getting Started: 68K Processors MRI Release*", Marzo 1996.

[4] Integrated Systems, Inc., "*pSOSytem Programmer's Reference: 68K Processors MRI Release*", Marzo 1996.

[5] Integrated Systems, Inc., "*pROBE+ User's Guide: 68K Processors MRI Release"*, Marzo 1996.

[6] Motorola Microprocessors and Memories Technologies Group "*MC68360 Quad Integrated Communications Controller User's Manual*".

[7] S. Gai, P.L. Montessoro, P. Nicoletti, "*Reti Locali: dal cablaggio all'internetworking*", Scuola Superiore G. Reiss Romoli, 1995.

[8] A. S. Tanenbaum, "*Computer Networks*", Second Edition, Prentice-Hall 1996.

[9] Intel, "*LXT908 Datasheet: Universal 3.3V 10BASET and AUI Transceiver*", Giugno 2001.

[10] J. Postel, "*RFC 792: Internet Control Message Protocol*", 09/01/1981.

[11] J. Postel, "*RFC 768: User Datagram Protocol*", 08/28/1980.

[12] J. Postel, "*RFC 792: Internet Protocol*", 09/01/1981.

[13] J. Postel, "*RFC 821: Simple Mail Transfer Protocol*", 08/01/1982.

[14] D. Crocker, "*RFC 822: Standard for the format of ARPA Internet text message"*, 08/13/1982.

[15] D. Plummer, "*RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol address to 48 bit Ethernet address for trasmission on Ethernet hardware"*, 11/01/1982.

[16] J. Mogul, J. Postel, "*RFC 950: Internet Standard subnetting procedure*", 08/01/1985.