di Andrea de Prisco

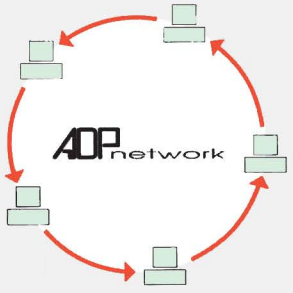
Facciamo un esempio: ADPnetwork!

Come esempio un tantino più «concreto» delle potenzialità offerte dalla programmazione multitask e, in particolare, dal linguaggio OCCAM dei transputer, questo mese rispolvereremo la rete ADPnetwork esposta lo scorso anno allo show-off dell'Amiga DevCon di Parigi e pochi mesi dopo presso lo stand di MCmicrocomputer allo SMAU.

Il progetto ADPnetwork, nato un po' per gioco un po' per scommessa, è partito in realtà nell'agosto 1989 con scopo essenzialmente didattico. Alla effettiva realizzazione avrebbero fatto seguito, come poi è effettivamente successo, una nutrita serie di articoli tecnici pubblicati sulla nostra rivista. Ma prima di immergerci nei processi e nelle varie funzionalità del progetto, nel sacrosanto motto di dare a Cesare quel che è di Cesare, vorremmo ricordare autori e ruoli dei progettisti hardware e software della rete. Capoprogetto, si sa, un certo ADP (uno... "a caso") ideatore del software di rete, dell'architettura "Token Free", oltre ad aver collaborato attivamente alla progettazione, ingegnerizzazione e realizzazione della scheda hardware basata su Transputer. L'opera, tutt'altro che secondaria, di rendere AmigaDOS compatibile l'intera rete è stata svolta dagli ottimi Marco Ciuchini e Andrea Suatoni che hanno scritto l'handler di rete grazie al quale era possibile vedere come un volume unico l'insieme di tutte le macchine in rete e come sottodirectory di queste le varie unità disponibili, stampanti comprese.

Nella fase finale del progetto, quando cioè occorreva aggiungere l'hardware per ottenere prestazioni altamente professionali, si sono aggiunti al team Giuseppe Cardinale Ciccotti e Luciano Macera con il preciso compito di progettare e realizzare una scheda di rete intelligente basata, come detto, su un Transputer.

Ma procediamo con ordine...



- Condivisione globale delle Risorse
- Tolleranza ai guasti e alla caduta momentanea del supporto fisico
- Architettura circolare
- Assenza di token
- Espansione trasparente
- Connessione punto a punto
- Accesso immediato alla rete

Figura 1

ADPnetwork: filosofia di progetto

La caratteristica principale dell'architettura ADPnetwork è la completa uniformità dell'insieme delle macchine che fanno capo a tale struttura. Ogni macchina può utilizzare qualsiasi risorsa disponibile (nel senso di «resa disponibile») sull'intera rete, comunicare con qualsiasi processo in esecuzione su qualsivoglia macchina, nonché sfruttare anche potenza di calcolo remota per bilanciare il carico di elaborazione su tutte le macchine disponibili.

L'architettura, come visibile in figura 1, è rigidamente circolare. Ogni nodo è collegato «punto a punto» con un nodo precedente e un nodo successivo. Dal primo riceverà l'intero flusso di informazioni, sul secondo inoltrerà i dati per altre macchine e reinoltrerà dati in transito per quella macchina.

Non esiste alcun token in circolo (da qui il nome di progetto «Token Free») e ogni macchina può accedere alla rete immediatamente, fatta salva l'ipotesi di

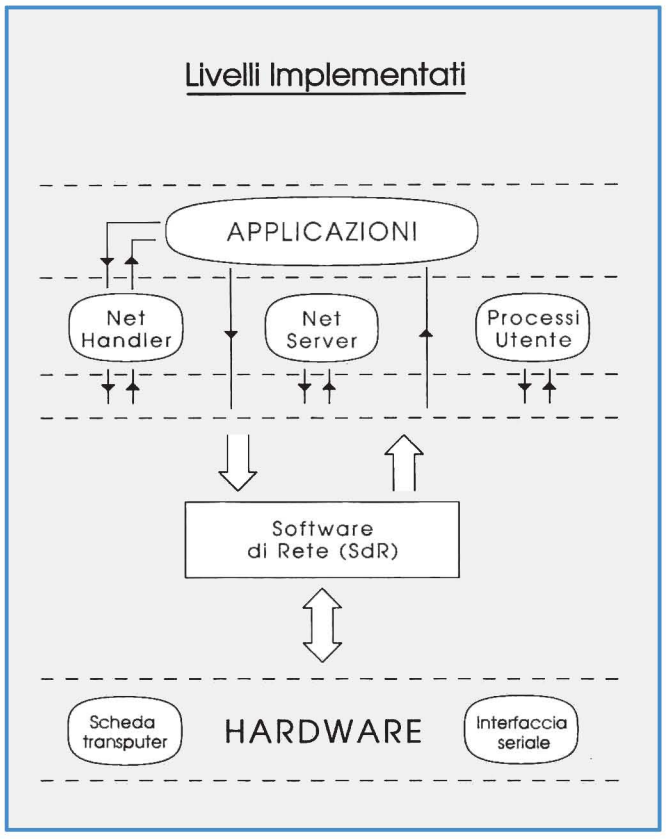


Figura 2

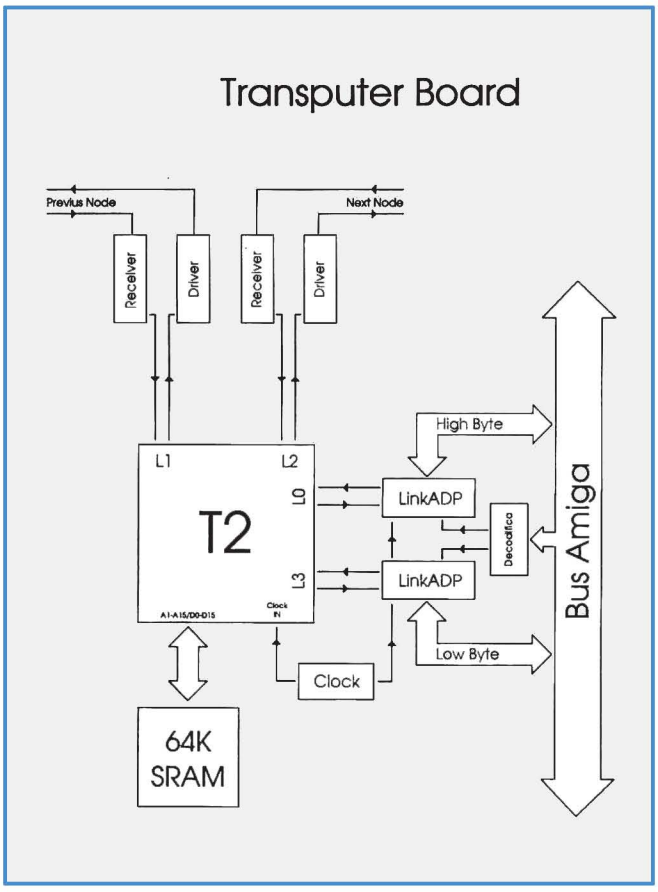


Figura 4

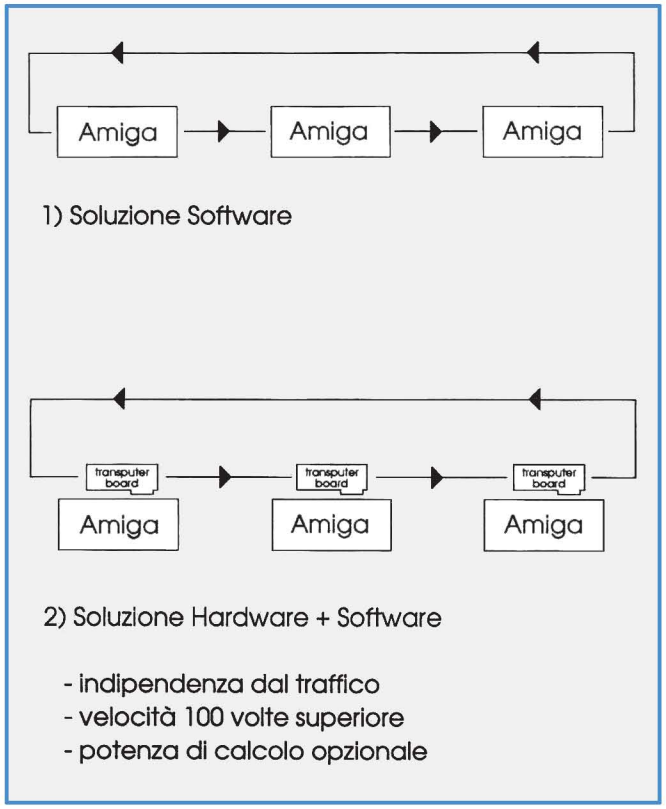


Figura 3

avere, in quell'istante, tutti buffer pieni (carico elevato).

Nonostante il fatto che l'anello (formato dalle 'n' connessioni punto a punto fra le 'n' macchine) sia unico è così possibile che nel medesimo istante fisico più macchine scambino dati tra loro.

Il software implementa, inoltre, vari livelli di tolleranza ai guasti, agli errori di trasmissione e alla caduta momentanea del supporto fisico. Grazie a questo ogni utente può entrare in rete in qualsiasi momento così come è possibile aggiungere o togliere nodi senza forzare sospensioni dell'attività di rete, al più provocando un leggero rallentamento durante l'operazione. I dati eventualmente persi saranno rispediti dai relativi mittenti che effettuano sempre e comunque un preciso controllo tra quanto spedito e quanto effettivamente arrivato a destinazione.

Come visibile in figura 2, ADPnetwork è implementata su vari livelli. Tutte le applicazioni in esecuzione sulle macchine possono accedere a tutti i file e a tutte le risorse disponibili sulle altre macchine. Analogamente i processi possono scambiare messaggi anche se

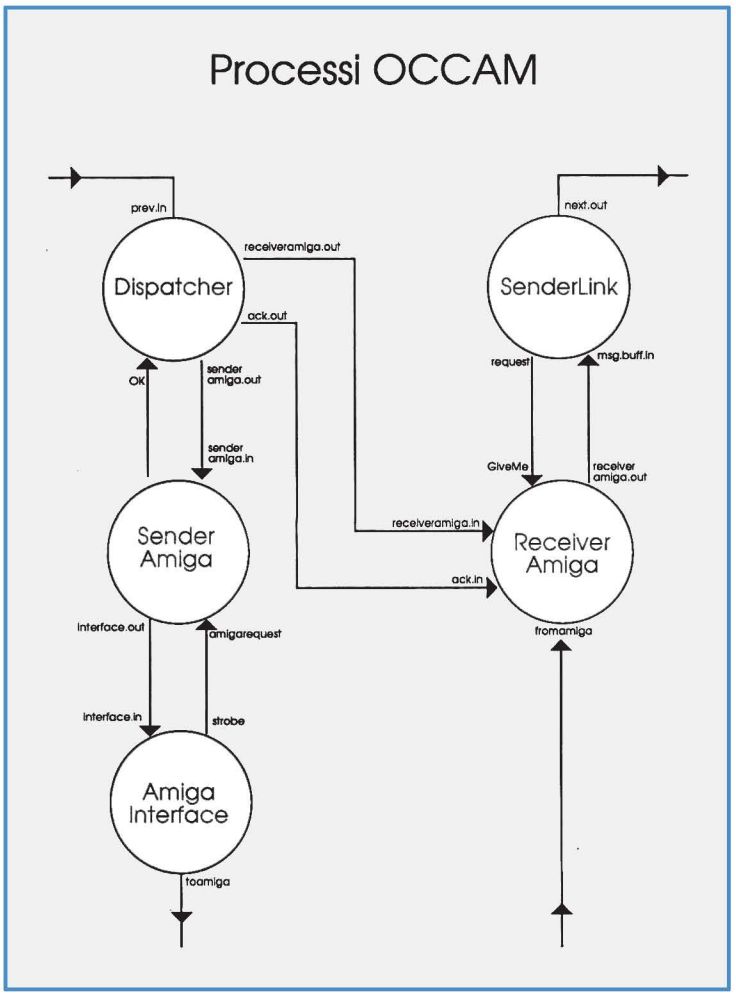


Figura 5

in esecuzione su macchine differenti.

Il Software di Rete si occupa di inoltrare e smistare dati in partenza, in arrivo e in transito, effettuando su questi l'impacchettamento, il controllo CRC, e generando i pacchetti di ACK per tutti i frame di rete giunti integri a destinazione. Per tutti i frame di rete di cui non si riceve l'ACK in un tempo «ragionevole» viene rieseguita una spedizione fino a completamento degli ACK mancanti.

La versione esposta a Parigi nel febbraio '90 nel corso della Amiga DevCon non disponeva ancora di hardware dedicato e realizzava il collegamento delle macchine attraverso l'interfaccia seriale standard RS-232. ADPnetwork «fase 2», esposta a SMAU '90, poteva invece contare sul nostro hardware dedicato basato su transputer.

Dal software all'hardware

Nessuna funzionalità è stata aggiunta alla versione hardware di ADPnetwork. Le migliorie riguardano solo le performance globali della rete, in termini di velocità di trasferimento tra nodi. Oltre a questo una scheda di rete intelligente ha potuto importare parte del software di rete in modo tale da non impegnare la potenza di calcolo delle macchine per tutti i messaggi in transito. In pratica, vedasi figura 3, la fase di riconoscimento ed eventuale smistamento dei pacchetti è demandata alla scheda di rete in modo che le macchine collegate siano coinvolte direttamente solo riguardo ai messaggi in partenza da quel nodo (impacchettamento) o in arrivo su quel nodo (ricomposizione). Ciò che viene trasferito da una macchina alla corrispondente scheda attraverso il bus di sistema sono solo pacchetti che hanno nel campo mittente o nel campo destinatario l'indirizzo fisico del nodo in questione.

Grazie a tutte le potenzialità offerte dal transputer la scheda di rete di ADPnetwork è «spaventosamente» semplice nonostante la sua intrinseca «intelligenza». In figura 4 è mostrato lo schema a blocchi dell'hardware: oltre al transputer T222 e ai suoi 64K ram sulla scheda sono presenti solo una coppia di integrati per la trasmissione differenziale tra nodi e due link adaptor verso il bus dell'Amiga. In questo modo il trasferimento dei dati tra Amiga e scheda avviene 16 bit la volta utilizzando in parallelo due dei quattro link fisici del transputer.

Non è presente nemmeno una rom, dal momento che il software di gestione della scheda viene caricato dall'Amiga stesso nel momento in cui si decide di entrare in rete.

Il software

In figura 5 sono mostrati i cinque processi OCCAM lanciati sulla scheda transputer che realizzano il riconoscimento, lo smistamento e la spedizione dei pacchetti in arrivo, in transito e in partenza. L'eseguibile di tutti i processi OCCAM non occupa più di 5 Kbyte, dunque i rimanenti 59K ram disponibili (un po' di meno per la verità) sono interamente occupati da buffer per i pacchetti, suddivisi anche per dimensione degli stessi. Dato che ad ogni pacchetto inviato, a destinazione avvenuta, viene generato un pacchetto di ACK corrispondente, si è scelto di dare massima priorità a quest'ultimi in modo da favorire il più possibile il completamento di ogni operazione nel più breve tempo possibile. Anche per i buffer si è lavorato nella medesima ottica: per i pacchetti di ACK sono stati predisposti buffer «a parte» con capacità ben superiori e ad accesso prioritario. Su ogni nodo i pacchetti in arrivo o in transito giungono sul canale «prev.in» e i pacchetti in partenza partono o ripartono dal canale «next.out». Il primo processo Dispatcher esegue prima un semplice CheckSum sull'header del pacchetto, poi il CRC sul suo corpo e solo dopo procede all'analisi vera e propria del contenuto. Se quel pacchetto è destinato al nodo in questione lo «passa» al processo SenderAmiga che è un primo processo buffer. In questo caso, se l'inserimento ha successo (buffer non pieno) il processo Dispatcher genera l'ACK e lo inoltra sulla rete verso il mittente del messaggio appena ricevuto.

Diversamente, se il pacchetto non è per il nodo in questione (sia esso un pacchetto valido o un ACK per un'altra macchina) è reinoltrato attraverso il secondo processo buffer ReceiverAmiga al quale confluiscono anche i pacchetti in partenza da quel nodo.

In che modo una macchina si accorge che un determinato partner non esiste (più)?

Semplicemente perché poco dopo vede arrivare al suo Dispatcher un pacchetto in cui si riconosce come mittente: il fatto stesso che sia ritornato vuol dire che tutte le altre macchine l'hanno rispedito, ovvero che non esiste (più) una macchina con l'identificatore specificato.

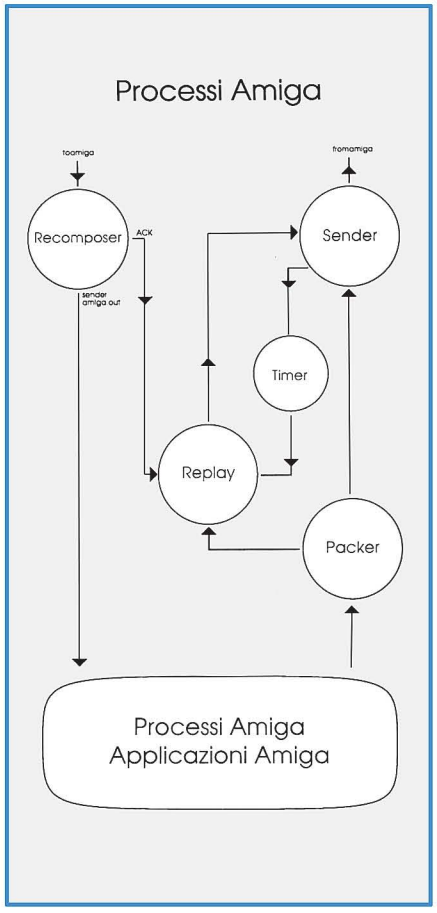


Figura 6

In figura 6 sono mostrati i processi in esecuzione sull'Amiga durante il funzionamento di rete. Le due frecce in alto, una entrante e l'altra uscente, indicano l'interfacciamento con la scheda di rete basata sul transputer.

Qui «ragioniamo» in termini di messaggi da spedire e di lunghezza arbitraria. Ogni processo Amiga che intende comunicare con qualsiasi altro processo remoto, non deve far altro che inviare al processo Packer il messaggio da spedire, la sua lunghezza e il destinatario dello stesso.

Questo primo processo, come dice «scherzosamente» il suo nome, esegue l'impacchettamento del messaggio e spedisce i vari pacchetti via via costruiti al processo Sender che si interfaccia con la scheda.

Su Amiga non è stato necessario implementare alcun processo buffer dal momento che le comunicazioni inter process possono avvenire anche in modo asincrono e quindi il buffer è automaticamente implementato dalla stessa coda messaggi che gestisce il kernel di Amiga su ogni porta di comunicazione creata.

Appena Packer ha terminato di impacchettare il suo messaggio manda al processo Replay una copia di questo che la conserverà fino a quando tutti i pacchetti non saranno giunti a destinazione. Parallelamente appena Sender ha inoltrato alla scheda transputer tutti i pacchetti del messaggio 'n' «carica» il processo Timer in modo da dare un messaggio di sveglia dopo un certo numero di secondi (tipicamente 2) al processo Replay. Altrettanto parallelamento (ricordiamo che Amiga è una VERA macchina multitasking) Replay riceve man mano anche i pacchetti di ACK che ritornano dalla rete. Allo scadere del Timer, Replay non fa altro che controllare quanti ACK ha ricevuto per quel messaggio ed, eventualmente, ritrasmette i pacchetti di cui non ha ricevuto l'ACK.

Spostiamoci ora sulla macchina destinataria. Lì, come detto, il processo OCCAM Dispatcher della scheda transputer genera l'ACK per ogni pacchetto arrivato integro e manda questo stesso verso il processo Amiga Recomposer.

Quest'ultimo processo ha un comportamento complementare al processo Packer. Man mano che arrivano pacchetti, questo ricompone il messaggio originario estraendo da ogni pacchetto la parte significativa e collocandola nella giusta posizione nel messaggio in via di ricostruzione. Non ha importanza l'ordine d'arrivo dal momento che in ogni pacchetto è indicato quale porzione del messaggio originario è contenuta nel vero e proprio corpo. Appena Recomposer si accorge di aver ricostruito interamente un messaggio lo inoltra al processo opportuno in esecuzione in quel momento su quella macchina.

```
Dispatcher ::

  SEQ
    RiceviPacchetto(prev.in,msg)
    IF
      (msg(dest) = MyName)
        SEQ
          SenderAmiga.IN ! msg
          OK ? TrueFalse
          IF
            (TrueFalse)
              SendAck(msg)
            TRUE
              SKIP
      TRUE
        Forward(msg)
```

Figura 7

Tanto il processo Recomposer quanto il processo Replay sono intrinsecamente paralleli potendo gestire contemporaneamente decine di messaggi in arrivo o in partenza. Ovvero fotografando idealmente lo stato di questi due processi in un qualsiasi istante potremmo ad esempio trovare che nel processo Recomposer vi sono 8 messaggi in via di ricostruzione (come tanti binari di una stazione in cui i vagoni appartenenti a più treni arrivano casualmente ed è necessario ricostruire i convogli originari prima di segnalare ai passeggeri l'effettivo arrivo) e provenienti da nodi diversi e 12 messaggi in spedizione (per altrettanti nodi, eventualmente) di cui non sono ancora arrivati tutti gli ACK.

Terminiamo qui

Ovviamente, per ragioni di spazio, non possiamo certo esaurire in una sola volta l'argomento ADPnetwork. Questo mese abbiamo descritto il funzionamento per così dire generale e mostrato a grandi linee i processi OCCAM e Amiga coinvolti nel «networking».

Il mese prossimo analizzeremo un po' più nei dettagli alcuni processi per mostrarvi dal di dentro qualcosa in più. Per non lasciarvi, però, completamente a bocca asciutta, in figura 7 vi anticipiamo, schematicamente, il funzionamento in OCCAM del processo Dispatcher in esecuzione sulla scheda transputer.

Dopo aver ricevuto, all'interno della funzione RiceviPacchetto (non mostrata), dal canale «prev.in» il pacchetto in arrivo e aver eseguito su questo checksum dell'header e CRC del corpo, Dispatcher controlla se si tratta di un pacchetto per quel nodo o per un altro nodo. Semplicemente confronta se il campo destinatario dell'header è uguale al proprio indirizzo di rete (MyName). In caso affermativo manda il pacchetto al processo SenderAmiga e attende da questo l'OK di avvenuto inserimento nella sua FIFO. Se tale OK è affermativo genera l'ACK e lo inoltra sulla rete, se è negativo (nel caso di buffer pieno) l'ACK non è generato. In questo caso il mittente non ricevendo l'ACK dopo l'intervallo di tempo prefissato dal processo Timer rispedirà il pacchetto ritentando la comunicazione.

Diversamente, ovvero se il campo destinatario non è uguale al proprio indirizzo di rete, Dispatcher reinoltra il pacchetto nuovamente verso la rete.

Per questo mese, come detto, è tutto: appuntamento dunque a dicembre per un'analisi un tantino più profonda dei processi di rete di ADPnetwork.

MC