Network Adaptors - Schede di Rete

Il Computer è composto da CPU, Memoria Interna (RAM) ed una Scheda di Rete.
Scheda di Rete è un sistema microprocessore che al suo interno ha una sua memoria e processore a sua volta, la differenza sta che il processore esegue un determinato algoritmo che si basa sul ricevere pacchetti dal sistema operativo del pc e trasmetterli tramite la tecnologia di connessione per la quale è stato progettato (cablata o non).

Come comunicano CPU e Scheda di Rete? e Come scheda di Rete trasmette i messaggi?

Scheda di Rete: è quindi composto da

- microprocessore SCO - che esegue un microcodice specializzato nella gestione dei pacchetti, organizza i frame ethernet da mandare, riceve pacchetti e interagisce con la memoria.
- al fine di eseguire questo codice bisogna che ci sia un'interfaccia per il Link e un'interfaccia al Bus del PC per poter comunicare sia con la linea (con il link) e sia con il PC (con il BUS).
- I messaggi (pacchetti) che passano tramite la scheda di rete potrebbero aver bisogno di buffer e quindi di una memoria per fare da tampone tra le diverse velocità della linea e del computer.

La CPU solitamente organizza le attività del calcolatore tramite un clock interno controllando il bus di memoria e il bus I/O.
Il SCO della scheda di rete per comunicare con la CPU ha bisogno di un protocollo che gestisca la comunicazione come ad esempio l'accesso alla memoria.

Dentro la CPU c'è un registro di stato dove vengono tenute una serie di informazioni.
Anche la SCO ha uno status register che permette a CPU e SCO di dialogare tra loro, perchè attraverso le modifiche dei bit dello status register i due comunicano. I bit possono essere letti e scritti da entrambe, CPU e SCO.
Es. CPU scrive, SCO legge e attiva meccanismo per avviare la richiesta.
SCO scrive, CPU legge e attiva meccanismo ...
Ogni bit ha uno specifico significato che servono quindi a sincronizzare i

due processori.

Lato HOST.
Per controllare cosa accade in CSR(Control Status Register) esistono due modi:

1. Busy Waiting - continua a testare CSR finchè non si rilevano modifche, ragionevole solo per CPU che non devono fare altre come ad esempio CPU di un Router. SCO può lavorare con Busy Waiting
2. Interrupt - colui che esegue modifica del control status register invia interrupt al destinatario. Il destinatario serve l'interrupt, legge CSR e capisce l'operazione da eseguire. Usato in tutte le comunicazioni classiche. CPU lavora cosi.

Trasferimento Dati tra Adaptor e Memoria.
I pacchetti che arrivano e che devono partire possono essere inviati tramite due modi:

1. Direct Memory Access - la cpu non viene coinvolta e si mette in alta impedenza, si utilizza un dispositivo del microprocessore dell'adapter per eseguire il trasferimento da adapter a memoria che viene chiamato DMA.
2. Programmed I/O - quando l'adapter avvisa di avere frame pronti, la cpu fa partire una routine che trasferisce i dati da adapter a memoria e quindi socket.

Questi metodi dipendono dalle architetture delle macchine.

Buffer Descriptor List.
La memoria dove allocare i frame p organizzata tramite buffer descriptor list ovvero un vettore di puntatori ad aree di memoria che descrivono la quantità di memoria che puntano.
Si tratta di una zona memoria condivisa dai vari processi che accedono per inserire frame da inviare e viene quindi gestita da un semaforo. Il numero massimo di processi che possono accedervi è 64.

Viaggio del messaggio in SCO.
Quando il messaggio è pronto all'interno della memoria del computer, la CPU esegue una modifica del CSR dei bit LE_TDMD e LE_INEA, di modo che la scheda di rete capisce che c'è un messaggio da inviare.
la SCO invia il messaggio sulla linea, allega header e footer e comincia ad inviare i bit fuori secondo lo standard di riferimento.
SCO modifica LE_TINT per avvisare la conclusa trasmissione e scatena

un interrupt per avvisare la CPU.
La cpu resetta LE_TINT e LE_INEA.

Device Driver di qualunque scheda è una collezione di procedure (di routine) che permettono di ancorare il sistema operativo all'hardware che altrimenti sarebbero indipendenti e scollegati.

La CPU per leggere il CSR pensa di leggere una locazione di memoria nella quale però in realtà c'e' il CSR.
Questo dettaglio viene definito dai device driver per decidere in che zona andrà messa quest'informazione.

Ogni volta che c'e' o una chiamata a socket (un processo vuole inviare informazioni) o informazioni ricevute dall'esterno allora parte un thread.
Ed essendo che non ci possono essere più di 64 di queste operazioni all'interno del Buffer Descriptor List allora c'e' un semaforo che gestisce questo limite e questi accessi.

MiddleWare

Ammettiamo di avere due sistemi operativi SO1 e SO2 magari diversi.
E' possibile scrivere un'applicazione che sia al di sopra dei sistemi e possa sfruttrare le risorse di entrambe?

Se guardiamo l'interfaccia dei sistemi operativi ed essi son diversi allora saranno diversi anche nell'application layer.
Si è proposto quindi di mettere uno strato ulteriore per fare da tramite tra l'applicazione ed il sistema operativo al fine di mascherare l'etereogenità dei sistemi di modo che per l'applicazione sia sempre uguale comunicare.
L'applicazione comunica quindi con un solo strato software = che si chiama MIDDLEWARE.

Da un punto di vista astratto, la divisione tra MiddleWare e Sistema Operativo in realtà non è fissa : il SO è solo una collezione di routine composta da diversi device driver dei quali ognuno gestisce un pezzo di hardware.
Il SO nel tempo ha dovuto introdurre nuove tecnologie che prima non esistevano e quindi anche device driver per gestirle (es. TCP, IP oggi sono

integrate ma negli anni 80’ c’erano diversi sistemi di rete e quindi il SO non si occupava di essi ed era l’utente che decideva a che pila affidarsi per inviare messaggi. Quando TCP IP ha sbaragliato gli avversarsi allora tutti gli SO hanno integrato le procedure per costruire pacchetti in TCP IP.)
Il sistema operativo è quindi in continua evoluzione.
I primi esempi di MiddleWare sono i remote control *** - che erano al di fuori degli SO mentre oggi son dentro.
Se quindi ci son delle routine utilizzate da tante applicazioni allora il SO ingloba il meccanismo “standardizzato".

MiddleWare nasce quindi per unire sistemi operativi eterogenei e per sviluppare applicazioni al di sopra e separate da queste piattaforme.

Client /Server Computing
Ci sono tanti client che chiedono informazioni ad un server che hanno una serie di servizi condivisi dai clienti.
I clienti son macchine leggere dal punto di capacità computazionale.
L’esempio più classico per questo tipo di comunicazione sono i DataBase - c'è infatti sempre necessità di condividere basi di dati.

Un’applicazione si suddivide in tre livelli applicativi - questi livelli possono essere o tutti sul server o sul client e quindi rendere uno o l’altro più pesanti. E’ chiaro che dove le risorse sono poche è bene spingere buona parte dell’applicazione tutta sul server. In Altri casi invece in cui si opera tra vari server, dove magari uno si comporta da client, i livelli vengono distribuiti tra i vari server.
Dove si trovano questi livelli dipende dal tipo di terminali che son coinvolti nell’interazione.

Quando un client fa una richiesta ad un server, esegue questa richiesta al di sopra del SO.
Ma poi le informazioni che sono invece raccolte nella memoria son rappresentate in un certo modo che dipende dall’architettura del computer. E quindi la rappresentazione di un'info all’interno di un PC diventa importante per l’interazione. Nonostante l’applicazione sia superiore.

Esisterà una logica del software nel client e nel server la quale sarà più o meno spostata da una parte o dall’altra a seconda dell’architettura e della richiesta:

Host Based processing- quel tipo di interazione in cui abbiamo il thin client ovvero non fa assolutamente nulla se non mostrare le informazioni. Tutta la logica è nel server il quale restituisce anche le informazioni da mostrare. (es. RDA Remote Desktop Access)

*I mainframe un tempo erano buone soluzioni, son stati sostituiti da database e oggi da cloud.

Server Based processing - man mano il client diventa più intelligente e pesante, la prima cosa che si trasferisce è il Presentation Logic ovvero come rappresentare i dati, mostrare la pagina in un certo modo. In quel caso deve avere potenza computazionale e grafica per farlo.

Client Based processing - il client diventa più pesante, dove abbiamo Presentation Logic e Application Logic , il quale è uno strato che è in entrambe i terminali sotto ai quali c'è il sistema operativo. Le due Application Logic devono comunicare tra loro e il modello di comunicazione dovrà essere decisamente più ricco e variegato rispetto a TCP /IP.
Ad esempio quando entri su twitter - vuoi cercare un hashtag ed escono le informazioni relative all'hashtag. Si tratta di una richiesta ad un server. La comunicazione per inviare la richiesta e ricevere i risultati in tempi rapidi è possibile tramite sistemi di comunicazioni che si chiamano PUBLISH SUBSCRIBE che si trovano sopra a TCP/IP e sono più potenti.

Cooperative processing -man mano che abbiamo client più pesanti possiamo spostare anche strato Database Logic sul client.

Architettura a tre livelli - è un'architettura nella quale abbiamo un client thin, un middle-tier sul quale mi appoggio per avere un servizio che viene dal backend.
Questa è la tipica struttura di client/server di media dimensione.
Noi client ci connettiamo ad un gateway
Il middle-tier se ci sono tante richieste dal gateway splitta le richieste + attiva politiche di sicurezza per proteggere il backend da accessi pericolosi.
E ci fa connettere al backend - dove ci sono i dati.

MiddleWare Servizi:

- Directory Service
- Security Service
- Time Service
- Altri servizi minori orientati alle applicazioni,

Un sistema di **Publish Subrscribe** è un sistema con
storage -
name server - come faccio a raggiungere informazione con quel nome specifico