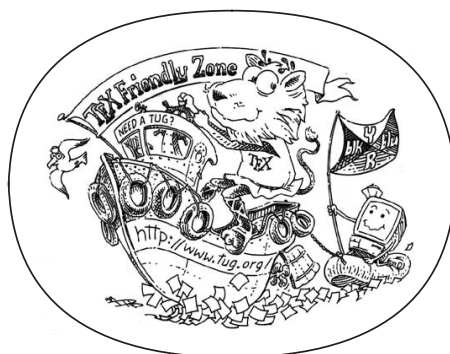


SISTEMI E TRASMISSIONE DELLE INFORMAZIONI

RICCARDO CEREGHINO



Appunti

June 2018 – aclassicthesis v4.6

INDICE

I INTRODUZIONE

0.1	Accenni al funzionamento di un calcolatore	3
0.1.1	Registri	3
0.2	Networking	3
0.2.1	TCP/UDP	3
0.2.2	Invio	3
0.2.3	Ricezione	3
0.3	trasmissione di datu su bus ring based	4
0.3.1	Ring	4
0.3.2	Commutazione	4
0.3.3	Routing	5

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

LISTINGS

ACRONYMS

Parte I

INTRODUZIONE

0.1 ACCENNI AL FUNZIONAMENTO DI UN CALCOLATORE

Un calcolatore è generalmente composto da:

- la *CPU*, esegue le istruzioni, tiene in memoria (*InstructionRegister*) l'istruzione corrente e la posizione dell'istruzione nella RAM (*P.C.*);
- la *RAM*, tiene in memoria le istruzioni da eseguire, certe sezioni sono riservate per specifiche funzioni (*stack*, *SP*, *BP*);
- dei moduli, come un disco fisso o altro;
- la *NIC*, che permette di inviare e ricevere pacchetti in rete.

0.1.1 *Registri*

0.2 NETWORKING

Vengono effettuate delle *syscall*, per eseguire operazioni necessario all'invio di dati. Per esempio la *syscall send*, copia dalla ram una certa sezione di memoria nel *NIC*, che verrà inviata in rete. ???

0.2.1 *TCP/UDP*

Il protocollo *TCP* garantisce l'invio del messaggio, dato che attende una risposta da parte del ricevente.

Il protocollo *UDP* invia datagrammi, ma non si è certi se la ricezione è avvenuta.

0.2.1.1 *Device driver*

E' un modulo che gestisce le richieste fuseche dei vari moduli, tra cui *NIC*, gestiscono le interruzioni del dispositivo.

0.2.2 *Invio*

Il *DMA* presente nel *NIC* è il componente che accede alla memoria.

L'unità di gestione della memoria *MMU*, permette di utilizzare indirizzi virtuali in indirizzi fisici, quindi mentre il processore utilizza indirizzi virtuali, il *DMA* del *NIC* utilizzerà indirizzi fisici.

Interrupt, buffer

0.2.3 *Ricezione*

Per la ricezione di pacchetti devono essere sempre disponibili dei buffer, su cui *NIC* può scrivere una volta ricevuto il pacchetto.

Il pacchetto in arrivo viene processato dal NIC quindi con il DMA inserito nel buffer, quindi un interrupt crea nuovi buffer per possibili nuovi messaggi, quindi il sistema legge il buffer copiandolo nel buffer dell'applicazione, direzionando quindi l'output verso l'utente corretto, quindi il buffer di ricezione può essere riutilizzato per nuovi pacchetti.

Il comportamento standard di un applicazione che implementa networking è un meccanismo bloccante per cui l'applicazione rimane in attesa fintanto che il buffer non è stato scritto al suo interno.

0.3 TRASMISSIONE DI DATI SU BUS RING BASED

La problematica principale del DMA di tipo bus mastering è che deve essere programmato in anticipo, ovvero è limitata da quanti buffer sono stati programmati per essere utilizzati dal DMA ed è dipendente dalla velocità del processore a liberare il buffer, per cui fintanto che il DMA è occupato, verranno persi i dati nel frattempo ricevuti dalla rete.

Quindi si implementa una struttura dati più complicata, integrando una piccola CPU all'interno della NIC (ring based).

0.3.1 *Ring*

Il ring è composto da un anello contenente più di un buffer, e due puntatori, un puntatore di inizio ed uno di fine gestiti da due processori diversi; la CPU si occupa di aggiungere elementi all'interno del ring quindi gestirà l'indice di fine, il puntatore al primo elemento viene gestito dal processore all'interno della NIC. Si ottiene una coda di buffer che si possono aggiungere o togliere a seconda se si sta gestendo un invio o ricezione di dati.

Quindi la NIC legge i buffer in memoria e usa un flag per impostare se il buffer è libero oppure se deve essere processato. Quindi l'unica comunicazione tra processore e DMA è la comunicazione della quantità di buffer che dovranno essere utilizzati al DMA.

0.3.2 *Commutazione*

Si rende necessario, per una questione di efficienza, un circuito fisico *R* attraverso cui trasmettere i pacchetti.

I pacchetti sono inviati come datagrammi, e nella struttura includono il destinatario e le informazioni trasmesse.

0.3.2.1 *Workflow*

La macchina che vuole inviare un datagramma *host* compila e viene mandato dal buffer in locale al buffer di ricezione di *R1*(router), il cui compito è di leggere il datagramma, ricavarne il destinatario, quindi di inviarlo, la macchina può quindi cancellare dal buffer il dato.

La parte ricevente ($R2$) copia il datagramma nel suo buffer di ricezione, quindi $R1$ può rimuovere dal proprio buffer il datagramma. Quindi $R2$ invia il datagramma all'effettivo destinatario con un altro *store and forward*.

Ogni passaggio di trasmissione del datagramma viene chiamato *hop*.

0.3.3 *Routing*

Identifichiamo per il processo di routing lo *host*, gli endpoint della trasmissione di pacchetti e i *router*, calcolatori. Più router possono essere collegati tra loro.