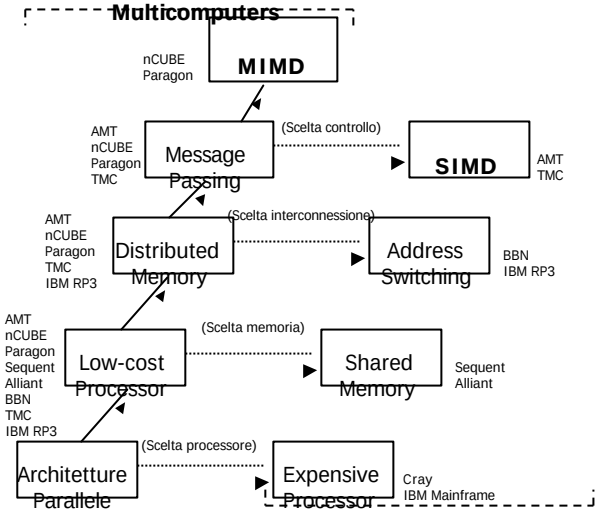
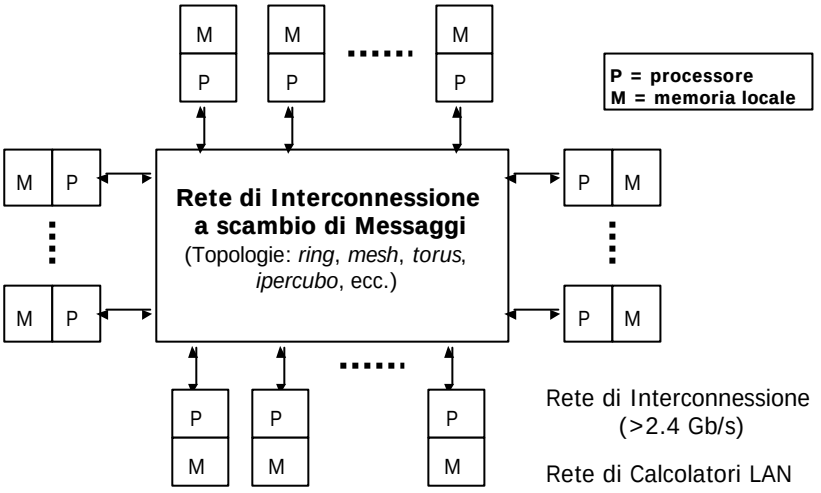
Architetture MIMD a memoria distribuita

(*Multicomputers*)

Scelte per arrivare ai MULTICOMPUTERS



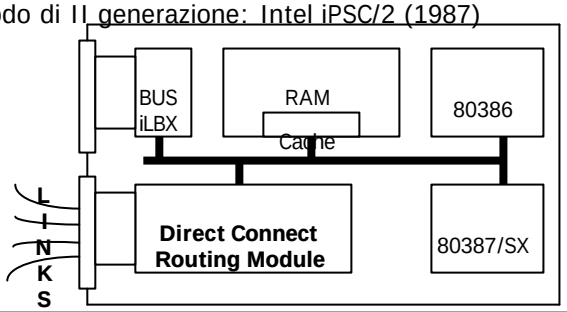
Modello Architetture MIMD *distributed memory*



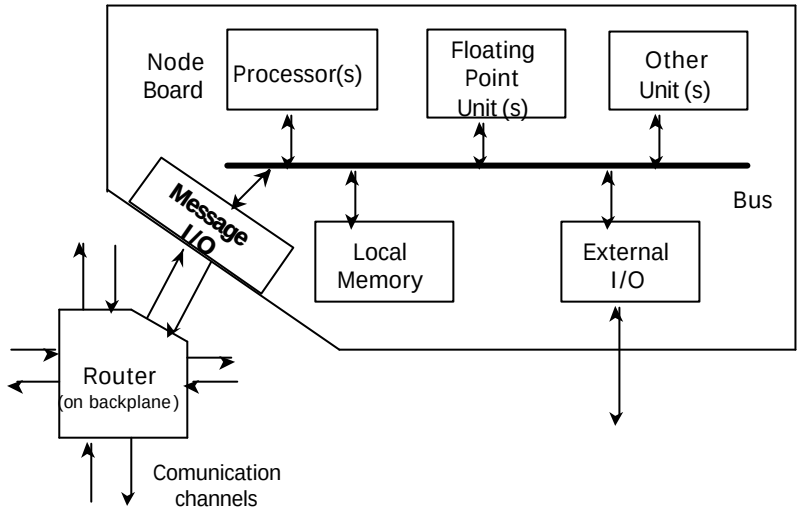
Multicomputer (MIMD a memoria distribuita)

- Composte da un numero variabile di elementi di elaborazione (detti anche *nodi*, in quanto composti da più componenti)
- I nodi sono tutti connessi mediante una rete di interconessione diretta
- Ciascun nodo contiene uno o più processori, una memoria locale, una cache, un canale di comunicazione con altri nodi e, nelle architetture più moderne, un canale di I/O (prevalentemente verso un disco)

Esempio di nodo di II generazione: Intel iPSC/2 (1987)



Esempio di nodo di III generazione Intel Paragon (1992)



Caratteristiche architetture MIMD *distributed memory*

- Potenzialmente molto più scalabili
- Non vi è memoria condivisa tra i diversi nodi
- Ciascun nodo esegue indipendentemente insiemi multipli di istruzioni utilizzando differenti insiemi di dati, memorizzati su spazi differenti
- I nodi scambiano informazioni attraverso la rete di interconnessione utilizzando qualche politica *message-passing*
- Tre sono le reti di interconnessione (commercialmente) più diffuse: **Ipercubi, Mesh, Torus.**

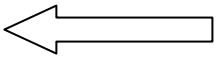
  Ipercubi
  - Intel iPSC/1, **iPSC/2**, iPSC/860
  - Ncube, Ncube-2

  Mesh 2D
  - Intel Delta, Intel Paragon
  - Basati su Transputer: Meiko Computing Surface, Parsytec

  Torus 3D
  - Cray T3D

Reti di Interconnessione Dirette

(elementi di caratterizzazione)

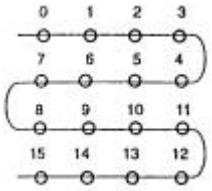
- **Topologia** 
- **Flow Control**
- **Politica di Switching**
- **Politica di Routing**

Topologia

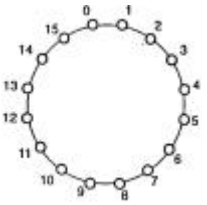
La topologia di une rete di interconnessione diretta, tipicamente modellata come un **grafo regolare**, definisce i canali (*archi del grafo*) mediante cui sono collegati i nodi di elaborazione (*nodi del grafo*).

- Ipercubo
- Mesh (2D – 3D)
- Torus (2D – 3D)

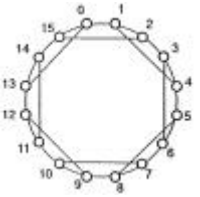
Esempi di topologie di interconnessione



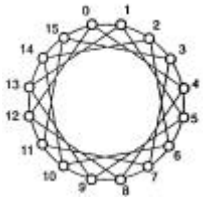
(a) Linear array



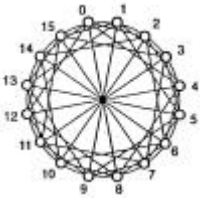
(b) Ring



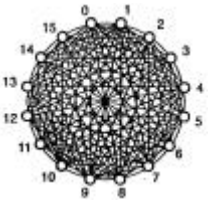
(c) Chordal ring di grado 3



(d) Chordal ring of degree 4 (come Illiac mesh)

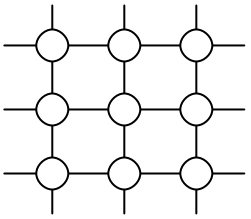


(e) Barrel shifter

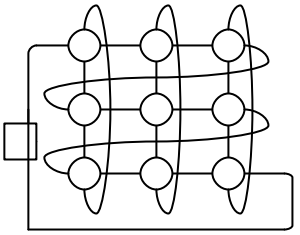


(f) Connessione completa

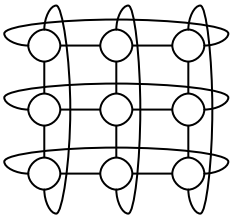
Topologie di interconnessione (bidimensionali)



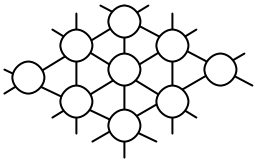
(a) Mesh



(b) Illiac mesh

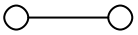


(c) Torus

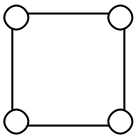


(d) Systolic array

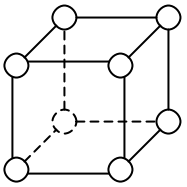
IPERCUBI



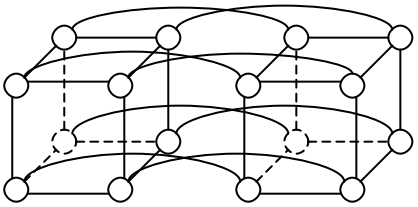
1 – Cube



2 – Cube



3 – Cube



4 – Cube

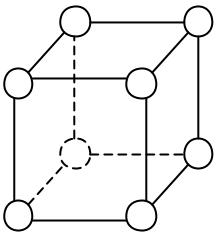
Dimensione ipercubo = d

Numero nodi = $2^d = N$

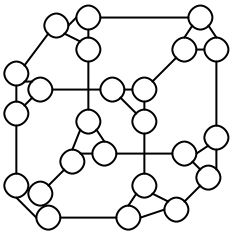
Diametro (*distanza massima*) = $\log_2 N = d$

Numerazione nodi → Codice binario di Gray
(due nodi connessi da un arco differiscono per un solo bit)

Altre forme di Ipercubo: *Cube Connected Cycles*

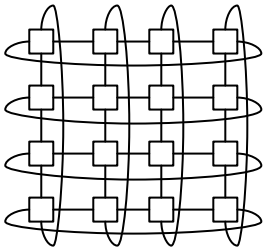


1 – Cube
Connected
Cycles (1 – CCC)

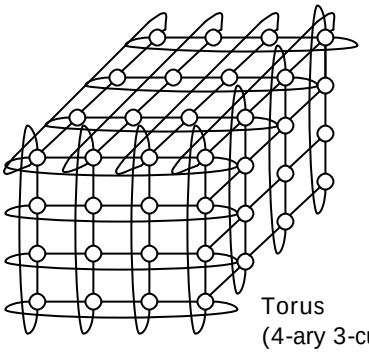


3 – Cube
Connected
Cycles (3 – CCC)

Famiglia delle reti “K-ary n-cube”



Torus
(4-ary 2-cube)



Torus
(4-ary 3-cube)

n → dimensione del cubo
K → radice (numero di nodi per ogni dimensione)
N → numero di nodi

$N = (K)^n$; $K = (N)^{1/n}$; $n = \log_k N$

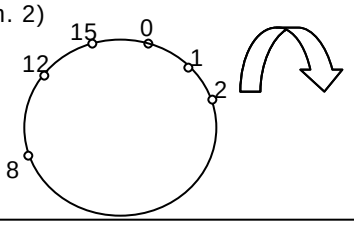
Peculiarità delle diverse Topologie di connessione

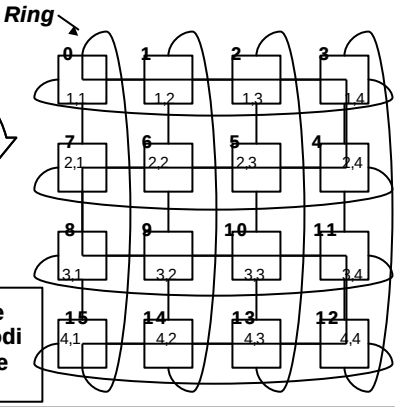
Topologia	Node Degree (*d*)	Network Diameter (*D*)	Numero link (*l*)	Simmetria	Note sulla dimensione della rete
Linear Array	*2*	*N-1*	*N-1*	*No*	*N nodi*
Ring	*2*	*ëN/2û*	*N*	*Si*	*N nodi*
Completely Connected	*N-1*	*1*	*N(N-1)/2*	*Si*	*N nodi*
Binary Tree	*3*	*2(h-1)*	*N-1*	*No*	*Altezza albero* $h = \acute{e}\log_2 N\grave{u}$
Star	*N-1*	*2*	*N-1*	*No*	*N nodi*
2D-Mesh	*4*	*2(r-1)*	*2N-2r*	*No*	*R x r Mesh* *dove* $r = (N)^{1/2}$
Illiac Mesh	*4*	*r-1*	*2N*	*No*	*Equivalente ad un chordal ring di dimensione* $r=(N)^{1/2}$
2D-Torus	*4*	*2ër/2û*	*2N*	*Si*	*r x r torus dove* $r = (N)^{1/2}$
Ipercubo	*n*	*N*	*nN/2*	*Si*	*N nodi,* $n=\log_2 N$ *(dimensione)*
CCC	*3*	*2k-1+ëk/2û*	*3N/2*	*Si*	$N = k \times 2^k$ *nodi con una dimensione di ciclo pari a* $k \geq 3$
K-ary *n*-cube	*2n*	*nëk/2û*	*nN*	*Si*	$N = k^n$ *nodi*

Problema dell' EMBEDDING di una topologia in un'altra

Il pattern delle comunicazioni spesso non coincide con la topologia dell'architettura. Pertanto, è necessario *mappare* il grafo delle comunicazioni nel grafo della topologia (***embedding***). L'embedding è tanto più efficace quanto meno aumenta il numero di hop nuovi da introdurre per collegare due nodi adiacenti nella topologia originaria.
In teoria, è possibile mappare una topologia di dimensione inferiore in una di dimensione superiore senza dover introdurre hop aggiuntivi.

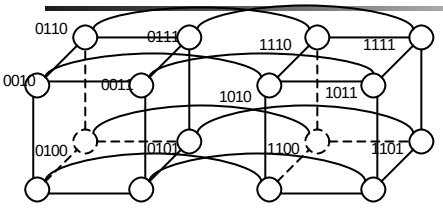
Embedding "perfetto": un ring (Dim. 1) in un torus (Dim. 2)



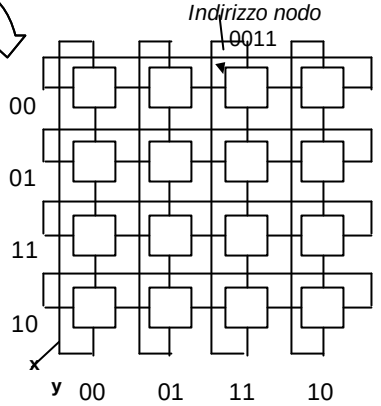


In pratica, bisogna determinare una numerazione dei nodi del grafo di destinazione tale che due nodi adiacenti nel grafo originario corrispondano a due nodi adiacenti nel grafo di destinazione.

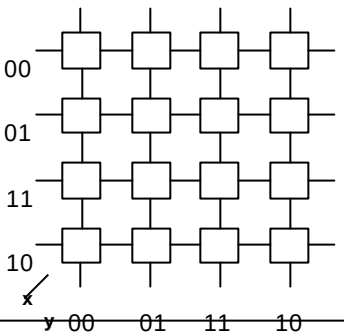
Embedding di un Ipercubo



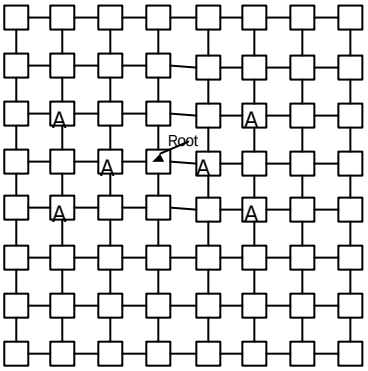
Ipercubo 4D in un Torus 2D



Ipercubo 4D in una Mesh 2D



Embedding di un Albero



Embedding di un Tree in una Mesh 2D

Quale Topologia?

Nella prima e seconda generazione di macchine parallele, la topologia prevalente era quella ad ipercubo (Intel iPSC/1, iPSC/2, iPSC/860, nCUBE, nCUBE-2)
Tuttavia, nella terza generazione, hanno preso il sopravvento architetture con dimensioni più limitate:

- Mesh 2D: Intel Paragon
- Torus 3D: Cray T3D
- Rete Omega (*indiretta*): IBM SP-2

PERCHE?

L'ipercubo ha molte proprietà che favoriscono le comunicazioni:

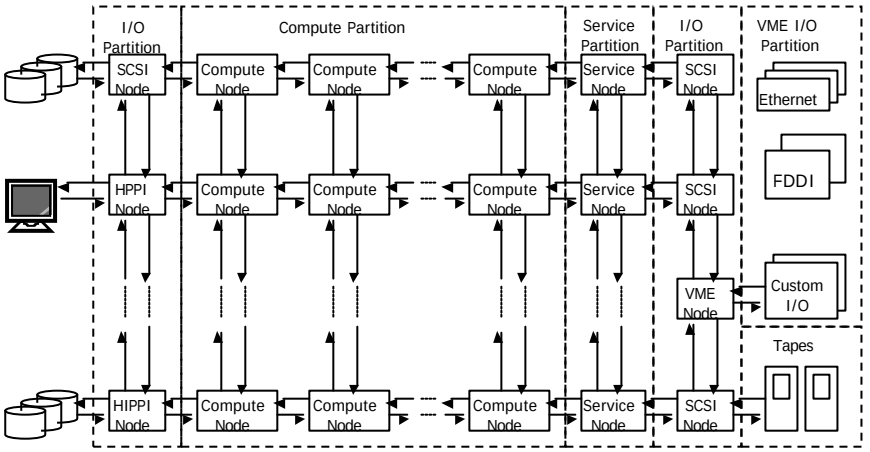
- topologia simmetrica
- diametro che cresce in modo logaritmico rispetto al numero di nodi
- semplicità nel routing

Ma l'ipercubo ha anche molti problemi:

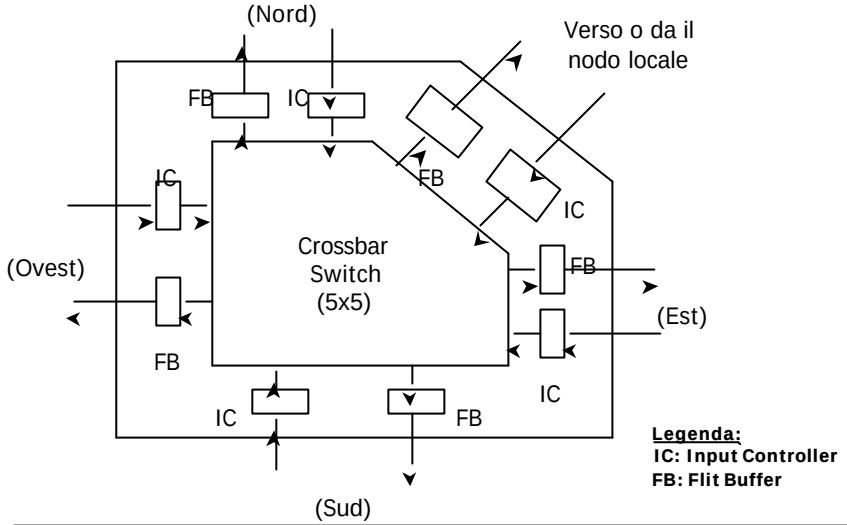
- scalabile solo con numero di nodi come potenza di 2
- dimensione massima limitata dai canali di comunicazione del router
- diametro della rete poco importante con le nuove politiche di routing

MESH-like

Architettura MESH della Intel Paragon
(Tipica macchina “number cranching”)

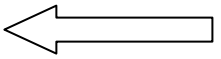


Architettura del Router dell’Intel Paragon



Reti di Interconnessione Dirette

(elementi di caratterizzazione)

- **Topologia**
- **Flow Control** 
- **Politica di Switching**
- **Politica di Routing**

Flow Control

Una rete di interconnessione consiste di molti canali e buffer che sono condivisi da tutti i messaggi che viaggiano nello stesso momento.
Il *flow control* determina l'allocazione di queste risorse ai messaggi che attraversano la rete, soprattutto nel momento in cui vi è un conflitto.

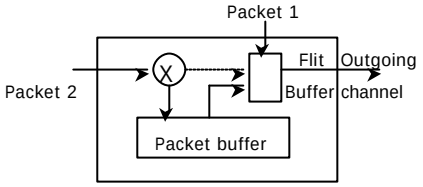
Varie alternative possibili nel caso di conflitti:

- Pacchetto eliminato
- Pacchetto bloccato sul posto
- Pacchetto bufferizzato
- Pacchetto ridirezionato

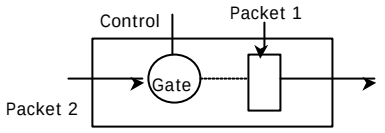
In pratica, il *flow control* determina quale link di output è selezionato per ogni pacchetto che arriva da un link di input e risolve i conflitti nel momento in cui più pacchetti in input richiedono lo stesso link di output

Flow Control (*cont.*)

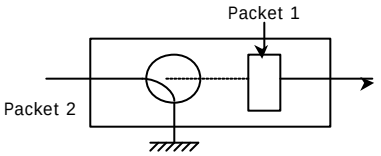
Il flow control non è del tutto indipendente dalla politica di switching. Ovvero alcuni meccanismi di flow control sono possibili soltanto con determinate politiche di switching.



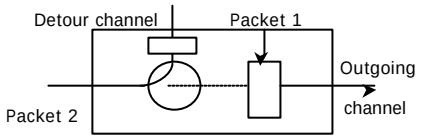
(a) Buffering (Virtual cut-through)



(b) Blocking flow control (Wormhole puro)



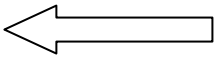
(c) Eliminazione e ritrasmissione (Drop & Retry)



(d) Deviazione dopo blocco

Reti di Interconnessione Dirette

(elementi di caratterizzazione)

- **Topologia**
- **Flow Control**
- **Politica di Switching** 
- **Politica di Routing**

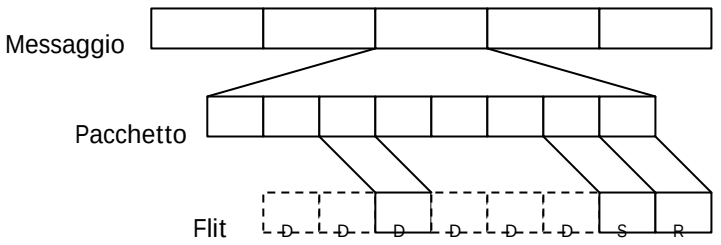
Politica di Switching

Mentre il flow control determina come un pacchetto usa i canali mentre attraversa i router, la politica di switching è il meccanismo effettivo che rimuove i dati dal canale di input e li indirizza verso il canale di output.

Quattro alternative principali:

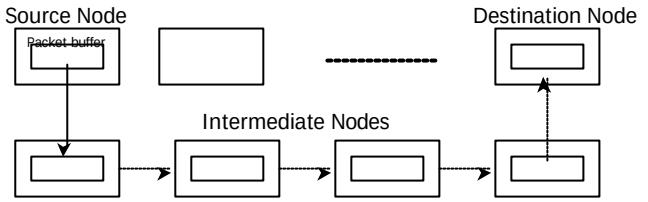
- Store-and-Forward
- Circuit-switching
- Virtual cut-through
- Wormhole routing

Messaggi, Pacchetti, Flits (*Flow Digits*)

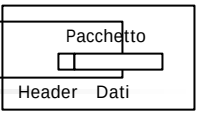


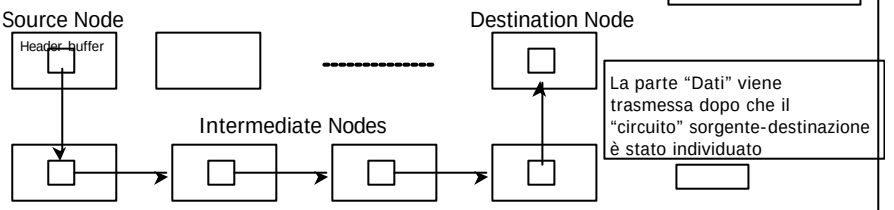
R: Routing Information
S: Sequence Number
D: Dati

Store – and – Forward

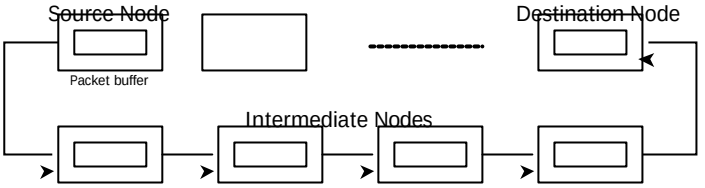


Circuit Switching

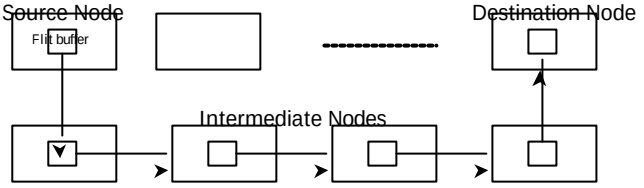




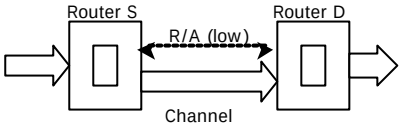
Virtual Cut-Through



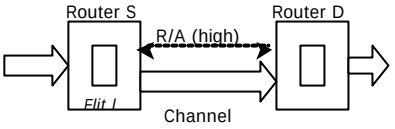
Wormhole Routing



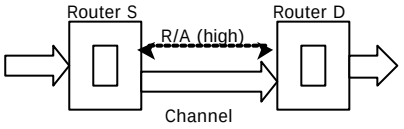
Protocollo di *handshaking* tra Wormhole Router



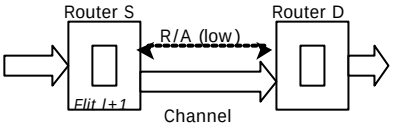
(a) D è pronto a ricevere un flit



(b) S è pronto ad inviare il flit *i*



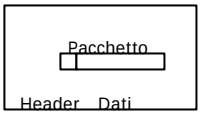
(c) Il flit *i* è ricevuto da D

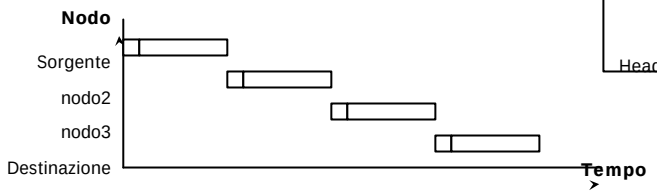


(d) Il flit *i* è ricevuto dal buffer di D
e il flit *i*+1 arriva al buffer di S

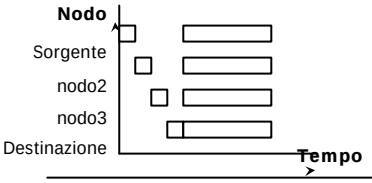
Tempi di trasmissione in assenza di conflitto

Store – and – Forward

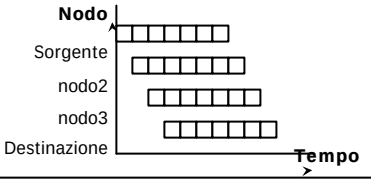




Circuit – Switching

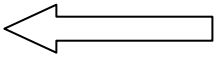


Wormhole / Virtual – Cut – Through



Reti di Interconnessione Dirette

(elementi di caratterizzazione)

- **Topologia**
- **Flow Control**
- **Politica di Switching**
- **Politica di Routing** 

Politica di Routing

In assenza di una interconnessione completa, la politica di routing determina il cammino (*path*) del messaggio dal nodo sorgente al nodo destinatario.

- ***Source routing****: il nodo sorgente seleziona l'intero cammino prima di inviare il messaggio.*
  (il cammino non può essere modificato dopo che il messaggio ha lasciato il nodo sorgente)
- ***Distributed routing:*** *ogni router intermedio decide se il messaggio deve essere consegnato al processo locale p o inviarlo ad un altro router.*
  In questo secondo caso, si applica una ***politica di routing*** *per stabilire a quale router confinante inviare il messaggio.*
  (La decisione deve essere presa nel minor tempo possibile; l' algoritmo deve poter essere facilmente implementato in hardware; la decisione non deve richiedere informazioni sullo stato globale della rete).

Politica di routing (*cont.*)

Routing deterministico

Il cammino è completamente determinato dall'indirizzo del nodo sorgente e del destinatario

Routing adattivo

Il cammino è stabilito sulla base di condizioni dinamiche della rete (nodi/link guasti o congestionati)

Il routing deterministico è sempre minimale. Il routing adattativo può essere minimale o non minimale.

Routing minimale

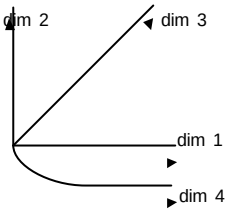
Il cammino selezionato è uno *shortest path* tra nodo sorgente e destinatario. (Ogni link attraversato porta il messaggio più vicino alla destinazione)

Routing non minimale

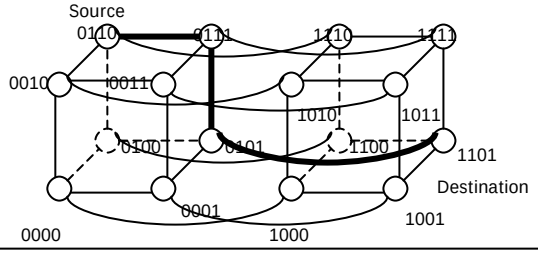
Consente di seguire cammini più lunghi (tipicamente in risposta a condizioni dinamiche della rete).

Routing deterministico (Minimale)

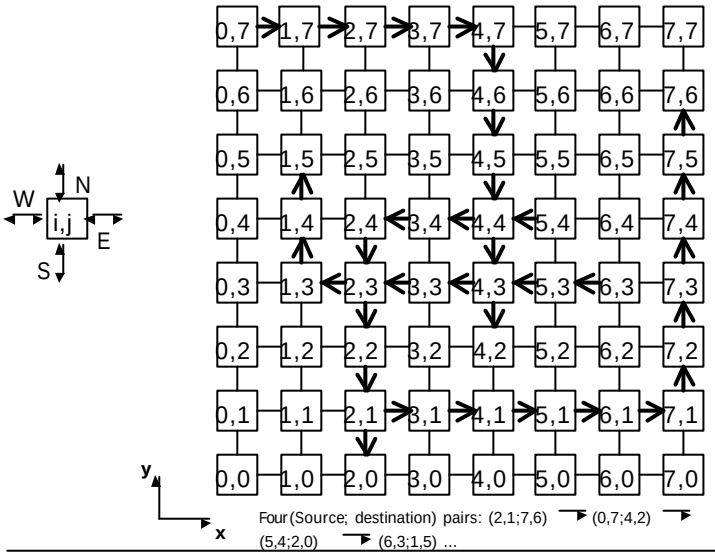
Es. : "Dimension Ordering" (Ipercubo) → E-cube



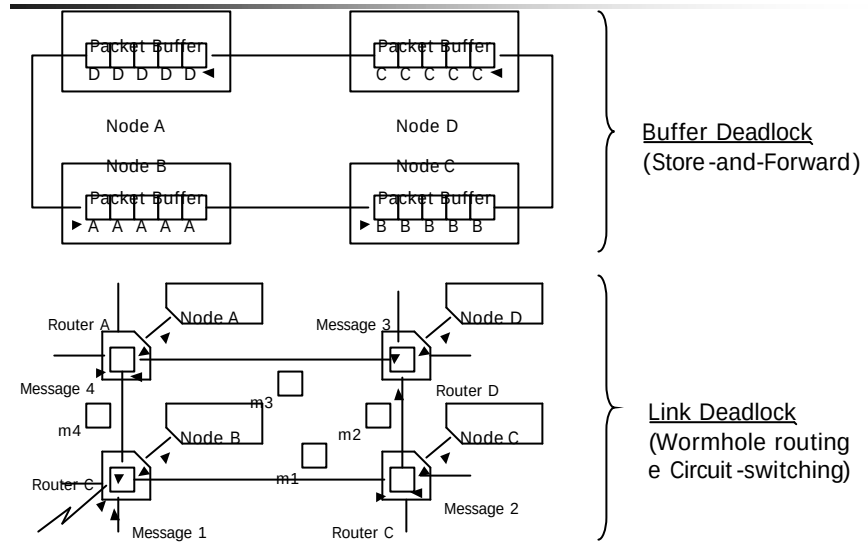
Source: s= 0110
Destination: d = 1101
Route:
0110 → 0111→ 0101→ 1101



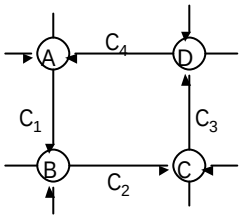
“Dimension Ordering” (Mesh) → X-Y Routing



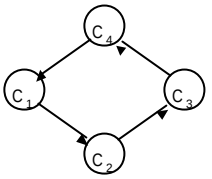
Problemi di deadlock



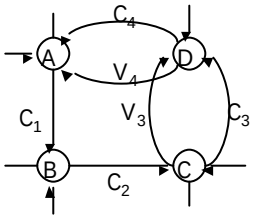
Evitare i Deadlock mediante i Virtual Channels



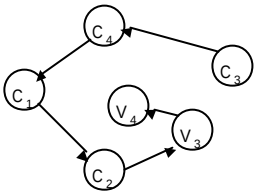
(a) Channel deadlock



(b) Channel-dependence graph contenente un ciclo

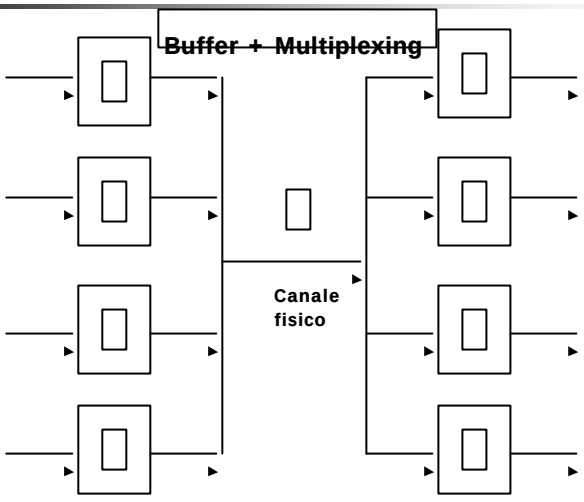


(c) Aggiunta di due canali virtuali (V_3,V_4)



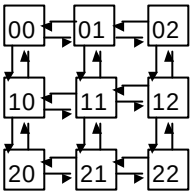
(d) Channel-dependence graph modificato con l'aggiunta dei due canali virtuali

Virtual Channels (realizzazione)

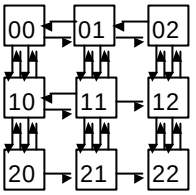


Vantaggi dei Virtual channels

Es. *Dimension Ordering* X-Y

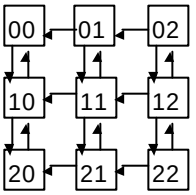


(a) Mesh originale **senza** virtual channel

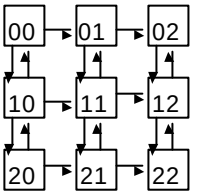


(b) **Aggiunta di due** virtual channels nella dimensione Y

Consentono comunicazioni contemporanee del seguente tipo:



(c) Messaggio *westbound*



(d) Messaggio *eastbound*