Reţele Multistagiu

O reţea multistagiu constă din mai multe nivele sau stagii de *switch*-uri (comutatoare) conectate între ele. Un *switch* asigură legături interne între mai multe intrări şi mai multe ieşiri.

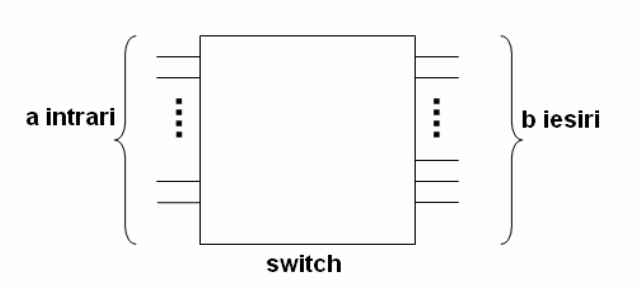

Figura 1. Un *switch* asigură conexiunea dintre *a* intrări şi *b* ieşiri.

Într-o reţea multistagiu, cu cât numărul de stagii este mai mare, cu atât întârzierea reţelei va fi mai mare. Alegând un anumit tipar de interconectare a *switch*-urilor, se pot realiza diferite topologii de reţele.

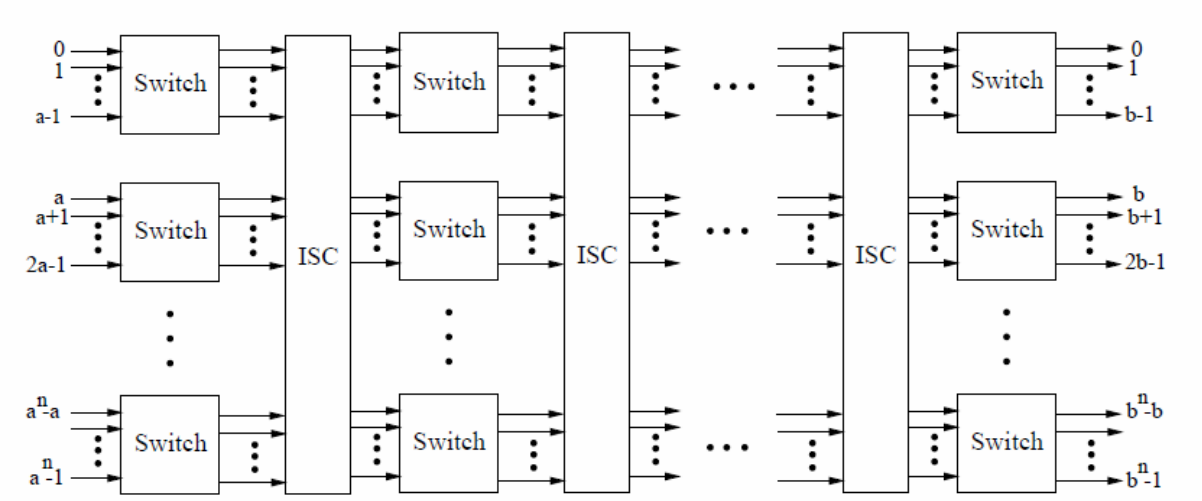

Figura 2. Generalizare a unei retele multistagiu interconectate (MIN), construita cu switch-uri a x b si un tipar pentru conexiunile interstagiu (ISC).

Un exemplu de reţele multustagiu sunt reţelele de Omega (fig. 3). Aici, conectarea *switch*-urilor realizează o distribuţie completă a semnalelor de intrare către ieşiri (de la orice intrare se poate ajunge la orice ieşire).

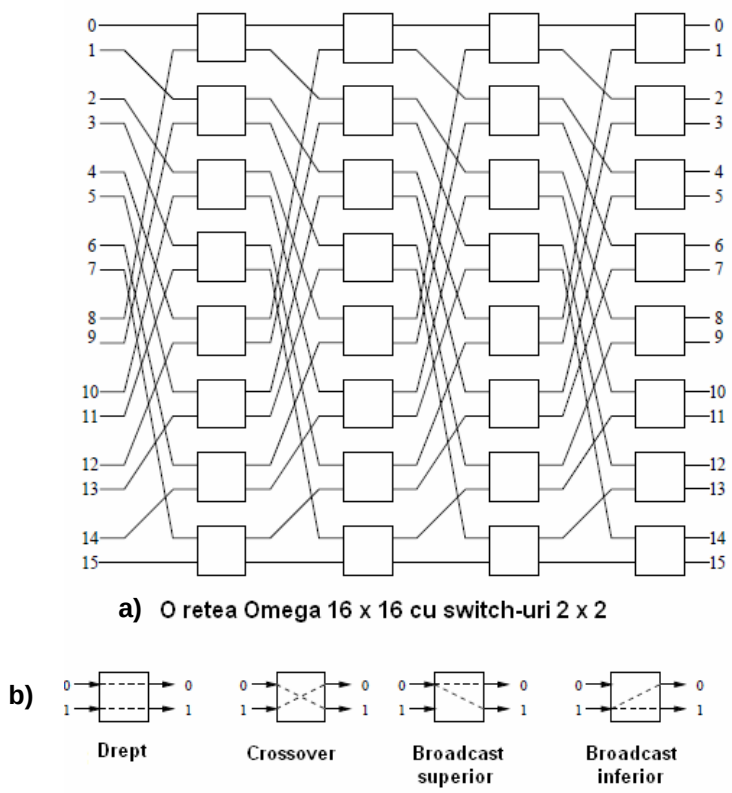

Figura 3. a) Exemplu de reţea Omega. b) Conexiuni aceptate ale *switch*-urilor

Reţelele multistagiu pot să fie **blocante** sau **neblocante**. Într-o reţea blocantă nu se pot realiza simultan toate legăturile de la intrare la ieşire din cauza unor conflicte de comutare internă a *switch*-urilor. La *switch*-urile unde apar astfel de conflicte de comutare, o parte din semnalele de la intrare sunt blocate până când canalele de ieşire necesare devin disponibile. Reţelele neblocante suportă orice conexiune fără blocare.
De exemplu, reţelele Omega sunt reţele blocante (fig. 4):

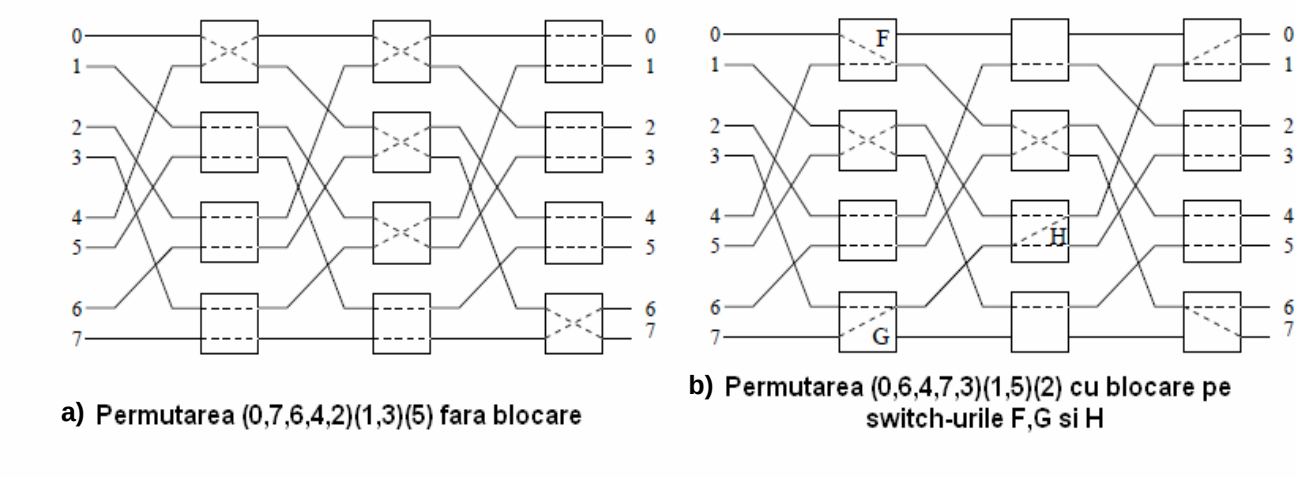

Figura 4. Exemplu de conexiuni într-o reţea Omega.
a) fără blocare; b) cu blocare

În figura 4 a), permutarea (0,7,6,4,2)(1,3)(5) semnifică următoarele conexiuni (neblocante):

0->7; 7->6; 6->4; 4->2; 2->0; 1->3; 3->1; 5->5.

În figura 4 b), conexiunile dorite sunt blocante:

0->6; 6->4; 4->7; 7->3; 3->0; 1->5; 5->1; 2->2.

Reţele *Crossbar*

O reţea de tip *crossbar* constă dintr-o matrice de comutatoare simple de tip *on/off* care pot realiza sau întrerupe o legătură de la o intrare către o ieşire:

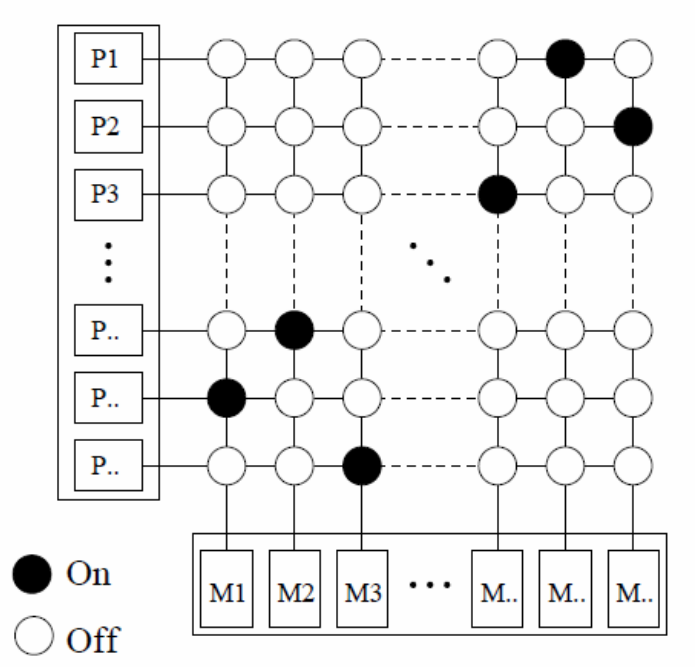

Figura 5. Reţea *Crossbar*

Într-o reţea de tip *crossbar*, pe o linie verticală nu putem avea mai multe *switch*-uri setate pe *on*, adică nu se poate ca mai multe procesoare să acceseze simultan un acelaşi bloc de memorie. Pe o linie orizontală putem avea mai multe *switch*-uri setate, rezultând în acest caz o emisie *multicast* de la un procesor la mai multe blocuri de memorie.

Avantaj: comunicarea prin *switch*-uri *crossbar* este neblocantă, adică toate combinaţiile de legături între procesoare şi memorii sunt posibile fără conflicte (respectând condiţiile anterioare).

Sunt posibile 3 implementări hardware pentru o reţea de tip *crossbar*(fig.6):

- folosind un lanţ de multiplexoare;
- folosind două reţele de semnale: una de intrări, alta de ieşiri, conectate prin circuite *three-state*;
- folosind un buffer de memorie în care intrările corespund cu procesoarele; fiecare element din buffer indică modulul de memorie RAM care poate fi accesat de procesorul respectiv.

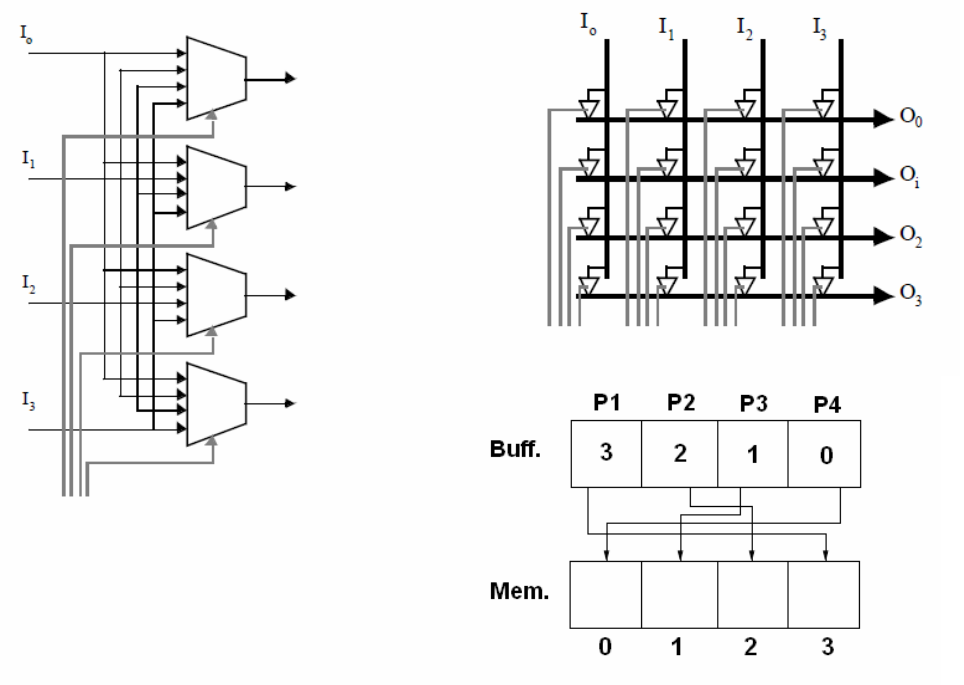

Figura 6. Implementări hardware pentru o reţea de tip *crossbar*

Comparaţie între reţelele directe şi cele indirecte:

a) La reţelele directe există canale de comunicaţie dedicate între procesoarele vecine. La reţelele indirecte nu există legături punct la punct între procesoare.

b) La reţelele directe este necesar un algoritm de rutare pentru ca informaţiile să ajungă de la sursă la destinaţie. La reţelele indirecte doar reţelele multistagiu necesită rutare. Aici *switch*-urile comutate într-un anumit mod vor asigura legătura între sursă şi destinaţie. Magistralele nu necesită rutare iar reţelele *crossbar* necesită comutarea unui singur *switch* pentru efectuarea conexiunii.

c) Reţelele directe pot fi uşor scalabile, adică se pot adăuga noduri în reţea cu modificări minime. Reţelele indirecte sunt mai greu scalabile, astfel magistralele pot ajunge la saturaţie când se depăşeşte un anumit număr de procesoare. La reţelele multistagiu şi *crossbar* creşterea numărului de procesoare devine scumpă datorită conexiunilor şi *switch*-urilor care trebuie adăugate în plus.

d) Reţelele directe folosesc de obicei comutarea prin pachete; reţelele *crossbar* şi magistralele folosesc comutarea prin circuite. La reţelele multistagiu, dacă

switch-urile sunt setate înaintea începerii comunicaţiei, atunci va fi realizată comutarea prin circuite. Dacă *switch*-urile pot comuta dinamic, în funcţie de ruta unui mesaj, atunci se poate implementa comutarea prin pachete.

Comutarea prin pachete (*packet switching*)

Reţelele care utilizează comutarea prin pachete folosesc *switch*-uri dedicate pentru transmiterea pachetelor. Aceste *switch*-uri transmit pachetele în reţea până când ele ajung la nodul destinaţie.

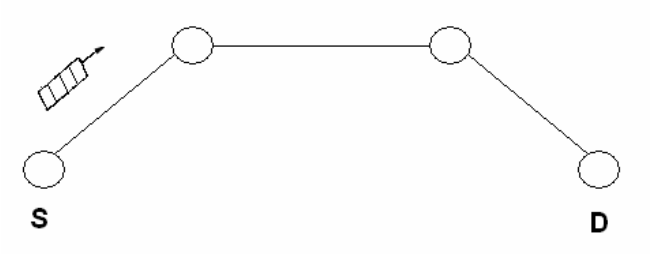

Figura 7. Transmiterea pachetelelor de la sursă la destinaţie

La reţelele directe, *switch*-urile fac parte din nodurile reţelei. Aici, *switch*-urile transmit pachetul mai departe dacă nodul curent este diferit de nodul destinaţie, sau copiază pachetul în memoria locală dacă nodul curent este acelaşi cu nodul destinaţie.

La reţelele indirecte, *switch*-urile nu fac parte din nodurile de procesare.

Există 3 modalităţi de comutare a pachetelor:

1) *Store & Forward*
2) *Wormhole routing*
3) *Virtual cut-through*

1) Store & Forward

Aici, pachetul este cea mai mică unitate de transfer a datelor. *Switch*-urile din reţea dispun de buffere pentru memorarea locală a pachetelor.

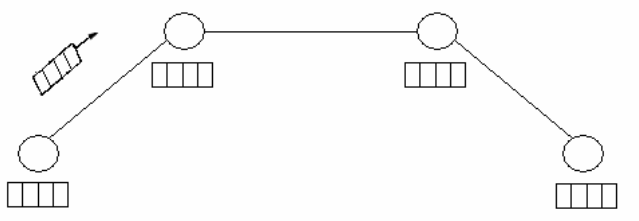

Figura 8. Transmisia ***Store & Forward***

În drumul de la sursă la destinaţie, un pachet ce tranzitează un nod va fi mai întâi memorat în întregime în nodul respectiv (etapa ***Store***), apoi va fi transmis mai departe către nodul destinaţie (etapa ***Forward***). Tehnica este simplu de implementat, dar toleranţa la erori nu e foarte bună.

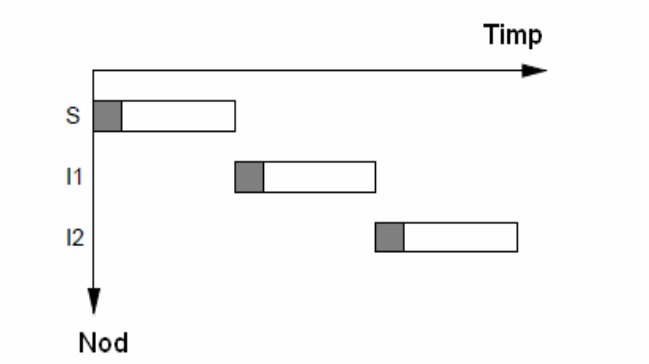

Figura 9. Diagrama de timp a transmisiei unui pachet - ***Store & Forward***

2) Wormhole routing

În cadrul acestei metode, pachetele sunt împărţite în blocuri de dimensiuni mici (de la 1 la câţiva octeţi). Blocul de date minimal care este transmis pe liniile de comunicaţie se numeşte **flit** (*Flow Control Digit*). Datele sunt conduse în reţea în manieră *pipeline* flit cu flit, toate flit-urile urmând acelaşi traseu. Când un nod recepţioneaza un flit, îl transmite imediat catre nodul următor şi apoi va recepţiona următorul flit.

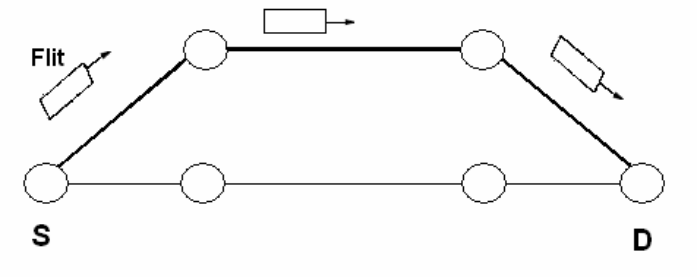

Figura 10. Transmisia ***Wormhole routing***

Informaţia de rutare e stocată în primele flit-uri din pachet. Pe toată durata transmiterii unui pachet, canalele de comunicaţie alocate la început rămân ocupate până la transmisia ultimului flit din pachet, care le eliberează.

Avantajul metodei: timpul de tranziţie al datelor în noduri scade, flit-urile având dimensiuni mai mici decât dimensiunea unui pachet.

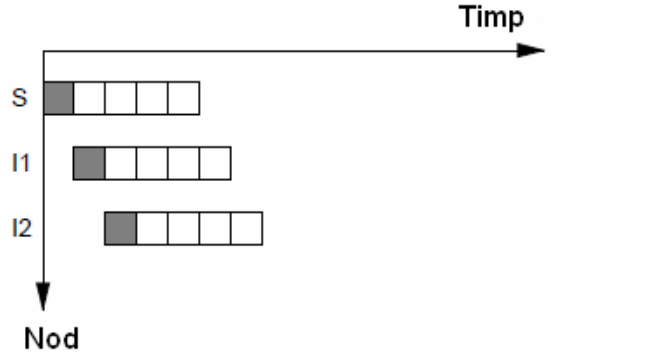

Figura 11. Diagrama de timp a transmisiei unui pachet - ***Wormhole routing***

Dezavantaj: metoda poate duce la saturarea reţelei. Astfel, atunci când *header*-ul unui pachet e blocat, rămân blocate toate canalele de comunicaţie alocate pentru transmisia pachetului, fiecare flit asteptând într-un nod intermediar. Aceasta situaţie poate duce de asemenea şi la blocarea altor pachete care circulă în reţea.

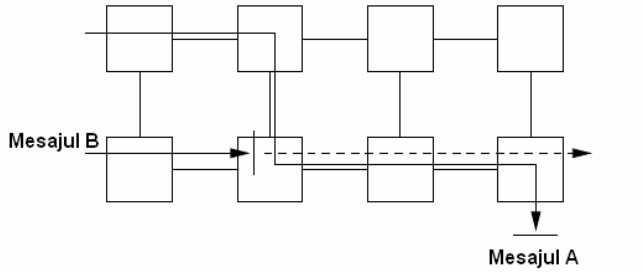

Figura 12. Exemplu de blocare a mesajelor la metoda ***Wormhole routing***

Timpii de transmisie a pachetelor pe liniile de comunicaţii se calculează astfel:

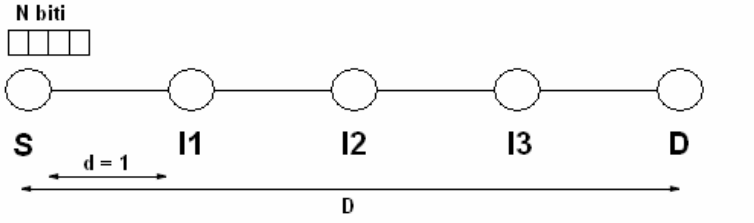

Figura 13. Transmiterea unui pachet – calculul timpilor de transmisie

- ***Store and forward***: - N = numărul de biţi dintr-un pachet
  - W = lăţimea de bandă a canalului

$$T1 = \frac{N}{W} \times D$$

- ***Wormhole routing***: - F = numărul de biţi dintr-un flit

$$T2 = \frac{N}{W} + \frac{F}{W} \times (D-1)$$

unde:

$\frac{N}{W}$ - timpul în care ultimul flit din pachet ajunge la nodul vecin I1

$\frac{F}{W} \times (D-1)$ - timpul după care ultimul flit (de dimensiune F) ajunge de la nodul I1 la destinaţie

3) Virtual cut-through

La această metodă, bufferele din fiecare nod au mărimea egală cu lungimea unui pachet. Transmisia pachetului se face *pipeline* flit cu flit ca şi la ***Wormhole Routing***. Dacă se blochează un *header* al unui pachet, transmisia celorlalte flit-uri continuă până când toate fliturile ajung în nodul care s-a blocat. Astfel canalele alocate se pot elibera, reducând gradul de saturare al reţelei.

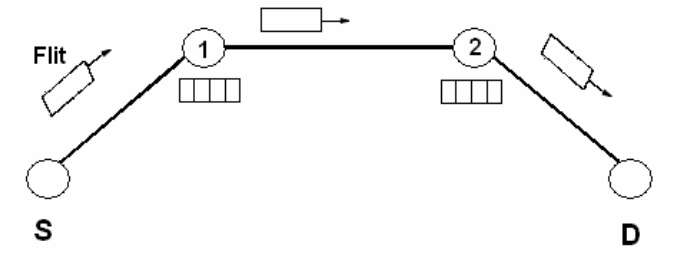

Figura 14. Metoda ***Virtual cut-through***

Pentru controlul fluxului de date (asigurarea faptului că datele nu se pierd in reţea), se poate folosi un protocol de tip *handshaking*, care este bazat pe o pereche de semnale ***Request/Acknoledge***:

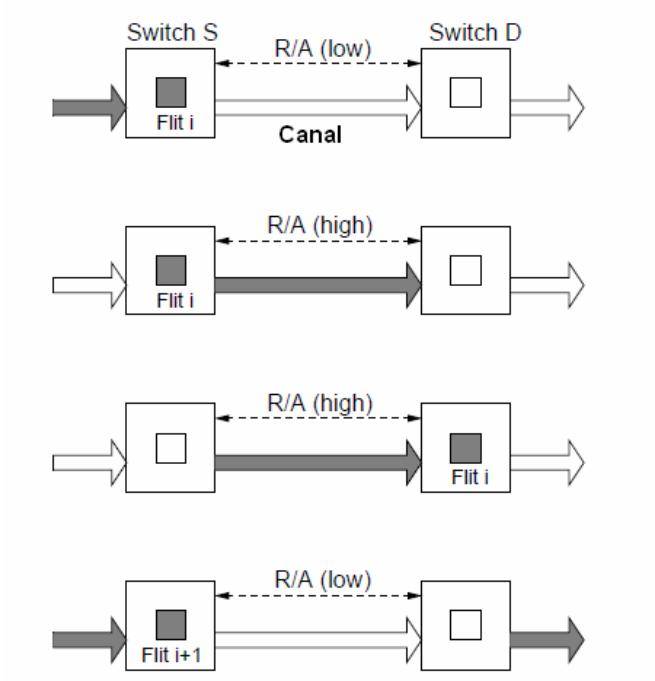

Figura 15. Protocol de transmisie bazat pe semnale ***Request/Acknoledge***