SPM 10 → Cyclic Connected Cube (CCC) (pg. 50): • se deosebește de hypercube prin faptul că vf. cubului au fost înloc. de inele ⇒ dc. $n=3$, inelul are 3 noduri (inelul are atâtea noduri cât n al n-Cubului);

- avem $n \cdot 2^n$ noduri;
- obs. că gradul nodurilor e întotdeauna n.
- dc. vreau să fol. numai p noduri din cele $n \cdot 2^n$, tb. să rez. ec $n \cdot 2^n = p$, unde $n \in N^*$

→ n-dymensional mesh (pg. 51): • fol. la pb. în spațiu (?);

- fol. la predicția meteo, etc.;
- pe fiecare din cele n dimensiuni avem k_i noduri ⇒ rețeaua are $k_{n-1} * k_{n-2} * \dots * k_0$ noduri;
- diam= $\sum_{i=0}^{n-1} k_i - 1$
- două noduri $X(x_0, \dots, x_{n-1})$ și $Y(y_0, \dots, y_{n-1})$ s.n. vecine dc. diferă printr-o sg. coordonată (de ex. j), și $x_j = y_j + 1$.

→ k-ary n-cube (pg. 54):

Metode de dirijare:

→ se lucrează cu pachete; pachetul pleacă din sursa S și tb. să ajungă la dest. D;

→ o posib. este tranzitarea a x noduri (nodurile traseului), iar fiecare nod intermediar așteaptă să primească tot pachetul, și apoi îl trimite mai departe ⇒ fiec. nod tb. să aibă un buffer ⇒ procedura hand-shake → se verifică dc. bufferul e disponibil, iar dc. nu e NU se face busy waiting, ci (urmând intermediar) come back later;

→ cererea de trimitere a unui pachet se înreg. la nodul solicitat; aici se face o listă, iar apoi, în ordinea priorităților pachetelor, buff.

sursa e informat că bufferul dest. va fi liber la mom. t_i;

→ apare overhead, însă transmisia este sigură;

→ în general transmisia este serială ⇒ apare o latență egală cu (nr. noduri intermediare)×(timpul necesar unui pachet să se propage de la nodul i la nodul j);

→ mec. s.n. store-foreward;

→ alternativa: wormhole → ideea de a descompune ~~mesajul~~ pachetul în unități mai mici, numite „flit" (to flit = to move quickly from one place to annother without stopping for long) ⇒ în loc să transmit un pachet complet și apoi să aștept ca totul să treacă, descompun pachetul în flits și, cum am primit o bucățică din mesaj o și transmit nodului urm.; în același timp primesc al 2-lea flit, îl dau mai departe etc. ⇒ similar cu struct ppl.;

→ tb. făcută f. clar o asociere a fliturilor la dif. mesaje, pt.că la un nod pot veni la un mom. dat m. multe flituri din mesg. diferite ⇒ flitul tb. să aibă o parte de info. care să spună din ce mesj. face parte ⇒ aparent redondant;

→ retele express ⇒ ∃ niște pcte. intermediare din care niște noduri sunt bypass-ate prin legături rapide;

Rețele dinamice de interconectare (Cap. 4):

→ prin intermediul unor comutatoare, leg. S–D se pot reconfigura cf. situației date;

→ un comutator are 2 fcții:
- straight: in0 → out0 && in1 → out1
- exchange: in0 → out1 && in1 → out0

→ m. multe trepte de conectare: m. multe comutatoare de același tip, corespunzător comandate ⇒ ca să transmit pachetul de la S la D tb. să fol. numele binar al out. și să fol. bitul coresp. pe treapta i (1 = exchange; 0 = straight);

→ și comutatoarele prezintă o latență;

→ pg. 5: 101 → 1 = exchg. → 0 = straight → 1 = exchg → d5 / ⇒

(în S0 aplic)

în S4 aplic 101 → 1 = exchg. → 0 = straight → 1 = exchg. → d6

⇒ o asemenea rețea se descrie prin permutările pe care le permite: (s_i, d_i) ≡ sursa i se leagă la dest. i ⇒ se poate ca anumite conexiuni să nu fie posibile ⇒ rețea cu blocaje;

→ în anumite sit. se poate ca prin rearanjare leg. care anterior era imposibilă să devină posibilă ⇒ rețea cu rearanjare;

ex.: am 6 conex., iar cu aranjarea actuală a 7-a e imposibilă; dar prin rearanjarea celor 6 se poate face posibilă și a 7-a conexiune, atunci avem rețea cu rearanjare;

→ rețele fără blocaje ⇒ (∀) ar fi situația conexiunilor la un mom. dat, (∀) nouă conexiune va fi posibilă în (∀) sit. ⇒ ⇒ rețea complexă, cu m. multe trepte de comutare și cu m. multe comutatoare pe fiecare treaptă;

→ clasif.:
- crossbar (din telefonie)
- single stage ⇒ recirculating
- multistage
  - concentrator
  - connector
    - blocking
    - rearrangeable
    - nonblocking

• crossbar ⇒ au n linii oriz. și m coloane verticale,
- n linii oriz. ↓ se montează procesoare
- m coloane verticale ↓ se montează memorii

→ la intersecția i, j ∃ conectare dc. vreau ca µP i să fie conectat la memoria j, altfel ∄ conectare

→ permite conectarea simultană a (∀) µP la (∀) memorie, cu cond. să nu am 2 cereri simultane la a. memorie (⇒ priorități, etc.)