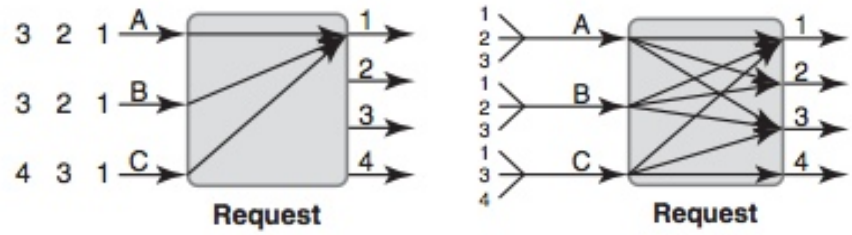
**Parallel Iterative matching:**

It is a natural consequence of VOQ.




**With out VOQ** **With VOQ**

Each cycle does consists of two operations,

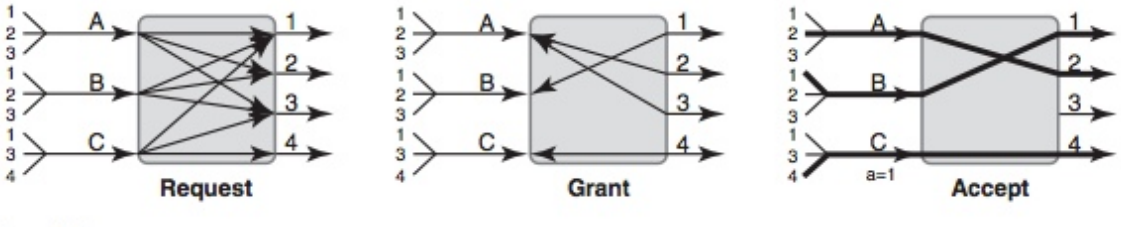
1) Switch packets based on previous decision (transfer)
2) Make decision for the next packets (matching)

Matching is a process of selecting an output port for each input port such that maximum number of packets can be transferred during the switching phase.

This matching process may take multiple iterations to reach the maximum match between the output and input ports. It is possible that some input ports may not be matched to any output port; this is based on the distribution of packets on the input queues.

Each **iteration** consists of three phases,
Request (all)
Grant (randomly)
Accept (randomly)

In PIM, this matching is done by **randomly** selecting input ports during the grant phase and **randomly** selecting output ports during the accept phase




In the above case, maximal match was obtained after a single iteration itself. However PIM may take O (log N) iterations to converge to a maximal match, where N is the number of input ports.

A single **Round** consists of multiple such iterations of matching, followed by the packet switching between the matched ports. The packet switching takes place once the maximal match is obtained. Each round is a cycle.

Two main difficulties with PIM is

1) It takes O (log N) iterations
2) It randomly selects input/output ports; this randomization is difficult

Difficulty in randomization:

Generating a random number is difficult, however it can still be obtained using some combinations of simple mathematical operations as the requirements for this random number is not as strict as in the security domain.

Ex: { 0,1,2,… 2^32 -1} % 137 where 137 is the input port number

The next difficulty lies in finding out the number of requests, suppose there are 256 ports and 137 of them are sending requests the difficulty is to know that there are 137 requests. Even this can be mitigated using the ”popcount” instruction using specialized software. However this can be slow if done using software.

So the **hardest part** is finding the kth request in the word, where k is your random number generated by your RNG.

So Randomization is not cheap.

**iSLIP:**

iSLIP overcomes the two difficulties faced by the PIM method. It does not use randomization; the underlying mechanism it uses for de-randomization is by using Programmable Priority Encoders.

It basically uses some extra memory for selecting ports during matching. Both input ports (Accept pointer) and output ports (grant pointer) remember the ports that they passionately matched with.

Passionately matched means – if the match succeeds during in the first iteration of the round for a particular pair of input and output ports.
i.e. Grant pointer: Accept port

Accept pointer: Input port - remembers whom it last passionately matched with (it may be last round or hundred rounds before). During the next iteration this algorithm start scanning from next to the passionately matched output port. This preserves fairness

**ISLIP algorithm:**

Each input port **‘requests’** for all where all means who's VOQ are non empty

Each output port **'grants'** to the first input port counted from grant pointer
Each input port **'accepts'** the first grant counted from the accept pointers
#IMP# Grant and Accept pointers will be updated only for the successful matching made during the first iteration