Online Routing on the Mesh and Offline Routing on the Benes Network

Mykola Protsenko

29.08.2008

Contents

1 Introduction

In this section we will give some general information about parallel computation and also some important definitions and notations.

1.1 Parallel computation models

A parallel machine consists of a set of processors $P = \{P_0, ..., P_{n-1}\}$ and communication mechanism. Generally there are 3 types of communication mechanisms:

- shared memory

- communication via bus
- communication via interconnection networks.

We will focus on machines with communication via interconnection networks.

In parallel machines with this type of communication processors are connected to *processor network (network)* according to *communication graph G = (P, E).*

The number of possible links int n-processors network is $\binom{n}{2}$. For instance, in network of 1024 processors it will be 523776 possible links. It is obviously impossible to implement such network, so real networks have constant degree.

We will also have to use generally wrong assumption, that all processors are synchronous, i. e. all processors begin next step of calculation simultaneously.

1.2 Notation and definitions

The following lines describe the notation used in this paper.

- $\mathbb{N} := \{0, 1, ...\}$, $\mathbb{N}^+ := \{1, ...\}$, $[n] := \{0, ..., n-1\}$.

  $\mathbb{Z}$ is the set of integers, $\mathbb{R}$ - the set of real numbers.

  $(a, b) := \{x \in \mathbb{R} | a < x < b\}$
- For a set X its power set is denoted as $\mathfrak{P}(X)$.
- $bin_d(n)$ denotes binary representation of n $\in \mathbb{N}$ using d bits, potentially with leading zeros. For example $bin_4(3) = 0011$.
- $(\overline{a})_n$ for $\overline{a} \in [n]^d$ denotes natural number which is represented as $\overline{a}$ to base n. For example $(201)_3 = 19$.
- Let $G = (V, E)$ be a graph. Then $\Gamma_G(v) := \{u | \{u, v\} \in E\}$; $U \subseteq V$: $\Gamma_G(U) := \bigcup_{v \in U} \Gamma_G(v)$.
- $\Gamma_k(U)$ is the **k-neighborhood** of U and is defined recursive as follows: $\Gamma_1(U) := \Gamma_G(U)$; $k > 1 : \Gamma_k(U) := \Gamma_{k-1}(U) \bigcup \Gamma_1(\Gamma_{k-1}(U))$.
- Let G be a graph and U a subset of vertices of G. $G\ |_U$ denotes a subgraph of G, which consists of vertices included by U and all edges that join two vertices in U.

  Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called **isomorphic** if exists a bijection between the vertex sets of G_1 and G_2: $\varphi : V_1 \to V_2$, with $\{u, v\} \in E_1 \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E_2$. This property is denoted as $G_1 \cong G_2$.

Now some common used definitions follows.

Definition 1 *Let $n, d \in \mathbb{N}^+$, $\overline{a} = (a_{d-1}, ..., a_0)$, $\overline{b} = (b_{d-1}, ..., b_0) \in [n]^d$. The **Hamming distance** between $\overline{a}$ and $\overline{b}$ is defined as $hamming(\overline{a}, \overline{b}) := \sum_{i=0}^{d-1} |a_i - b_i|$.*

Definition 2 *For a graph $G = (V, E)$ and $x, y \in V$ is $dist_G(x, y)$ the length of* ***shortest path*** *between x and y, i. e. the number of edges in this path. The* ***diameter*** *of a graph is $diam(G) = max\{dist_G(x, y)|x, y \in V\}$.*

Now we will define what the routing is.

Definition 3 (Routing protocols) *Let $M = (P, E)$ be a network, where $P = [N]$ is the set of processors, and $f : [N] \times [p] \to [N]$ some function and let $x_{0,0}, ..., x_{0,p-1}$, $x_{1,0}, ..., x_{1,p-1}, ..., x_{N-1,0}, ..., x_{N-1,p-1}$ be messages.*

We say ***"M is routing $x_{0,0}, ..., x_{N-1,p-1}$ according to f"*** *when: if processor i saves the package $(i, f(i, k), x_{i,k})$ than the message $x_{i,k}$ becomes known for processor $f(i, k)$. This process is called* ***function routing.*** *Is $p = 1$ then f describes a permutation on $[N]$ so we speak of* ***permutation routing*** *and leave the second parameter of f.*

A ***routing protocol for*** *M consists of protocols for all processors* i. *Each processor* i *has a* ***buffer*** *to store packages.*

In each ***routing step*** *every processor i does following:*

- i *chooses one package from his buffer;*
- i *selects one of his neighbors j;*
- i *sends the package to j, it means that* i *saves the package in buffer of* j *and deletes it from it's own buffer.*

M works ***synchronous****, i. e. all processors $0, ..., N - 1$ start the t-th routing step simultaneously. The* ***routing time*** *is the number of routing steps, the* ***buffer size*** *is the maximum amount of packages that can be stored in a buffer at the same time.*

Routing with preprocessing *(also* ***off-line routing****) is a routing protocol which depends on f. At the beginning of routing some computation is performed depending on f to generate protocols for processors* i.

In ***routing without preprocessing (on-line routing)*** *the routing protocol is independent from f.*

2 Permutation routing on the mesh networks

In this section we will consider grid networks family. This networks have regular structure so they are inexpensive to build as long as their dimension is not too big.

Definition 4 *Let $a, b \in \mathbb{N}^+$. The* ***(n,d)-grid*** *M(n,d) is a network with set of processors $P = \{\overline{a}|\overline{a} \in [n]^d\}$ and communication graph $G = (P, E)$, $E = \{\{\overline{a}, \overline{b}\}|\overline{a}, \overline{b} \in [n]^d, hamming(\overline{a}, \overline{b}) = 1\}$ This network is called d-dimensional grid with edge length n. The edge connecting two vertices $\overline{a}$ and $\overline{b}$ with $|a_i - b_i| = 1$ is called an edge in* ***Dimension i****.*

Properties of $M(n, d)$

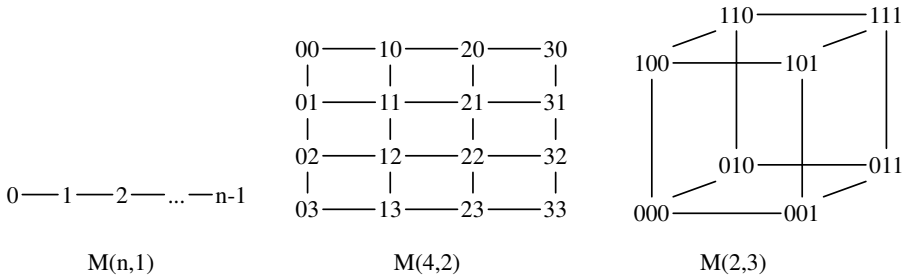

Figure 1: Some examples of M(n,d)

1. $M(n,d)$ has n^d nodes and $dn^d - dn^{d-1}$ edges.
2. $dist(\overline{a},\overline{b}) = hamming(\overline{a},\overline{b})$;
3. $diam(M(n,d)) = (n-1)\cdot d$
4. $M(n,d)\mid_{\{\overline{a}|a_i=l\}} \cong M(n,d-1)$ for $d>0$ and fixed i,l
5. $M(n,d)\mid_{\{i\overline{b}|i\in[n]\}} \cong M(n,1)$ for fixed $\overline{b}\in[n]^{d-1}$

2.1 Online routing on linear array

Theorem 1 (Online routing on linear array) *Permutation routing without preprocessing in $M(n,1)$ can be performed with routing time $2(n-1)$ and buffer size 3.*

Proof:
Let π be any permutation on $[n]$. The following pseudo code describes an algorithm that in first phase sends packages which destination is located to the left hand from start, and int second phase all another packages.

for all processors $i \in [n]$ do

PHASE 1:
    for t:=1 to n-1 do
        for all packages $(j,\pi(j),x_j)$ do
            if $i \le \pi(j)$ then do nothing
            else send package to processor i-1
        done
    done
{Result: all packages with $\pi(j) \le j$ have reached their destination.}

PHASE 2:
    for t:=1 to n-1 do

```
            for all packages (j, π(j), x_j) do
                if i ≥ π(j) then do nothing
                else send package to processor i+1
            done
        done
{Result: all packages have reached their destination}
done
```

In both phases any processor i stores in it's buffer at most 3 packages: package i, package j with $\pi(j) = i$ and some other package that must be transferred. It means that needed buffer size is 3.

The routing time is obvious.

□

2.2 Online routing on 2D array

Theorem 2 (Online routing on 2D array) *Permutation routing without preprocessing in $M(n, 2)$ can be performed with routing time at most $4(n-1)$ and buffer size at most* n.

Proof:

The idea is that we route in 2 phases: first "horizontal" and then "vertical". Packages with farthermost destination have higher priority, so each node sends forward that package, whose destination node is located farthermost from this node. Due to this is routing time for each dimension at most $2(n-1)$.

The maximum buffer size is validated as follows. The most buffer size is needed in case, when during first phase of routing all n packages of one row are concentrated in one node to be then distributed between nodes of a row. In that case the node must be able to store in buffer n packages at the same time.

□

3 Digression: graph theory

In next sections we will use some theorems of graph theory. So we will present them in this section. First, we need some new definitions.

Definition 5 (Bipartite graph) *A graph $G = (V, E)$ is a bipartite graph with vertex classes V_1 and V_2 if $V = V_1 \bigcup V_2, V_1 \bigcap V_2 = \emptyset$ and every edge joins a vertex of V_1 to a vertex of V_2.*

Definition 6 (Hypergraph) *A hypergraph H is a pair (V,E) such that $V \bigcap E = \emptyset$ and $E \subset \mathfrak{P}(V)$. V is a set of nodes and E is a set of hyperedges.*
*H is **r-uniform** if all hyperedges connect exact r vertices.*
*H is **r-regular** if all vertices are connected to exact r hyperedges.*

Definition 7 (Matching) *Given a graph G = (V,E), a* ***matching*** *M in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. A* ***perfect matching*** *is a matching which covers all vertices of the graph. That is, every vertex of the graph is incident to exactly one edge of the matching.*

3.1 Marriage theorem

Theorem 3 (Frobenius/Hall, marriage theorem) *Let* $G = (V, E)$ *be a bipartite graph with* $V = V_1 \bigcup V_2$ *and* $|V_1| = |V_2|$*, then*

$$G \text{ contains a perfect matching } \Leftrightarrow \forall S \subset V_1 : |S| \leq |\Gamma_G(S)| (*)$$

Proof:
"$\Rightarrow$":
this is obvious.

"$\Leftarrow$":
In this proof we shall use the matchmaking terminology, so let V_1 be a set of girls and V_2 a set of boys. We shall apply induction on $m = |V_1| = |V_2|$.

For $m = 1$ the condition is clearly sufficient, so we assume that $m \geq 2$ and the condition is sufficient for smaller values of m.

Suppose first that any k girls ($1 \leq k < m$) know at least *k+1* boys. Then we arrange one marriage arbitrarily. The remaining (sets of) girls and boys still satisfy the condition (*), so the other *m-1* girls can be married off by induction.

Suppose now that for some k, $1 \leq k < m$, there are k girls who know exactly k boys altogether. These girls can clearly be married of by induction. What about the other girls? We can marry them off (again by induction) if they also satisfy the condition (*), provided that we do not count the boys who are already married. But the condition is satisfied, since if some l girls to be married know fewer than l remaining boys, than these girls together with the first k girls would know fewer than $k+l$ boys.

□

3.2 Coloring theorem

Using marriage theorem we can prove following theorem.

Theorem 4 (Coloring theorem) *Every bipartite n-regular graph can be decomposed into n perfect matchings, that is this graph is* ***n-edge-colorable.***

Proof:
In this proof we will apply induction on n.

For $n = 1$ we have a graph with every node incident to exact one edge. This graph is a matching and is obviously 1-edge-colorable.

Now let $n > 1$. We will show, that for $\forall X \subset V_1$: $|\Gamma(S)| \geq |S|$. S is incident to exact $|S| \cdot n$ edges. Each node of $\Gamma(S)$ can be an endvertex of at most n of this edges, so $|\Gamma(S)| \geq |S|$.

According to marriage theorem G has a perfect matching M. We color all edges of this matching in color *n-1*. Since $G \backslash M$ is a (n-1)-regular-graph, we can color it by induction in *n-1* colors.

□

We can also apply the marriage theorem on not regular graphs:

Corollary 1 *Each bipartite graph with maximum grad c can be decomposed into n perfect matchings and is c-edge-colorable.*

4 Permutation networks

Now we can construct some special permutation networks.

Definition 8 (Permutation network) *Let $n, d \in \mathbb{N}^+$. The (n,d)-**Permutation network** (n,d)-PN is a network with processors set $P = \{(l, a) | l \in \{-d, ..., -1, 1, ..., d\}, a \in [n]^d\}$ and communication graph (P, E),*

$$E = \{\{(l, \overline{a}), (l+1, \overline{a}')\} | l < -1, a_i = a_i' for \forall i \neq |l| - 1\}$$
$$\bigcup \{\{(l, \overline{a}), (l-1, \overline{a}')\} | l > 1, a_i = a_i' for \forall i \neq l - 1\}$$
$$\bigcup \{\{(-1, \overline{a}), (1, \overline{a}')\} | a_i = a_i' for \forall i \neq 0\}$$
$$\bigcup \{\{(l, \overline{a}), (l+1, \overline{a})\} | -d \leq l < d\}$$

*The **sources** are processors $\{Q_{\overline{a}} | Q_{\overline{a}} = (-d, \overline{a}), \overline{a} \in [n]^d\}$ and the **sinks** are processors $\{Q_{\overline{a}} | Q_{\overline{a}} = (d, \overline{a}), \overline{a} \in [n]^d\}$.*

We call $Q_{\overline{a}}$ a $(\overline{a})_n$-th source, sinks are called analogical.

*For $l \in \{-d, ..., -1, 1, ..., d\}$ the processors $\{(l, \overline{a}) | \overline{a} \in [n]^d\}$ are called processors on **level** l.*

One of the most famous and frequently used networks is the butterfly network.

Definition 9 (Butterfly network) *The **d-dimensional butterfly network** BF(d) comes up from (2,d)-PN, if the set of processors is reduced to $\{(l, \overline{a}) | l \in \{-1, 1, ..., d\}, \overline{a} \in [n]^d\}$. The processor -1 is then renamed to 0. If the nodes of level 0 are identified with those of level d, then such network is called a butterfly network with **wrap-around-edges**.*

The (2,d)-PN is also called *Beneš* network or **Waksman** network.

Properties of (n,d)-PN and BF(d):

Figure 2: (2,3)-PN

1. (n,d)-PN has $2d \cdot n^d$ processors;
2. (n,d)-PN has depth 2d-1;
3. let $t < d$, $\overline{b} \in [n]^{d-t}$ and $P_{\overline{b}} = \{(l, \overline{a}) | l \in \{-t, ..., -1, 1, ...t\}, \overline{a} = \overline{b}\overline{c} for \overline{c} \in [n]^t\}$. Then is $(n,d) - PN \mid_{P_{\overline{b}}} \cong (n,t) - PN$. This gives us a recursive decomposition of (n,d)-PN. For $i \in [n]$ and $t = d-1$ denote B^i the subnetwork $(n, d-1) - PN \mid_{P_i}$;
4. BF(d) has $(d+1) \cdot 2^d$ processors;
5. BF(d) has depth d;
6. for every processor $(0, \overline{a})$ and $(d, \overline{b})$ there is exact one shortest path from $(0, \overline{a})$ to $(d, \overline{b})$.

In following subsections we will discuss some routing algorithms for butterfly and (n,d) permutation networks.

4.1 Congestion in butterfly network

The **bit-reversal** permutation is a permutation that reverses the bits of a number: let $\overline{a} = (a_{d-1}, ...a_1, a_0)$, then $\pi(\overline{a}) = (a_0, a_1, ..., a_{d-1})$.

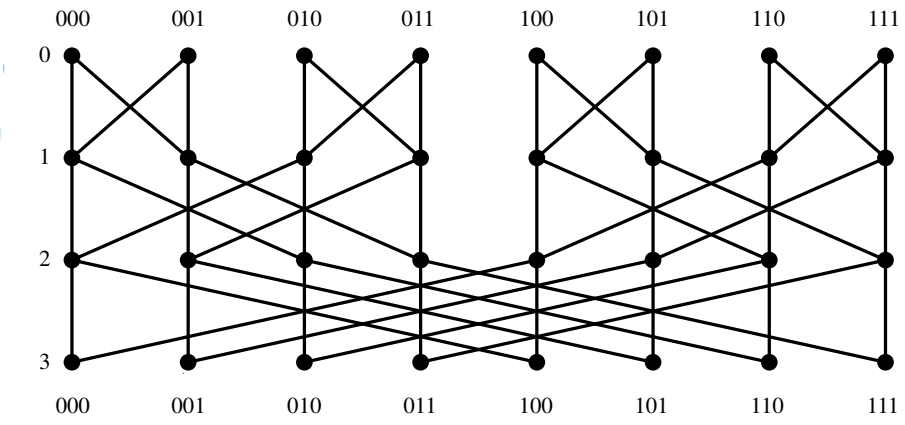

Figure 3: BF(3).

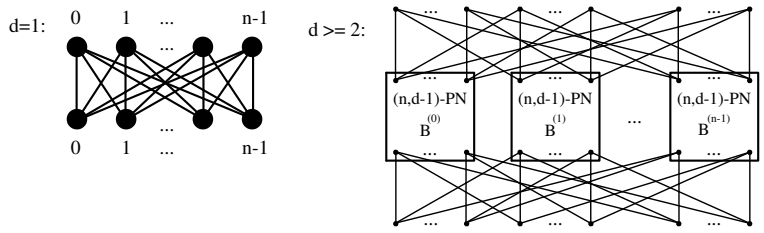

Figure 4: Recursive decomposition of PN

The routing of bit-reversal permutation as a good instance of worst case congestion (figure 5).

Red lines mark paths of packages from processors 0000, 0001, 0010, 0011. The red numbers near nodes denote the amount of packages that will be transferred through this nodes. The red marked node (2, 0000) is a **"hot spot"**, i.e. node with maximum congestion.

4.2 Offline routing in general case permutation network

First we will show that for any permutation there is a set of disjoint paths from sources to sinks, assuming that messages are to be transferred from sources to sinks.

Lemma 1 *For any permutation π on $[n]^d$ there is in (n,d)-PN a set W of n^d disjoint paths $W_{\overline{a}}$ beginning in $Q_{\overline{a}}$ and ending in $S_{\pi(\overline{a})}$, with length $2d-1$.*

Proof:

Let π be any permutation on $[n]^d$. The idea of this proof is that we construct a set of paths recursively using decomposition of (n,d)-PN described in feature nr. 3 of PN.

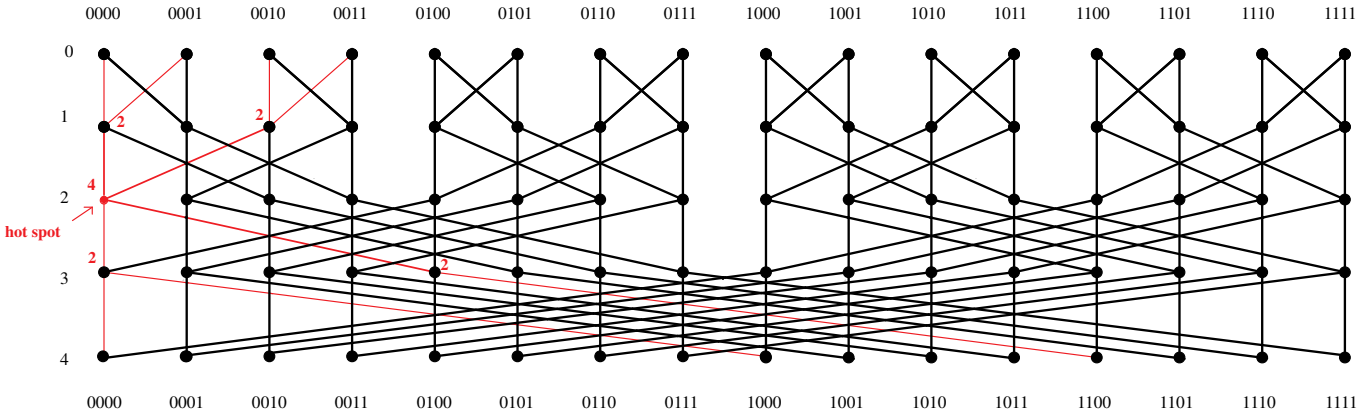

Figure 5: Routing of bit-reversal permutation on BF(4)

For $d = 1$ it is easy to find disjoint paths, because (n,1)-PN is a complete bipartite graph with n nodes in both parts. In that case set of paths $W = \{\{Q_{\overline{a}}, S_{\pi(\overline{a})}\} | \overline{a} \in [n]^d\}$. The length of each path is obvious $1 = 2 \cdot 1 - 1$.

Suppose now $d > 1$. For every path $W_{\overline{a}}$ we have to choose a partition $B^{(i)}$, through which this path will go, keeping in mind that following two conditions must be satisfied:

1. Any source of $B^{(i)}$ is incident to exact one path.
2. Any sink of $B^{(i)}$ is incident to exact one path.

If we could satisfy this conditions, then we can implement permutations on $[n]^{d-1}$ by induction.

Now consider a hypergraph $H_\pi = ([n]^d, E_1 \bigcup E_2)$, also called a **routing hypergraph** for π. H_π has following hyperedges:

- for $\forall \overline{b} \in [n]^{d-1}$ is a set of vertices $e_1(\overline{b}) = \{(j\overline{b}) | j \in [n]\}$ a type-1 hyperedge $e_1(\overline{b}) \in E_1$.
- for $\forall \overline{b} \in [n]^{d-1}$ is a set of vertices $e_2(\overline{b}) = \{\overline{a} \in [n]^d | (j\overline{b}) = \pi(\overline{a}), j \in [n]\}$ a type-2 hyperedge $e_2(\overline{b}) \in E_2$.

This hypergraph has paths $W_{\overline{a}}$ as nodes. Type-1 hyperedge contains the paths $W_{\overline{a}}$ whose sources $Q_{\overline{a}}$ are connected to the same sources of $B^{(i)}$. And type-2 hyperedge connects all the paths whose sinks are connected to the same sinks of partitions $B^{(i)}$. So what we need is to make sure that no two paths connected by type-1 or type-2 hyperedge are going through the same partition $B^{(i)}$. We can assign to each of n partitions $B^{(i)}$ one of n colors. If we could color the vertices of H_π in n colors, then we could lay any path through partition of same color, so both conditions would be satisfied.

Note that H_π can have multiple edges, because same vertices can be connected with both type-1 and type-2 hyperedges. H_π is n-uniform and 2-regular.

Let us show that H_π is n-colorable.

First we construct for $H_\pi = ([n]^d, E_1 \bigcup E_2)$ a bipartite graph $G_\pi = (E_1 \bigcup E_2, E_G)$. Note, that due to multiple edges of H_π some vertices in G_π can appear "double", as vertices from E_1 and as vertices from E_2. The nodes $e_1 \in E_1$ and $e_2 \in E_2$ are connected via one edge for every $v \in e_1 \bigcap e_2$ thus there are also multiple edges. Each vertex e of G_π has degree n, so G_π is n-regular. Now let $e \in E_1$, then there is for $\forall v \in e$ exact one $e^{(v)} \in E_2$ with $v \in e^{(v)}$. The same argumentation pertains to E_2.

G_π is bipartite and n-regular, and according to coloring theorem is n-edge-colorable. It means that H_π n-vertex-colorable is, because each edge in G_π corresponds to some vertex of H_π.

Let ϕ be some such n-coloring. We lay the path $W_{\overline{a}}$ through partition network $B^{(\phi(\overline{a}))}$. The coloring is a guarantee that there are no collisions and $B^{(\phi(\overline{a}))}$ have to implement only one permutation on $[n]^{d-1}$, what is possible by induction.

The length of path is $1 + 2(d-1) - 1 + 1 = 2d - 1$.

□

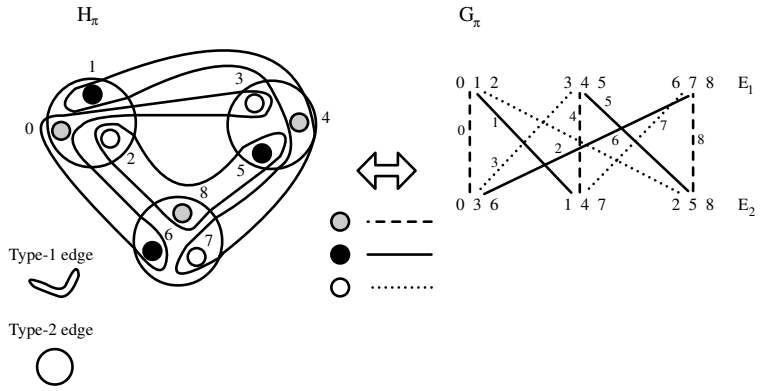

Figure 6: Example of H_π and G_π

Example:

Consider following permutation $\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 1 & 0 & 3 & 2 & 6 & 5 & 7 \end{pmatrix}$
which we want to route on (2,3)-PN. The corresponding graphs H_π and G_π are shown in figure 7, and a possible set of disjoint paths in figure 8.

The only thing we are missing now is some coloring algorithm for routing hypergraph. Such algorithms are expensive, if n is not a power of two.

Theorem 5 *(n,d)-PN can route with preprocessing according to any permutation π on $[n]^d$, sending messages from $Q_{\overline{a}}$ to $S_{\pi(\overline{a})}$ for $\forall \overline{a} \in [n]^d$. The routing time is $(2d-1) + (s-1)$, assuming that every message consists of* s *parts.*

Proof:

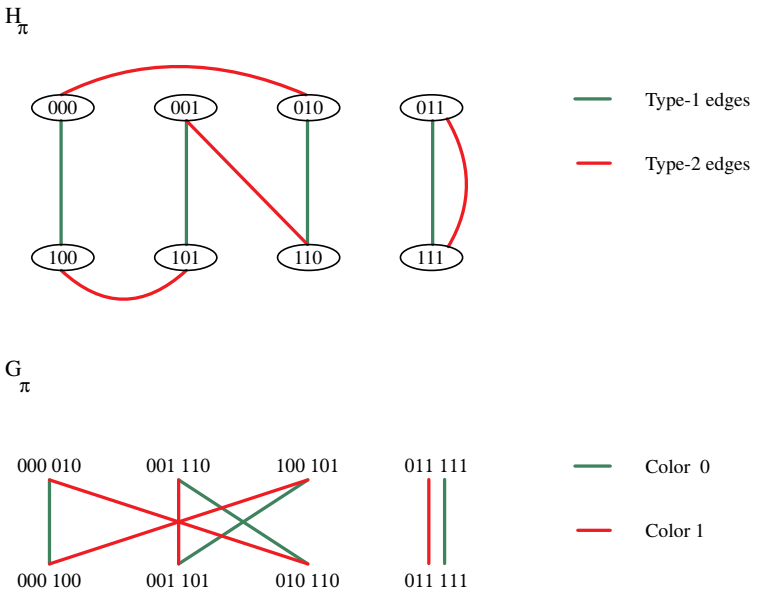

Figure 7: Concrete example of H_π and G_π

We construct the system of paths W according to proof of previous lemma. The s single messages will be transferred from $Q_{\overline{a}}$ to $S_{\pi(\overline{a})}$ through (n,d)-PN along the $W_{\overline{a}}$. There will be no congestion, because all pathes are disjoint, thus $2d + s - 2$ routing steps are needed. This process is called **pipelining**.

□

5 Conclusion

Unsolved problems

In theorem 6 we have proven that (2,d)-PN for all permutations $\pi : [2]^d \to [2]^d$ has a system of disjoint paths. This network consists of two copies of BF(d) connected inverse to each other. The question is, is it really necessary to connect BF networks in that way? It is possible to show, that 4 consecutively connected butterfly networks can route any permutation along disjoint paths. D. Parker has proven that also 3 copies are sufficient. It is easy to see, that one copy not enough is, but whether 2 copies are sufficient is unknown.

Some other communication problems

There are few more essential problems of communication in networks, as:

- The **distribution routing**, in which input packages can be sent to many destinations.

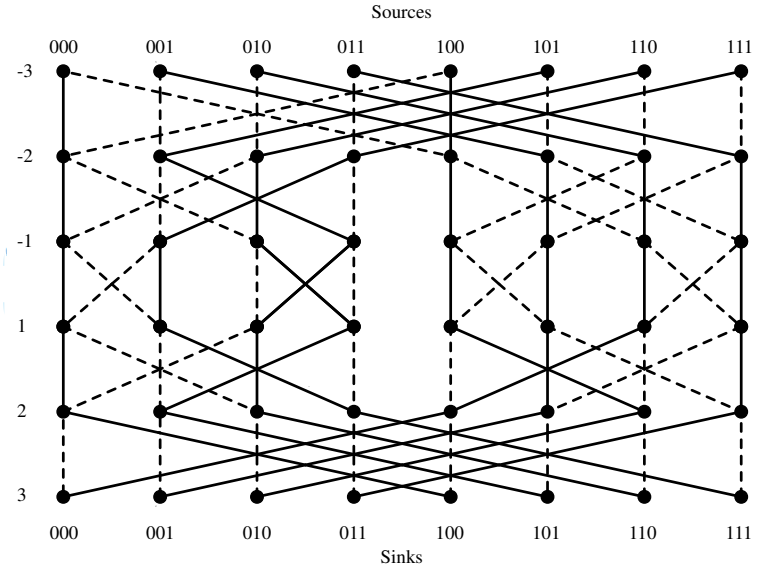

Figure 8: Example of a set of disjoint paths

- The **relation routing** which is kind of fusion of distribution routing and function routing.
- The **broadcasting**, where one of processors possesses a message which is to send to all another processors, and **gossipping**, which is a generalization of this process, so every processor has a message he must send to all another ("everyone becomes everything").
- The **token distribution** (also **load balancing**), where the packages (called **token**) have no destination address and are transmitted equally through entire network.

The most important feature of permutation routing is that many of above mentioned problems can be reduced on permutation routing or permutation routing is an important part of their solutions.

References

- Friedhelm Meyer auf der Heide. *Kommunikation in Parallelen Rechenmodellen.*
- B. Bollobăs. *Modern Graph Theory.*
- http://en.wikipedia.org