ALL-TO-ALL COMMUNICATION

- **All-to-all communication in a network:**
  - **All-to-all personalized exchange:**
    every node sends a distinct message to every other node.
  - **All-to-all broadcast:**
    every node sends the same message to all other nodes.
    Special case of all-to-all personalized exchange, with less time complexity.
- **Applications of all-to-all communication**
  - **Matrix multiplication, LU-factorization, Householder transformations, matrix transposition, and fast Fourier transform (FFT).**

All-to-all broadcast

- **Networks considered:**
  - **Hypercube networks:**
    shorter communication delay but poorer scalability
  - **Mesh and torus networks:**
    bounded node degree and better scalability
- **Assumptions:**
  - **Full-duplex communication channel**
  - **Each node has all-port capability**
  - **Messages broadcast are of the same length**
- **Goal:**
  Achieve maximum degree of parallelism in message transmission.

Previous Related Work:

All-to-all broadcast algorithms in a torus:

- **Saad and Schultz:**
  Each node sends the message along horizontal ring, and then vertical ring.
- **Calvin, Perennes, and Trystram:**
  Recursive algorithm.

In these algorithms, some degree of parallelism was achieved among different nodes, but no time overlap for the messages passing through the same node.

The new algorithm:

- **Further explores the parallelism in message transmission and achieves some degree of parallelism among the messages passing through the same node**
- **Optimal in message transmission time**
- **Total communication delay close to the lower bound of all-to-all broadcast**
- **Conceptually simple, and symmetrical for every message and every node**
- **Easy implementation in hardware**

Properties of 2D Meshes and Tori

- **Each node** (x, y) **has up to four neighbors:** $(x-1, y)$**,** $(x+1, y)$**,** $(x, y-1)$**, and** $(x, y+1)$
- **Distance between** $P_1 = (x_1, y_1)$ **and** $P_2 = (x_2, y_2)$**:** $dist(P_1, P_2)$
  - **Mesh:**

$$dist(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$$

  - **Torus:**

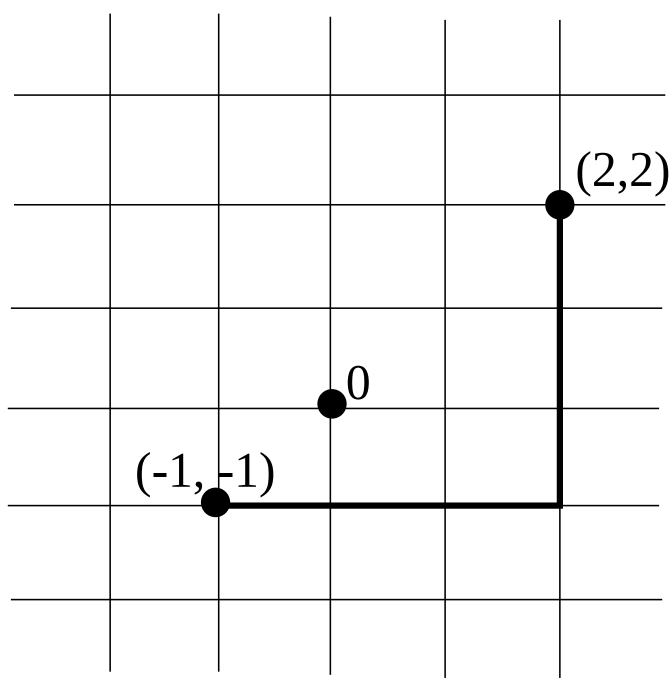
$$\begin{aligned} dist(P_1, P_2) &= \min\{|x_1 - x_2|, n - |x_1 - x_2|\} \\ &+ \min\{|y_1 - y_2|, n - |y_1 - y_2|\} \end{aligned}$$

- **d-neighbor:**

  The distance between two nodes equals d.

- **Example:**

  Node $(-1,-1)$ and node $(2,2)$ in a mesh are 6-neighbors



- **Diameter:**
  Maximum distance between any two nodes in the network

  - $n \times n$ **mesh:** $2(n-1)$
  - $n \times n$ **torus:** $2\lfloor \frac{n}{2} \rfloor$

- **Circle centered at a node:**
  The set of nodes with an equal distance to the node

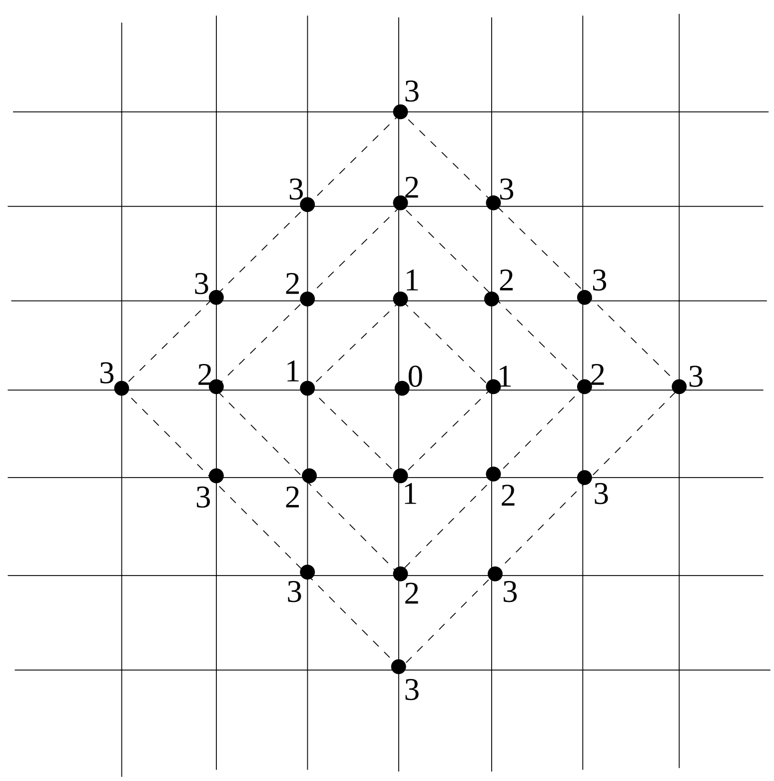
- **Radius of circle:**
  The distance to the center

- **Perimeter of circle:**
  Cardinality of the node set of the circle

- **In an infinite mesh, the perimeter of a circle with radius d is**

$$C_d = \begin{cases} 1 & \textbf{if } d = 0 \\ 4d & \textbf{if } d \geq 1 \end{cases}$$

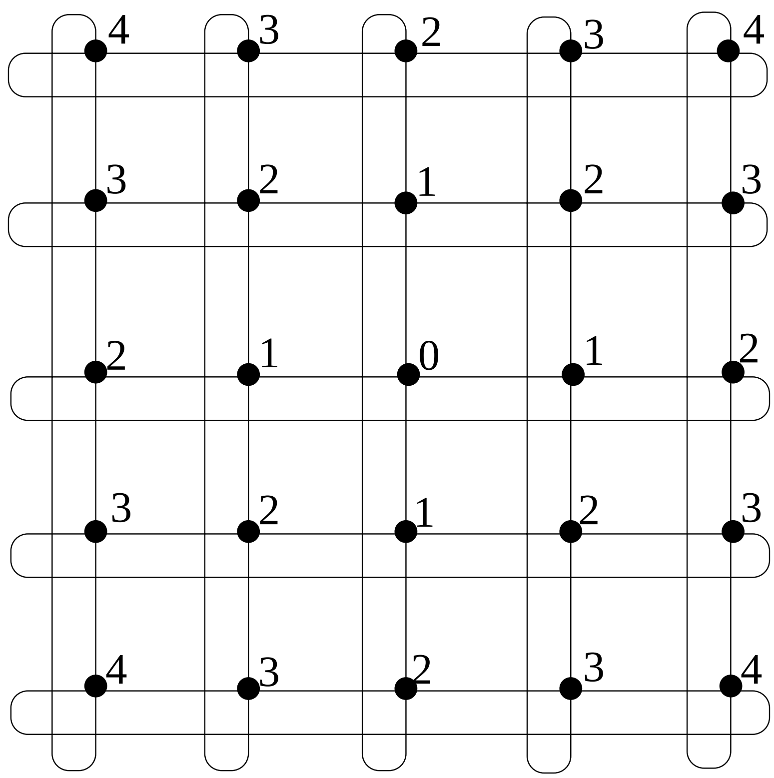
- **Circles centered at the same node with different radii in a mesh**



- **In a $(2k+1) \times (2k+1)$ torus, the perimeter of a circle with radius d, where $0 \leq d \leq 2k$, is**

$$C_d = \begin{cases} 1 & \textbf{if } d = 0 \\ 4d & \textbf{if } 1 \leq d \leq k \\ 4(2k+1-d) & \textbf{if } k+1 \leq d \leq 2k \end{cases}$$

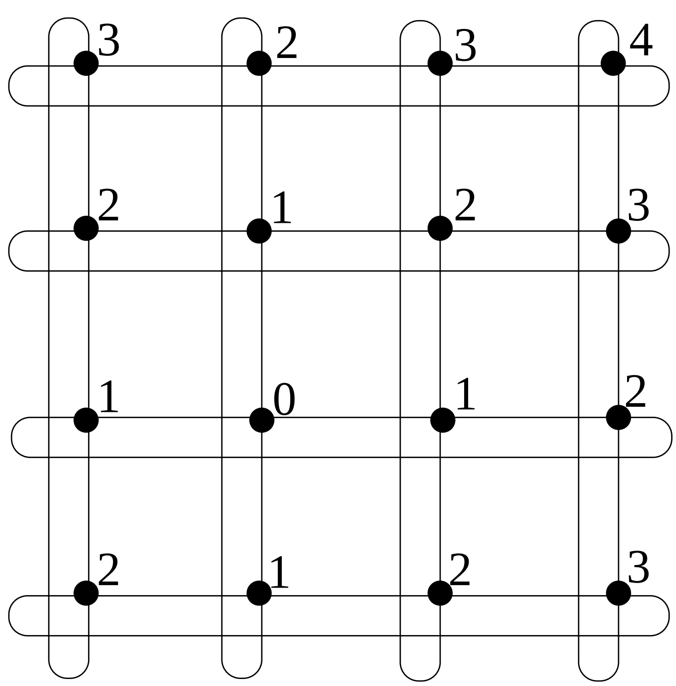
- **Circles centered at a node with different radii in a 5×5 torus**



- **In a $2k \times 2k$ torus, the perimeter of a circle with radius d, where $0 \le d \le 2k$, is**

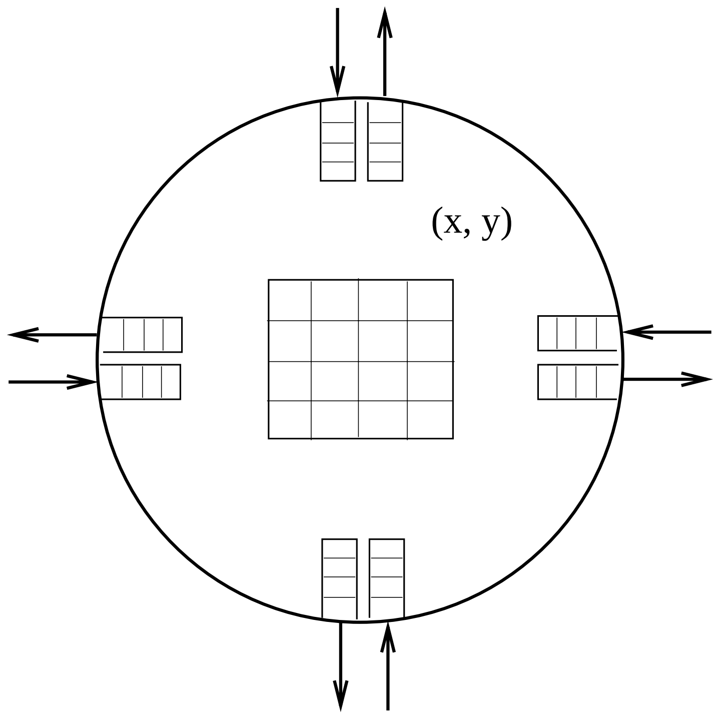
$$C_d = \begin{cases} 1 & \textbf{if } d = 0 \\ 4d & \textbf{if } 1 \le d \le k-1 \\ 4k-2 & \textbf{if } d = k \\ 4(2k-d) & \textbf{if } k+1 \le d \le 2k-1 \\ 1 & \textbf{if } d = 2k \end{cases}$$

- **Circles centered at a node with different radii in a 4×4 torus**



Broadcasting in a Mesh or Torus

- **Buffer structure of a node:**



- **Logical format of a message:**

Source node address	Message content

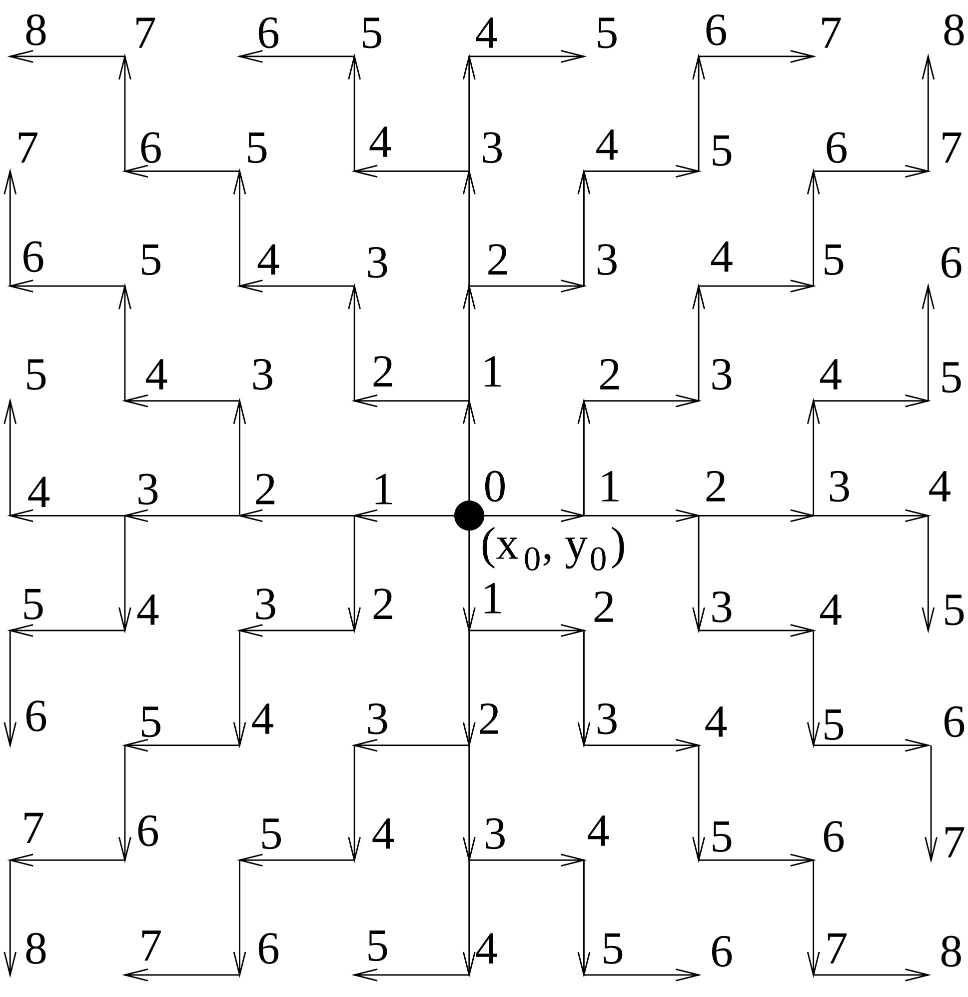
- **Lower bound for broadcast**
  - α: **startup time**
  - δ: **switching time**
  - γ: **transmission time**
  - L: **the number of bytes per message**

Theorem 1 ***(1) The maximum communication delay of one-to-all broadcast is at least*** $\alpha+\delta+2(n-1)L\gamma$ ***in an*** $n\times n$ ***mesh, and is at least*** $\alpha+\delta+2\lfloor\frac{n}{2}\rfloor L\gamma$ ***in an*** $n \times n$ ***torus. (2) The maximum communication delay of all-to-all broadcast in all-port model is at least*** $\alpha+\delta+\frac{n^2-1}{2}L\gamma$ ***in an*** $n \times n$ ***mesh, and is at least*** $\alpha+\delta+\frac{n^2-1}{4}L\gamma$ ***in an*** $n \times n$ ***torus.***

Broadcast Pattern

- **Basic idea:**
  - **Controlled message flooding**
  - **Use a specially designed spanning tree rooted at the source node (broadcast pattern)**
  - **Logical broadcast phases:**
    In phase d, the message originating from a node reaches all its d-neighbors.
- **Consider an infinite mesh first**

- **Broadcast pattern from source node** (x_0, y_0)



- **Formal description of broadcast pattern**
  - **Define three functions:**

$$U(x) = \begin{cases} 1 & \textbf{if } x > 0 \\ 0 & \textbf{if } x < 0 \end{cases}$$

$$I(x) = \begin{cases} 1 & \textbf{if } x > 0 \\ -1 & \textbf{if } x < 0 \end{cases}$$

$$mod_2(x) = x \bmod 2$$

  - **When a broadcast message originating from node (x_0, y_0) reaches node (x, y), it continues to broadcast to the neighbors of node (x, y) as follows.**

Case 1: $x = x_0$ **and** $y = y_0$**.**

(x, y) **broadcasts the message to all of its four neighbors** $(x, y+1)$, $(x, y-1)$, $(x+1, y)$, **and** $(x-1, y)$**.**

Case 2: Either $x = x_0$ **or** $y = y_0$ **but not both.**

(x, y) **multicasts the message to its two neighbors** (x_1, y_1) **and** (x_2, y_2)

2.1: $x = x_0$ **and** $y \neq y_0$**, i.e. along the** y**-axis.**

x_1, y_1, x_2, **and** y_2 **satisfy**

$$
\begin{aligned}
x_1 &= x + mod_2(y - y_0 + U(I(y - y_0))) \\
y_1 &= y + I(y - y_0) \times mod_2(y - y_0 + 1 - U(I(y - y_0))) \\
x_2 &= x - mod_2(y - y_0 + U(-I(y - y_0))) \\
y_2 &= y + I(y - y_0) \times mod_2(y - y_0 \\
&\quad +1 - U(-I(y - y_0)))
\end{aligned}
$$

2.2: $x \neq x_0$ **and** $y = y_0$**, i.e. along the** x**-axis.**

x_1, y_1, x_2, **and** y_2 **satisfy**

$$
x_1 = x + I(x - x_0) \times mod_2(x - x_0 + U(I(x - x_0)))
$$

$$
\begin{aligned}
y_1 &= y + mod_2(x - x_0 + 1 - U(I(x - x_0))) \\
x_2 &= x + I(x - x_0) \times mod_2(x - x_0 + U(-I(x - x_0))) \\
y_2 &= y - mod_2(x - x_0 + 1 - U(-I(x - x_0)))
\end{aligned}
$$

Case 3: $x \neq x_0$ **and** $y \neq y_0$**.**

(x, y) **sends the message to its neighbor** (x_3, y_3) **where** x_3 **and** y_3 **satisfy**

$$
\begin{aligned}
x_3 &= x + I(x - x_0) \times mod_2((x - x_0) + (y - y_0) + \\
&\quad U(I(x - x_0) \times I(y - y_0))) \\
y_3 &= y + I(y - y_0) \times mod_2((x - x_0) + (y - y_0) + \\
&\quad 1 - U(I(x - x_0) \times I(y - y_0)))
\end{aligned}
$$

Case 3 can be viewed as a generic form for all cases if letting $I(0)$ **take** 1 **and** -1 **respectively.**

- **Observations:**
  - **Message is sent out in the direction leaving the origin**
  - **Message is broadcast to all its four neighbors from a node with the same x-coordinate and y-coordinate as the origin**
  - **Message is multicast to its two neighbors from a node with the same x-coordinate or y-coordinate as the origin**
  - **Message is sent to only one of its neighbors from a node with a different x-coordinate and y-coordinate from the origin**

Theorem 2 ***Each node in an infinite mesh can be reached exactly once by using the broadcast pattern.***

- **For a mesh, the broadcast pattern is trimmed at the boundary of the mesh**
- **For a torus, perform additional check:**

Additional check for a torus:

For any node (x_1, y_1) which is a neighbor of node (x, y) chosen from the broadcast pattern,

if $dist((x_0, y_0), (x, y)) \geq dist((x_0, y_0), (x_1, y_1))$
then remove (x_1, y_1) from the list of the neighbors to which node (x, y) relays the message originating from node (x_0, y_0).

All-to-All Broadcast Algorithm

- **Buffers are used to resolve the possible contention of multiple messages**
- **Higher level description of the algorithm**

All-to-All Broadcast Algorithm:

for each node (x, y) in the network **do in parallel**
  for each input and output buffer **do in parallel**
    case of an output buffer:
      repeat
        remove a message from the buffer;
        send it to corresponding neighboring node
          (enter into the input buffer of the neighbor);
      until no more message arrives at the buffer;
    end case;
    case of an input buffer:
      repeat
        remove a message from the buffer;
        extract source address of the message, say, (x_0, y_0);
        copy the message content to (x_0, y_0) entry of the
          buffer matrix;
        calculate the addresses of the neighbors to be
          multicast by using the broadcast pattern;
        for a torus, perform the additional check in Table 1;
        multicast the message to the output buffers
          connected to the corresponding neighbors;
      until no more message arrives at the buffer;
    end case;
  end for;
end for;
End

- **Observations:**
  - **Each node is continuously receiving (and sending) messages from (and to) its four neighbors**
  - **In each buffer, messages in earlier phase d' always arrive before those in a later phase d'', for any $d' < d''$**
  - **In phase d, the message originating from a node reaches all its d-neighbors, and the messages originating from all d-neighbors of a node reach the node**
- **Question:**

  Whether the incoming (and outgoing) messages are uniformly distributed to the four input (and output) buffers at each node

Theorem 3 ***According to the broadcast pattern, in phase d of all-to-all broadcast in an infinite mesh, each node (x_0, y_0) receives $4d$ messages originating from all its $4d$ d-neighbors via the four input channels of node (x_0, y_0), with d messages from each input channel, and at the end of the phase, it will send $4(d+1)$ messages to its four 1-neighbors via the four output channels, with $d+1$ messages on each output channel.***

Proof. By induction on phase d.

- **Need to show in phase k the $4(k+1)$ outgoing messages are uniformly distributed to the four output buffers at node (x_0, y_0).**
- **Consider the $4k$ k-neighbors of node (x_0, y_0). They can be grouped into four groups:**

  Group I: $(x_0 + l, y_0 + k - l)$ **for** $0 \leq l \leq k$

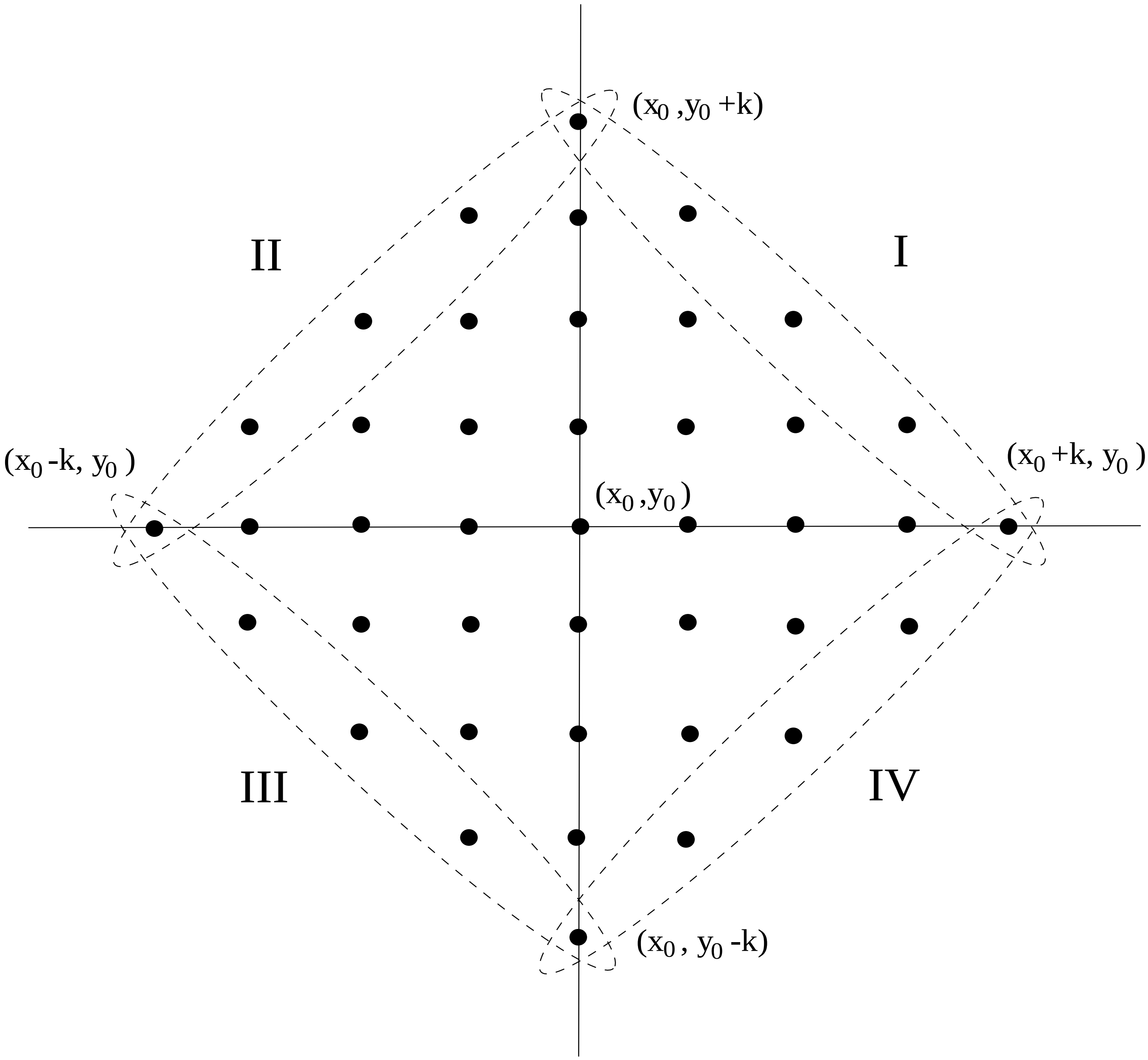
  Group II: $(x_0 - k + l, y_0 + l)$ **for** $0 \leq l \leq k$

  Group III: $(x_0 - k + l, y_0 - l)$ **for** $0 \leq l \leq k$

  Group IV: $(x_0 + l, y_0 - k + l)$ **for** $0 \leq l \leq k$

  Four special nodes (x_0, y_0+k), (x_0-k, y_0), (x_0, y_0-k) and (x_0+k, y_0) each is counted twice and thus each group has $k+1$ nodes.

$4k$ k-neighbors of node (x_0, y_0) $(k = 4)$:



By the broadcast pattern, the following 1-neighbors of (x_0, y_0) are chosen to relay messages from Groups I, II, III, and IV, respectively.

$$\begin{cases} (x_0 - 1, y_0) & \textbf{if } k \textbf{ is even} \\ (x_0, y_0 - 1) & \textbf{if } k \textbf{ is odd} \end{cases}$$

$$\begin{cases} (x_0, y_0 - 1) & \textbf{if } k \textbf{ is even} \\ (x_0 + 1, y_0) & \textbf{if } k \textbf{ is odd} \end{cases}$$

$$\begin{cases} (x_0 + 1, y_0) & \textbf{if } k \textbf{ is even} \\ (x_0, y_0 + 1) & \textbf{if } k \textbf{ is odd} \end{cases}$$

$$\begin{cases} (x_0, y_0 + 1) & \textbf{if } k \textbf{ is even} \\ (x_0 - 1, y_0) & \textbf{if } k \textbf{ is odd} \end{cases}$$

Corollary 1 ***In all-to-all broadcast using the algorithm in Table 2 in an*** $n \times n$ ***torus, each channel between two adjacent nodes in the network relays*** $\frac{n^2-1}{4}$ ***messages.***

Proof. The total number of messages through a single channel is

$$\sum_{d=1}^{2\lfloor \frac{n}{2} \rfloor} \frac{C_d}{4} = \frac{1}{4}\left(\sum_{d=0}^{2\lfloor \frac{n}{2} \rfloor} C_d - 1\right) = \frac{n^2-1}{4}$$

Corollary 2 ***In all-to-all broadcast using the algorithm in Table 2 in an*** $n \times n$ ***mesh, each channel between two adjacent nodes in the network relays at most*** $\frac{n^2-1}{2}$ ***messages.***

Delay Analysis in All-to-All Broadcast

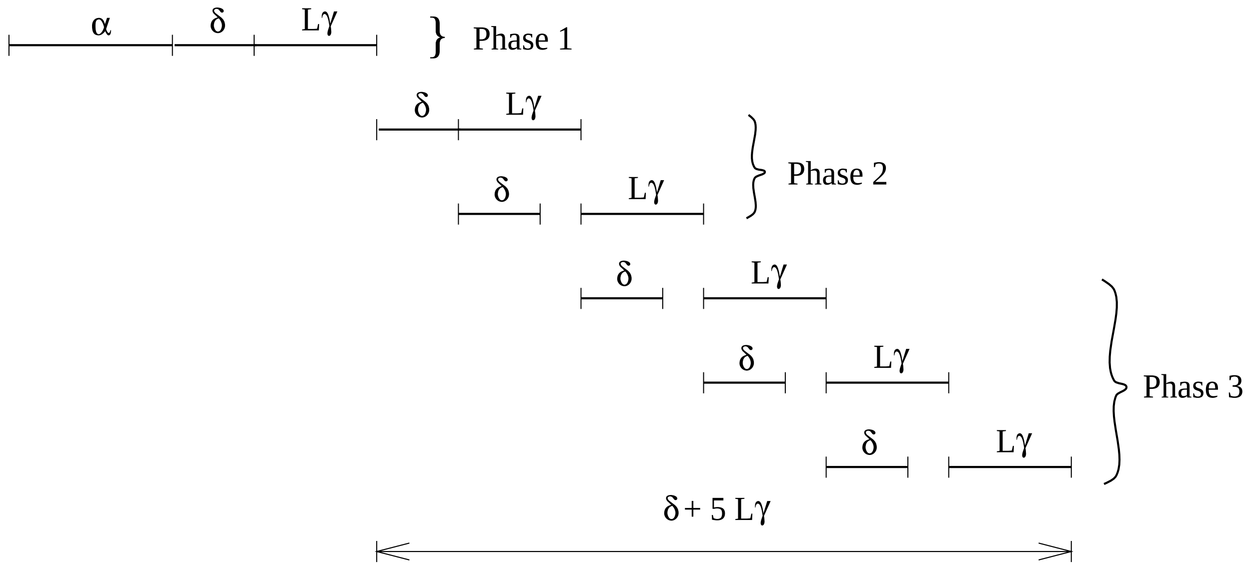
- **Switching time overlapped with transmission time**
- **Messages in any direction are pipelined**

Theorem 4 ***The total delay of the all-to-all broadcast algorithm for an*** $n \times n$ ***torus is no more than***

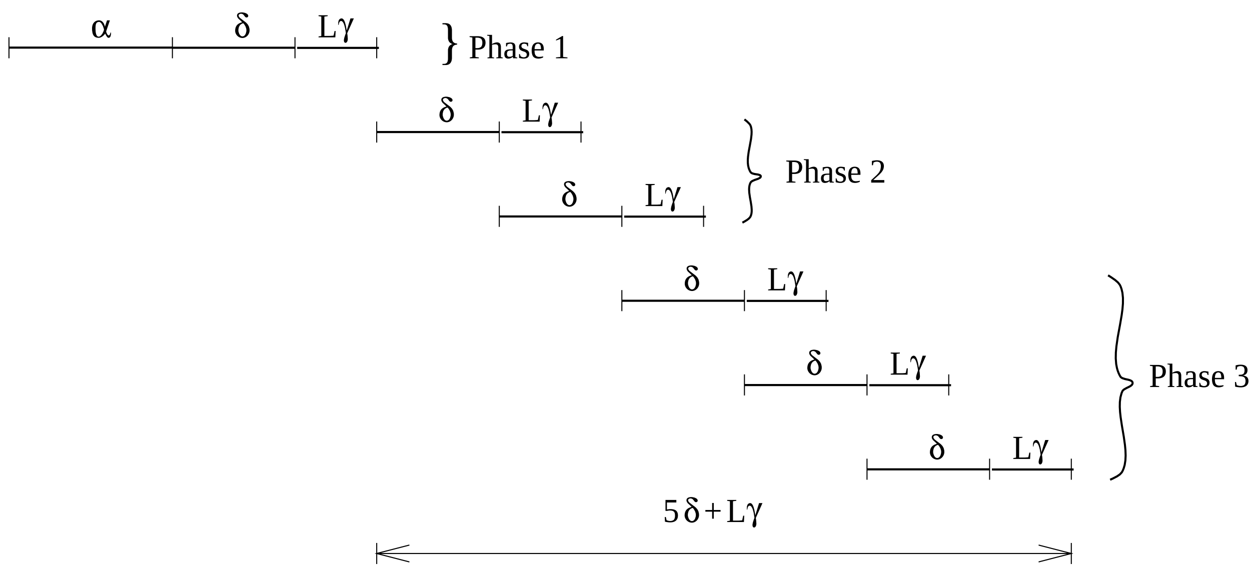
$$\alpha + \delta + L\gamma + \left(\frac{n^2 - 1}{4} - 1\right)\max\{\delta, L\gamma\} + \min\{\delta, L\gamma\}$$

where α**,** δ**,** γ **and** L **are the startup time, the switching time, the transmission time per byte, and the number of bytes per message, respectively.**

The total delay of the first three phases of all-to-all broadcast. (a) $\delta \leq L\gamma$. (b) $\delta > L\gamma$.



(a)



(b)

Corollary 3 ***The total delay of the all-to-all broadcast algorithm for an*** $n \times n$ ***torus is no more than***

$$\begin{cases} \alpha + 2\delta + \left(\frac{n^2-1}{4}\right) L\gamma & \textbf{\textit{if}}\ \delta \leq L\gamma \\ \alpha + 2L\gamma + \left(\frac{n^2-1}{4}\right) \delta & \textbf{\textit{if}}\ \delta > L\gamma \end{cases}$$

- **Observations:**
  - **The new algorithm is optimal in transmission time**
  - **The total communication delay is close to the optimal value within only a small constant δ, for $\delta \leq L\gamma$.**
  - **Easily implemented in hardware so that the algorithm can achieve the optimum in practice.**

Comparisons

- **Saad and Schultz[12]'s horizontal/vertical algorithm for a torus:**
  $2\alpha + 2\lfloor \frac{n}{2} \rfloor \delta + (\frac{n^2-1}{2}) L\gamma$.
- **Calvin, Perennes,and Trystram[15]'s recursive algorithm for a torus:**
  $2 \log_5(n^2)\alpha + \frac{3}{2}(n-1)\delta + \frac{n^2-1}{2} L\gamma$.
- **The transmission time of the both algorithms is twice as long as the new algorithm.**
- **Both algorithms cannot take advantage of overlapping of the switching time and the transmission time as does the new algorithm.**
- **When $\delta > 2L\gamma$, both algorithms take less time than the proposed algorithm. However, in general we can have $\delta < L\gamma$.**

Summary:

A new all-to-all broadcast algorithm in all-port 2D mesh and torus networks.

- **The algorithm can be implemented in either packet-switched networks or virtual cut-through and wormhole-switched networks.**
- **The new algorithm utilizes a controlled message flooding based on a specially designed broadcast pattern.**
- **The new algorithm takes advantage of overlapping of message switching time and transmission time, and achieves optimal transmission time for all-to-all broadcast.**

- **In most cases, the total communication delay is close to the lower bound of all-to-all broadcast within a small constant range.**
- **The algorithm is conceptually simple, and symmetrical for every message and every node, so that it can be easily implemented in hardware and achieves the optimum in practice.**
- **The algorithm can also be extended to multi-dimensional meshes and tori.**

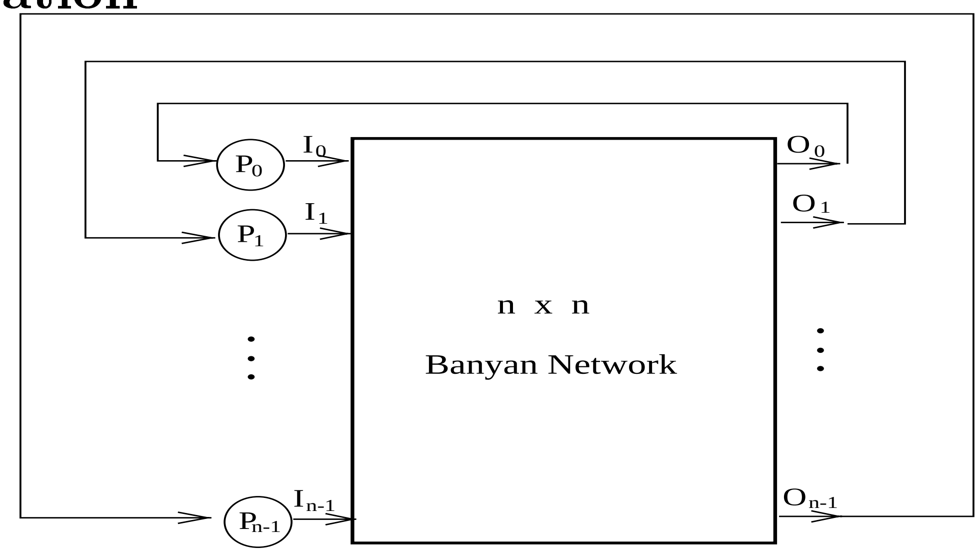
All-to-all personalized exchange

Previous work

- **All-to-all personalized exchange in an n-node hypercube**
  - **One-port model: time $O(n \log n)$**
  - **All-port model: time $O(n)$**
- **All-to-all personalized exchange in an n-node mesh/torus network**
  - **Two dimensional: time $O\left(n^{\frac{3}{2}}\right)$**
  - **k dimensional: time $O\left(n^{\frac{k+1}{k}}\right)$**
- **Drawback:**
  - **Hypercubes: poor scalability due to the unbounded node degrees**
  - **Meshes and tori: longer communication delay due to the limitations of the topologies**

All-to-all personalized exchange in multistage interconnection network

- **Given n processors $P_0, P_1, \ldots, P_{n-1}$, an $n \times n$ MIN can be used for inter-processor communication**

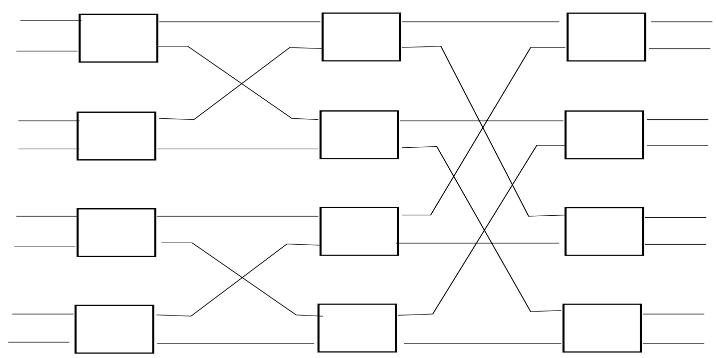


  - **Crossbar: hardware cost is too high**
  - **Benes network: not every permutation is easily routable**
  - **Banyan network: self-routing unique path network, fast switch setting: $O(n)$ time for all-to-all personalized exchange, single input/output port per processor.**

Network Structure and Permutation

- **An $n \times n$ banyan network composed of $\log n$ stages of 2×2 switches**

  E.g. An 8×8 banyan network



- **A permutation is a full one-to-one mapping between the network inputs and outputs.**

- **A permutation in $n \times n$ network denoted as**

$$\rho = \begin{pmatrix} 0 & 1 & \ldots & n-1 \\ a_0 & a_1 & \ldots & a_{n-1} \end{pmatrix}$$

  where $a_i \in \{0, 1, \ldots, n-1\}$ for $0 \le i \le n-1$, and $a_i \neq a_j$ for $i \neq j$

- **Identity permutation is denoted as I**

- **Each stage can be considered as a shorter $n \times n$ network**
- **Each set of interstage links can also be as a shorter $n \times n$ network**
- σ_i**: the stage permutation of stage** i
- τ_i**: the interstage permutation between stage** i **and stage** $i+1$**, expressed as the following mapping:**

$$p_{m-1}p_{m-2}\cdots p_{i+2}p_{i+1}p_i\cdots p_1p_0 \xrightarrow{\tau_i}$$
$$p_{m-1}p_{m-2}\cdots p_{i+2}p_0p_i\cdots p_1p_{i+1},$$

i.e., the function of swapping bit 1 **for bit** $i+2$

- **The overall permutation of a banyan network is the composition:**

$$\sigma_{m-1}\tau_{m-2}\sigma_{m-2}\cdots\tau_0\sigma_0$$

Realizing All-to-All Personalized Exchange in Banyan Networks

- **Lower Bound for All-to-All Personalized Exchange**
  - **The maximum communication delay of all-to-all personalized exchange in an $n \times n$ banyan network of $\log n$ stages is at least $\Omega(n + \log n)$.**
- **All-to-All Personalized Exchange Algorithm Using a Latin Square**
  - **A Latin square is defined as an $n \times n$ matrix**

$$\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{bmatrix}$$

in which the entries $a_{i,j}$'s are numbers in $\{0, 1, 2, \ldots, n-1\}$ and no two entries in a row (or a column) have the same value.

All-to-all personalized exchange algorithm (ATAPE) f

begin

Step 1. for each processor j $(0 \le j \le n-1)$

  do in parallel

   1.1 for each $a_{i,j}$ $(0 \le i \le n-1)$ **in the**

    Latin square do in sequential

     prepare a personalized message

     from processor j **to processor** $a_{i,j}$**;**

     insert the message into the

     message queue j**;**

Step 2. for each processor j $(0 \le j \le n-1)$

  do in parallel

   2.1 for each message with destination

    address $a_{i,j}$ $(0 \le i \le n-1)$ **in the**

    message queue j **do in sequential**

     send the message destined to $a_{i,j}$

     through input j **of the network;**

end;

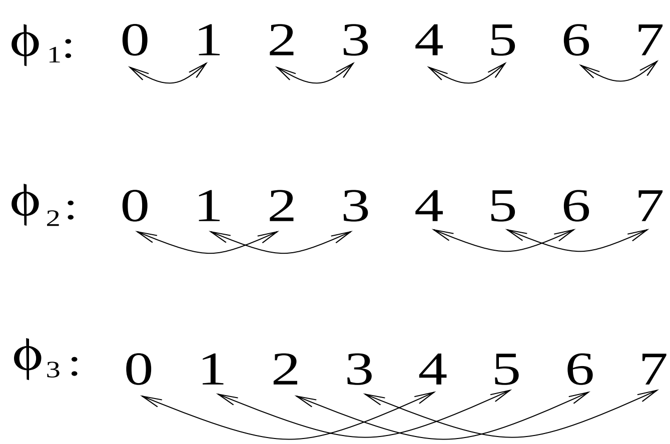
Time complexity of ATAPE: $O(n + \log n)$

Two Methods for Constructing a Latin Square

- **A set of basic permutations** ϕ_i $(1 \leq i \leq m)$
  - **Definition:**

$$p_{m-1}p_{m-2}\cdots p_i p_{i-1} p_{i-2}\cdots p_1 p_0 \xrightarrow{\phi_i}$$
$$p_{m-1}p_{m-2}\cdots p_i \bar{p}_{i-1} p_{i-2}\cdots p_1 p_0$$

  - **Properties:**
    * $\phi_i \phi_j = \phi_j \phi_i$, **for** $1 \leq i, j \leq m$
    * $\phi_i \phi_i = I$, **for** $1 \leq i \leq m$
  - **Example: Basic permutations for an** 8×8 **mapping. Each arc represents a mapping between two numbers**



- **The first construction of a Latin square**
  - **Given** m **basic permutations** $\phi_1, \phi_2, \ldots, \phi_m$**, construct** $\Psi = \{\phi_{i_1}\phi_{i_2}\cdots\phi_{i_k} | m \geq i_1 > i_2 > \cdots > i_k \geq 1$ **and** $1 \leq k \leq m\}$
  - **Note that** $|\Psi| = n - 1$**.**
  - **Let** $\rho_1, \rho_2, \ldots, \rho_{n-1}$ **be the** $n-1$ **permutations in** Ψ**, and** $a_0, a_1, \ldots, a_{n-1}$ **be a list of numbers such that** $\{a_0, a_1, \ldots, a_{n-1}\} = \{0, 1, \ldots, n-1\}$**. Then the following matrix is a Latin square.**

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ \rho_1(a_0) & \rho_1(a_1) & \rho_1(a_2) & \cdots & \rho_1(a_{n-1}) \\ \rho_2(a_0) & \rho_2(a_1) & \rho_2(a_2) & \cdots & \rho_2(a_{n-1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho_{n-1}(a_0) & \rho_{n-1}(a_1) & \rho_{n-1}(a_2) & \cdots & \rho_{n-1}(a_{n-1}) \end{bmatrix}$$

- **The second construction of a Latin square**
  - **An algorithm to build the Latin square row by row in an iterative fashion**
  - **Minimum complexity** $O(n^2)$
  - **The algorithm description**

Algorithm LatinSquare (List $\{a_0, a_1, \ldots, a_{n-1}\}$**) /*main*/**

begin

  List $BL \leftarrow$ **List** $\{\}$**;**

  BuildBasicList(m**); /*** $m = \log n$ ***/**

  BuildLatinSquare(BL**,** $\{a_0, a_1, \ldots, a_{n-1}\}$**);**

end;

Function BuildBasicList(int k**)**

begin

  if ($k == 1$**)**

    BL.append(ϕ_1**);**

    return;

  end if

  BuildBasicList($k-1$**);**

  BL.append(ϕ_k**);**

  BuildBasicList($k-1$**);**

end;

Function BuildLatinSquare(List $\{\phi_{k_1}, \phi_{k_2}, \ldots, \phi_{k_{n-1}}\}$**,**

    List $\{a_0, a_1, \ldots, a_{n-1}\}$**)**

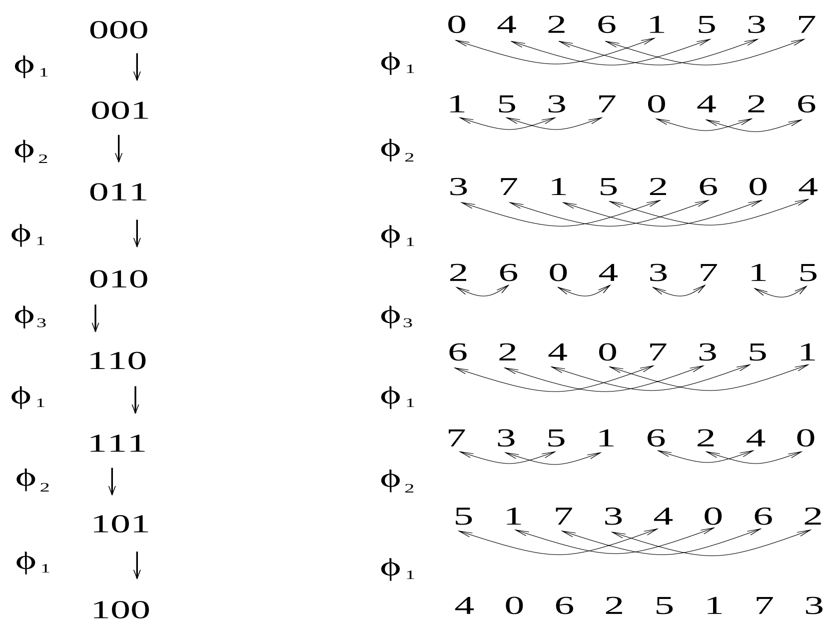
begin

  for $i = 0$ **to** $n-1$ **do**

- **The matrix generated by the second construction is a Latin square**
- **An example of the second construction**

  (a) The 3-bit Gray code sequence generated by applying the basic permutation list to number 0.

  (b) An 8×8 Latin square generated by the algorithm



(a) (b)

- **The Latin squares generated by the two constructions are equivalent.**

Generating Permutations for All-to-All Personalized Exchange in Banyan Networks

- **Generate admissible permutations to form the Latin square needed in algorithm ATAPE**
- **Method: simply let each stage permutation** $\sigma_i = \phi_1$ **or** I**.**
- **Proof sketch:**

  Let $\tau = \tau_{m-2}\tau_{m-3}\ldots\tau_1\tau_0$

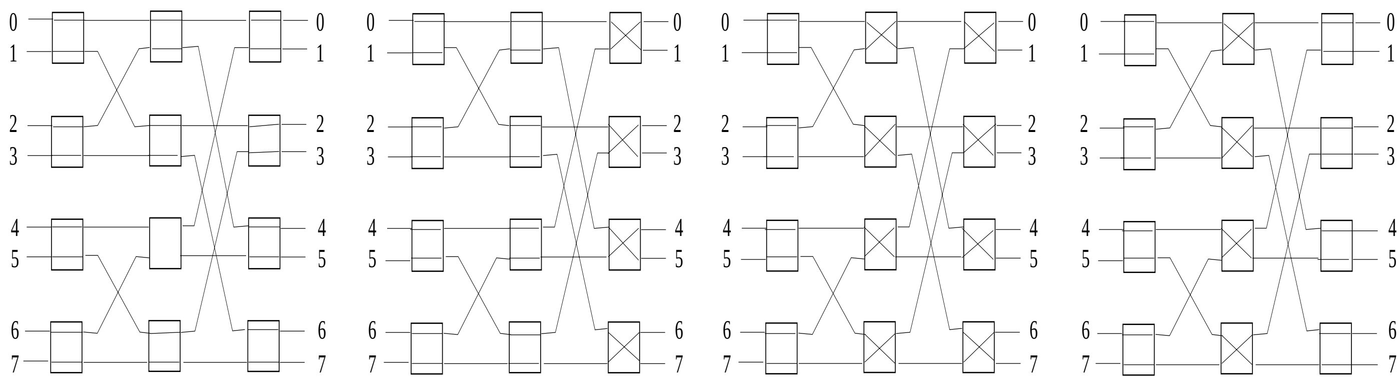
  Apply $(\tau_{m-2}\tau_{m-3}\cdots\tau_{m-i-1})\phi_1 = \phi_{m-i+1}(\tau_{m-2}\tau_{m-3}\cdots\tau_{m-i-1})$

  All such admissible permutations form a Latin square as that in the first construction method.
- **By the equivalence of the two constructions, the Latin square can be constructed by simply using the second algorithm** $LatinSquare($**List** $\{\tau(0), \tau(1), \ldots, \tau(n-1)\})$

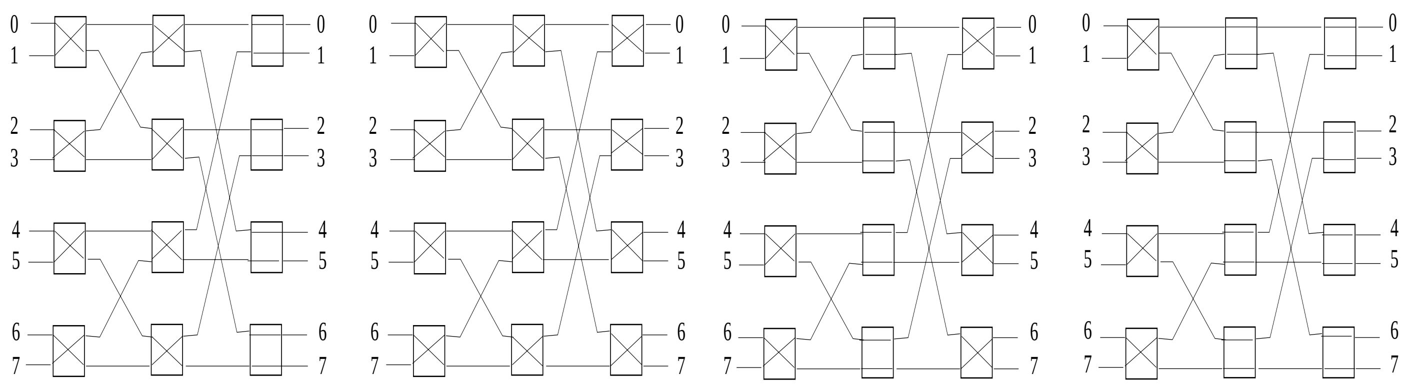
- **An example of 8×8 banyan network**
  - **The generated Latin square**

$$\begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\ 1 & 3 & 5 & 7 & 0 & 2 & 4 & 6 \\ 3 & 1 & 7 & 5 & 2 & 0 & 6 & 4 \\ 2 & 0 & 6 & 4 & 3 & 1 & 7 & 5 \\ 6 & 4 & 2 & 0 & 7 & 5 & 3 & 1 \\ 7 & 5 & 3 & 1 & 6 & 4 & 2 & 0 \\ 5 & 7 & 1 & 3 & 4 & 6 & 0 & 2 \\ 4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \end{bmatrix}$$

  - **All possible switch settings, in which each stage is set to either ϕ_1 or I.**



(a) III 02461357 (b) IIϕ_1 13570246 (c) I$\phi_1\phi_1$ 57134602 (d) Iϕ_1I 46025713



(e) $\phi_1\phi_1$I 64207531 (f) $\phi_1\phi_1\phi_1$ 75316420 (g) ϕ_1Iϕ_1 31752064 (h) ϕ_1II 20643175

Time Complexity and Comparisons

Network type	**Node degree**	**Diameter/ No. stages**	**Comm. delay**
Hypercube 1-port model	$\log n$	$\log n$	$O(n \log n)$
Hypercube all-port model	$\log n$	$\log n$	$O(n)$
2D mesh/torus	**4**	$O(n^{\frac{1}{2}})$	$O(n^{\frac{3}{2}})$
3D mesh/torus	**6**	$O(n^{\frac{1}{3}})$	$O(n^{\frac{4}{3}})$
Banyan	**1**	$\log n$	$O(n)$

Summary

- **An optimal all-to-all personalized exchange algorithm for banyan networks is presented.**
- **The new algorithm is based on a special Latin square, which corresponds to a set of admissible permutations of a banyan network.**
- **The off-line Latin square construction algorithm needs to be run only once at the time a network is built.**
- **The all-to-all personalized exchange implemented in banyan networks is favorably compared with other type of networks in terms of the number of I/O ports per processor and the communication delay.**
- **The proposed approach can be similarly applied to other unique-path, self-routing multistage networks.**