

Switch Fault Tolerance in a Mirrored K-Ary N-Tree

Yamin Li

Dept. of Computer Science

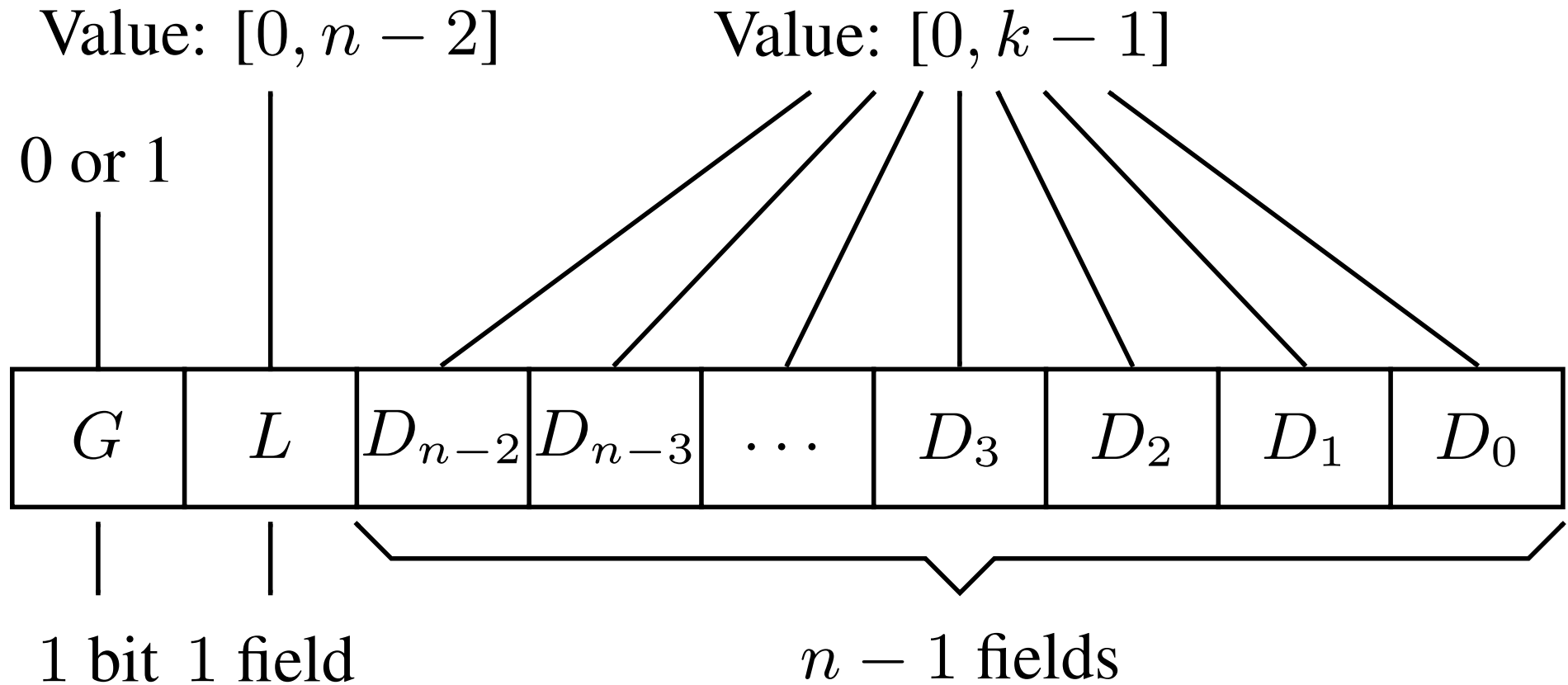
Hosei University

<http://yamin.cis.k.hosei.ac.jp/>

Motivations

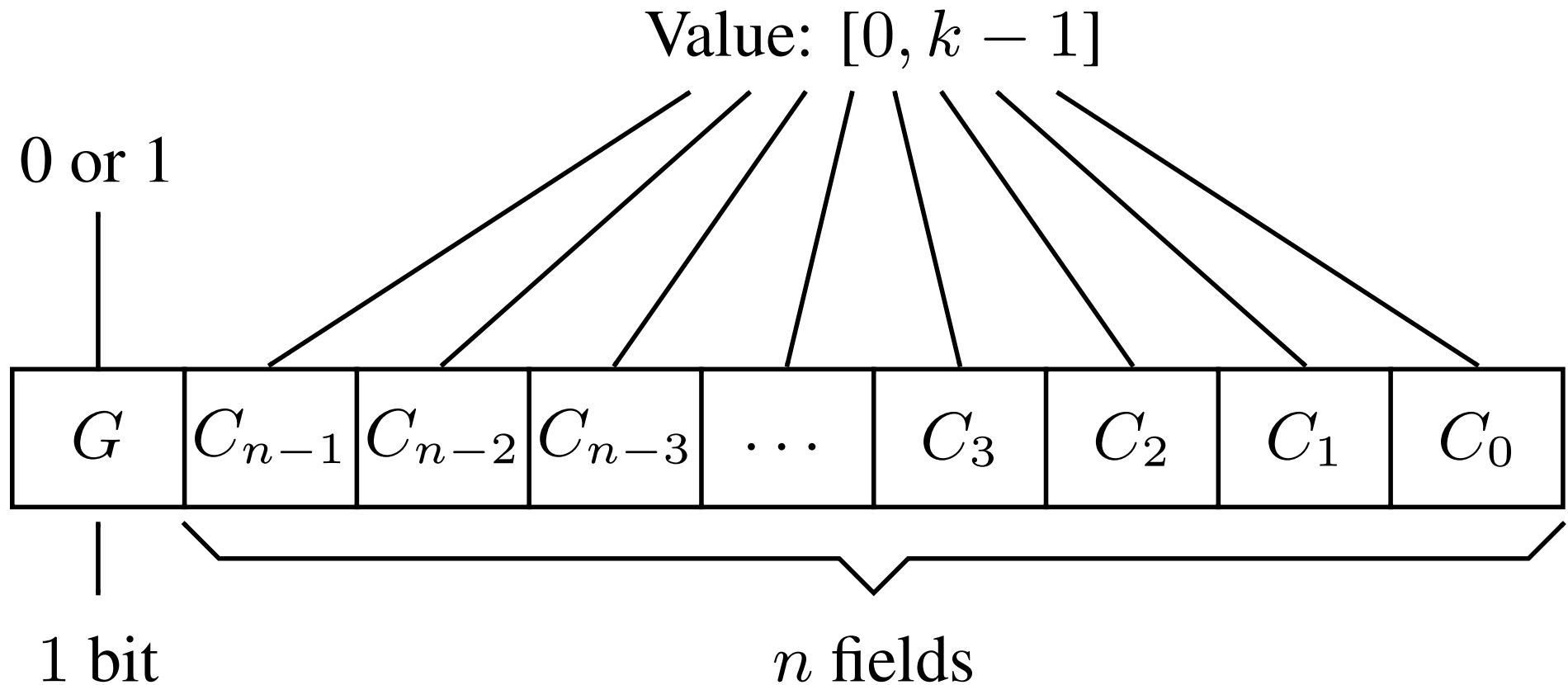
- The supercomputer scales up year by year and the cost of INs becomes high (www.top500.org)
- We propose a MiKANT - Mirrored K-Ary N-Tree
 - Symmetric (all switches have the same radix)
 - Low hardware (switch and link) cost
 - Short average distance $\Rightarrow$ short packet latency
 - Multiple paths from source to destination
- As the scale of MiKANT becomes large, the probability of switch failure increases
- We give three deadlock-free switch fault tolerant routing algorithms for MiKANT and evaluate their performance

Switch Address Format of MiKANT(n, k)



G : Group ID; L : Level (Stage) ID; D : Switch ID

Node Address Format of MiKANT(n, k)



G : Group ID; C : Node ID; $C_{n-2} \dots C_0$: Switch ID

Switch-Switch Connections

A switch

$$\langle G, L, D_{n-2}, \dots, D_{L+1}, D_L, D_{L-1}, \dots, D_0 \rangle$$

will connect to switches

$$\langle G, L + 1, D_{n-2}, \dots, D_{L+1}, *, D_{L-1}, \dots, D_0 \rangle$$

if $0 \leq L \leq n - 3$; otherwise ($L = n - 2$) to switches

$$\langle \overline{G}, L, *, D_{n-3}, \dots, D_1, D_0 \rangle$$

Switch-Compute Node Connections

There is a link between a switch

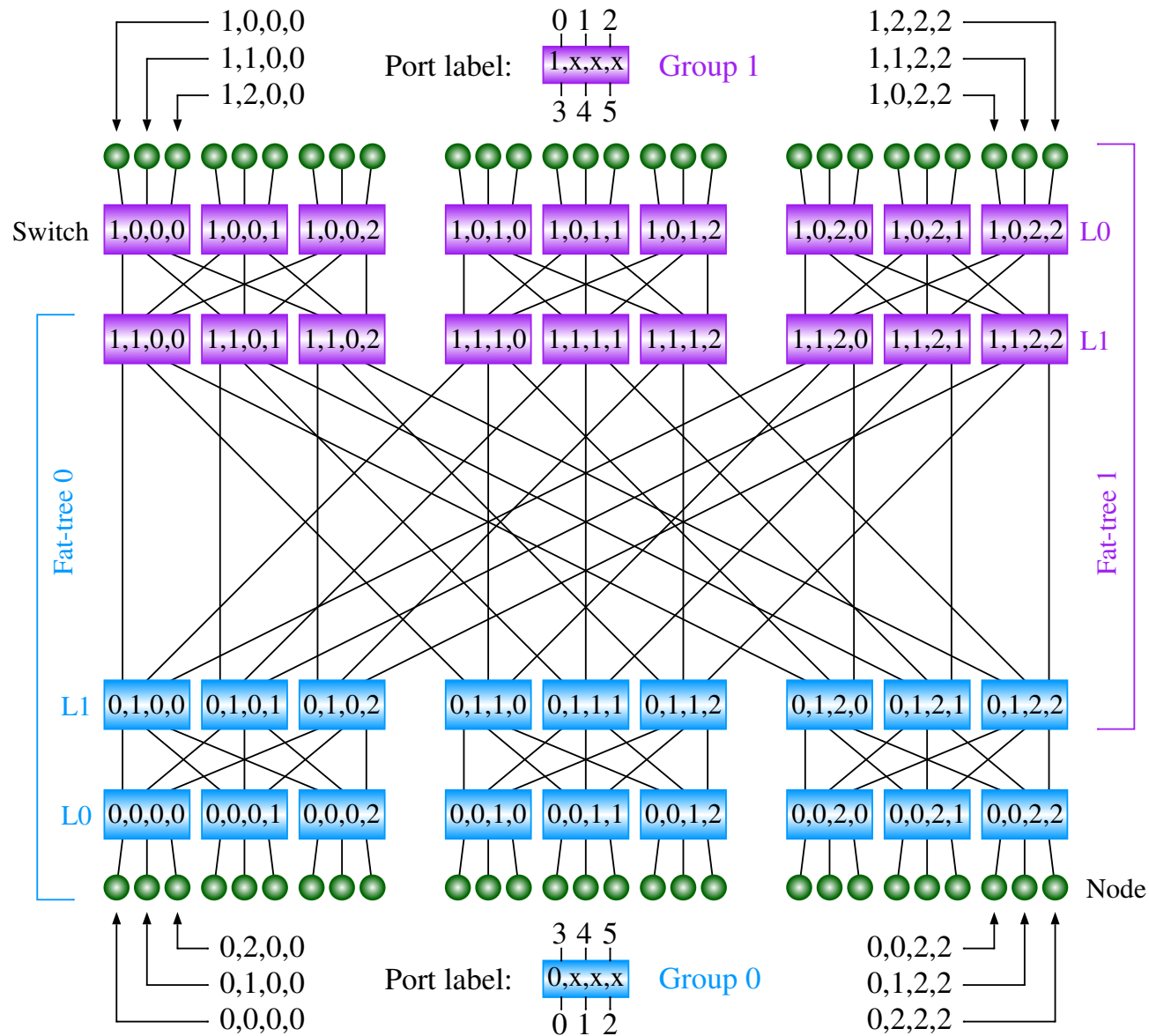
$$\langle G, 0, D_{n-2}, \dots, D_1, D_0 \rangle$$

and a compute node

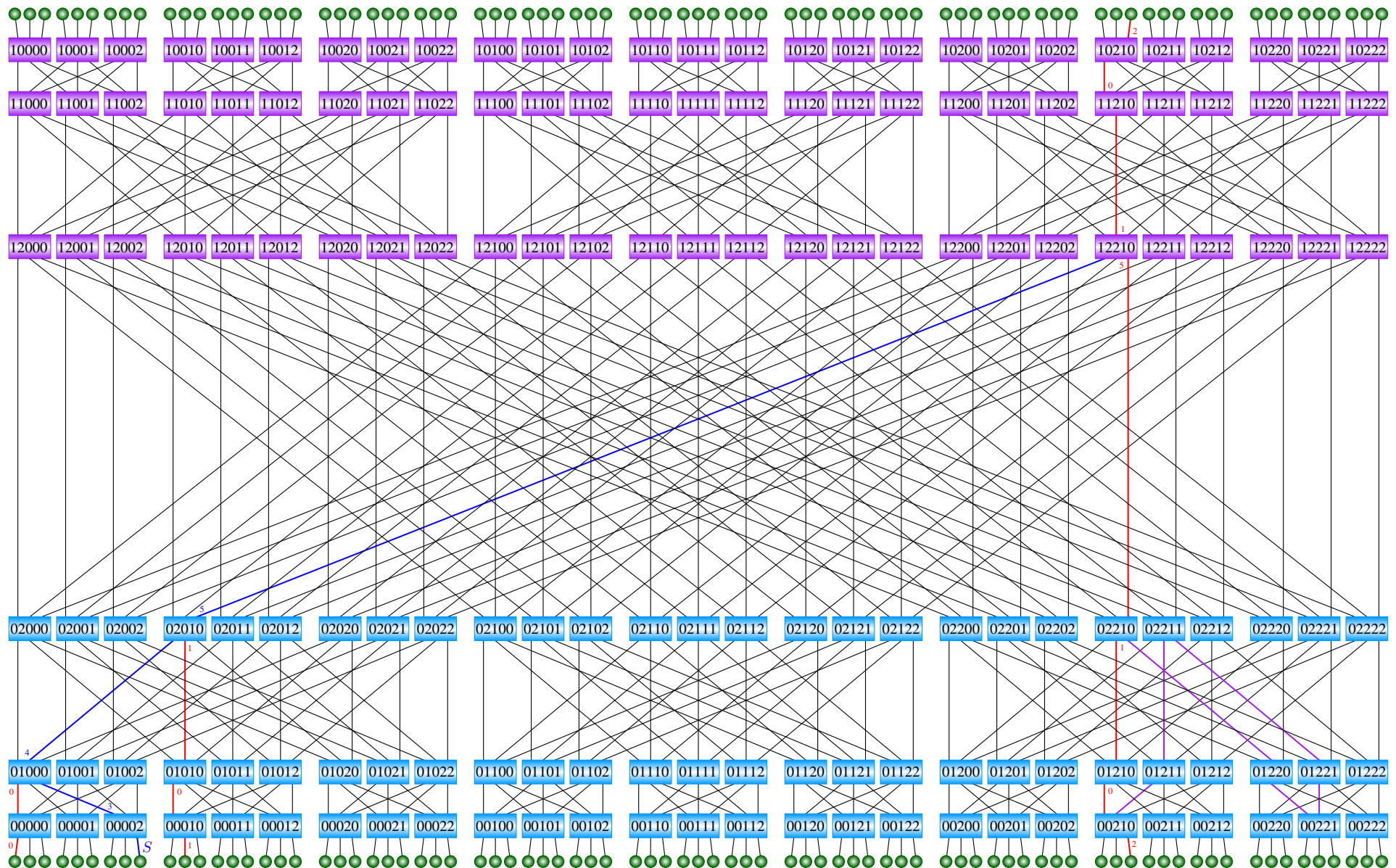
$$\langle G, C_{n-1}, C_{n-2}, \dots, C_1, C_0 \rangle$$

if $D_i = C_i$ for all $i \in \{n-2, \dots, 1, 0\}$

MiKANT(3,3)



MiKANT(3,4)



Comparison of Topological Properties

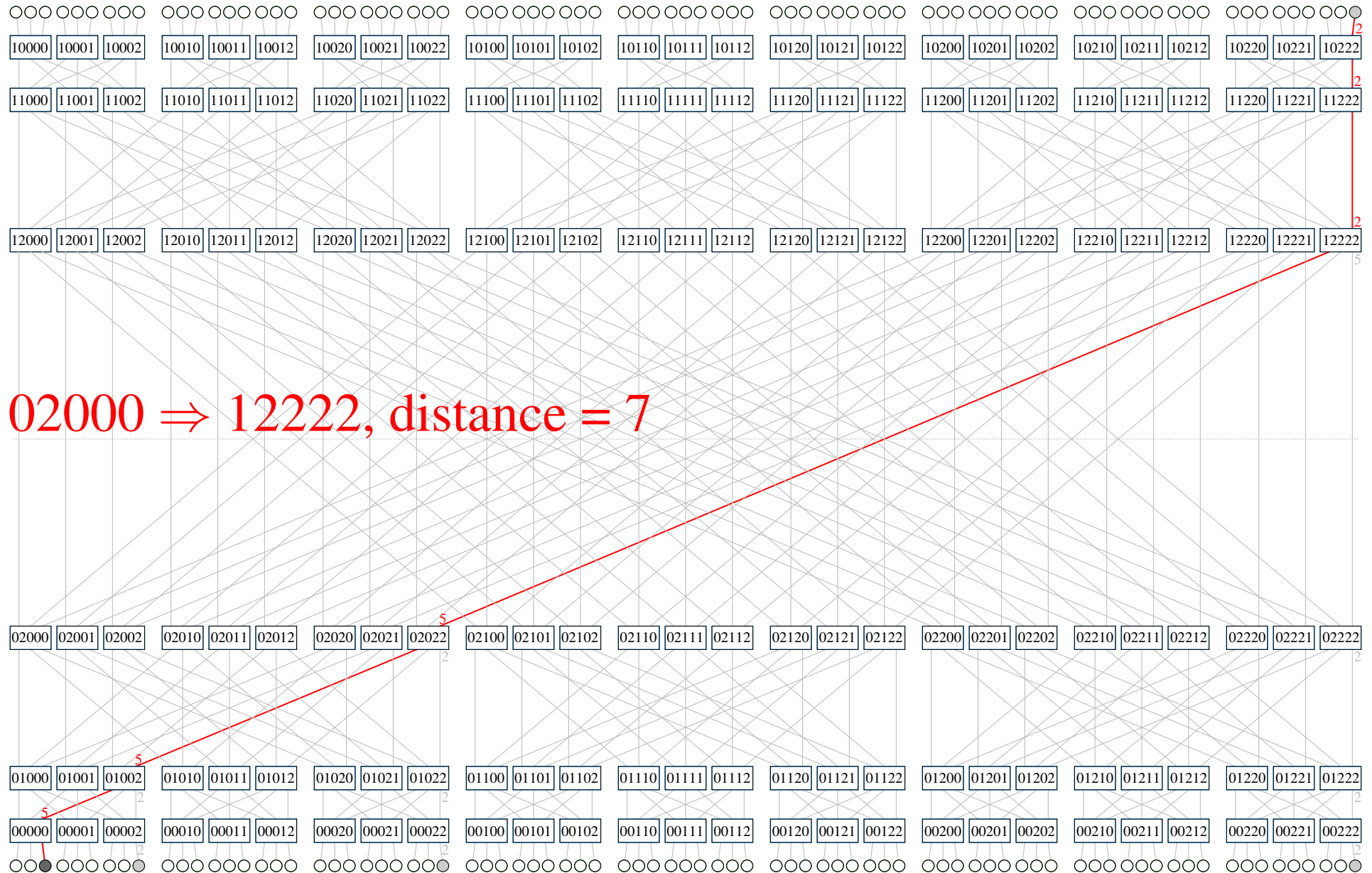
	Classical k -ary n -tree	Clos k -ary n -tree	Mirrored k -ary n -tree
#nodes	k^n	$2k^n$	$2k^n$
#switches	nk^{n-1}	$(2n-1)k^{n-1}$	$(2n-2)k^{n-1}$
#links	nk^n	$2nk^n$	$(2n-1)k^n$
Radix	$2k$	$2k$	$2k$
Diameter	$2n$	$2n$	$2n$
Bisection	$k^n/2$	$k^n/2$	$k^n/2$
Ave. dist	$2n - \frac{2}{k-1} + \frac{2}{(k-1)k^n}$	$2n - \frac{1}{k-1} + \frac{1}{(k-1)k^n}$	$2n - \frac{1}{k-1} + \frac{1}{(k-1)k^n} - \frac{1}{2}$

Minimal Per-Hop Deterministic Routing

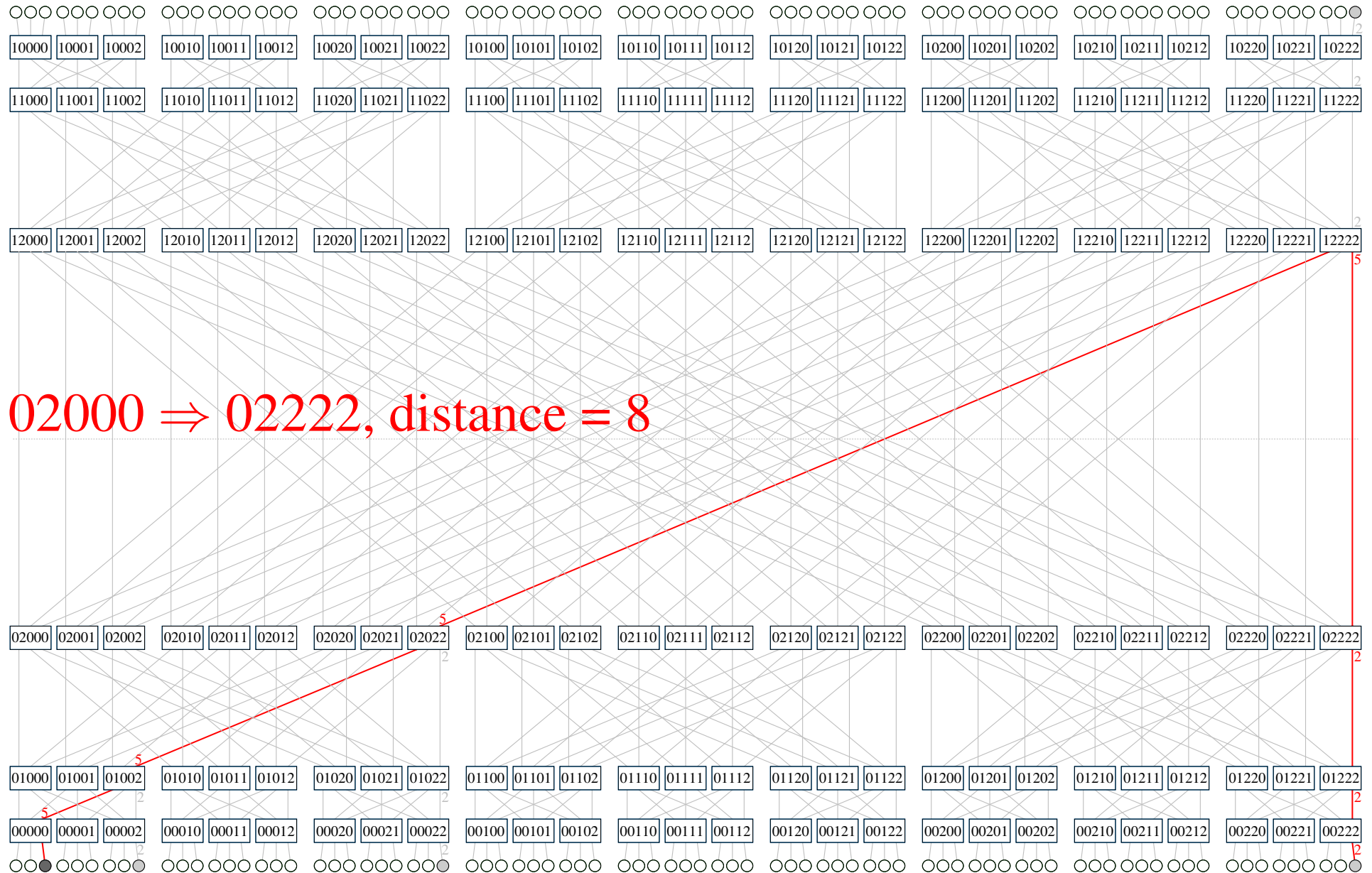
Algorithm 1 MiKANT_Routing (*packet*)

```
Input: packet =  $\langle T, data \rangle$ ;          /* received packet which will be sent to  $T$  */
 $T = \langle G_T, T_{n-1}, T_{n-2}, \dots, T_1, T_0 \rangle$ ;      /* destination node ID */
 $W = \langle G_W, L_W, W_{n-2}, \dots, W_1, W_0 \rangle$ ;      /* my switch ID */
if ( $G_W \neq G_T$ )                                     /*  $W, T$ : different groups */
    send packet to  $T_{L_W}^+$  port;                    /* increasing stage */
else                                                  /*  $W, T$ : same group */
    if ( $W_{n-2}, \dots, W_0 \neq T_{n-2}, \dots, T_0$ )    /* to nearest common ancestor (NCA) */
        send packet to  $T_{L_W}^+$  port;                /* increasing stage */
    else                                              /* to destination from NCA */
        if ( $L_W \neq 0$ )                             /* not a stage 0 switch */
            send packet to  $T_{L_W}^-$  port;            /* decreasing stage */
        else                                       /* a stage 0 switch */
            send packet to  $T_{n-1}^-$  port;          /* to destination node */
        endif
    endif
endif
endif
```

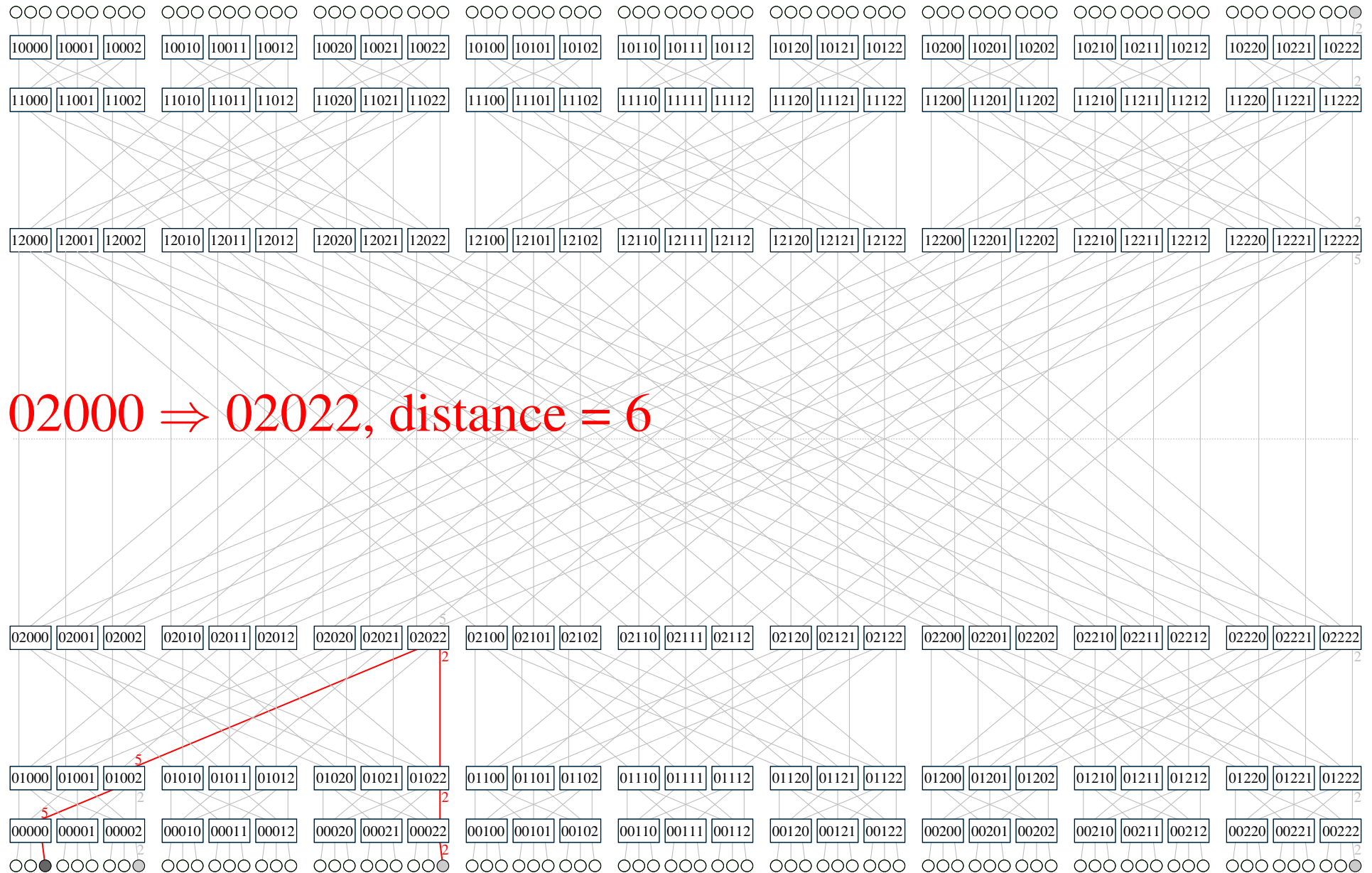
MiKANT(3,4) Routing Example 1



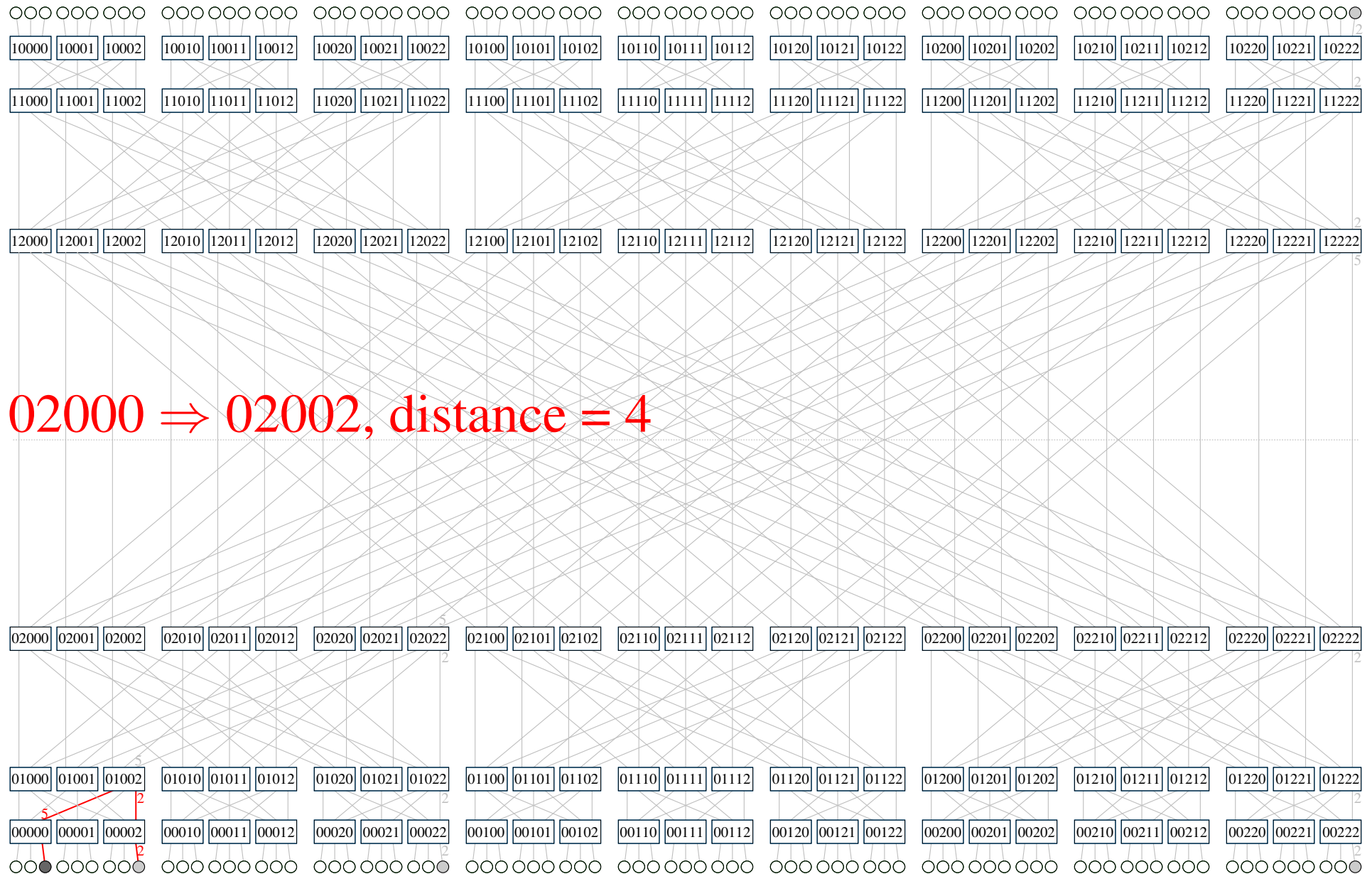
MiKANT(3,4) Routing Example 2



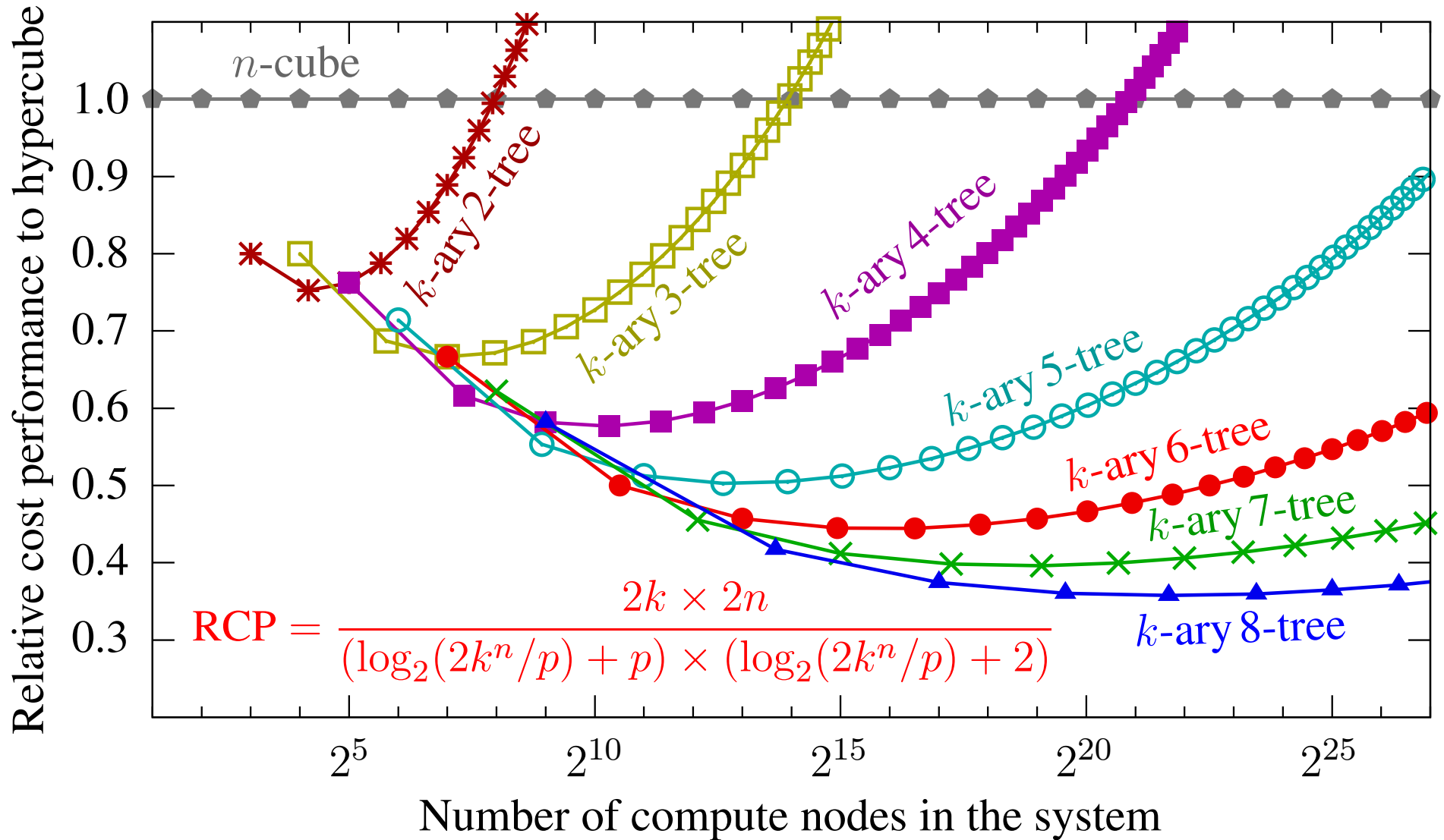
MiKANT(3,4) Routing Example 3



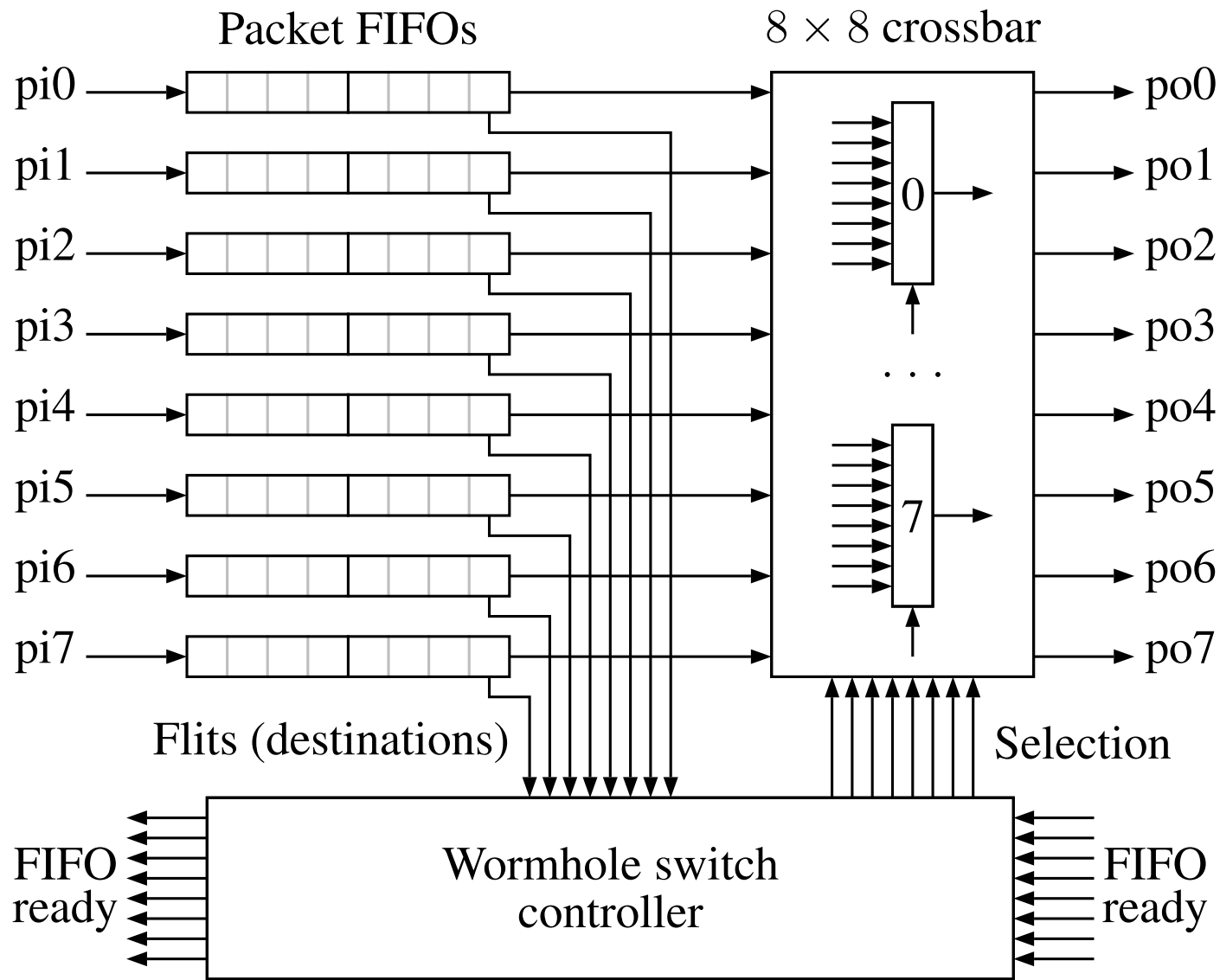
MiKANT(3,4) Routing Example 4



Relative Cost Performance to Hypercube



Block Diagram of Switch for Simulation

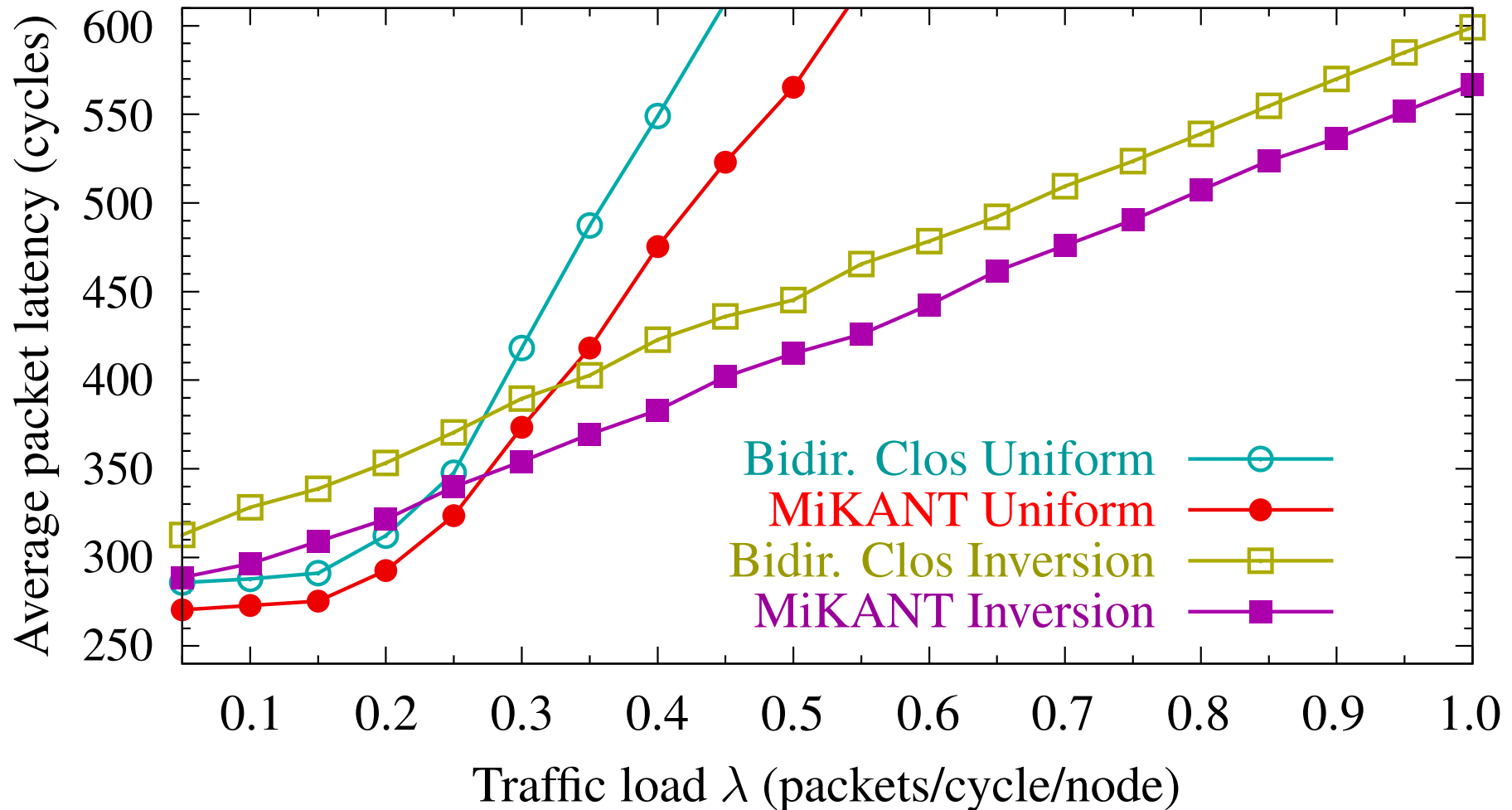


Synthetic Simulation

- Evaluate average packet latencies
- Network size: $k = 4$ and $n = 5$
- Use the minimal per-hop deterministic routing algorithm
- Simulate with two synthetic traffic patterns
 - **Uniform**: Destinations are randomly assigned
 - **Bit-inversion**: Destinations = bit-inversion of sources
- Traffic load λ in the range of 0.05 and 1.00
- Terminate when at least 200 packets of each source node reached their destinations
- Use the Wormhole switching routing technique (32-bit/flit)

Average Packet Latencies (Clock Cycles)

MiKANT has shorter packet latency than Clos network



Fault Tolerance — One-Port Algorithm

Algorithm 2 MiKANT_Fault_Tolerant_Routing_One_Port (*packet*)

Input: $packet = \langle T, data \rangle$;

$T = \langle G_T, T_{n-1}, T_{n-2}, \dots, T_1, T_0 \rangle$;

$W = \langle G_W, L_W, W_{n-2}, \dots, W_1, W_0 \rangle$;

if ($G_W \neq G_T$)

send *packet* to U via $T_{L_W}^+$ port;

if (U is faulty)

send *packet* to U via $((T_{L_W} + 1) \bmod k)^+$ port;

if (U is faulty)

 routing fails

endif

endif

else

if ($W_{n-2}, \dots, W_{L_W} \neq T_{n-2}, \dots, T_{L_W}$)

send *packet* to U via $T_{L_W}^+$ port;

if (U is faulty)

send *packet* to U via $((T_{L_W} + 1) \bmod k)^+$ port;

if (U is faulty)

 routing fails

endif

endif

else

if ($L_W > 0$)

send *packet* to U via $T_{L_W-1}^-$ port;

if (U is faulty)

 routing fails

endif

else

send *packet* to T via T_{n-1}^- port;

endif

endif

endif

/* received packet which will be sent to T */

/* destination node ID */

/* my switch ID */

/* W, T : different groups */

/* increasing level */

/* W, T : same group */

/* going to NCA */

/* increasing level */

/* going to destination from NCA */

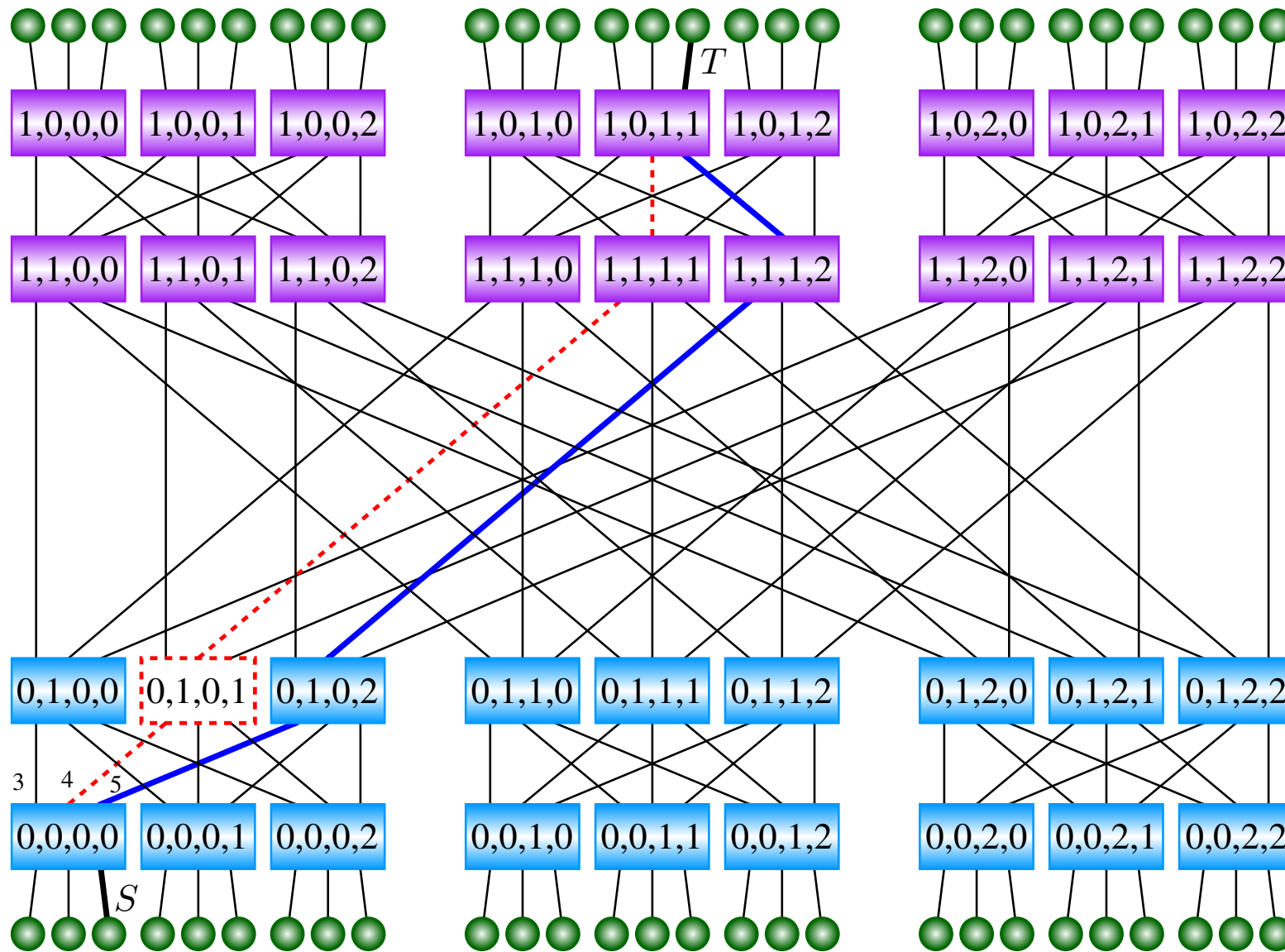
/* not a level 0 switch */

/* decreasing level */

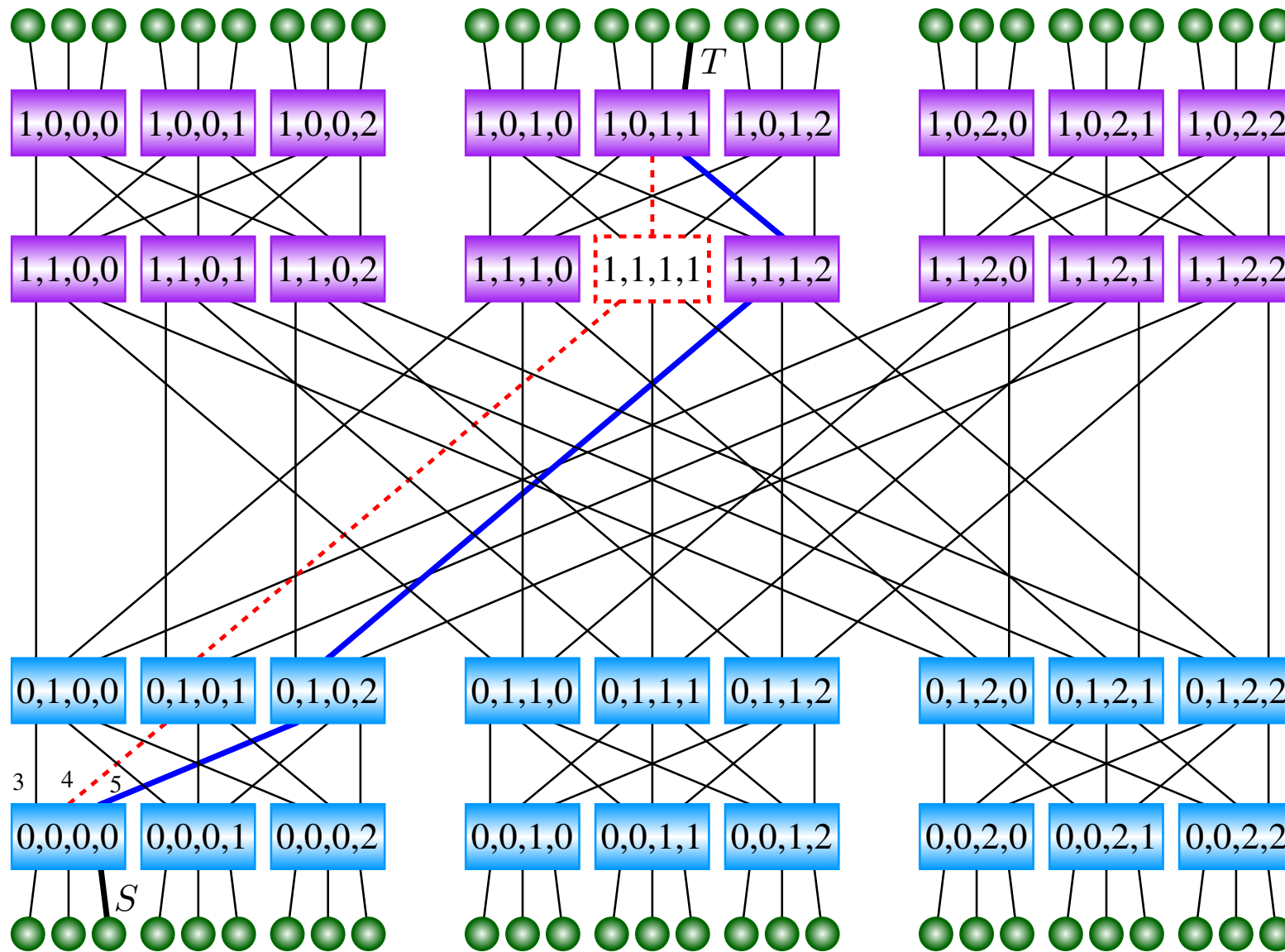
/* a level 0 switch */

/* to destination node */

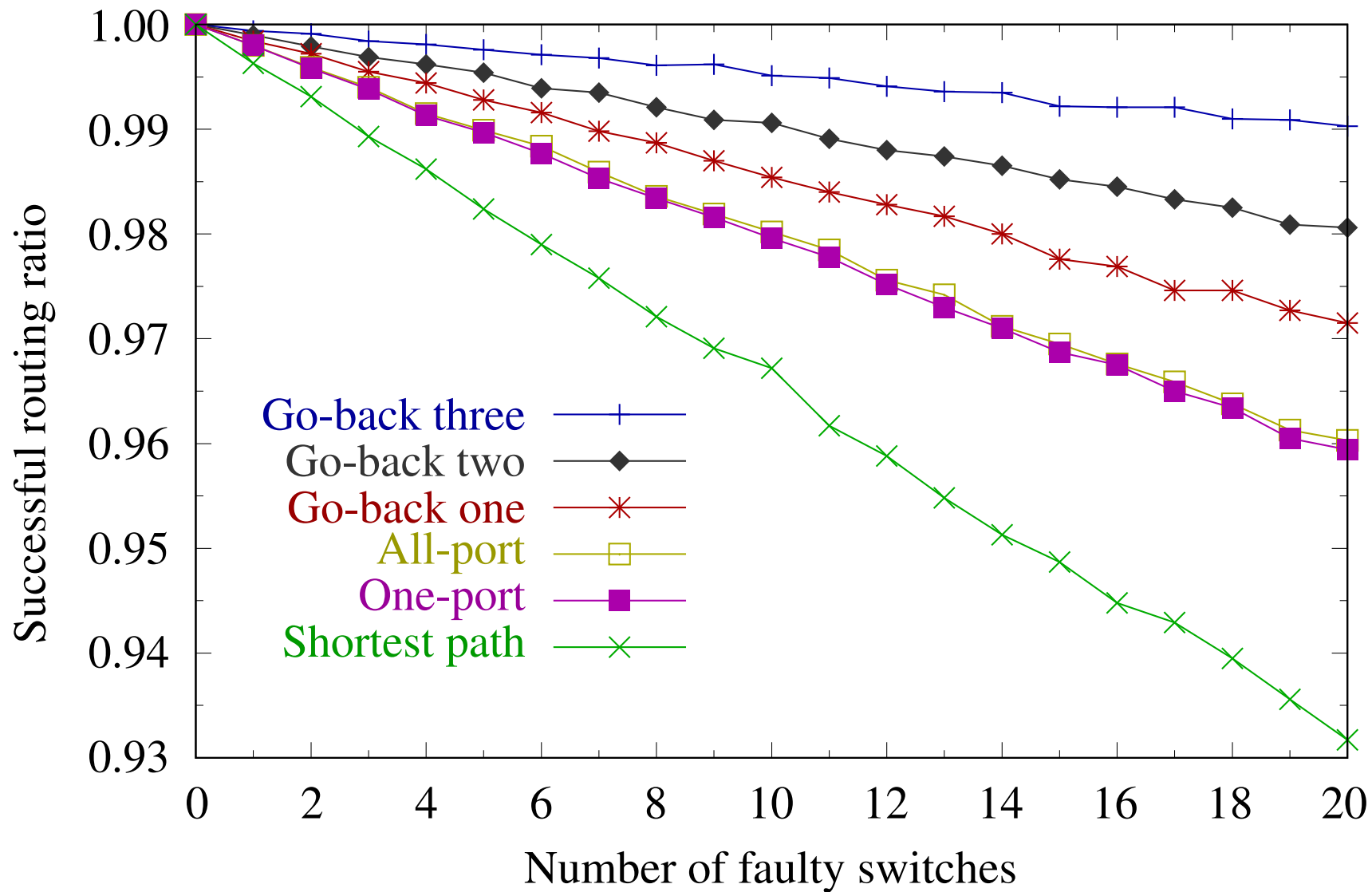
Fault Tolerance — One-Port Algorithm



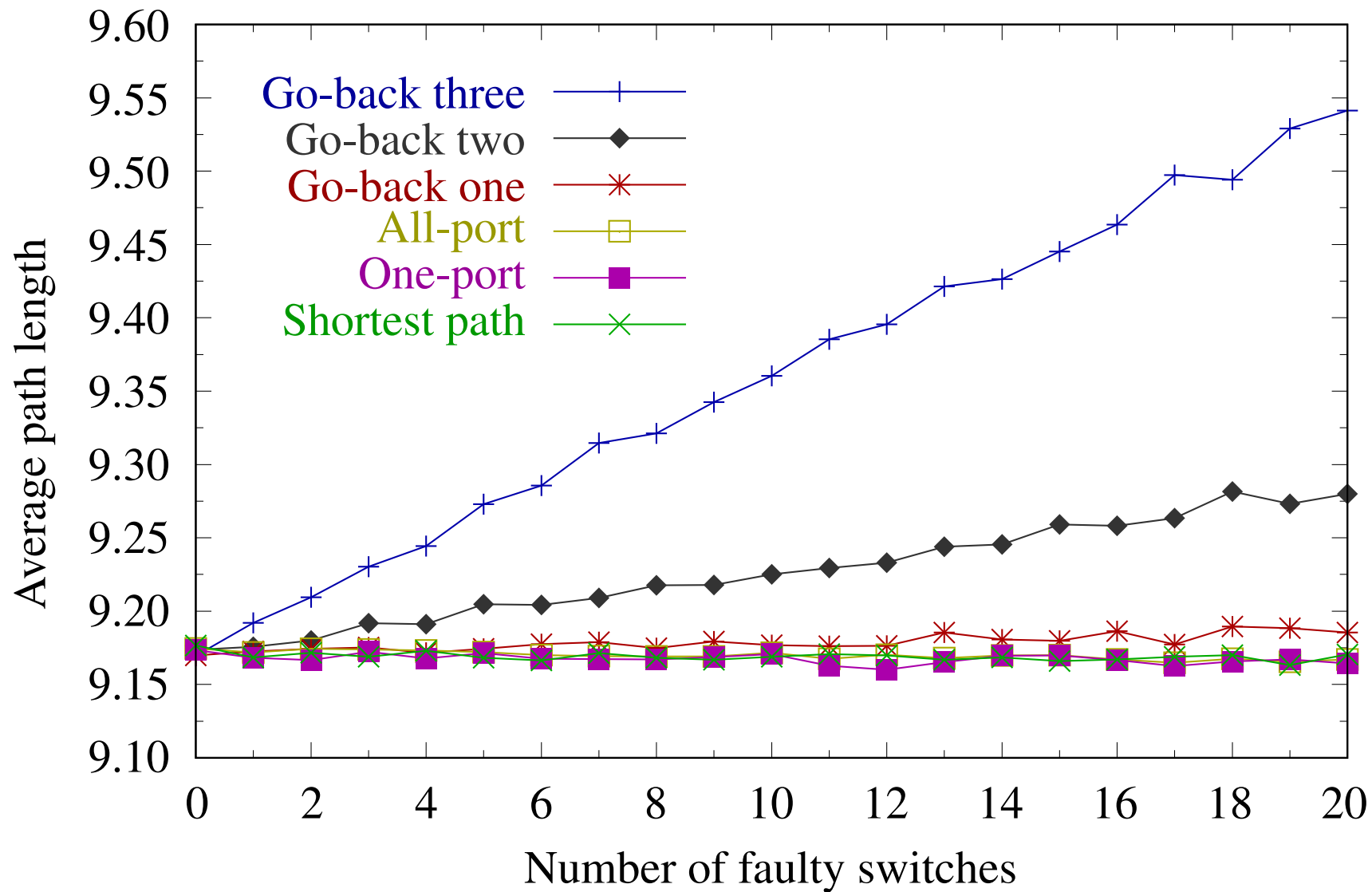
Fault Tolerance — Go Back Algorithm



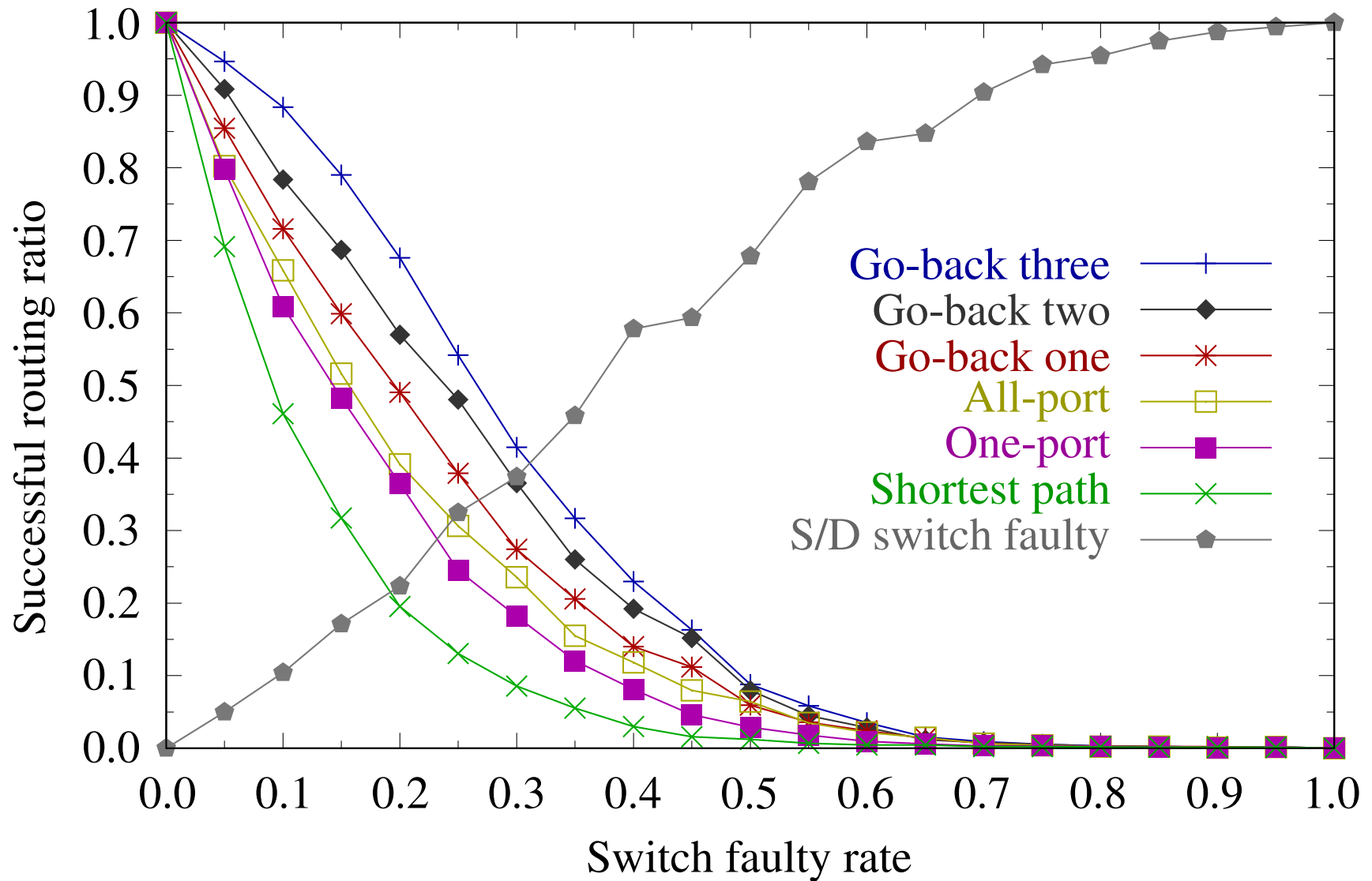
Fault Tolerance — Performance



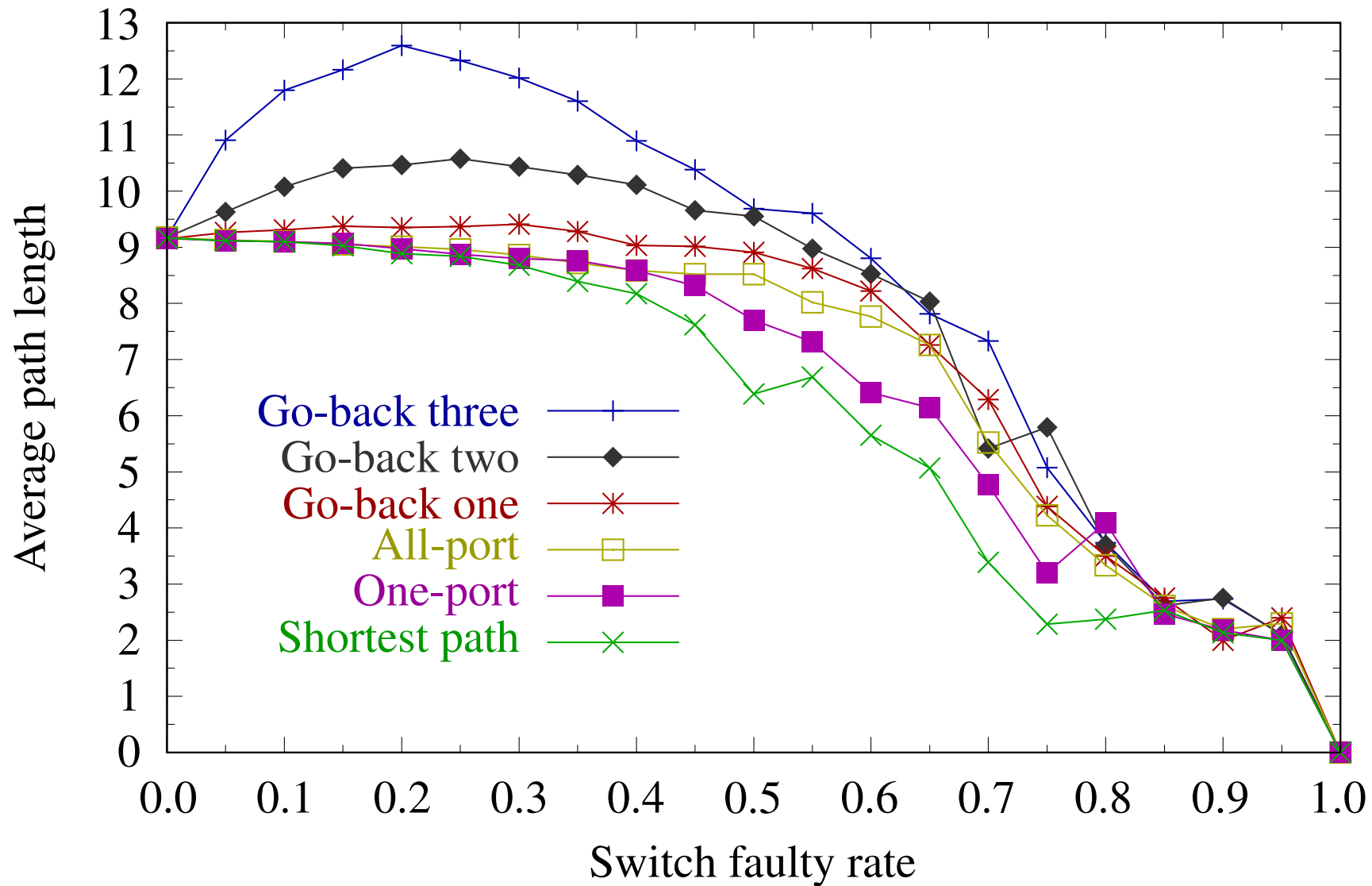
Fault Tolerance — Path Length



Fault Tolerance — Performance



Fault Tolerance — Path Length



Conclusions and Future Work

- Fat-tree network are widely used in supercomputers
- We propose a MiKANT - Mirrored K-Ary N-Tree
 - Symmetric (all switches have the same radix)
 - Low hardware (switch and link) cost
 - Short average distance $\Rightarrow$ short packet latency
- The proposed three simple fault tolerant routing algorithms are deadlock-free and can find a path between source and destination nodes at high probability
- Future research work
 - Link Fault tolerant routing in MiKANT
 - Implementation of MiKANT on NoC