Proceedings of *The 18th IEEE International Conference on Scalable Computing and Communications (ScalCom 2018)*, October 8-12, 2018, Guangzhou, China

MiKANT: A Mirrored K-Ary N-Tree for Reducing Hardware Cost and Packet Latency of Fat-Tree and Clos Networks

Yamin Li[*]　　Wanming Chu[†]

Interconnection networks based on the fat-tree topology are widely used in high-performance parallel supercomputers. In a classical fat-tree, the radix of root switches is less than that of other switches. Fat-tree is a folded version of a Clos network. A Clos network uses the same radix switches in all stages. However, fat-tree or Clos network has a high switch cost and great packet latency.

This study proposed a variant of the fat-tree, named Mirrored k-ary n-tree (MiKANT), that doubles the number of compute nodes of the fat-tree by adding a few switches and making all the switches have a same radix. Fig. 1 shows a MiKANT(k,n) with $k=3$ and $n=3$. Compared to the classical fat-tree and Clos network, MiKANT not only reduces the numbers of switches and links so that it can be implemented at lower hardware cost, but also makes the network average distance shorter for achieving higher communication performance.

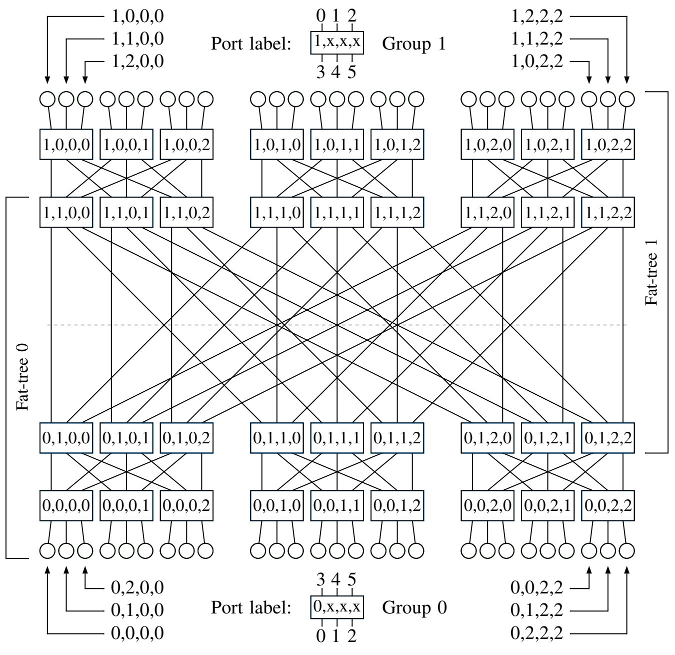

Fig. 1. A Mirrored 3-ary 3-tree

In this study, we described the structure of MiKANT, examined its topological properties (see TABLE I), gave a minimal per-hop deterministic routing algorithm (see **Algorithm 1**), and evaluated the cost performance through analytical and synthetic simulations.

[*] Department of Computer Science, CIS, Hosei University, Tokyo, Japan.
[†] Division of Information Systems, University of Aizu, Fukushima, Japan.

TABLE I. Comparison of topological properties

	Fat-tree	Bidir. Clos	MiKANT
# of nodes	k^n	$2k^n$	$2k^n$
# of switches	nk^{n-1}	$(2n-1)k^{n-1}$	$(2n-2)k^{n-1}$
# of links	nk^n	$2nk^n$	$(2n-1)k^n$
Radix	$2k$	$2k$	$2k$
Diameter	$2n$	$2n$	$2n$
Bisect. width	$k^n/2$	$k^n/2$	$k^n/2$

Algorithm 1 MiKANT_Routing (*packet*)

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

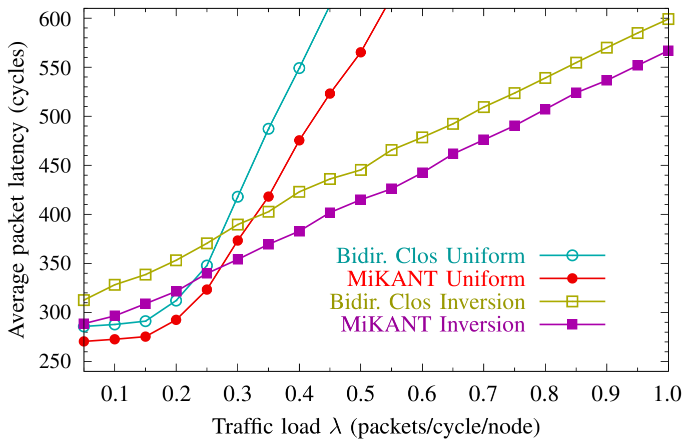

Fig. 2. 4-ary 5-tree average packet latencies

The analytical simulation results show that MiKANT reduces the average distance by about 0.5, saves 6.3% to 25.0% links and 12.5% to 50.0% switches, and improves performance of 9.1% to 41.4%, compared to the fat-tree, when n is in the range of 2 and 8. Our synthetic simulation results (see Fig. 2) also show that the MiKANT achieves much lower average packet latencies than the Clos network at two traffic patterns.