k-ary n-cubes versus Fat Trees

Weihaw Chuang and Wilson Chin
EE482 Research Paper, Spring 1999

Introduction

In this paper, we examine performance aspects of fat-tree topologies and compare these to k-ary n-cube topologies. In particular, we apply the analytic techniques developed for cube topologies upon fat trees to understand better the characteristic of that topology. The organization of this paper is as follows: Section 1 contains an overview of k-ary n-cube and fat-tree topologies. Section 2 contains a discussion of baseline assumptions used in modeling these network topologies. Section 3 contains an analysis of network performance in terms of bisection bandwidth and network latency. Lastly we conclude in Section 4 with a general overview and criticism of the papers we read.

1. Topology Overview

Fat Tree Topology

Fat tree network topologies was first presented in the literature in [Lei85]. They are represented by a tree with the intuitive characteristic that the edge capacity get "thicker further from the leaves" meaning that the capacity of the network closer to the root is larger than that towards the leaves. Fat trees are organized with processing elements at the leaves, and router nodes at the root and intermediate nodes. Messages entering a routing node from the parent link may leave from any of the children links; messages entering from a child link may leave through either another child link or the parent link. In contrast to meshes and tori, fat trees are acyclic and therefore deadlock free.

Let us now characterize the basic properties of fat trees. For this paper we describe fat trees as k-ary n-trees where the use of k and n is analogous to k-ary n-fly butterfly networks. For fat trees, k represents the number of logical children nodes of a given parent node, while n represents height of the tree. N represents the number of processor nodes found at the leaves, $N = k^n$. Edge capacities grow with the height of the tree, and growth rates are constrained by a volume-universality criterion (discussed below). Increasing capacity implies a proportionate increase in pins for routers closer to root, which in practice may result in an unpackagable number of pins. This is handled in two ways: First, the actual

implementation of the fat tree is usually done through a hyper-tree topology detailed in the next section. Second, the parent edge capacity of routers in fat trees may be concentrated. Concentration in a fat tree network means the parent edge capacity of a node is less than the sum of the children edge capacities.

We first consider the fat tree construction based on the node design in [Lei85] and the interpretation by [DeHon90]. We consider designs that are *volume-universal*, in which successive levels towards the root of the fat tree increase in bandwidth at a rate that conserves the relationships given by

$$B \propto a$$

$$v \propto \text{Number of Terminals}$$

where

$B \equiv$ Bisection bandwidth
$v \equiv$ volume
$a \equiv$ Surface Area enclosed by given v
$a \propto v^{2/3}$

For a k-ary fat tree, where k is the number of child links of a tree node, the volume increases between stages according to the relationship

$$v_{i+1} = kv_i$$

where the tree level i is numbered from leafnode=0 to root=n.

The bisection bandwidth must increase by at least

$$B_{i+1} = k^{2/3} B_i$$

from level to level in order to preserve volume-universality. Volume universality of fat trees is the key idea of [Lei85] which makes use of this assumption to prove a communication latency relationship between fat trees and any other network with equal volume-cost. Leiserson proves that it takes $O\left(t \lg^3 n\right)$ time to route a message that would route in t time in any other network with equal volume-cost. The implication is that universal fat trees can simulate any other routing topology in polylogarithmic time for the same amount of hardware volume proving that fat-trees are close to ideal in a theoretical-computational sense.

[Lei85] applies proves volume-universality on a binary (k=2) fat tree. But his conclusions should be applicable to any k-ary tree as long as surface area to communication rate relationship should still hold.

Hyper-Tree

Hyper-trees are a variation of the fat-tree topology that follows realistic pin constraints. Unlike [Lei85] fat trees, router nodes have up to k many parent edges, and the parents notes and children within the local sub-tree are fully bipartite. In other words, each child node sees the same set of parent nodes as the other children, and the parent nodes see the same children. Concentration is allowed. The number of internal nodes is $N_{\text{int}} = nk^{n-1}$ assuming no concentration and the number of terminal nodes is $N = k^n$. Hence we find that N_{int} the total number of internal nodes in the fat tree without concentration is

$$N_{\text{int}}(k,n) = \left(\frac{n}{k} + 1\right)k^n.$$

This equation provides an upper bound on the router node count. One problem with the hyper-tree is that, though it is referred to several times, a strong definition does not exist in the literature.

k-ary n-cube Topology

k-ary n-cubes are networks with *n* dimensions and *k* nodes in each dimension. The width of links between nodes is uniform across all nodes in the network. We consider only tori, in which each row of *k* nodes in each dimension is connected in a ring.

2. Baseline modeling assumptions

Preliminary investigations in the literature assumed that

- network latency can accurately be measured by hop count (a constant)
- link width is independent of dimensionality

These assumptions tended to favor high-dimension, low hop count topologies (such as hypercubes).

As networks grow in size, constraints imposed by packaging and cost become significant. Dally introduces the concept of a constant bisection constraint [Dal90] to capture the cost related to the complexity imposed by wiring, in order to more fairly compare cost-equivalent networks. Agarwal introduces the concept of a constant node size constraint, which captures the costs related to packaging pin-count limitations [Aga91]. Others examine advancements in signaling and switching technology, which affect the underlying latency models of components that make up the network. For instance, Scott in [Sco94] investigates the impact of

pipelining packet transmissions, which allows the removal of transmission time from cycle time, which makes high-dimension (high-length) topologies somewhat less unattractive than in [Dal90] and [Aga91].

We examine networks using a combination of the constant bisection bandwidth constraint and the constant node size constraint. We also briefly look at wire delay, and draw some conclusions about the effect of wire delay on performance.

3. Network Performance

We begin by examining the bisection bandwidth of each topology. We then examine latency and network performance in terms of latency in the presence of contention.

k-ary n-cube Bisection Bandwidth

Following the analysis in [Dal90], the bisection bandwidth of a k-ary n-cube torus with W-wide communication channels is

$$B = \frac{2WN}{k}, N = k^n$$

Dally introduces the concept of comparing networks with *constant bisection*: cost of a network is directly related to the number of wires crossing the bisection of the network, so to compare networks of equivalent cost, we should hold the bisection of that network constant.

Applying this *constant bisection* constraint and normalizing to the case of the 2-ary n-cube (hypercube) with W=1 channels, the channel width of an arbitrary k-ary n-cube with the same cost as the baseline hypercube becomes a function of the topology k and n [Dal90]

$$W(k,n) = \frac{k}{2} = \frac{1}{2} N^{\frac{1}{n}}$$

Agarwal [Agarwal91] argues that packaging limitations (pins per chip or pins per board) impose a hard limit on the total number of wires connected to a node, and this node size constraint (rather than the bisection bandwidth) may limit the design of the network. Because of this physical limitation, we should

also consider the case where total channel width out of any given node remains fixed as other topology parameters are varied. Applying this *constant node size* constraint and normalizing to the case of the 2-ary n-cube, we get the relationship:

$$NodeSize(k,n) = 2nW = const \equiv NodeSize(2,n)$$

$$W = \frac{NodeSize(2,n)}{2n}$$

In both models, the link bandwidth decreases with higher dimension. Link bandwidth falls off more rapidly with the *constant bisection bandwidth* constraint, compared to the *constant node size* constraint. For example, possible k-ary n-cube torus networks with 4096 nodes may be implemented in the following ways, depending on which of these is most constraining.

4096-node network, maximum bisection bandwidth=4096

k	n	W, link width	Node Size	B, bisection bandwidth
2	12	1	24	4096
4	6	2	24	4096
8	4	4	32	4096
16	3	8	48	4096
64	2	32	128	4096

4096-node network, maximum node size=24

k	n	W, link width	Node Size	B, bisection bandwidth
2	12	1	24	4096
4	6	2	24	4096
8	4	3	24	3072
16	3	4	24	2048
64	2	6	24	768

We observe the following:

- Dally's paper does not constrain node size, which may result in large node sizes for low-dimension networks for a given bisection bandwidth. Such nodes are likely to be much more expensive to implement than smaller nodes.
- While Scott and Agarwal investigate models based on each of these constraints separately; they do not compare the resulting network performance of these models on an *equal cost* basis.

Fat Tree and Hypertree Bisection Bandwidth

Using a similar framework, we can examine the bisection bandwidth in the context of fat trees and hypertrees. We first consider the fat tree construction based on the node design in [Leiserson85] and the interpretation by [DeHon]. We consider designs that are volume-universal, in which successive levels towards the root of the fat tree increase in bandwidth at a rate, which conserves the relationships given by

$$B \propto a$$

$$v \propto \text{Number of Terminals}$$

For a k-ary fat tree, where k is the number of child links of a tree node, the volume increases between stage according to the relationship

$$v_i = kv_{i-1}$$

The bisection bandwidth must increase by at least

$$B_i = k^{2/3} B_{i-1}$$

from level to level in order to preserve volume-universality.

We have the following relationships

$$levels = \log_k N - 1$$

$$N_{\text{int}}(k,n) = \sum_{i=0}^{levels-1} k_i$$

$$NodeSize_i = k \cdot W_i + k^{2/3} W_i$$

$$B_i = \left(k^{2/3} + k\right) W_i \Rightarrow W_i = \frac{B}{k^{2/3} + k}, \text{bidirectional links}$$

4096-node network, flat fat tree, fixed max bisection bandwidth

k	Levels, *n*	routers	Root Node Size	Root Bisection Bandwidth	Leaf-node Node Size	Link Width, leaf-node	leaf-node Bisection Bandwidth
2	12	2047	4096	4096	25.4	7	25.4
4	6	1365	4096	4096	40.3	6.2	40.3
8	4	585	4096	4096	64	5.4	64
16	3	273	4096	4096	102	4.6	102
64	2	65	4096	4096	256	3.2	256

For the hypertree, we fix the router size and use multiple routers for each logical node. Given a growth rate of G per level ($G \geq k^{2/3}$ to preserve volume universality) we have a router count of

$$N_{\text{int}}(k,n) = \sum_{i=0}^{i=n-1} G^i k^{n-i}$$

4096-node network, hyper tree with fixed node size=24; minimum $G = k^{2/3}$

k	Levels, *n*	Node Size, Root	Root Bisection Bandwidth	Leaf-node Node Size	leaf-node Link Width	leaf-node Bisection Bandwidth
2	12	162x24	3870	24	6.7	24
4	6	102x24	2438	24	3.7	24
8	4	64x24	1536	24	2	24
16	3	41x24	968	24	1.07	24
64	2	X	X	24	<1	X

k-ary n-cube Baseline Latency
The baseline latency model is

$$T_{;} = T_{net} + T_{node}$$

In [Dal90] concentrates on deriving T_{net} which is a latency model based on serialization delay and routing delay as the routing delay is well documented in other papers. [Aga91] contributes T_c contention term to model blocking in terms of latency but we don't consider this at this point. To simplify our work we deal with T_{net}:

$$T_{net} = T_c\left(D + \frac{L}{W}\right)$$

D is average hop count with uniform distribution of destinations, and $\frac{L}{W}$ is the serialization delay if the packet size is larger than the width of the wire. Dally's [Dal90] models T_c channel time as either a constant, short wire or long wire delay in order to characterize the range of physical wire models. Constant delay is provided to clarify examples though is unrealistic. For short wires, delay is essentially based on capacitive loading of the wire dependent on wire length l and is logarithmic of l:

$$t_s = te \log_e Kl$$

For very long wires, wire delay looks like a transmission line, which has a delay proportionate to the speed of light; hence delay is proportionate to wire length l.

$$t_l = \frac{l\sqrt{e}}{c}$$

Dally then normalizes away the coefficients to find T_c channel time based on n and k using the following length equation.

$$l = \frac{T^w(n)}{T^w(2)} = \sqrt{k}^{n-2} = k^{\frac{n}{2}-1}$$

We use Agarwal explanation of the derivation to understand how length is related to n and k. The right hand side of the equation is trying to map an n dimensional network onto a 2-dimension physical plane. Each network dimension contributes factor k nodes, and $\sqrt{k}$ distance to each dimension in the physical plane. The $T^w(2)$ term normalizes out a baseline 2-D distance between nodes. Note this is generalized to the z-dimensional case as follows: $l = k^{\frac{n}{z}-1}$

For some source node, the mean number of hops to a destination node assuming uniform distribution of destinations is

$$D = \left(\frac{k-1}{2}\right)n$$

We use the above planar length model to find T_c in terms of k an n, and subsequently T_{net} from either the linear or logarithmic delay model. Using a logarithmic delay model (applicable to short wires),

$$T_c \propto 1 + \log_\lambda l = 1 + (\frac{n}{2} - 1)\log_\lambda k$$

hence

$$T_{net} = \left(1 + (\frac{n}{2} - 1)\log_\lambda k\right)\left(\left(\frac{k-1}{2}\right)n + \frac{L}{W}\right)$$

Using a linear delay model (applicable to long wires),

$$T_c \propto l = k^{\frac{n}{2}-1}.$$

hence

$$T_{net} = \left(k^{\frac{n}{2}-1} \right)\left(\left(\frac{k-1}{2} \right) n + \frac{L}{W} \right)$$

In order to validate my own calculations, we plotted out the equation for constant bisection bandwidth obtaining similar but not exactly the same results as Dally. Dally finds that for N=256,16384 that the minimum latency points are n=2,4. Our results concur giving us confidence that the following spreadsheet models are in working order[1].

Fat tree Latency

Following the above method we build a latency model for fat trees. We find the average hop count D, assuming uniform distribution, using the observation that packets as they are routed up the tree will take a child exit proportionate to the size of the sub-tree. This is summed up as

$$D = 2\sum_{i=1}^{n} \left(\frac{1}{k} \right)^{n-i} \left(\frac{k-1}{k} \right) i$$

Traffic through the root node dominates as illustrated in Figure 1.

n	k	D
1.00	16384.00	2.00
2.00	128.00	3.98
3.00	25.40	5.92
4.00	11.31	7.80
5.00	6.96	9.60
6.00	5.04	11.50
7.00	4.00	13.33
8.00	3.36	15.00
9.00	2.94	16.00
10.00	2.64	18.00
11.00	2.42	20.00
12.00	2.24	22.00
13.00	2.11	24.00
14.00	2.00	26.00

Figure 1. Fat Tree Average Hop Count

To simplify our Fat Tree latency approximation, we simplify D to

$$D = 2n$$

at the cost some additional error in the range of 0 to +2 which should be conservative since we are adding to hop count.

The wire length derivation for fat trees below is motivated by the analysis method used on the k-ary n-cube topology in [Dal90]. Since we are concerned with a physical model of a fat tree, we use a hyper-tree. First we observe that starting from the root, the k sub-trees of the root will only communicate within that sub-tree and to its parent. This partitioning is recursively applied. We use this observation to find the wire distance for each stage. We approximate this by first finding the number of channels C between each stage i in a sub-tree where i=0 at the terminals, and assuming no concentration.

$$C_i = k\left(k^{i-1}\right) = k^i$$

As wires have a cross section, the number of channels-in ratio to the channels at root is proportional to the surface area as follows

$$A_i \propto C_i = k^{i-n}$$

We normalize out the k^n term. To find length *l* of stage *i* (i=0 level is the terminal nodes parent channels; n represents the height of the tree), we take the diagonal of this to get

$$l_i \propto k^{i/2}$$

Next we use the observation that communication delay at the root dominates, having the worst-case wire length.

$$l_i \propto k^{n/2}$$

We normalize this to a plane resulting in the following approximate relationship

$$l = k^{n/2-1}$$

Because of the normalization step taken above, this length approximation is most likely not directly compatible with k-ary n-cube.

From this we can find T_c the channel delay as derived above and T_{net} .
For logarithmic delay (short wire)

---

[1] We would provide graphs of this, except that we can't get the Excel graphing function to show the correct results!?! All we can do is provide the raw data.

$$T_{net} = \left(1 + (\frac{n}{2} - 1)\log_{\lambda} k\right)\left(2n + \frac{L}{W}\right)$$

For linear delay (long wire)

$$T_{net} = \left(k^{\frac{n}{2}-1}\right)\left(2n + \frac{L}{W}\right)$$

If we hold bisection bandwidth B = NW constant and normalize it to N by setting B = N, we find that W=1. For fat trees, the constant-bisection constraint provides freedom on k and n so long as it obeys N. Hence graphing T_{net} would seemingly lead us to find that the lowest possible n (a "flat" network) will give us the lowest latency. For physical implementation this is unrealistic, and we should apply the constant-node constraint from Agarwal's paper.

For constant bisection bandwidth where $W = 1$, we find T_{net}

For logarithmic delay (short wire)

$$T_{net} = \left(1 + (\frac{n}{2} - 1)\log_{\lambda} k\right)(2n + L)$$

For linear delay (long wire)

$$T_{net} = \left(k^{\frac{n}{2}-1}\right)(2n + L)$$

For constant node size, where $W = \frac{NodeSize(k,n)}{2n}$, we find T_{net}

For logarithmic delay (short wire)

$$T_{net} = \left(1 + (\frac{n}{2} - 1)\log_{\lambda} k\right)\left(2n + \frac{2nL}{NodeSize}\right)$$

For linear delay (long wire)

$$T_{net} = \left(k^{\frac{n}{2}-1}\right)\left(2n + \frac{2nL}{NodeSize}\right)$$

And finally, we have the baseline latency given by

$$T_{base\_latency} = T_{node} + T_{net} = T_{node} + \left(k^{\frac{n}{2}-1} \right) \left(2n + \frac{2nL}{NodeSize} \right)$$

We can see from the following charts that cubes and trees have similar absolute minimal latencies using the following formula. However, the dimensions at the minimal latencies differ:

K-ary N-Cube for const node

N	256				
n	k	w	D	L/W	Tnet
1	256.00	320.00	127.50	3.13	130.63
2	16.00	160.00	15.00	6.25	21.25
3	6.35	106.67	8.02	9.38	17.40
4	4.00	80.00	6.00	12.50	18.50
5	3.03	64.00	5.08	15.63	20.70
6	2.52	53.33	4.56	18.75	23.31
7	2.21	45.71	4.23	21.88	26.10
8	2.00	40.00	4.00	25.00	29.00
9	1.85	35.56	3.83	28.13	31.96
10	1.74	32.00	3.71	31.25	34.96
11	1.66	29.09	3.61	34.38	37.98
12	1.59	26.67	3.52	37.50	41.02
13	1.53	24.62	3.46	40.63	44.08
14	1.49	22.86	3.40	43.75	47.15
15	1.45	21.33	3.35	46.88	50.23

K-ary N-Cube for const node

N	16384				
n	k	w	D	L/W	Tnet
1	16384.00	320.00	8191.50	3.13	8194.63
2	128.00	160.00	127.00	6.25	133.25
3	25.40	106.67	36.60	9.38	45.97
4	11.31	80.00	20.63	12.50	33.13
5	6.96	64.00	14.91	15.63	30.54
6	5.04	53.33	12.12	18.75	30.87
7	4.00	45.71	10.50	21.88	32.38
8	3.36	40.00	9.45	25.00	34.45
9	2.94	35.56	8.73	28.13	36.85
10	2.64	32.00	8.20	31.25	39.45
11	2.42	29.09	7.79	34.38	42.16
12	2.24	26.67	7.47	37.50	44.97
13	2.11	24.62	7.21	40.63	47.84
14	2.00	22.86	7.00	43.75	50.75
15	1.91	21.33	6.82	46.88	53.70

K-ary N-Tree for const node size

N	256

L 150
NodeSize 640

n	k	w	D	L/W	Tnet
1	256.00	1.25	2.00	120.00	122.00
2	16.00	20.00	4.00	7.50	11.50
3	6.35	50.40	6.00	2.98	8.98
4	4.00	80.00	8.00	1.88	9.88
5	3.03	105.56	10.00	1.42	11.42
6	2.52	126.99	12.00	1.18	13.18
7	2.21	144.92	14.00	1.04	15.04
8	2.00	160.00	16.00	0.94	16.94

K-ary N-Tree for const node size
N 16384
L 150
NodeSize 640

n	k	w	D	L/W	Tnet
1	16384.00	0.02	2.00	7680.00	7682.00
2	128.00	2.50	4.00	60.00	64.00
3	25.40	12.60	6.00	11.91	17.91
4	11.31	28.28	8.00	5.30	13.30
5	6.96	45.95	10.00	3.26	13.26
6	5.04	63.50	12.00	2.36	14.36
7	4.00	80.00	14.00	1.88	15.88
8	3.36	95.14	16.00	1.58	17.58
9	2.94	108.86	18.00	1.38	19.38
10	2.64	121.26	20.00	1.24	21.24
11	2.42	132.44	22.00	1.13	23.13
12	2.24	142.54	24.00	1.05	25.05
13	2.11	151.69	26.00	0.99	26.99
14	2.00	160.00	28.00	0.94	28.94

This table does not fully take into account the increased wire latencies of the fat tree, which in general must traverse much longer distances. In general, the worst-case wire length, assuming a hollow-cube construction [DeH90], is proportional to

$$l_{\max} \propto \sqrt[3]{k^n}$$

so as the number of terminals grows, the worst-case wire length grows much more rapidly in the fat tree.

k-ary n-cube latency with contention

Following [Dal90] and [Aga91], the latency of messages with contention due to randomly distributed traffic of different k-ary n-cube implementations of an N-node is given by

$$T_{contention\_latency} = T_{base\_latency} + T_c$$

where the total waiting time due to contention is derived in [Dal90]. We reproduce a numerical example from [Dal90] below to give the reader a sense of the latency-contention trends, but the reader is referred to [Dal90] for details of analysis.

k	n	Max throughput	Latency λ=0.1	Latency λ=0.2	Latency λ=0.3
2	12	.41	241	288	357
4	6	.36	135	181	287
8	4	.31	79.9	112	245
16	3	.31	55.2	70.3	135
64	2	.35	70.7	73.1	78.6

Fat Tree latency with contention

From [Lei85], the latency of a fat tree with the same traffic as an equivalent ($N_{fattree}=N_{cube}$) k-ary n-cube is provably within a polylogarithmic factor of the latency of that k-ary n-cube.

$$T_{latency,\,fattree} \propto L_{latency,k-ary,n-cube} \cdot \lg^3 N$$

However, Leiserson does not investigate the details of construction that make up this proportionality constant, which is critical for comparing real systems. For this investigation, we follow the analysis in [Dal90].

Following the assumption made above in the baseline latency calculations, we assume that for our worst-case latency, we need to pass upward through all levels of the fat tree before descending towards the destination endpoint. We assume a random uniform distribution of destinations for any given source node. Note that as a consequence, the probability of routing from a child-link to a parent-link is much higher towards the leaves of the tree, where the subset of nodes reachable by the sub-tree rooted at that level is small compared to the set of all possible destinations.

If we consider a k-way node on level *i* (*i*=0 at the leaf-level of the tree; *i*=*n* at the root level of the tree), we have the following probabilities:

Out of the k^{n-i} sub-trees at the current level i, k^{n-i} -1 sub-trees are reachable only via a parent link. The

probability of a message entering from a child link and exiting on a parent link (Pcp) is

$$P_{cpx} = \frac{k^{n-i} - 1}{k^{n-i}} \text{ (not normalized); } P_{cp} = \frac{P_{cpx}}{P_{cpx} + (k-1)P_{ccx}} \text{ (normalized)}$$

Out of the k^{n-i+1} sub-trees at the level below i, k sub-trees are reachable via a lateral route from child-node to child-node. The probability of a message entering from a child link and exiting on a child link (Pcc) is

$$P_{ccx} = \frac{k}{k^{n-i+1}} \text{ (not normalized); } P_{cc} = \frac{P_{ccx}}{P_{cpx} + (k-1)P_{ccx}} \text{ (normalized)}$$

Downward packets are evenly distributed among children links, since destinations are assumed to be uniformly distributed. The probability of a message entering from a parent link and exiting on a child link (Ppc) is

$$P_{pc} = \frac{1}{k}$$

Because of the asymmetry of probabilities, we consider the route into two parts: the up-route from the source to the root, and the down-route from the root to the destination. Probabilities of collision for each of these routes are

$$P_{collision\_up} = 1 - (1 - P_{cp})^{k-1}$$

$$P_{collision\_down} = 1 - (1 - P_{cd})^{k-1} P_{pd}$$

We assume a blocking flow control in which the rate of packets λ is equal across all logical links, and compute the effective waiting time T_w, which we add to the baseline latency to give us the resultant latency for a given λ.

To a first-order approximation, the expected waiting time given a collision is

$$E(T_w \mid collision) = T_{i+1} / 2$$

The waiting time of a message each level up is

$$T_{w\_up,i} = P_{collision\_up} E(T_w \mid collision)\ lT_{up,i+1}$$

and each level down is

$$T_{w\_down,i} = P_{collision\_down} E(T_w \mid collision)\ lT_{down,i-1}$$

Waiting time at each node is given by

$$T_{up,i} = T_{up,i+1} T_{w\_up,i}$$

$$T_{down,i} = T_{down,i-1} T_{w\_down,i}$$

$$T_{down,n} = T_{up,n}$$

The network saturates when the duty factor is one.

$$lT_{up,0} = 1$$

$$l_{sat} = \frac{1}{T_{up,0}}$$

Total time waiting due to contention is

$$T_c = \sum_{i=0}^{n} T_{w\_up,i} + T_{w\_down,i}$$

and latency becomes

$$T_{latency,contention} = T_{latency,baseline} + T_c$$

We observe that P_{cp} is close to 1 for levels near the bottom, while P_{pc} is low at 1/k, so contention for upward paths is high compared to the downward paths. We see that waiting time is dominated by contention for upward channels, and can be improved by increasing growth rate of the link bandwidth towards the root. In [DeH90], DeHon proposes a hybrid fat-tree by changing the linear upward path into a tree-like structure and using a greater link-width growth rate to reduce hop-count and upward contention. However, as the network is scaled, the bottleneck is still the upward links toward the root.

Conclusions

k-ary n-cubes Versus Fat Trees

- Our calculations show that the fat tree is competitive with, and in some cases achieves lower latencies than, the k-ary n-cube. However, the routing lengths grow much more rapidly in the fat tree compared

to the k-ary n-cubes, as the number of endpoints grows. This effect is a key limiter to the performance of fat trees for larger *N*.

- Logical nodes of fat trees are nonuniform in capacity or connectivity (depending on whether fat trees or hypertrees are used) across the network, which results in some non-uniformity (and potentially added cost) in construction. Low-dimension cubes map naturally into a 2- or 3- dimensional system of boards and chips.
- Routing decisions in Fat Trees are less complex than routing decisions in k-ary n-cubes, which may result in faster node latencies for Fat Trees.
- Given the comparable first-order performance of k-ary n-cubes and fat trees, constraints of physical implementation (effects like wire delay and packaging on practical bandwidth and latency) indicate that as networks scale in size, the k-ary n-cube topologies will outperform Fat Trees.

Comments on Papers

- When examining latency, Leiserson [Lei85] assumes off-line scheduling in which the traffic pattern is known beforehand, and can be optimally scheduled. A subsequent paper by Greenberg and Leiserson proves a similar latency relationship for on-line scheduling.
- Lei85 paper takes a very different approach than the k-ary n-cube papers as it tries to prove communication properties of fat trees in theoretical sense, as opposed to finding performance metrics. Since its a dated paper, it uses dated technique (store and forward flow control, SAF), and assumes that an arbitrary SAF zero load message flight time is the unit measure of time. To be fair, it acknowledges that it is not an implementation paper, and leaves that for future research.
- Cost-Performance: From the literature surveyed by this paper, the high-performance interconnection field is lacking a universal methodology for measuring cost-performance of different network topologies in a uniform way. Ultimately of interest to network designers is an analysis of performance of different topologies on an equal-cost basis, or equivalently an analysis of cost of different topologies on an equal-performance basis. Dally in [Dal90] takes an equal-cost approach by introducing a constant bisection bandwidth constraint. Later papers by [Aga91] and [Sco94] introduce a constant node-size constraint to model packaging limitations, but consider each these constraints separately and do not compare networks across these constraints using equal cost or equal performance. Lei[85] uses

a mathematical proof to show the cost-performance relationship between fat-trees and arbitrary other networks, but his assumptions of volume-universality do not address package interface constraints.

- Modeling Assumptions: Network modeling assumptions vary widely from paper to paper, which largely reflects evolutions in packaging, signaling and switching technology, new ideas in routing and flow control, and application-specific focus. While each new innovation is modeled in comparison with a prior baseline model, the lack of a uniform baseline set of assumptions makes it difficult to compare the cost-performance benefit of innovations.
- Performance metrics: Performance metrics used in each paper varies widely, making direct performance comparisons of different topologies and different innovations difficult. [Dal90] uses Chaos Normal Format for the set of networks investigated, while [Aga91] reports latency with respect to network request rate. [Lei85] uses a mathematical proof to show that performance of all universal fat trees are within a poly-logarithmic factor of any other equivalent-cost network. [DeH90] derives analytical models of latency and analyses performance in terms of probability of success given offered traffic rates. Given this non-uniform body of data, it is not possible to directly compare the merits of one network with another.

References

k-ary n-cube references

[Dal90] Dally, William J. "Performance analysis of k-ary n-cube Interconnection Networks". *IEEE Transactions on Computers.* Vol. 36 No. 5, June 1990, pp 547-553.

[Aga91] Agarwal, Anant. Limits on Interconnection Network Performance. *IEEE Transactions on Parallel Distributed Systems*, October 1991. pp 398-412.

[Sco94] Scott, Steven L. and Goodman, James R. "The Impact of Pipelined Channels on k-ary n-Cube Networks". *IEEE Transactions on Parallel and Distributed Systems.* Vol. 5 No. 1, January 1994.

Fat tree references

[Lei85] Leiserson, Charles E. "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing". IEEE Transactions on Computers. Vol. 34 No 10, October 1985, pp 892-901.

[DeH90] DeHon, Andre. "Fat-Tree Routing for Transit." *MIT, A.I. Technical Report No. 1224*. February 1990.