Lecture 10: Fat-tree and Dragonfly Networks

Abhinav Bhaele, Department of Computer Science
Summary of last lecture

- Key requirements of HPC networks
  - extremely low latency, high bandwidth, scalable
  - low network diameter, high bisection bandwidth

- Torus networks (less common now)
  - Network diameter grows as $O(\sqrt[3]{N})$ where N is the number of nodes

- Different types of routing algorithms:
  - Shortest path vs. non-minimal
  - Static vs. dynamic
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth

Router/switch radix = number of ports = k

Compute Nodes
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth

Router/switch radix = number of ports = k
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth

Router/switch radix = number of ports = $k$
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth

Router/switch radix = number of ports = $k$
Fat-tree network

- Most popular network topology

- Low network diameter, high bandwidth

Router/switch radix = number of ports = k
Pod = group of switches = k/2 switches
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth

Router/switch radix = number of ports = k
Pod = group of switches = k/2 switches
Fat-tree network

- Most popular network topology
- Low network diameter, high bandwidth

Router/switch radix = number of ports = k
Pod = group of switches = k/2 switches
Fat-tree network

- Most popular network topology
  - Low network diameter, high bandwidth

Router/switch radix = number of ports = $k$
Pod = group of switches = $k/2$ switches
Max. number of pods = $k$
Fat-tree networks on the top500 list

- Infiniband EDR/FDR
- Intel Omni-Path

https://www.top500.org/statistics/list/
Routing on a fat-tree

- Until recently, most fat-tree installations used static routing
  - Destination-mod-k (D-mod-k) routing
- Adaptive routing is now starting to be used
Variations on a full bandwidth fat-tree

Single-rail single-plane fat-tree
Variations on a full bandwidth fat-tree

Single-rail single-plane fat-tree (tapered)
Variations on a full bandwidth fat-tree

Dual-rail single-plane fat-tree
Variations on a full bandwidth fat-tree

Single-rail single-plane fat-tree
Variations on a full bandwidth fat-tree

Dual-rail dual-plane fat-tree
Dragonfly network
IBM PERCS network

- All-to-all connections within each group

One supernode in the PERCS topology
IBM PERCS network

- All-to-all connections within each group

One supernode in the PERCS topology
Cray Aries network

- Row and column all-to-all connections within each group

Aries Router

Compute Nodes

Abhinav Bhaetele, CMSC714
Cray Aries network

- Row and column all-to-all connections within each group

![Diagram of Cray Aries network](image)
Cray Aries network

- Row and column all-to-all connections within each group

Figure 3 illustrates a four-group Cray Cascade installation. Ninety-six routers are connected together to form a group, with six routers in each column connected in an all-to-all manner by links. Routers in different groups are connected together via inter-group blue links leading to a shortest path of at most five hops.

In adaptive routing schemes, routers can dynamically choose between multiple paths for routing to minimize hotspots. In the destination mod k or D-mod-k algorithm, the next upward link in the path is chosen at each step, leading to a sequence of links going down to the destination node. A commonly-used static routing algorithm is the nearest common ancestor, followed by a sequence of links.

Traffic in current fat-tree networks is usually forwarded using a static routing algorithm, meaning that all messages between a given pair of nodes take the same (shortest) path through the fat-tree every time. Each path consists of at most three routers, so if there are no conflicts, a message is routed in at most two hops (on the black and/or green links). If congestion does not exist, some paths are minimal in the number of routers. However, if congestion occurs, this scheme can fail. When multiple applications are using non-local communication patterns, network contention can occur regardless. For example, contention can occur for the global links if multiple applications execute concurrently and contend for shared resources. In this work, we study the effects of network contention on fat-tree and dragonfly machines. While the fat-tree topology has been shown to have good performance, more recent approaches such as the dragonfly topology have become popular for interconnection networks in post-petascale supercomputing. Applications can be assigned to the same routers within a group, and even if both have localized (e.g., nearest-neighbor) communication patterns, traffic that is routed indirectly through a given group can also conflict with jobs scheduled to that group on the local links. As shown in Section I, jobs in HPC systems typically compete for the shared system interconnect, degrading communication performance. In certain architectures, for example the IBM Blue Gene machines, jobs are always placed so that such placements might lead to system fragmentation and hence lowered system utilization, and most modern machines are designed to mitigate real-time congestion.

Therefore, inter-switch traffic belonging to different jobs may need to be distributed across the network to avoid congestion. As mentioned above, dragonfly machines typically use a rank-2 or rank-1 network topology, which does not suffer from this problem and is believed to have good performance. In this scheme, the next upward link in the path is chosen at each step, leading to a sequence of links going down to the destination node. A commonly-used static routing algorithm is the nearest common ancestor, followed by a sequence of links.

In adaptive routing schemes, routers can dynamically choose between multiple paths for routing to minimize hotspots. In the destination mod k or D-mod-k algorithm, the next upward link in the path is chosen at each step, leading to a sequence of links going down to the destination node. A commonly-used static routing algorithm is the nearest common ancestor, followed by a sequence of links.
Network comparisons

<table>
<thead>
<tr>
<th>Network topology</th>
<th>#nodes/router</th>
<th>#links/router</th>
<th>Maximum system size (#nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All-to-all (A2A) dragonfly</td>
<td>k/4</td>
<td>k/2 (L), k/4 (G)</td>
<td>$(k/2 + 1)^2 \times (k/4 + 1) \times k/4$</td>
</tr>
<tr>
<td>Row-column (RC) dragonfly</td>
<td>k/6</td>
<td>2k/3 (L), k/6 (G)</td>
<td>$(k/3 + 1)^4 \times (k/6 + 1) \times k/6$</td>
</tr>
<tr>
<td>Express mesh (3D, gap=1)</td>
<td>k/4</td>
<td>3k/4</td>
<td>$(k/4 + 1)^3 \times k/4$</td>
</tr>
<tr>
<td>Fat-tree (three-level)</td>
<td>k/2</td>
<td>k/2</td>
<td>$k/2 \times k/2 \times k$</td>
</tr>
</tbody>
</table>

![System size comparison](image)
Questions

Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing

- How do you use a partial concentrator graph to construct a good concentrator switch?

- The paper says the capacities of the channels of a universal fat-tree grow exponentially as we go up the tree from the leaves. If so, we must have a large number of wires for the top layers in a big fat tree, which may lead to higher costs in my view. So how can we manage the costs of building a fat-tree network?

- How does fat tree compare with the dragonfly network? Under what kind of circumstance, we prefer one to another?
Questions

Technology-Driven, Highly-Scalable Dragonfly Topology

• It's said in figure 6(b), the effective radix is 32, which I understand as \( a=8, \ p=2, \ h=2 \) and \( k'=a(p+h)=32 \). But it says the radix of each router \( k=7 \), which I don't get it. According to the formula, \( k \) should be \( a+p+h-1=11 \). So why does it say \( k=7 \) here?

• In the part introducing the credit round-trip latency technique, it says "the credit is delayed by \( td(O) - \min[td(o)] \). Where does the little \( o \) come from?

• Is there any hardware technology that supports advanced congestion look ahead nowadays?
Questions?

Abhinav Bhatele
5218 Brendan Iribe Center (IRB) / College Park, MD 20742
phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu