Interconnection Network Design

Vida Vukačinović

1 Introduction

Parallel computer networks are interesting topic, but they are also difficult to understand in an overall sense. The topological structure of these networks has elegant mathematical properties, and there are deep relationships between these topologies and the fundamental communication patterns of important parallel algorithms. In this paper we would like to provide a holistic understanding of parallel computer networks and its topologies.

2 Basic Definitions

We would firstly provide a set of basic definitions that underlie all networks. Formally, a parallel machine interconnection network is a graph, where the vertices $V$ are processing hosts or switch elements connected by communication channels $C \subseteq V \times V$. A channel is a physical link between host or switch elements, including a buffer$^1$ to hold data as it is being transferred. It has a width $w$ and a signaling rate $f = \frac{1}{\tau}$ (for cycle time $\tau$), which together determine the channel bandwidth$^2$ $b = wf$. The amount of data transferred across a link in a cycle is called a physical unit, or phit. Switches connect a fixed number of input channels to a fixed number of output channels. This number is called the switch degree. Hosts typically connect to a single switch but can be multiply connected with separate channels. Messages are transferred through the network from a source host node to a destination host node along a path, or route, comprised of a sequence of channels and switches.

A network is characterized by its topology, routing algorithm, switching strategies and flow control mechanism.

- The topology is the physical interconnection structure of the network graph.

- The routing algorithm determines which routes messages may follow through the network graph. In a parallel machine, we are only concerned with the routes from a host to a host. The routing algorithms restrict the set of possible paths to a smaller set of legal paths. There are many different routing algorithms which are offering different performance trade-offs.

- The switching strategy determines how the data in a message traverses its route. There are basically two switching strategies. In circuit switching, the path from the source to the destination is established and reserved until the message is transferred over the circuit. In packet switching the message is broken into a sequence of packets. Each packet contains routing and sequencing information as well as data. Packets are individually routed from the source to the destination.

---

$^1$Buffer is a temporary location to store or group information in hardware or software. Buffers are used whenever data is received in sizes that may be different than ideal size.

$^2$Bandwidth refers to how much data is transmitted over a given period of time. It is measured in bits per second in digital devices and in Hertz in analog devices.
The flow control mechanism determines when the message, or portion of it, moves along its route. Flow control is necessary whenever two or more messages attempt to use the same channel at the same time.

An important property of a topology is the diameter of the network, which is the length of the maximum shortest path between any two nodes. The routing distance between a pair of nodes is the number of links traversed en route. This is at least as large as the shortest path between the nodes and it may be larger. The average distance is simply the average of the routing distance over all pairs of nodes and this is also the expected distance between a random pair of nodes. A network is partitioned if a set of links or switches are removed such that some hosts are no longer connected by routes.

Figure 1: Typical packet format

A packet consists of three parts, illustrated in Figure 1, a header, a payload and a trailer. The header is the front of the packet and usually contains the routing and control information so that the switches and network interface can determine what to do with the packet as it arrives. The payload is the part of the packet containing data transmitted across the network. The trailer is the end of the packet and typically contains the error-checking code.

3 Basic Communication Performance

It is useful to understand how four major aspects of network design (topology, routing algorithm, switching strategy and flow control) interact to determine the performance and functionality of the overall communication subsystem. Some of these interactions are explained.

3.1 Latency

The time to transfer \( n \) bytes of information from its source to its destination has four components as follows.

\[
\text{Time}(n)_{S-D} = \text{Overhead} + \text{Routing Delay} + \text{Channel Occupancy} + \text{Contention Delay}
\]

The \textit{Overhead} is associated with getting the message into and out of the network on the ends of the actual transmission. The \textit{Channel Occupancy} is associated with each step along the route. The occupancy of each link is influenced by the channel width, the signaling rate and the amount of the control information, which is in turn influenced by the topology and routing algorithm. The \textit{Routing delay} refers to the length of time required to move a message from source to destination through the network. It is a function of the routing distance \( h \) and the delay \( \Delta \) incurred at each switch as part of selecting the correct output port. The overall delay is strongly affected by switching and routing strategies.

With packet-switched, \textit{store-and-forward} routing, the entire packet is received by a switch before it is forwarded on the next link, as illustrated in Figure 2. This strategy is used in most wide area networks and was used in several early parallel computers. The unloaded network latency for an \( n \)-byte packet with store-and-forward routing is

\[
T_{sf}(n, h) = h \left( \frac{n}{b} + \Delta \right)
\]
Figure 2: Store-and-forward and cut-through routing for packet-switched networks. A four-flit packet traverses three hops from source to destination under store-and-forward and cut-through routing. Cut-through achieves lower latency by making the routing decision (gray arrow) on the first flit and pipelining the packet through a sequence of switches. Store-and-forward accumulates the entire packet before routing it toward the destination.

where $b$ is the raw bandwidth of channel and $\Delta$ is the additional routing delay per hop. With circuit switching, we expect a delay proportional to $h$ to establish the circuit. After this time, the data should move along the circuit in time $\frac{n}{b}$ plus an additional small delay proportional to $h$. Thus, the unloaded latency in units of the network cycle time $\tau$ for an $n$-byte message traveling distance $h$ in a circuit-switched network is

$$T_{CS}(n, h) = \frac{n}{b} + h\Delta.$$  

In this case, the setup and routing delay is independent of the size of the message. Thus, circuit switching is traditionally used in telecommunications networks since the call setup is short compared to the duration of the call.

It is also possible to retain packet switching and reduce the unloaded latency from that store-and-forward routing, where the delay is the product of the routing distance and the occupancy for the full message. A long message can be fragmented into several small packets, which flow through the network in a pipelined fashion. In this case the effective routing delay is proportional to the packet size rather than the message size. This is the approach adopted for traditional data communication networks such as the internet.

In parallel computer networks, the idea of pipelining the routing is carried much further. With packet switched, cut-through routing, the switch makes its routing decision after inspecting only the first few flits of the header and allows the remainder of the packet to cut-through from the input channel to the output channel, as indicated by Figure 2. In this case, the transmission of even a single packet is pipelined. For cut-through routing, the unloaded latency has a form similar to the circuit switch case, although the routing coefficient $\Delta$ may differ.

$$T_{ct}(n, h) = \frac{n}{b} + h\Delta$$

With cut-through routing a single message may occupy the entire route from the source to the destination, much like circuit switching. The head of the message establishes the route as it moves toward its destination, and the route clears as the tail moves through.

Of course, the reason that networks are so interesting is that they are not simple pipeline but a big system of many pipelines. The whole motivation for using a network rather than bus is to allow multiple data transfers to occur simultaneously. This means that one message flow
may collide with others and contend for resources. If two messages attempt to use the same channel at once, one must be deferred. Clearly, contention increases the communication latency. Exactly how it increases latency depends on mechanisms used for dealing with contention within the network, which in turn differ depending on basic network design strategy. For example, with packet switching, contention may be experienced at each switch. Switches have limited buffering for packets, so under sustained contention the buffers in a switch may fill up. In traditional data communication networks, the links are long and little feedback occurs between the two ends of the channel, so the typical approach is to discard the packet. Thus, under contention the network becomes highly unreliable, and sophisticated protocols are used at the nodes to adapt the requested load to what the network can deliver without a high loss rate.

In parallel computer networks, a packet headed for a full buffer is typically blocked in place, rather than discarded. This requires a handshake between the output port and input port across the link, called link-level flow control. Under sustained congestion, the sources experience back pressure from the network (when it refuses to accept packets), which causes the flow of data into the network to slow down to a rate that can move through the bottleneck. Increasing the amount of buffering within the network allows contention to persist longer without causing back pressure at the source, but it also increases the potential queuing delays within the network when contention does occur.

We can see that all aspects of network design; link bandwidth, topology, switching strategy, routing algorithm, and flow control; combine to determine the latency per message. It is also clear that a relationship exists between latency and bandwidth. If the communication bandwidth demanded by the program is low compared to the available network bandwidth, collisions will be few, buffers will tend to be empty and latency will stay low. As the bandwidth demand increases, latency will increase due to contention.

### 3.2 Bandwidth

Network bandwidth is critical to parallel program performance, in part because higher bandwidth decreases occupancy, in part because higher bandwidth reduces the likelihood of contention and in part because a large volume of data may be pushed around without waiting for transmission of individual data items to be completed.

Bandwidth can be looked from two points of view; the global aggregate bandwidth available to all the nodes through the network and the local individual bandwidth available to a node. If the total communication volume of a program is \( M \) bytes and the aggregate communication bandwidth of the network is \( B \) bytes per second, then clearly the communication time is at least \( \frac{M}{B} \) seconds. On the other hand, if all of the communication is to or from a single node, this estimate is far too optimistic and the communication time would be determined by the bandwidth through that single node.

If many of the nodes are communicating at once, it is useful to focus on the global bandwidth that the network can support rather than only the bandwidth available to each individual node. The most common notion of aggregate bandwidth is the *bisection bandwidth* of the network, which is the sum of the bandwidths of the minimum set of channels that, if removed, partition the network into two equal unconnected sets of nodes. This is a valuable concept because, if the communication pattern is completely uniform, half of the messages are expected to cross the bisection in each direction. However communication is not necessarily distributed uniformly over the entire machine. If communication is localized rather than uniform, bisection bandwidth will give a pessimistic estimate of communication time. An alternative notion of global bandwidth in this case would be the sum of the bandwidth of the links from the nodes to the network. The concern with this notion of global bandwidth is that internal structure of the network depends on the communication pattern (in particular, it depends on how far the packets travel).

A lot of factors affect the network’s *saturation point*, which represents the total channel
bandwidth it can usefully deliver. As illustrated in Figure 3, if the offered bandwidth (bandwidth demand placed on the network by the processor) is moderate, the latency remains low and the delivered bandwidth increases with the offered bandwidth. At some point, demanding more bandwidth only increases the contention for resources and the latency increases dramatically. The network is essentially moving as much traffic as it can, so additional requests just get queued up in the buffers. Increasing offered bandwidth does not increase what is delivered. Parallel machines are tried to be designed so that the network stays out of saturation, either by providing ample communication bandwidth or by limiting the demands placed by the processors.

4 Organizational Structures

This section outlines the basic organizational structure of a parallel computer network. Scalable interconnection networks are composed of three basic components: links, switches and network interfaces. A basic understanding of these components, their performance characteristics and their inherent costs is essential for evaluating network design alternatives.

4.1 Links

A link is a cable of one or more electrical wires or optical fibers with a connector at each end that is attached to a switch or network interface port. It allows an analog signal to be transmitted from one end and received at the other. Essential logical properties of the links can be characterized along three independent dimensions; length, width and clocking.

- A short link is one in which only a single logical value can be on the link at any time. A long link is viewed as transmission line where a series of logical values propagate along the link simultaneously at a fraction of the speed of light.

- A narrow link is one in which data, control and timing information are multiplexed onto each wire, such as on a single serial link. A wide link is one that can simultaneously transmit data and control information. In either case, network links are typically narrower than to internal processor datapaths.

- Clocking may be synchronous or asynchronous. In the synchronous case, the source and destination operate on the same global clock. In the asynchronous case, the source encodes its clock in some manner within the analog signal that is transmitted and the destination recovers the resource clock from the signal and transfers the information into its own clock domain.
4.2 Switches

A switch consists of a set of input ports, a set of output ports, an internal crossbar connecting each input to every output, internal buffering and control logic to effect the input/output connection at each point in time. Usually the number of input ports is equal to the number of output ports, which is called the degree of the switch. Each output port includes a transmitter to drive the link. Each input port includes a matching receiver. The input port has a synchronizer in most designs to align the incoming data with the local clock domain of the switch. The complexity of the control logic depends on the routing algorithm. Many evaluations of network treat the switch degree as its cost. This is clearly a major factor, but although the cost of some parts of the switch is linear in the degree (transmitters, receivers and port buffers), the internal interconnect cost may increase with square of the degree.

4.3 Network Interfaces

The network interface contains one or more input/output ports. The network interface, or host nodes, behave quite differently than switch nodes and may be connected via special links. The network interface formats the packets and construct the routing and control information. It may perform end-to-end error checking and flow control. Its cost is influenced by its storage capacity, processing complexity and number of ports.

5 Interconnection Topologies

This section covers the set of important interconnection topologies. Each topology is a class of network scaling with the number of host nodes $N$, so the key characteristics of each class are understood as a function of $N$. The main characteristics of these topologies are listed in the table 1 at the end of this section.

5.1 Fully Connected Network

A fully connected network is essentially a single switch, which connect all inputs to all outputs. The diameter is 1 link. The degree is $N$. The loss of the switch wipes out the whole network. One such network is simply a bus. Bus has the nice property that the cost scales as $O(N)$. Unfortunately, only one data transmission can occur on a bus at once, so the total bandwidth is $O(1)$, as is the bisection. Another fully connected network is a crossbar. It provides $O(N)$ bandwidth, but the cost of the interconnect is proportional to the number of cross-points $O(N^2)$. An example are individual switches, which are often fully connected internally and provide the basic building blocks for larger networks.

5.2 Linear Arrays and Rings

A simple network is a linear array of nodes numbered 0, . . . , $N − 1$ and connected by bidirectional links. The diameter is $N − 1$, the average distance is roughly $\frac{2}{3}N$ and the removal of a single link partitions the network, so the bisection width is 1 link. Routing in such a network is trivial since there is exactly one route between any pair of nodes. Since there is a unique route between a pair of nodes, clearly the network provides no fault tolerance. The network consists of $N − 1$ links and can easily be laid out in $O(N)$ space using only short wires.

A ring or torus of $N$ nodes can be formed by simply connecting the two ends of an array. With unidirectional links, the diameter is $N − 1$, the average distance is $\frac{N}{2}$, the bisection width is 1 link and there is one route between any pair of nodes. With bidirectional links, the diameter is $\frac{N}{2}$, the average distance is $\frac{N}{4}$, the degree of the node is 2 and the bisection is 2. There are two routes (two relative addresses) between pairs of nodes, so the network can function with
degraded performance in the presence of a single fault link. The network is easily laid out with \( O(N) \) space using only short wires, as indicated by Figure 4.

![Linear array](Figure 4: Linear and ring topologies.)

Although these one-dimensional networks are not scalable in any practical sense, they are an important building blocks conceptually and in practice.

5.3 Multidimensional Meshes and Tori

A \( d \)-dimensional array consists of \( N = k_{d-1} \cdot k_{d-2} \cdot \ldots \cdot k_0 \) nodes, each identified by its \( d \)-vector of coordinates \((i_{d-1}, i_{d-2}, \ldots, i_0)\), where \( 0 \leq i_j \leq k_j - 1 \) for \( 0 \leq j \leq d - 1 \). Figure 5 shows the common cases of two and three dimensions. For simplicity we will assume the length along each dimension is the same, so \( N = k^d \) (\( k = \sqrt[d]{N} \)). This is called a \( d \)-dimensional \( k \)-ary mesh. Each node is a switch addressed by a \( d \)-vector of radix \( k \) coordinates and is connected to the nodes that differ by one in precisely one coordinate. The node degree varies between \( d \) and \( 2d \) (nodes in the middle have full degree and the corners have minimal degree).

![Figure 5: Mesh, torus and cube topologies.](image)

For a \( d \)-dimensional \( k \)-ary torus, every node has degree \( 2d \) (in the bidirectional case) and is connected to nodes differing by one \( \text{mod} \ k \) in each dimension.

The \( d \)-dimensional \( k \)-ary unidirectional torus is a very important class of networks, often called a \( k \)-ary \( d \)-cube, employed widely in modern parallel machines.

To define the routes from node \( A \) to node \( B \) in a \( d \)-dimensional array, let \( R = (b_{d-1} - a_{d-1}, b_{d-2} - a_{d-2}, \ldots, b_0 - a_0) \) be the relative address of \( B \) from \( A \). A route must cross \( r_i = b_i - a_i \) links in each dimension \( i \), where the sign specifies the appropriate direction. In general, we may view the source and destination points as corners of subarray and follow any path between the corners that reduces the relative address from the destination at each hop.

The diameter of the mesh is \( d(k-1) \). The average distance is simply the average distance in each dimension, roughly \( d^2 k \). If \( k \) is even, the bisection of a \( d \)-dimensional \( k \)-ary array is \( k^{d-1} \) bidirectional links. This is obtained by simply cutting the array in the middle by a (hyper)plane perpendicular to one of the dimensions. (If \( k \) is odd, the bisection may be a little bit larger.)

It is clear that a two-dimensional mesh can be laid out with \( O(N) \) space in a plane with short wires and three-dimensional mesh in \( O(N) \) volume in free space.

5.4 Trees

In meshes, the diameter and average distance increases with the \( d \)th root of \( N \). Many other topologies exist where the routing distance grows only logarithmically. The simplest of these is a tree. A binary tree has degree 3. Typically, trees are employed as indirect networks with hosts as the leaves, so for \( N \) leaves the diameter is \( 2 \log N \). In this case, we may treat the binary address of each node as a \( d = \log N \) bit vector specifying a path from the root and so on down the levels of
the tree. The levels of the tree correspond directly to the "dimension" of the network. Formally, a complete indirect binary tree is a network of \(2N - 1\) nodes organized as \(d + 1 = \log_2 N + 1\) levels. Host nodes occupy level 0 and are identified by a \(d\)-bit address \(A = a_{d-1}, \ldots, a_0\). A switch node is identified by its level \(i\) and its \(d - i\) bit address \(A^{(i)} = a_{d-1}, \ldots, a_i\). A switch node \([i, A^{(i)}]\) is connected to a parent \([i + 1, A^{i+1}]\) and two children \([i - 1, A^{i}[0]]\) and \([i - 1, A^{i}[1]]\) where the vertical bars indicate bitwise concatenation. There is a unique route between any pair of nodes by going up to the least common ancestor, so no fault tolerance is present. The average distance is almost as large as the diameter. The bisection width of the tree is 1 link. Removing a single link near the root bisects the network. One virtue of the tree is the ease of supporting broadcast of multicast operations from one node to many.

![Figure 6: Binary trees.](image)

Clearly, by increasing the branching factor of the tree the routing distance is reduced. In a \(k\)-ary tree, each node has \(k\) children, the height of the tree is \(d = \log_k N\), and the address of a host is specified by a \(d\)-vector of radix \(k\) coordinates describing the path down from the root.

One potential problem with trees is that they seem to require long wires. After all, when we draw a tree in a plane, the lines near the root usually grow exponentially in length with the number of level as illustrated in the top of the Figure 6. In the Figure 6 is also illustrated how is the problem solved for 16-node tree.

### 5.5 Butterflies

The constriction at the root of the tree can be avoided if there are "lots of roots". This is provided by an important logarithmic network called a butterfly. Given \(2 \times 2\) switches, the basic building block of the butterfly is obtained by simply crossing one of each pair of edges, as illustrated in the Figure 7. This is a tool for correcting one bit of relative address (going straight leaves the bit the same, crossing flips the bit). These \(2 \times 2\) butterflies are composed into a network of \(N = 2^d\) nodes in \(\log_2 N\) levels of switches by systematically changing the cross edges as shown by the 16-node butterfly illustrated in Figure 7. This configuration shows an indirect network with unidirectional links going upward so that hosts deliver packets into level 0 and receive packets from level \(d\). Each level corrects one additional bit of the relative address. Each node at level \(d\) forms the root of a tree with all the hosts as leaves and from each host is a tree of routes reaching every node at level \(d\).

A \(d\)-dimensional indirect butterfly has \(N = 2^d\) host nodes and \(d2^{d-1}\) switch nodes of degree 2. The diameter is \(\log N\). In fact, all routes are \(\log N\) long. The bisection is \(N/2\). There is exactly one route from each host output to each host input, so no fault tolerance is present in the basic topology. But, there are many proposals for making the butterfly fault tolerant by just adding a few extra links. One simple approach is to add an extra level to the butterfly so there are two routes to every destination from every source.
The butterfly appears to be a qualitatively more scalable network than meshes and trees because each packet crosses $\log N$ links and there are $N \log N$ links in the network, so on average it should be possible for all the nodes to send messages anywhere all at once.

The second problem is that even though the butterfly is thought to have enough links, the topology of the butterfly will not allow an arbitrary permutation of $N$ messages among the $N$ nodes to be routed without conflict. A path from an input to an output blocks the paths of many other input/output pairs because there are shared edges. However, if two butterflies are laid back to back so that a message goes forward through one and in the reverse direction through the other, then for any permutation there exists a choice of intermediate positions that allows a conflict-free routing of the permutation. This back-to-back butterfly is called a Benes network (Benes 1965; Leighton 1992) and it has been extensively studied because of its elegant theoretical properties. It is often seen as having little practical significance because it is costly to compute the intermediate positions and permutations has to be known in advance. On the other hand, there is another interesting theoretical result that says that on a butterfly any permutation can be routed with high probability by first sending every message to a random intermediate node and then routing the messages to the desired destination (Leighton 1992). These two results come together in a very nice practical way in the fat-tree network.

A $d$-dimensional $k$-ary fat-tree is formed by taking a $d$-dimensional $k$-ary Benes network and folding it back on itself at the high-order dimension, as illustrated in Figure 7. The collection of
switches at level $i$ is viewed as $N^{d-i}$ “fat nodes” of $2^{i-1}$ switches. The edges of the forward-going butterfly go up the tree toward the roots and the edges of the reverse butterfly go down toward the leaves. The route from $A$ to $B$, pick a random node $C$ in the least common ancestor fat node of $A$ and $B$ and take the unique tree route from $A$ to $C$ and unique route back down from $C$ to $B$. Let $i$ be the highest dimension of difference in $A$ and $B$. Then there are $2^i$ root nodes to choose from, so the longer the routing distance the more traffic can be distributed. This topology has the bisection of the butterfly, the partitioning properties of the tree and allows essentially all permutations to be routed with very little contention.

5.6 Hypercubes

It may seem that the straight edges in the butterfly are target for potential optimization since they take a packet forward to the next level in the same column. Consider what happens if we collapse all the switches in a column into a single log $N$ degree switch. this has brought us full circle. It is a $d$-dimensional 2-ary torus. Actually, we need to split the switches in a column in half and associate them with the two adjacent nodes. It is called a hypercube. Each of the $N = 2^d$ nodes is connected to the $d$ nodes that differ by exactly one bit in address. The relative address specifies the dimensions that must be crossed to go from $A$ to $B$. The length of the route is equal to the number of ones in the relative address. The dimensions can be corrected in any order. The butterfly routing corresponds exactly to dimension order routing. Fat-tree routing corresponds to picking a random node in the subcube defined by the high-order bit in the relative address, sending the packet ”up” to the random node and back ”down” to the destination. Observe that the fat-tree uses distinct sets of links for the two directions, so to get the same properties we need a pair of bidirectional links between nodes in the hypercube.

<table>
<thead>
<tr>
<th>Network topology</th>
<th>diameter</th>
<th>degree</th>
<th>bisection (links)</th>
<th>average distance</th>
<th>routs</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear array</td>
<td>$N - 1$</td>
<td>1-2</td>
<td>1</td>
<td>$\frac{1}{2}N$</td>
<td>1</td>
</tr>
<tr>
<td>unidirected ring</td>
<td>$N - 1$</td>
<td>2</td>
<td>1</td>
<td>$\frac{N}{2}$</td>
<td>1</td>
</tr>
<tr>
<td>bidirected ring</td>
<td>$\frac{N}{2}$</td>
<td>2</td>
<td>2</td>
<td>$\frac{N}{2}$</td>
<td>2</td>
</tr>
<tr>
<td>$d$-dim. $k$-ary mesh</td>
<td>$d(k - 1)$</td>
<td>$d-2d$</td>
<td>$k^{d-1}$</td>
<td>$d^2_k$</td>
<td></td>
</tr>
<tr>
<td>uni. $d$-dim. $k$-ary torus</td>
<td>$d(k - 1)$</td>
<td>$2d$</td>
<td>$k^{d-1}$</td>
<td>$d^2_k$</td>
<td></td>
</tr>
<tr>
<td>binary tree</td>
<td>$2\log_2 N$</td>
<td>3</td>
<td>1</td>
<td>$2 \log_2 N$</td>
<td>1</td>
</tr>
<tr>
<td>$d$-dim butterfly</td>
<td>$\log_2 N$</td>
<td>4</td>
<td>$N^{1/2}$</td>
<td>$\log_2 N$</td>
<td>1</td>
</tr>
<tr>
<td>Benes network</td>
<td>$2 \log_2 N$</td>
<td>4</td>
<td>$N^{1/2}$</td>
<td>$2 \log_2 N$</td>
<td>$N$</td>
</tr>
</tbody>
</table>

Table 1: The table of main network topologies and their characteristics.

6 Conclusion

Parallel computer networks present a rich and diverse design space. In network design, we would like to achieve that the solution is near optimum in a global sense rather than optimized for a particular component of interest. Nevertheless, no clean consensus exists in the field on the appropriate cost model for networks since trade-offs can be made between very different technologies. For example, bandwidth of the links may be traded against complexity of switches.

We know that a lot of basic questions like the degree of the switch, properties of the link, etc. depend heavily on the cost model of the network. The network is characterized by its topology which has been a point of lively debate over the history of parallel architectures. We know that once the degree of the switch is determined, the space of candidate networks is relatively constrained. For example, if our primary concern is the routing distance, then we would maximize the dimension and build a hypercube. This would be the case with store-and-forward routing, assuming that the degree of the switch and the number of links were not a
significant cost factor. However, with cut-through routing and a more realistic hardware cost model, the choice is much less clear. If the number of links or the switch degree is the dominant cost, we would minimize the dimension and built a mesh.

The assumed communication pattern influences the decision too. If we look at the worst case traffic pattern, we will prefer high dimensional networks where essentially all the paths are short. If we look at patterns where each node is communicating with only one or two near neighbors, we will prefer low dimensional networks since only few of the dimensions are actually used.

Basically, as with most aspect of architecture, we have to define the cost model and performance model and then optimize the design accordingly.

References