MITOCW | MIT6_004S17_20-02-06_300k

Let's wrap up our discussion of system-level interconnect by considering how best to connect N components that need to send messages to one another, e.g., CPUs on a multicore chip.

Today such chips have a handful of cores, but soon they may have 100s or 1000s of cores.

We'll build our communications network using point-to-point links.

In our analysis, each point-to-point link is counted at a cost of 1 hardware unit.

Sending a message across a link requires one time unit.

And we'll assume that different links can operate in parallel, so more links will mean more message traffic.

We'll do an asymptotic analysis of the throughput (total messages per unit time), latency (worst-case time to deliver a single message), and hardware cost.

In other words, we'll make a rough estimate how these quantities change as N grows.

Note that in general the throughput and hardware cost are proportional to the number of point-to-point links.

Our baseline is the backplane bus discussed earlier, where all the components share a single communication channel.

With only a single channel, bus throughput is 1 message per unit time and a message can travel between any two components in one time unit.

Since each component has to have an interface to the shared channel, the total hardware cost is O(n).

In a ring network each component sends its messages to a single neighbor and the links are arranged so that its possible to reach all components.

There are N links in total, so the throughput and cost are both O(n).

The worst case latency is also O(n) since a message might have to travel across N-1 links to reach the neighbor that's immediately upstream.

Ring topologies are useful when message latency isn't important or when most messages are to the component that's immediately downstream, i.e., the components form a processing pipeline.

The most general network topology is when every component has a direct link to every other component.

There are O(N**2) links so the throughput and cost are both O(N**2).

And the latency is 1 time unit since each destination is directly accessible.

Although expensive, complete graphs offer very high throughput with very low latencies.

A variant of the complete graph is the crossbar switch where a particular row and column can be connected to form a link between particular A and B components with the restriction that each row and each column can only carry 1 message during each time unit.

Assume that the first row and first column connect to the same component, and so on, i.e., that the example crossbar switch is being used to connect 4 components.

Then there are O(n) messages delivered each time unit, with a latency of 1.

There are N**2 switches in the crossbar, so the cost is O(N**2) even though there are only O(n) links.

In mesh networks, components are connected to some fixed number of neighboring components, in either 2 or 3 dimensions.

Hence the total number of links is proportional to the number of components, so both throughput and cost are O(n).

The worst-case latencies for mesh networks are proportional to length of the sides, so the latency is O(sqrt n) for 2D meshes and O(cube root n) for 3D meshes.

The orderly layout, constant per-node hardware costs, and modest worst-case latency make 2D 4-neighbor meshes a popular choice for the current generation of experimental multi-core processors.

Hypercube and tree networks offer logarithmic latencies, which for large N may be faster than mesh networks.

The original CM-1 Connection Machine designed in the 80's used a hypercube network to connect up to 65,536 very simple processors, each connected to 16 neighbors.

Later generations incorporated smaller numbers of more sophisticated processors, still connected by a hypercube network.

In the early 90's the last generation of Connection Machines used a tree network, with the clever innovation that the links towards the root of the tree had a higher message capacity.

Here's a summary of the theoretical latencies we calculated for the various topologies.

As a reality check, it's important to realize that the lower bound on the worst-case distance between components in our 3-dimensional world is O(cube root of N).

In the case of a 2D layout, the worst-case distance is O(sqrt N).

Since we know that the time to transmit a message is proportional to the distance traveled, we should modify our latency calculations to reflect this physical constraint.

Note that the bus and crossbar involve N connections to a single link, so here the lower-bound on the latency needs to reflect the capacitive load added by each connection.

The winner?

Mesh networks avoid the need for longer wires as the number of connected components grows and appear to be an attractive alternative for high-capacity communication networks connecting 1000's of processors.

Summarizing our discussion: point-to-point links are in common use today for system-level interconnect, and as a result our systems are faster, more reliable, more energy-efficient and smaller than ever before.

Multi-signal parallel buses are still used for very-high-bandwidth connections to memories, with a lot of very careful engineering to avoid the electrical problems observed in earlier bus implementations.

Wireless connections are in common use to connect mobile devices to nearby components and there has been interesting work on how to allow mobile devices to discover what peripherals are nearby and enable them to connect automatically.

The upcoming generation of multi-core chips will have 10's to 100's of processing cores.

There is a lot ongoing research to determine which communication topology would offer the best combination of high communication bandwidth and low latency.

The next ten years will be an interesting time for on-chip network engineers!