

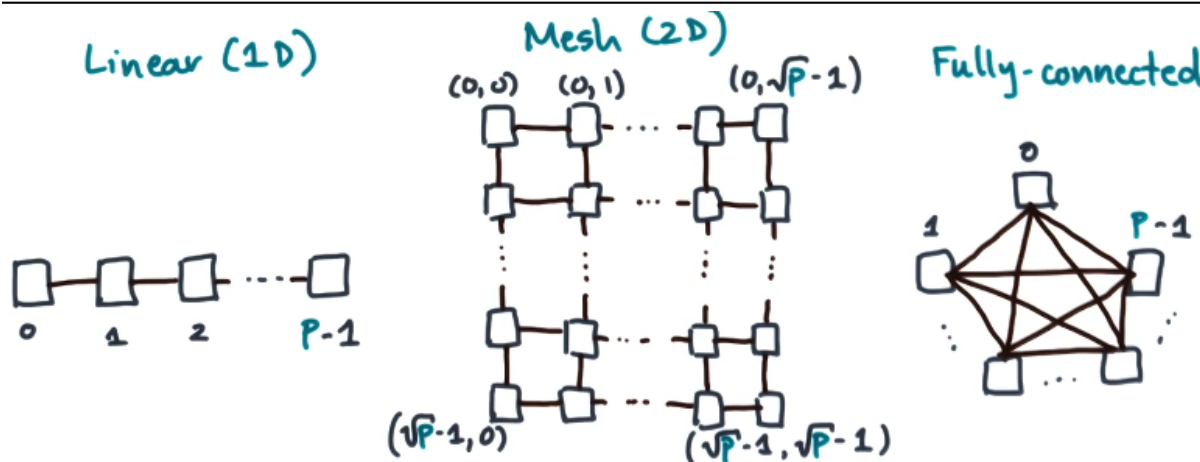
# Topology

## Introduction

1. Reasoning about distributed algorithms requires some consideration of the underlying network
  - If an algorithm was optimized for a fully-connected model, will it still perform well under a linear model?
2. Networks had become fast enough that we stopped worrying about the latency, leading to models like alpha-beta
  - However, as processors become faster and networks larger, we will need to consider the network topology once again
    - Begins to matter again at ~1 billion processors

## Introduction to Network Models - Links and Diameter

1. Links ( $\lambda$ )
  - Linear:  $P-1$
  - Mesh:  $2 * (\sqrt{P}-1) * \sqrt{P} \sim 2P$
  - Fully-connected:  $P * (P-1) / 2 \sim P^2$
  - Links are a proxy for cost; more links = more expensive
2. Diameter ( $\delta$ ) - Longest shortest path
  - Linear:  $P - 1$
  - Mesh:  $2 * (\sqrt{P} - 1)$
  - Fully-connected: 1
  - Proxy for the maximum distance any message must travel



Network Examples

## Improve the Diameter of a Linear Network

1. Add a link (edge) to reduce the diameter of this network by half
  - Add the link between nodes 0 and 7

## Improve the Diameter of a 2-D Mesh

1. Given a 2D mesh network with 16 nodes, where will adding links cut the diameter in half?
  - From  $(0,0)$  to  $(\sqrt{P}-1, \sqrt{P}-1)$  and  $(0, \sqrt{P}-1)$  to  $(\sqrt{P}-1, 0)$
  - A ring of links connecting  $(0,0)$  to  $(\sqrt{P}-1, \sqrt{P}-1)$  and  $(0, \sqrt{P}-1)$  to  $(\sqrt{P}-1, 0)$
  - Wraparound links from left to right, top to bottom

2. All three options reduce the diameter by a factor of (roughly) 2

## Bisection (Band)Width

1. Bisection width: Minimum links to remove to cut the network in half
  - Equality is measured in number of nodes
2. For a linear network with 8 nodes, bisection width is 1
  - $B(P) = 1$
3. For a mesh network with 16 nodes, bisection width is 4
  - $B(P) = \sqrt{P}$  for a mesh
4. For a fully-connected network, there are roughly  $P^2/2$  links
  - $B(P) = P^2/4$
5. Bisection width is important for an all-to-all personalized exchange
  - Send message from every node to every other node
6. Bisection bandwidth is link speed (Beta) times bisection width
  - If not all links have equal speed, look for a set of nodes that cuts the network in half and minimizes total bandwidth

## Improve the Bisection of a 2-D Mesh

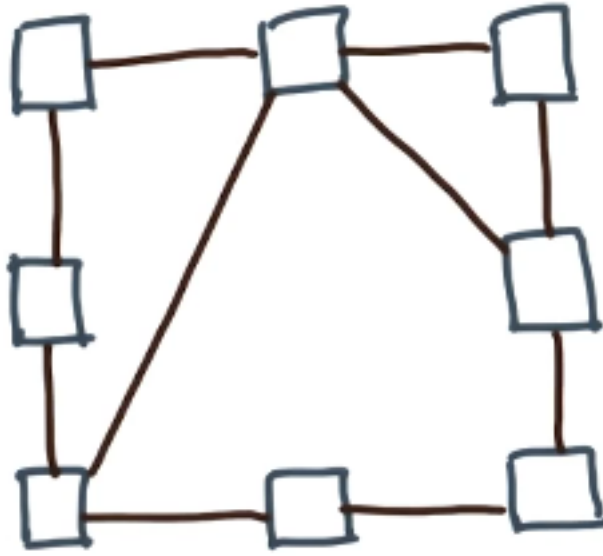
1. Given a 2D mesh network with 16 nodes, where will adding links double the bisection width?
  - From (0,0) to ( $\sqrt{P}-1, \sqrt{P}-1$ ) and (0, $\sqrt{P}-1$ ) to ( $\sqrt{P}-1, 0$ )
  - A ring of links connecting (0,0) to ( $\sqrt{P}-1, \sqrt{P}-1$ ) and (0, $\sqrt{P}-1$ ) to ( $\sqrt{P}-1, 0$ )
  - Wraparound links from left to right, top to bottom
2. Only the third option doubles the bisection width

## Some Other Network Topologies

1. Tree: Compute nodes are leaves of the tree, carrier nodes don't do any actual computation (typically more links at higher levels of the tree to improve BW)
  - Links:  $P$
  - Diameter:  $\log(P)$
  - Bisection: 1
2. d-dimensional mesh or torus:  $P^{1/d}$  nodes per dimension (extension of a mesh to higher dimensions)
  - Links:  $dP$
  - Diameter:  $d * P^{1/d} / 2$
  - Bisection:  $2 * P^{(d-1)/d}$
  - Many of the world's top supercomputers use low-dimensional toroidal networks
3. Hypercube:  $\log(P)$  dimensional torus
  - Links:  $P * \log(P)$
  - Diameter:  $\log(P)$
  - Bandwidth:  $P/2$
  - Much more expensive in terms of number of wires, but lower diameter and larger bisection width
  - Build a hypercube by copying the topology of the previous dimension and connecting corresponding nodes

## Diameter and Bisection

1. Consider the following network:



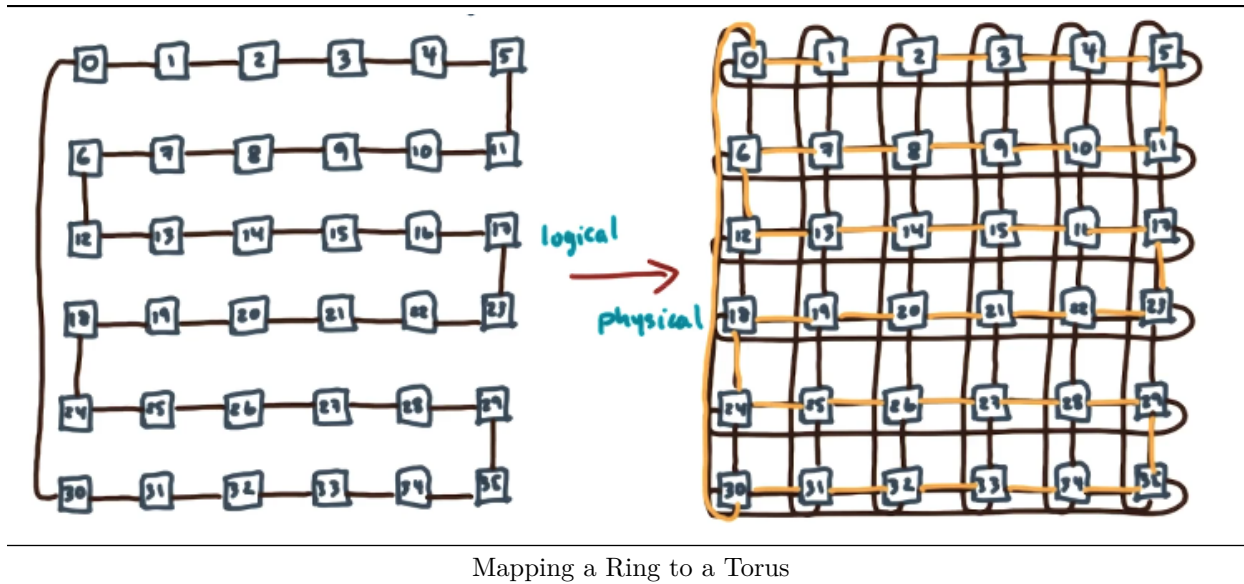
Network Examples

---

2. Answer the following questions:
  - How many nodes does it have?
    - $P = 8$
  - How many links does it have?
    - Links = 10
  - What's its diameter?
    - Diameter = 3
  - What's its bisection width?
    - Bisection width = 3

## Mappings and Congestion

1. Scatter/gather implementations were designed to avoid contention for links
  - If we design an algorithm for a linear network then try to run it on a different topology, how well would it work?
    - Need to map between the networks
  - If we map a ring to a 2D torus, we observe that the edges of the torus are a superset of the edges of the ring
    - This means if there's no contention on the ring, there will be no contention on the torus either
    - The opposite isn't necessarily true; there might be contention on the ring even if there wasn't on the torus
2. Congestion: Maximum number of logical edges that map to a given physical edge
  - Congestion (ring to torus) = 1
  - Congestion (torus to ring)  $\geq 6$



## 2-D to 1-D Congestion

1. Consider the mapping from a 2D torus (logical) to a 1D ring (physical)
2. What is the congestion of this mapping?
  - $\sqrt{P} + 2$  (two comes from the wraparound edges)

## A Lower Bound on Congestion

1. Consider two hypothetical networks: one logical, one physical
  - Suppose we find a bisection in the physical network
    - $B_x$  is the number of physical edges cut
  - Bisecting the physical network implies a cut in the logical network as well
    - $L$  is the number of logical edges cut
  - Congestion ( $C$ )  $\geq L / B_l$
  - $L \geq B_l$
  - $C \geq L / B_x \geq B_l / B_x$
2. Congestion for ring to 2D torus
  - $B_x = 2$
  - $B_l = 2 * \sqrt{P}$
  - $C \geq B_l / B_x = \sqrt{P}$ 
    - Truth:  $C = \sqrt{P} + 2$

## Congestion Lower Bounds

1. For which pair of networks might a congestion-free mapping be possible? (logical  $\rightarrow$  physical)
  - fully-connected  $\rightarrow$  hypercube
  - hypercube  $\rightarrow$  butterfly (true)
  - butterfly  $\rightarrow$  3D torus
  - complete binary tree  $\rightarrow$  ring (true)
  - none of these

## Exploiting Higher Dimensions

1. Consider an allgather primitive
  - Tree-based:  $T(n, P) = \alpha * \log(P) + Bn$

- Bucketing:  $T(n,P) = \alpha * P + Bn$
2. Can we do better if the underlying topology is a 2D mesh instead of ring?
    - First, do allgathers along each row
      - $T_{row}(m * \sqrt{P}; \sqrt{P}) = a * \sqrt{P} + B * m * \sqrt{P}$
    - Then, perform allgathers of entire rows along each column
      - $T_{col}(n; \sqrt{P}) = a * \sqrt{P} + Bn$
    - $T_{row} + T_{col} = 2 * a * \sqrt{P} + B * m * (P + \sqrt{P})$ 
      - Alpha now scales with  $\sqrt{P}$ , B is basically optimal

## 2D Broadcast

1. Consider a broadcast on a mesh
  - Scheme 1:
    - Tree-based broadcast in each row
    - Tree-based broadcast in each col
  - Scheme 2:
    - Scatter in all rows
    - Scatter in all cols
    - Bucket allgather in cols
    - Bucket allgather in rows
2. Which scheme sends fewer messages?
  - Scheme 1: Alpha is proportional to  $\log(P)$  for a tree-based approach while bucket-allgather is proportional to  $P$

## All to All Personalized Exchange

1. Consider a distributed matrix transpose where each node has a column of data
  - Every node wants to send data of size  $m$ 
    - $n = m * P$
  - Distance the  $i$ th message needs to travel is  $\min(i, P-i)$  if we assume a ring buffer
    - Average distance =  $\sum(\min(i, P-i)) / (P-1) = P/4$
  - Traffic volume =  $P * m * (P-1) * (P/4)$
  - Total bandwidth =  $P / B$
  - Time = Volume / Speed  $\geq B * n * (P-1) / 4$
2. Algorithm for all-to-all exchange: Each node sends all the data that should be distributed each timestep (circshift)
  - Round  $i$ : send  $m(P - i)$  words
  - Total time =  $(a + B * m * (P/2)) * (P - 1) = a * (P-1) + B * n * (P-1)/2$

## All to All in Higher Dimensions

1. Which network has the best chance to reduce the asymptotic running time to  $O(a * \log(P) + B * n * \log(P))$ 
  - Complete binary tree
  - $d$ -dimensional torus
  - Hypercube
  - Fully-connected
2. All-to-all is intrinsically bisection limited; only hypercube and fully-connected networks have linear bisection widths or better

## Conclusion

1. Idea of congestion: Concept lets you design for one topology and then estimate whether it will map well or poorly to another

2. Exploit high-dimensional networks: There are algorithmic scalability gains from using a 3D mesh network instead of 2D