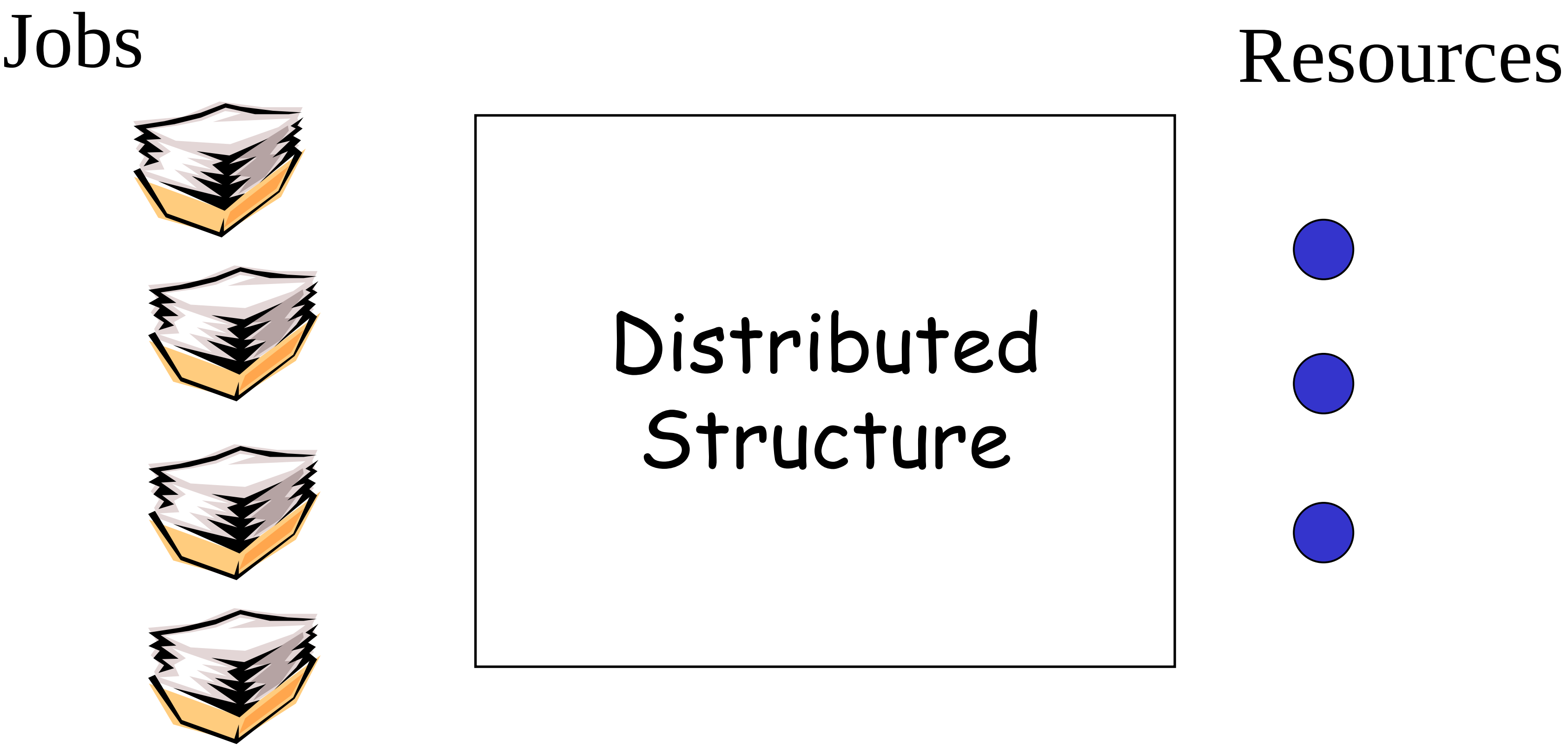
Adaptive Counting Networks

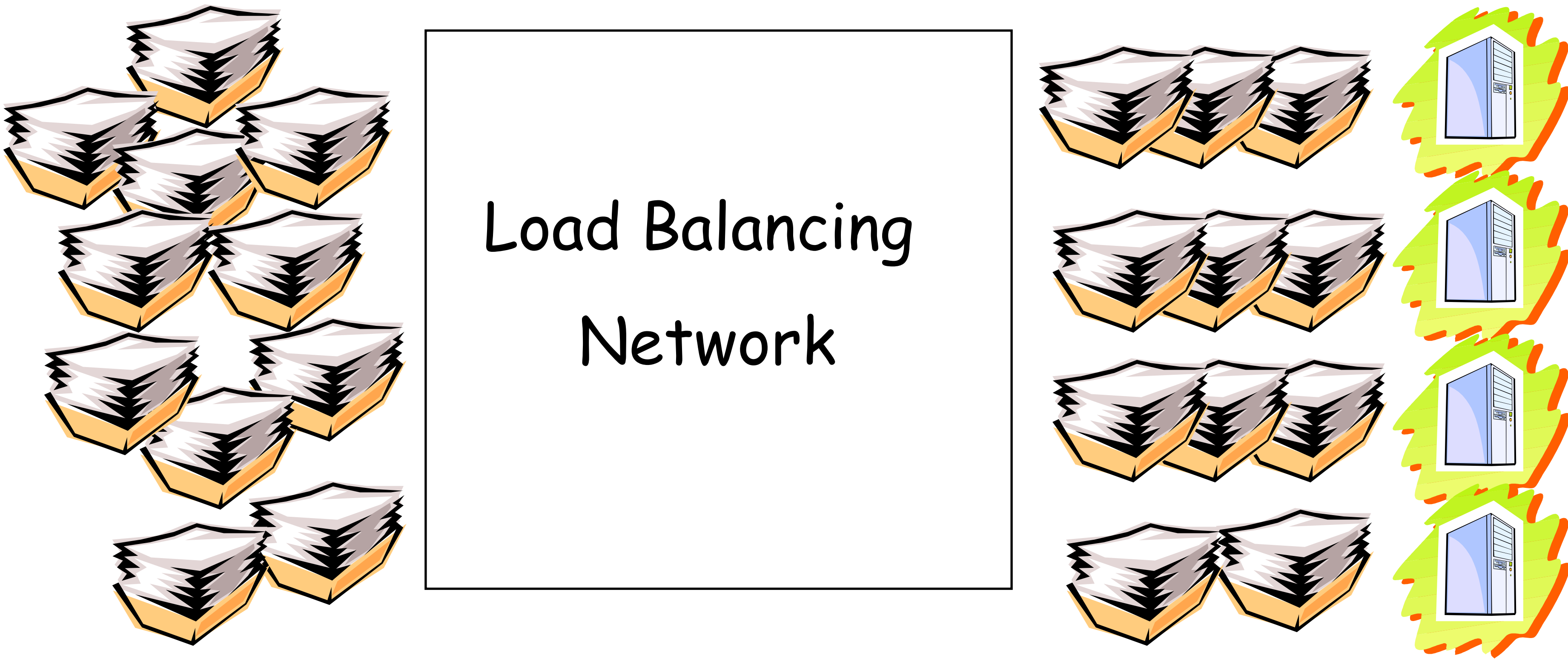
Srikanta Tirthapura
Elec. And Computer Engg.
Iowa State University

Example: Producer - Consumer



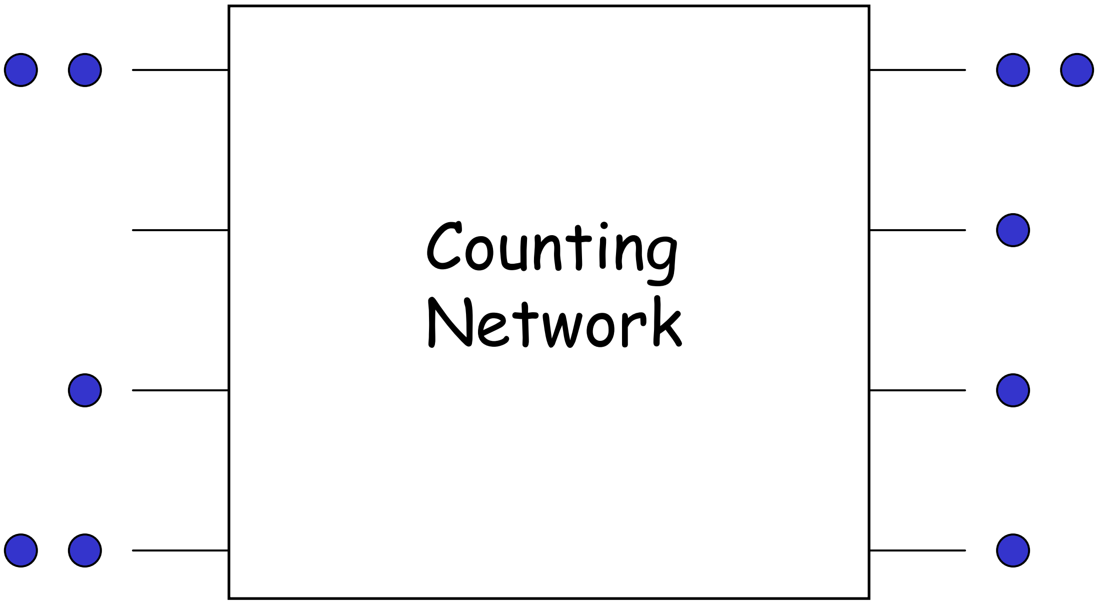
Centralized Solutions don’t scale, look for distributed solutions

Distributed Load Balancing

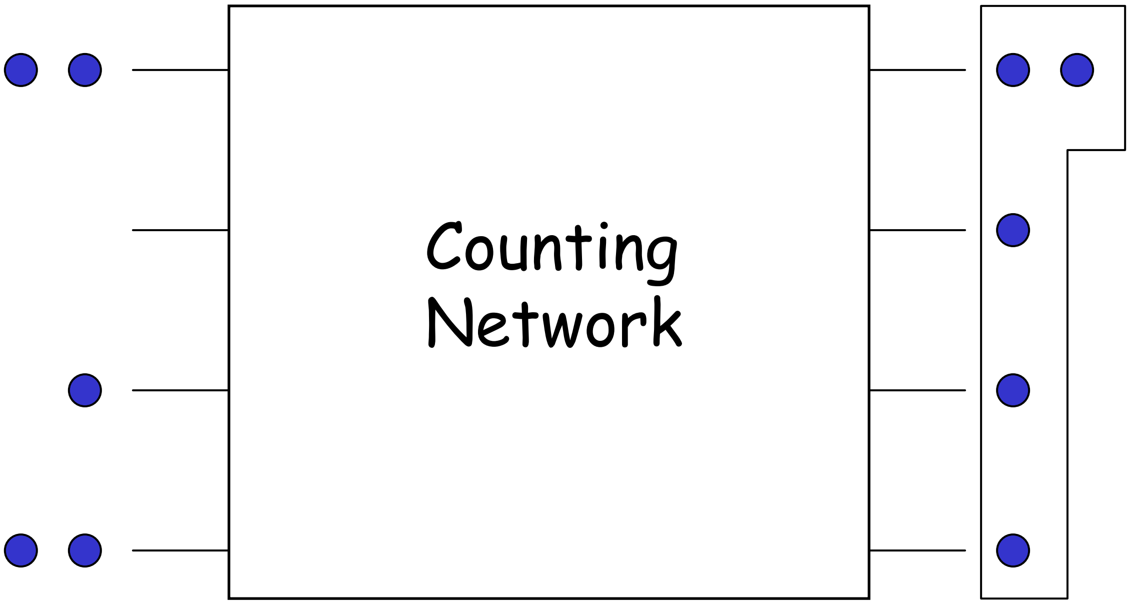


Routing Tasks to Processors

Counting Network



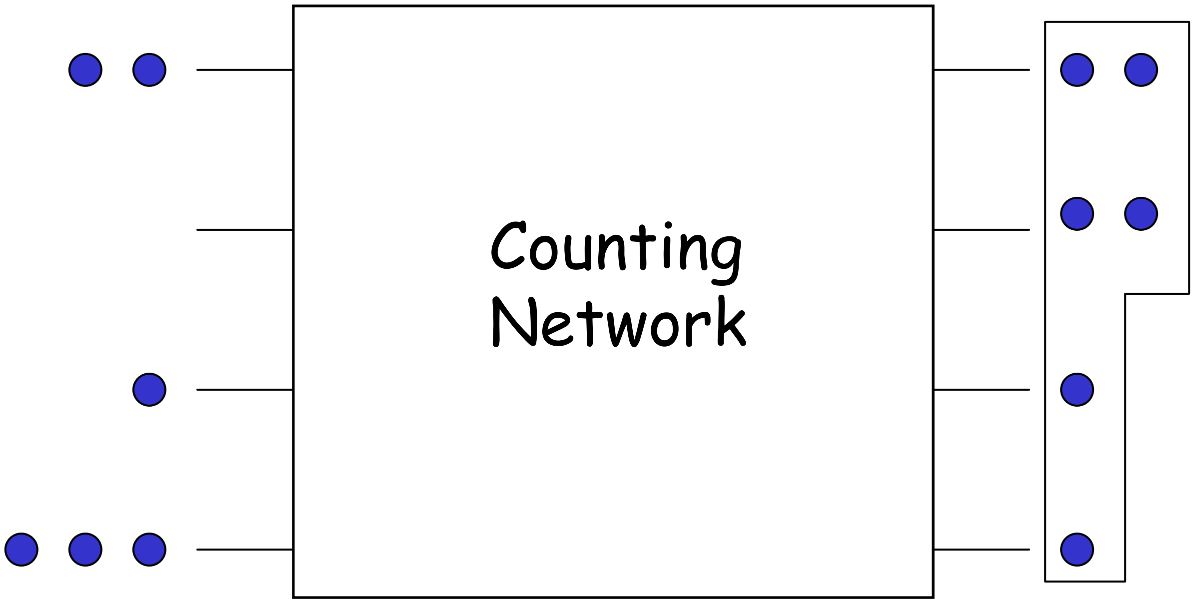
Counting Network: Step Property



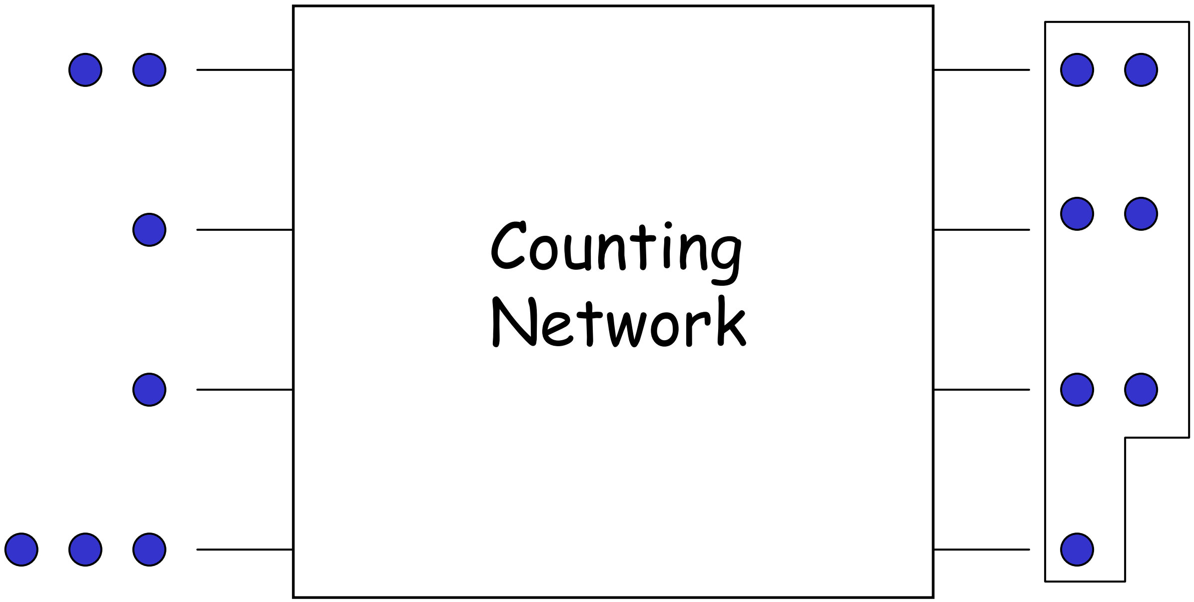
Input Tokens (imbalanced)

Output Tokens (balanced)

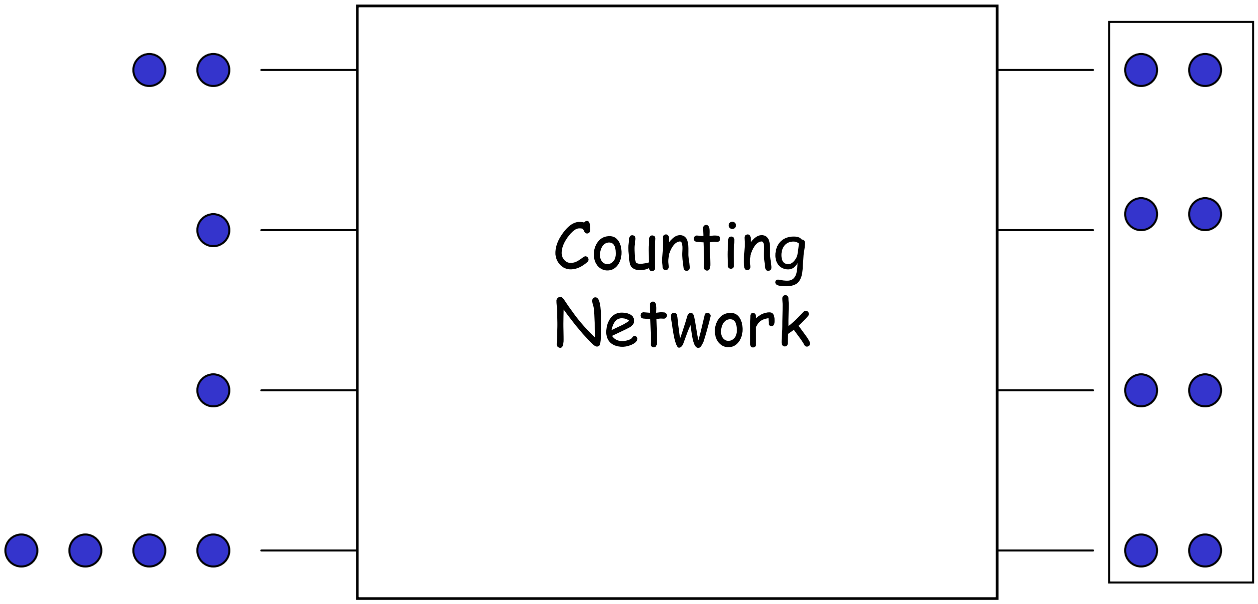
Step Property



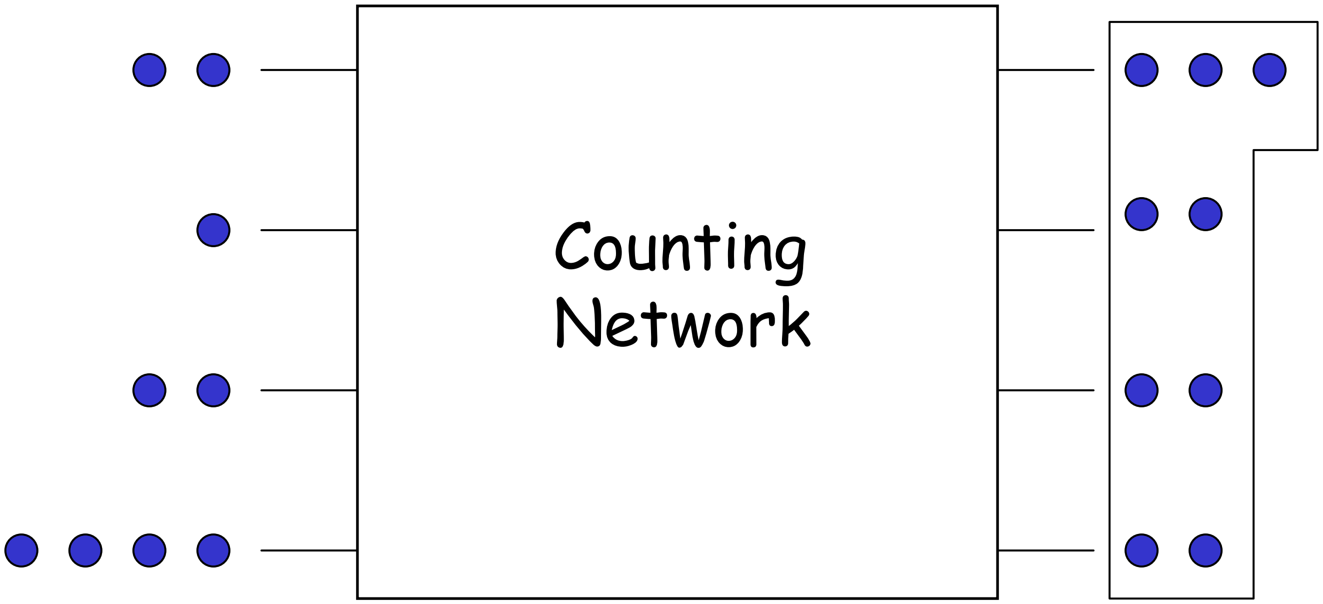
Step Property



Step Property



Step Property



Applications

- Load Balancing
- Producer-Consumer solved using two back-to-back counting networks
- Shared Counters in a Distributed System

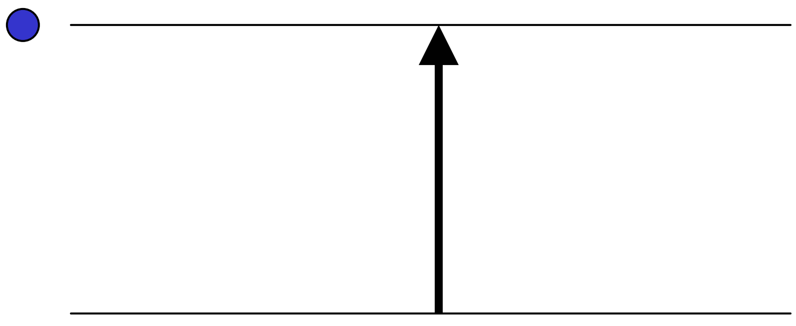
Counting Network Construction

- Bitonic network, Periodic network (Aspnes, Herlihy, Shavit – 1991)

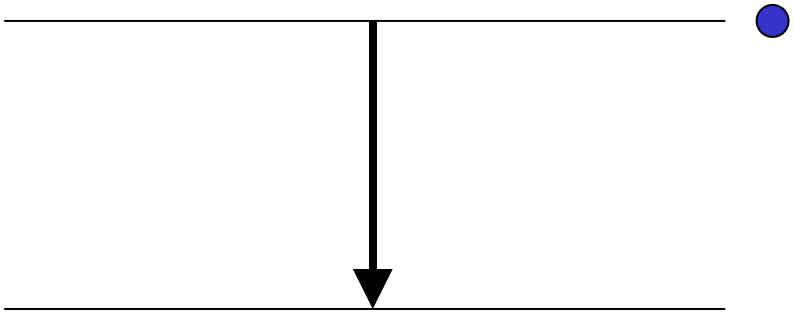
- Network of basic elements called balancers

- State of the system distributed over the network
  - No sequential bottleneck

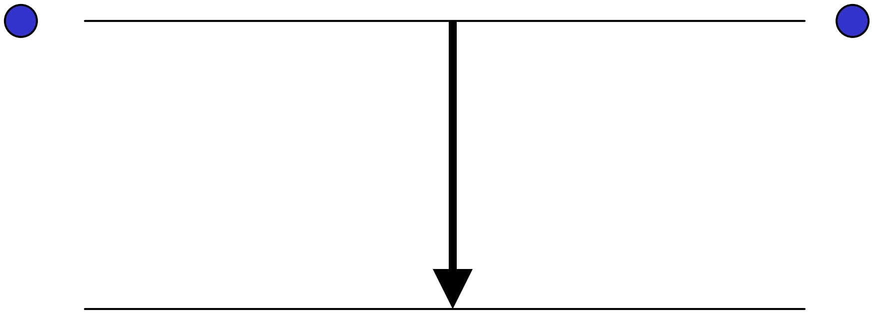
Balancer



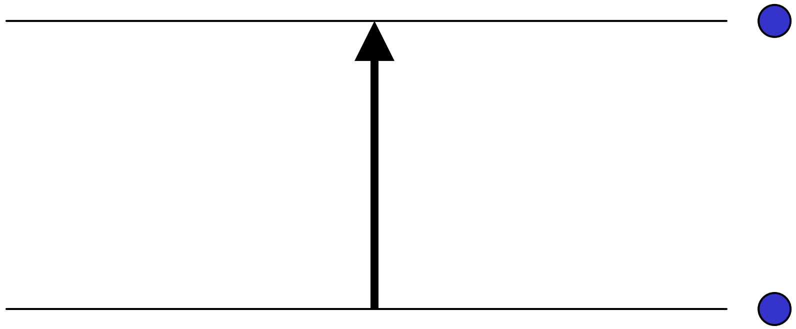
Balancer



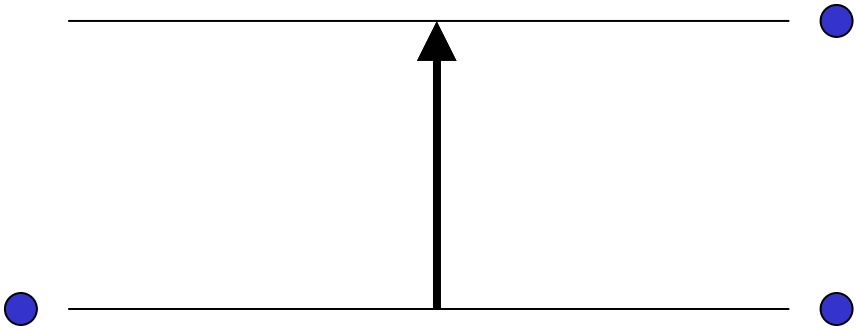
Balancer



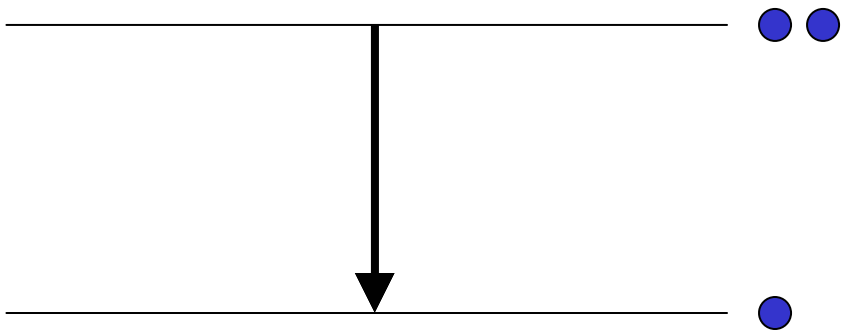
Balancer



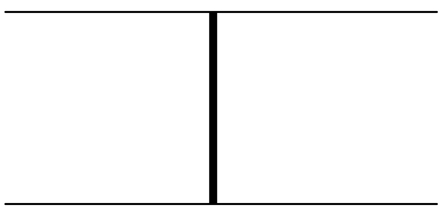
Balancer



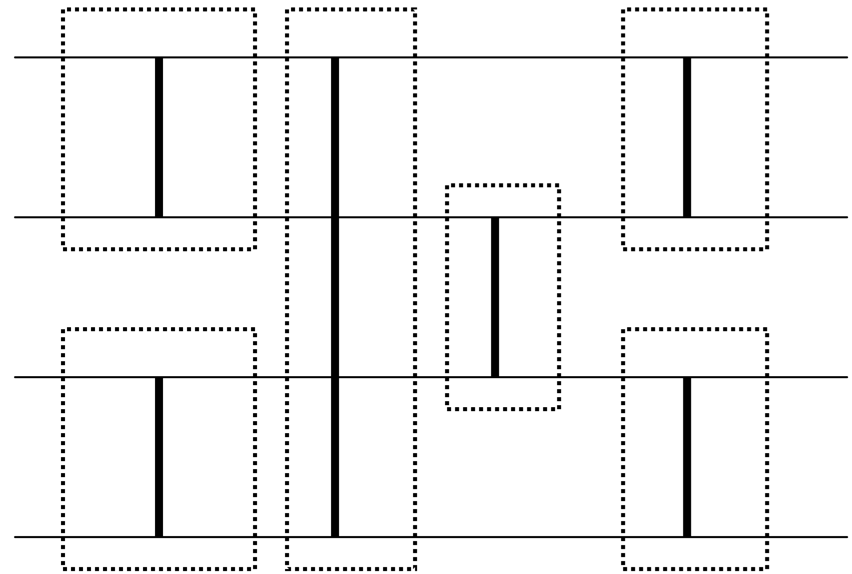
Balancer



Scalable Construction

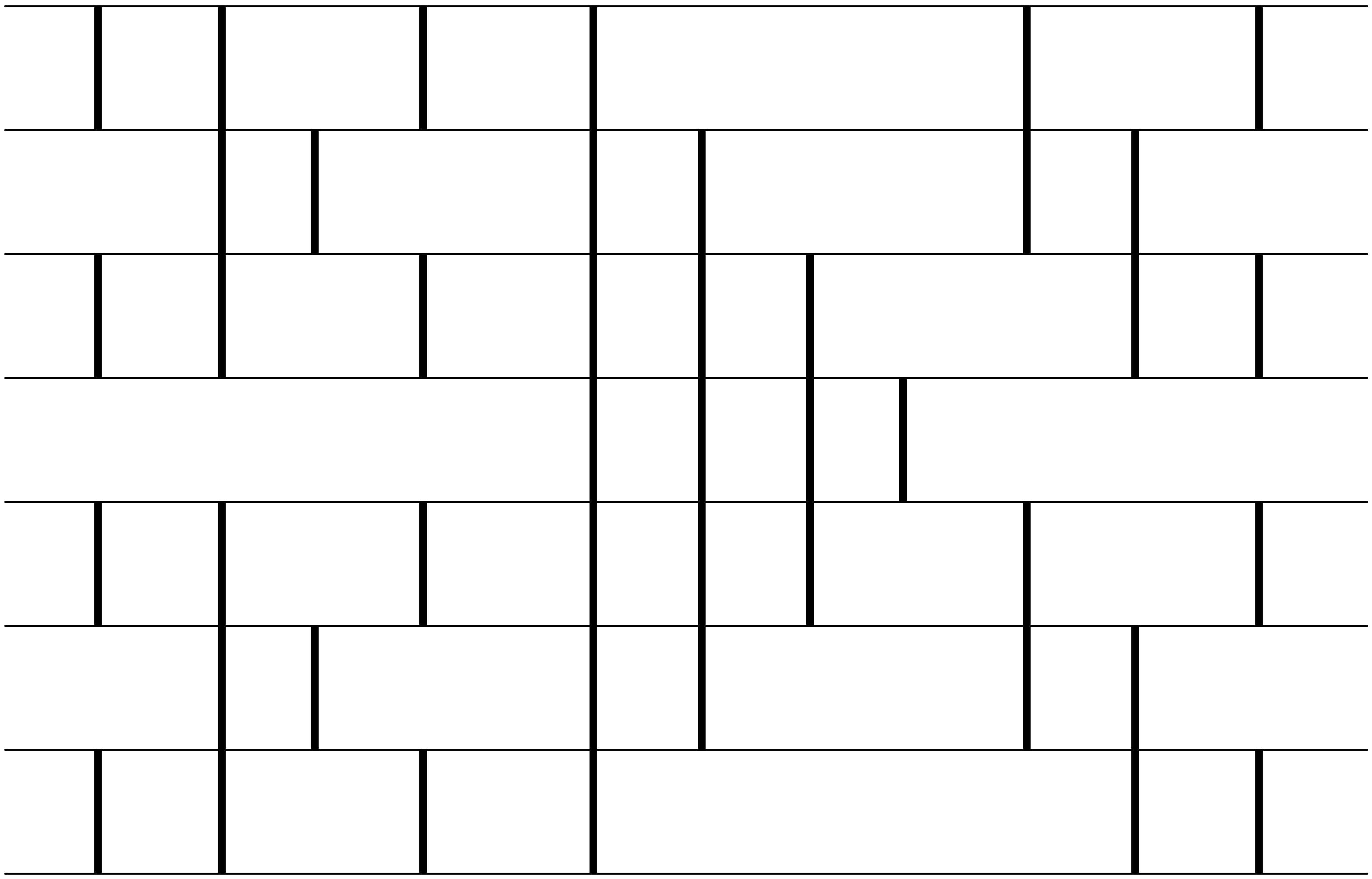


Bitonic[2]

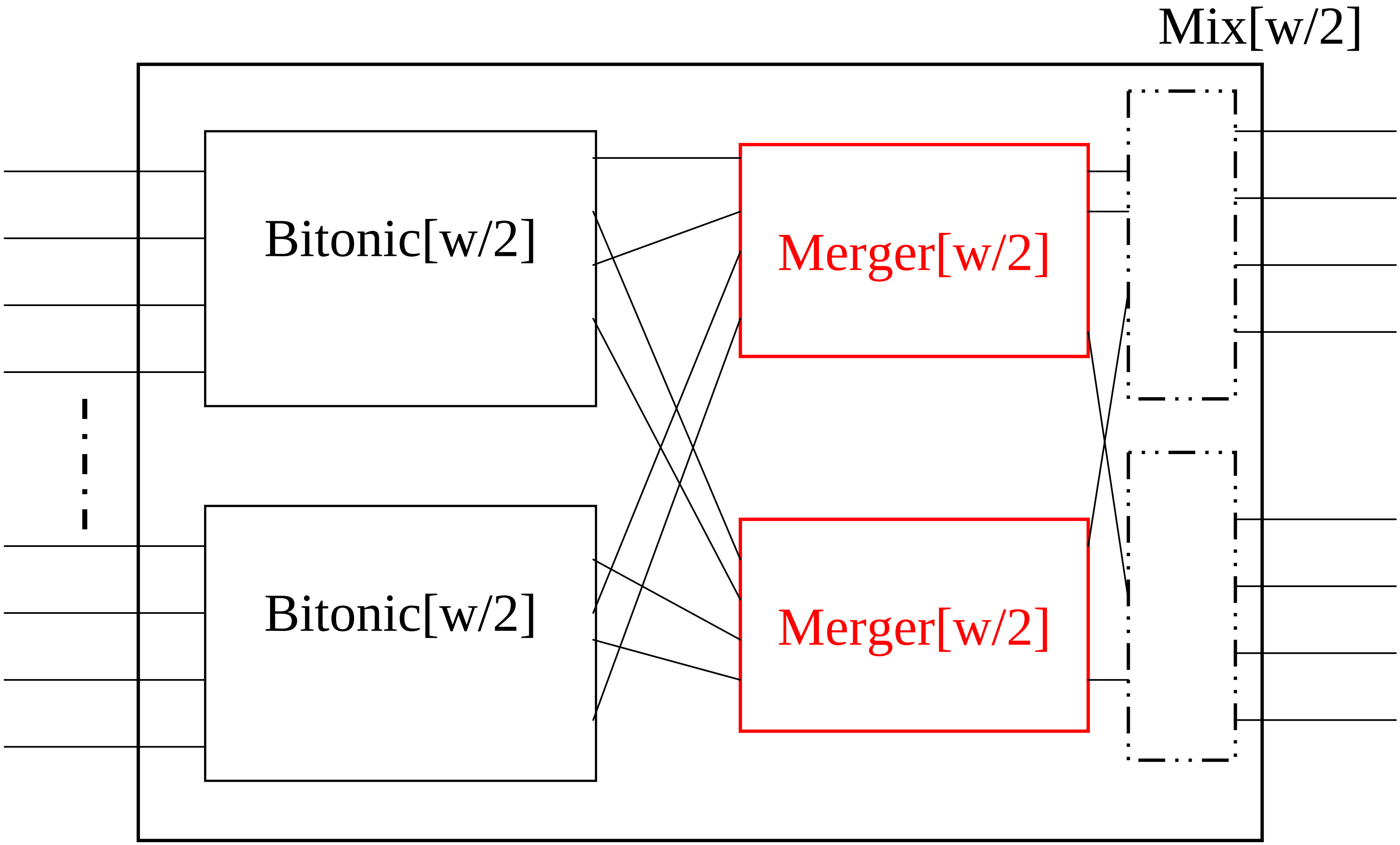


Bitonic[4]

Bitonic[8] Network



Recursive Construction of Bitonic[w]



Overlay Networks

- Plan: Counting network as a peer-to-peer overlay network
  - Balancers → nodes of the network
  - Wires → communication links between nodes

- Structured peer-to-peer network

1. Efficient lookup service
   - Plaxton et. al., Chord, CAN, etc
2. Good local estimates of network size
   - Manku, Viceroy, Horowitz-Malkhi, …

Problem

- All Current Constructions of counting networks are Static
  - **Degree of parallelism (width) has to be decided in advance**
- System size changes with time!
- Does not scale with the underlying network size
- Bad:
  - Width 64 network for a system with 20 nodes
  - Width 4 network with 1000 nodes
- Question: How to build an adaptive counting network (or your favorite distributed data structure)?

Adaptive Counting Network

Degree of parallelism tunes itself to current network conditions

- As underlying physical network expands and contracts, so will the counting network
- Expansion and contraction are local operations (no central control)
- Decision of when to expand and contract also local

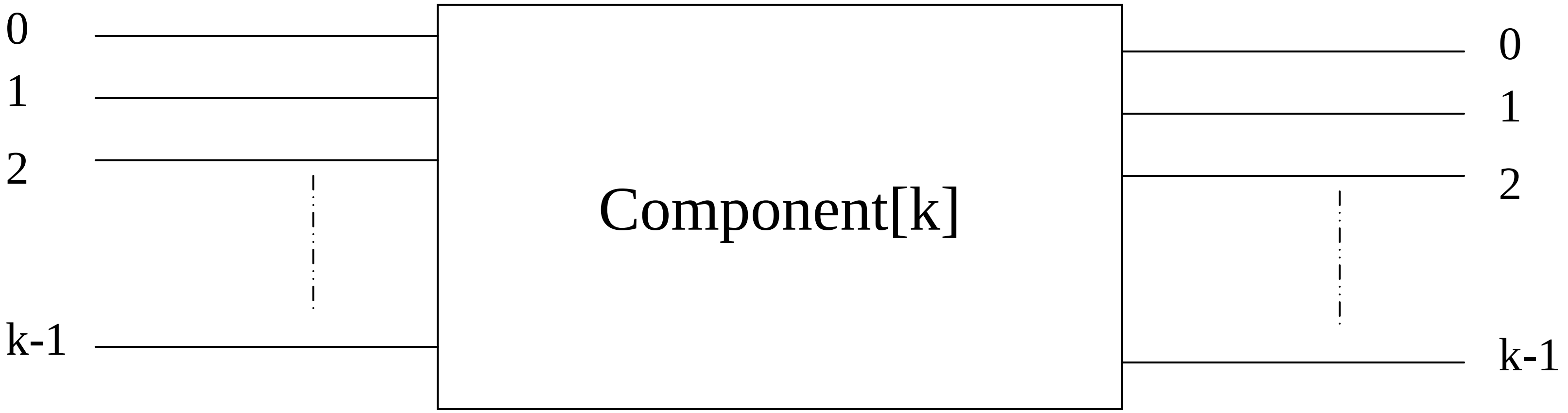
Solution Ideas for Bitonic Network

1. Network built using variable sized components rather than fixed sized balancers

2. Network size changes with underlying physical network size
    1. Expand: A component splits into more components
    2. Contract: Many components merge into a single one

3. Distributed Decisions for Splitting and Merging
    1. Sense current network conditions using Distributed Network Size Estimation

Component



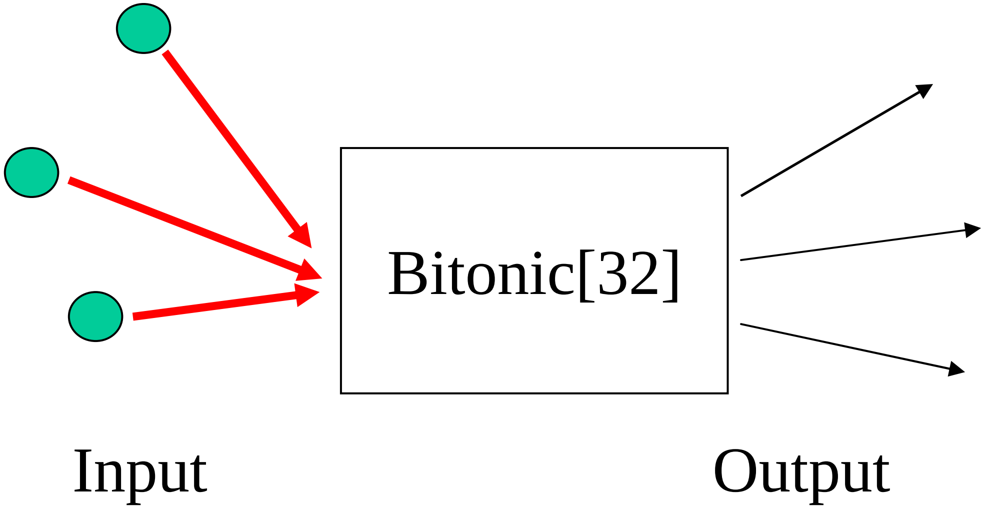
j th input token leaves on wire (j mod k)

Can be implemented trivially on a single node

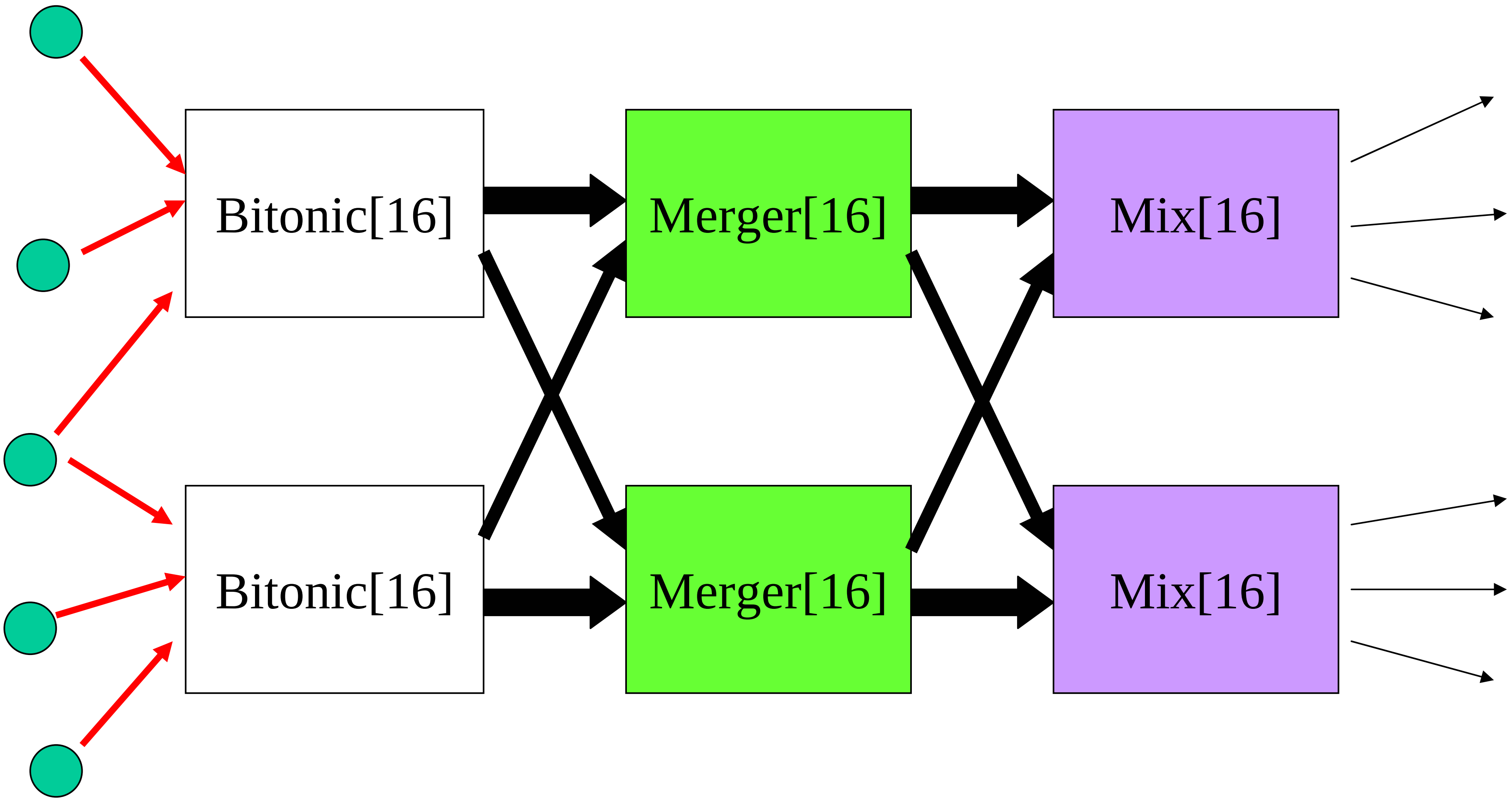
Adaptive Bitonic Network

- Choose a maximum width for the network
  Suppose maximum width = 32

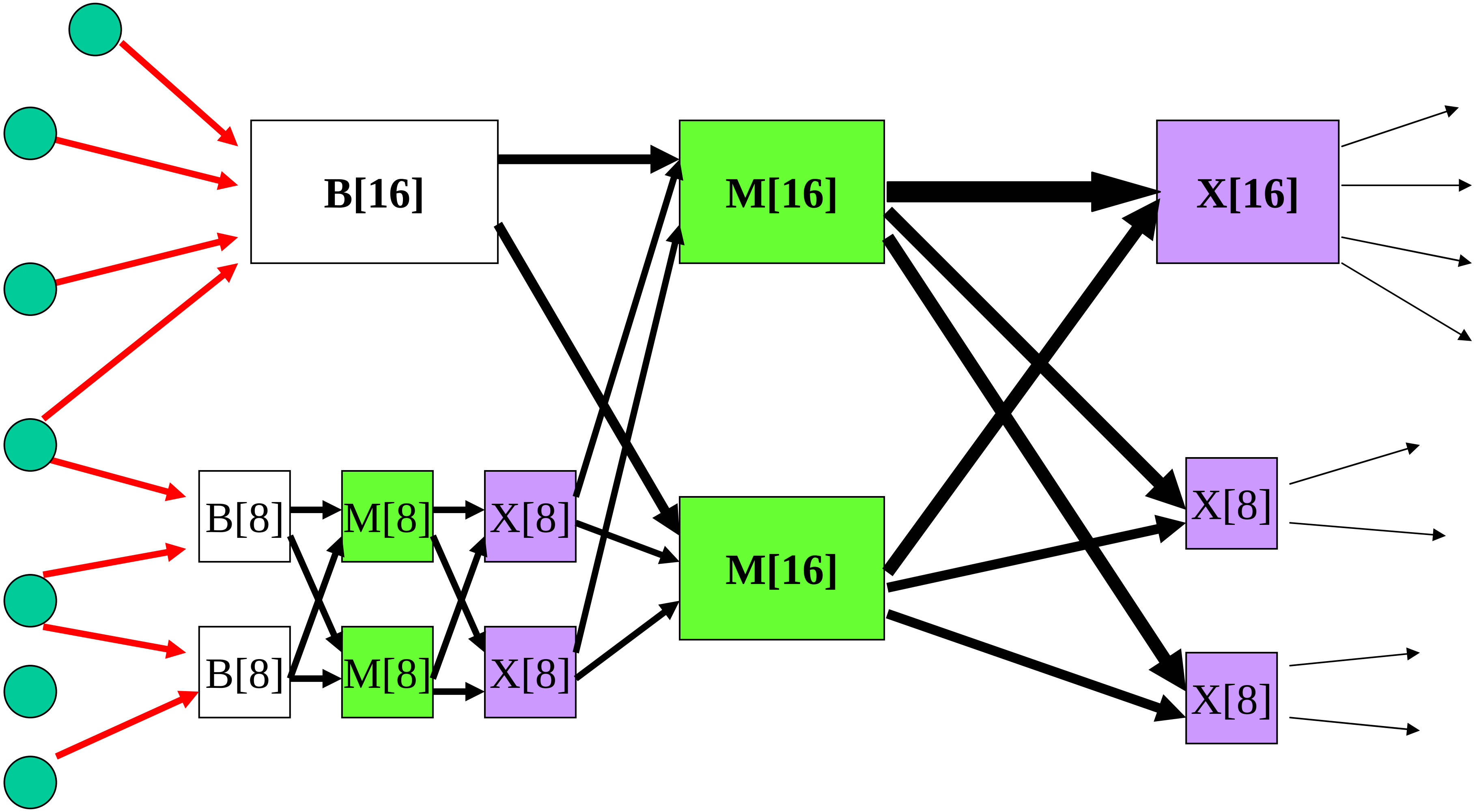
- Initially the whole network is implemented as a single component



Load Increases: Split Components



More Splits – "Irregular" Network



On a single node, each component can be implemented trivially

Flexibility

- Using components rather than balancers allows many more possibilities
- Network can morph into the best possible implementation for the current conditions

When to Split and Merge?

- Decision local to each node
- Possible Strategies:
  - Based on Load experienced by a node
  - Based on Estimate of network size
- Our Recipe (yields provable theoretical bounds):
  - Locally estimate network size
  - If network size estimate > threshold, then split
  - If network size estimate < threshold, then merge
  - Threshold varies with the component

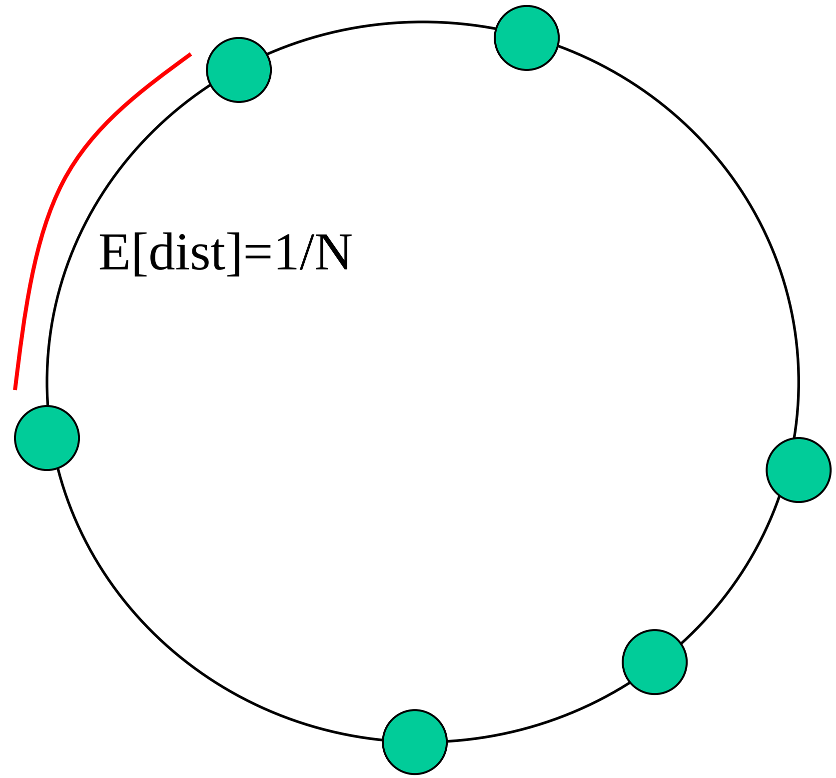
Network Size Estimation

N = number of nodes

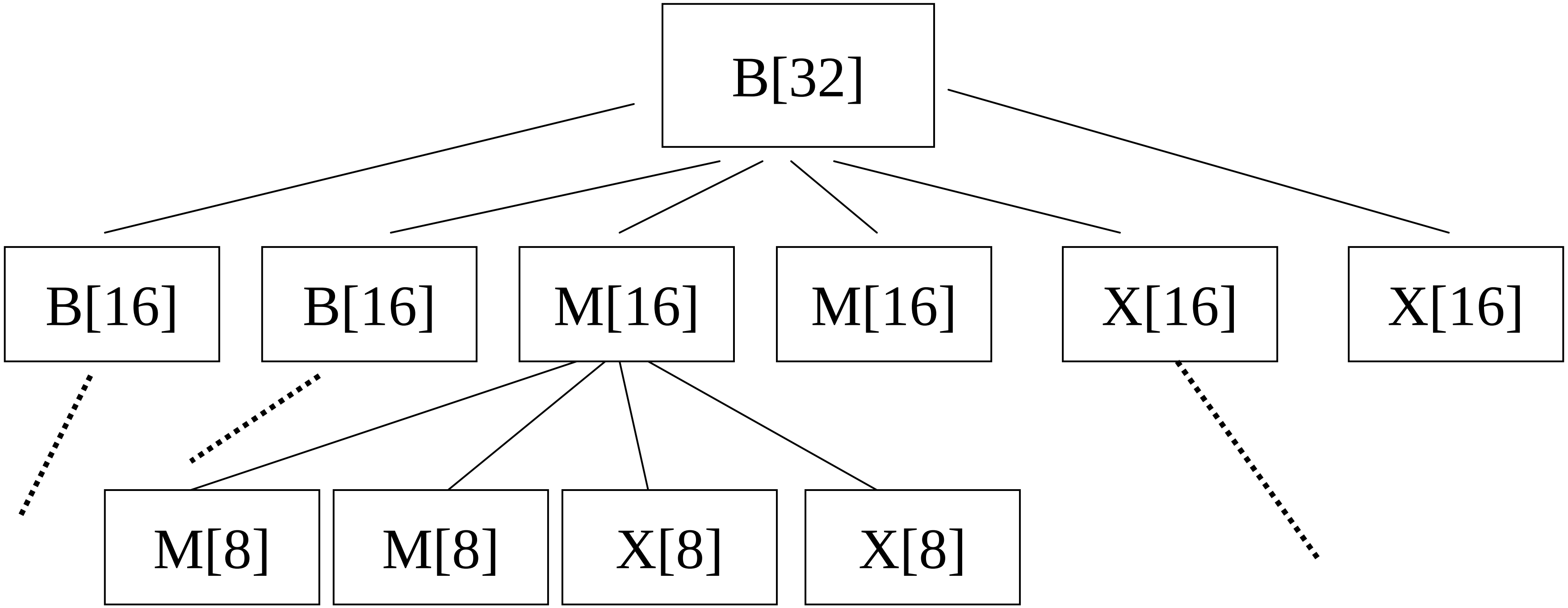
- Each node uses local estimate of physical network size

- Example: Chord p2p system
  - Nodes organized in a ring
  - Rough estimate = 1/(distance to successor)
  - Better estimate = k/(distance to kth successor)

- Local (inaccurate) estimates are enough for our purposes
  - Local Decisions are approximate, but aggregate of decisions is "pretty good"



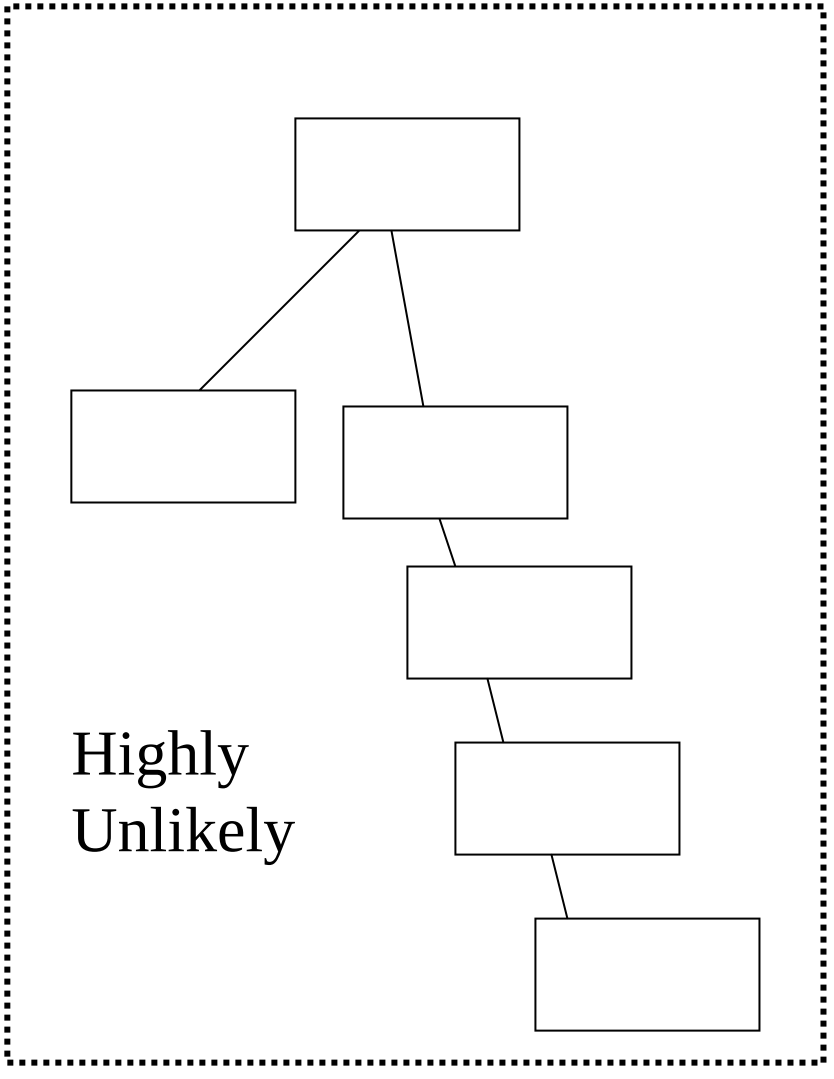
Component Hierarchy

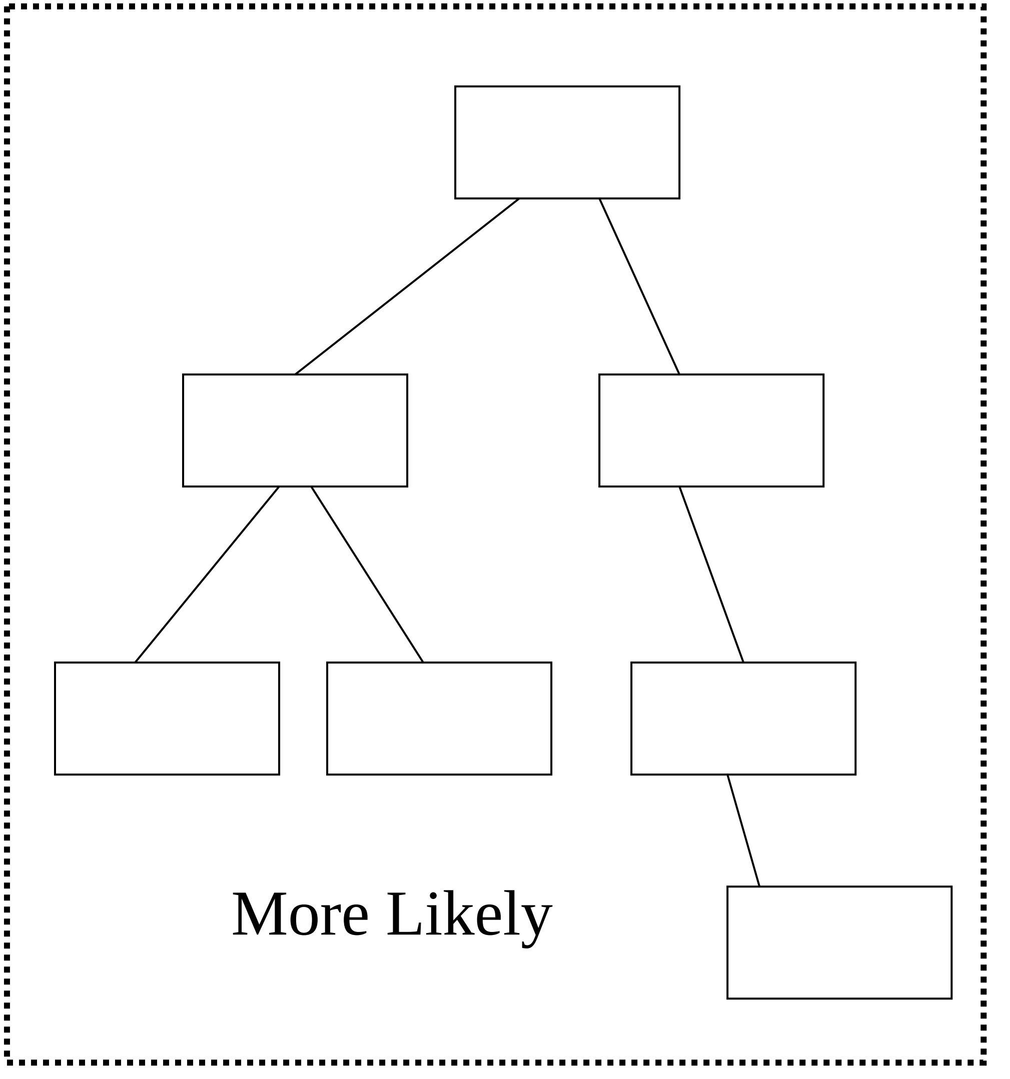


Intuition: N < 6 nodes, level 1 is ideal
N = 6 to 24 nodes, level 2 is best
N = 24 to 80, level 3 is best

We show that the level estimate of every component is close to the “optimal”

Balanced Hierarchy





Our Results for Bitonic Network

Definitions:

- Effective Width = number of edge disjoint paths from input to output
- Effective Depth = longest path from input to output

Our Results for Bitonic Network

Adaptive Network

If N = number of nodes currently in the physical network

With high probability,

- Total Number of Components = O(N)
- Effective width $O\left(\frac{N}{\log^2 N}\right)$
- Effective Depth $O(\log^2 N)$

Static Network

- Total number of components= $O(w \log^2 w)$
- Effective width = w is a constant
- Effective depth = $O(\log^2 w)$

Conclusions

- Counting networks built out of variable width components rather than fixed width balancers
- Distributed Decisions expand and contract the Network
- Final Network is provably tuned to the current network conditions (assuming a structured p2p overlay)
- Applies to any distributed data structure
  - That can be decomposed recursively
  - Needs to resize dynamically in response to system load

How to Locate Components?

- Each component has a name, derived from its position in the recursive decomposition
- Lookup component location by name (using the distributed hash table)
- If output component changes during execution, then re-compute location

Acknowledgments

- Thanks to Costas Busch for help with the presentation