CSC2/458 Parallel and Distributed Systems
Distributed Systems: Background

Sreepathi Pai

March 20, 2018

URCS

Outline

Definitions and Goals

Networking Background

Some Real-world Distributed Systems

Fundamental Algorithms

Outline

Defining Distributed Systems

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

Leslie Lamport (1987)

What we will use for this course

A distributed system is a set of processes that are:

- autonomous
- share nothing (as opposed to shared memory)
- communicate via messages

[Important: a process in a distributed system may be on the same machine or on different machines.]

Some organizations of distributed systems

- Clusters
  - Most popular in supercomputing
  - Usually restricted to a single site
  - Very homogeneous
- Grids
  - Roughly, *federated* clusters
  - Usually, multiple operators
  - Rarely homogeneous
- "Cloud"
  - Roughly a grid run by a single operator
  - Relies heavily on hardware/software virtualization

Goals of (some) Distributed Systems

- Performance
  - Problem size
  - Data size and location
- Availability
  - Service can tolerate failure of one or more components
- Scaling out: add more machines to handle greater loads
  - (as opposed to scaling up, which is upgrading a single machine)
  - largely an industry term

Outline

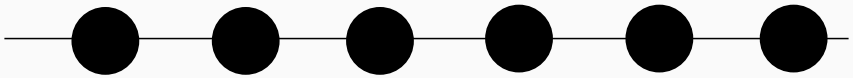
Topology

- Networking *topology* determines how computers are connected to each other.
  - i.e. the interconnection network
- Two important properties of interconnection networks:
  - Bisection bandwidth
  - Diameter (or Hop count)
- Interconnection networks exist inside single computers too:
  - Connect different cores, caches and memory
  - usually called Networks on Chip (NoCs)

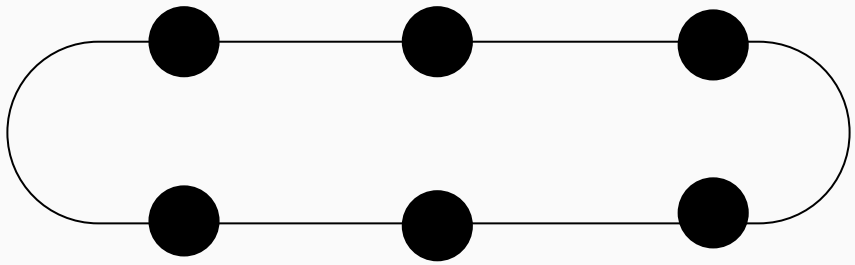
Definitions

- Bisection Bandwidth
  - Bandwidth across minimum number of links cut to split the network into two parts
- Diameter
  - Maximum number of hops (intermediate nodes) between any two nodes

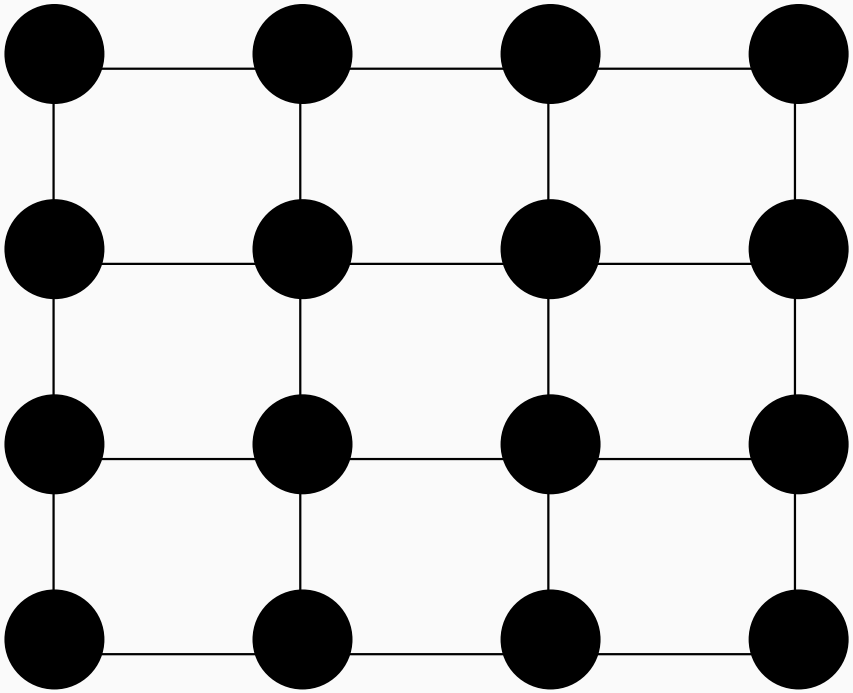
Linear



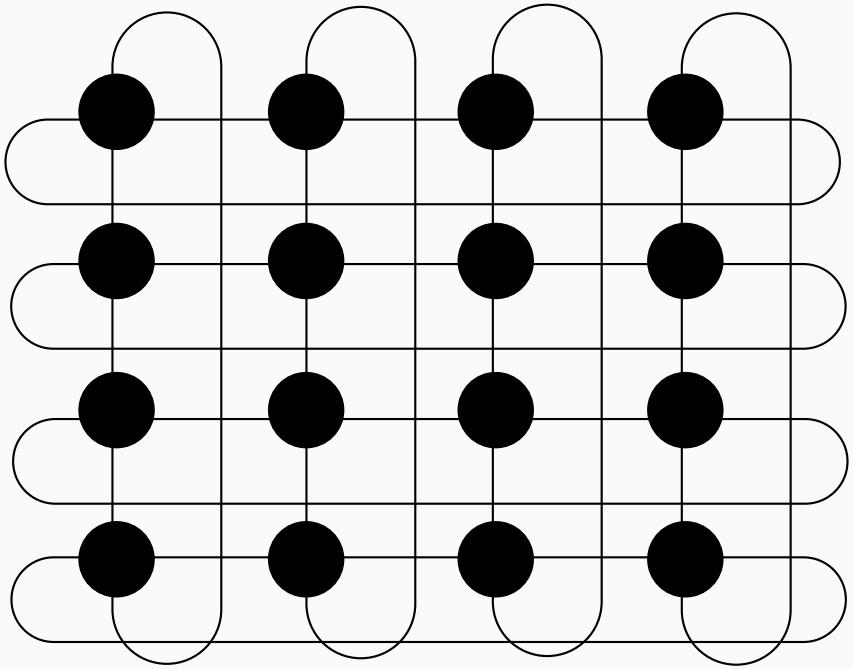
Ring



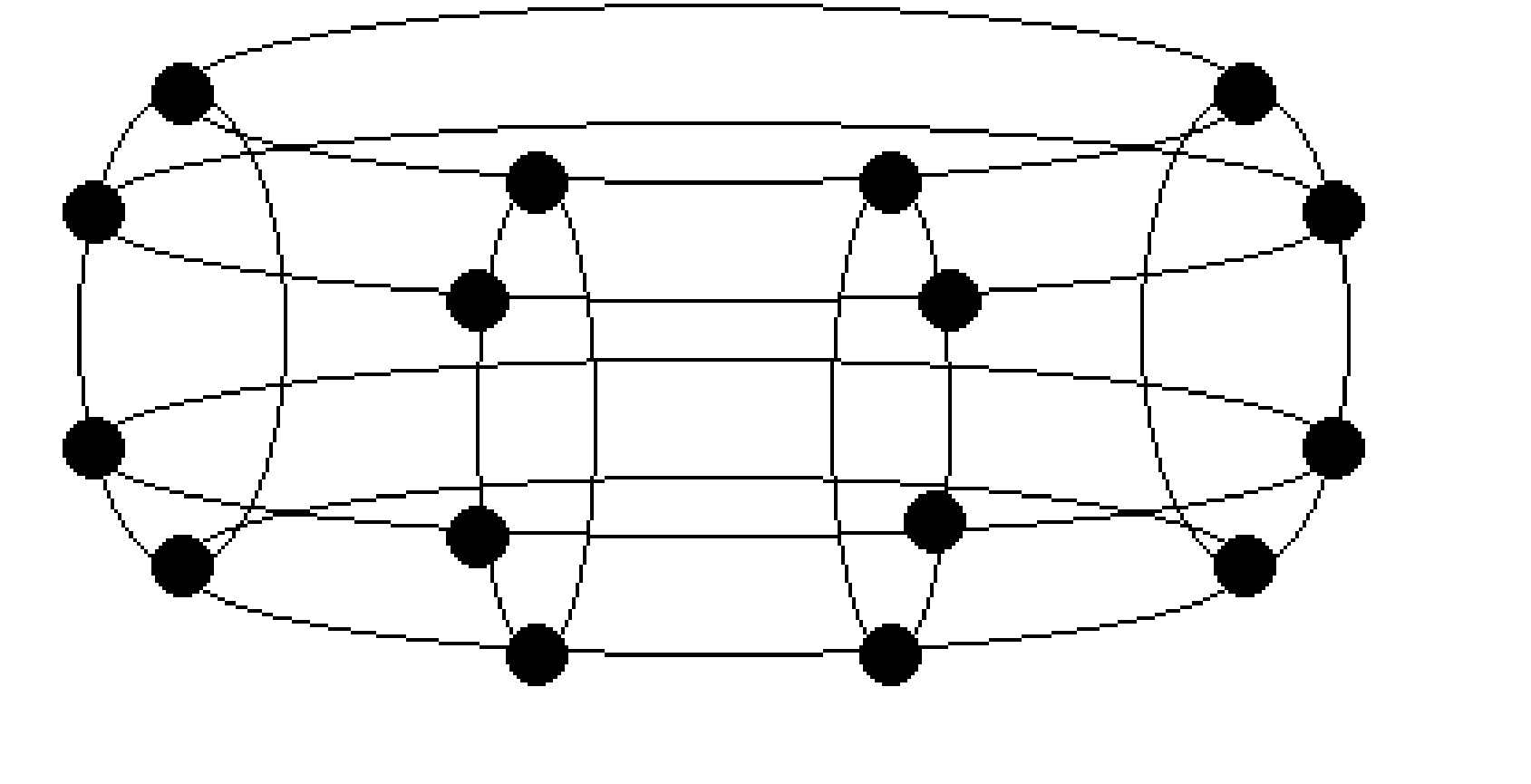
Mesh Network



Torus



Torus as a Doughnut



Source: `http://users.ecs.soton.ac.uk/bim/notes/aca/lecture/revise/netrev.html`

Other network topologies

- Higher-dimensional torus
- Hypercubes
- Trees (contain switches at interior nodes)
    - Fat trees

Adapting an application to the topology of the network

- Many distributed scientific applications exhibit communication locality
  - Communicate with only neighbours, for example
- This communication can be taken into account when mapping processes to compute nodes
- Many algorithms can also be optimized for a particular topology

Layer 4 protocols

- (Partial) OSI Model
  - Layer 1 is physical layer (e.g., Fibre)
  - Layer 2 is data link layer (e.g., Ethernet)
  - Layer 3 is transport layer (e.g., IP)
  - Layer 4 is network layer (e.g., TCP, UDP)

TCP and UDP

- IP and UDP
  - Datagram
  - Connectionless
  - Unreliable
  - Unordered
  - how is UDP different from IP?
- TCP
  - Stream-based (not datagram)
  - Connection-oriented
  - Reliable
  - Ordered
- SCTP
  - Newer protocol, combines TCP/UDP

Basic communication primitives

- Unicast (or point-to-point)
  - Messages from one process to another process
- Broadcast
  - Messages from one process to all others
- Multicast
  - Messages from one process to a group of processes
  - Can be implemented more efficiently than broadcast

Outline

Memcached

- Key–Value store
  - Stores `value` corresponding to a particular `key`
  - akin to a Python dictionary
- Used by server applications to cache data
  - e.g. from database queries
- Each process (usually on a separate machine) stores a portion of the dictionary
- Uses hash function on key to select a process

Amazon S3

- Object store
  - Unlike memcached, intended for storage
- Store and retrieve items using keys
- Eventually Consistent

Eventual consistency

Verbatim from Amazon S3 manual:

- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.
- A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.

Google Cloud Spanner

Cloud Spanner is the only enterprise-grade, globally-distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale.

- Google Cloud Spanner Marketing Materials

Outline

Leader Elections

- Many "distributed" algorithms are centralized
  - One server, others are clients
  - Can simplify implementation
- How to pick a server from among n processes?
- How to pick a server if current server disappears?

Mutual Exclusion

How can n processes co-ordinate to allow one of them to have exclusive access to a resource?

- Imagine a database that can be updated by only process at a time

Deadlock Detection and Recovery

How can n processes detect (and resolve) a deadlock?

- Imagine a distributed database where each record in a query needs to be locked before being updated
- Two queries may end up deadlocked

Termination Detection

When can n processes determine that they are done and quit?

- Note that there might be messages *in flight*

Checkpointing and Recovery

Checkpointing means saving the state of a process to storage so that if the process fails, it can be recovered by restarting and reading from the previous checkpoint.

- In distributed algorithms, state is globally distributed
- How to checkpoint a distributed algorithm?
- How to recover from a checkpoint in a distributed fashion?

Consensus (under uncertainty)

How can n processes agree on something in the presence of failure?