Passing Messages while Sharing Memory

Naama Ben-David

Based on joint work with
Marcos Aguilera, Irina Calciu, Rachid Guerraoui, Virendra Marathe,
Erez Petrank, Sam Toueg, Igor Zablotchi

Distributed Computation

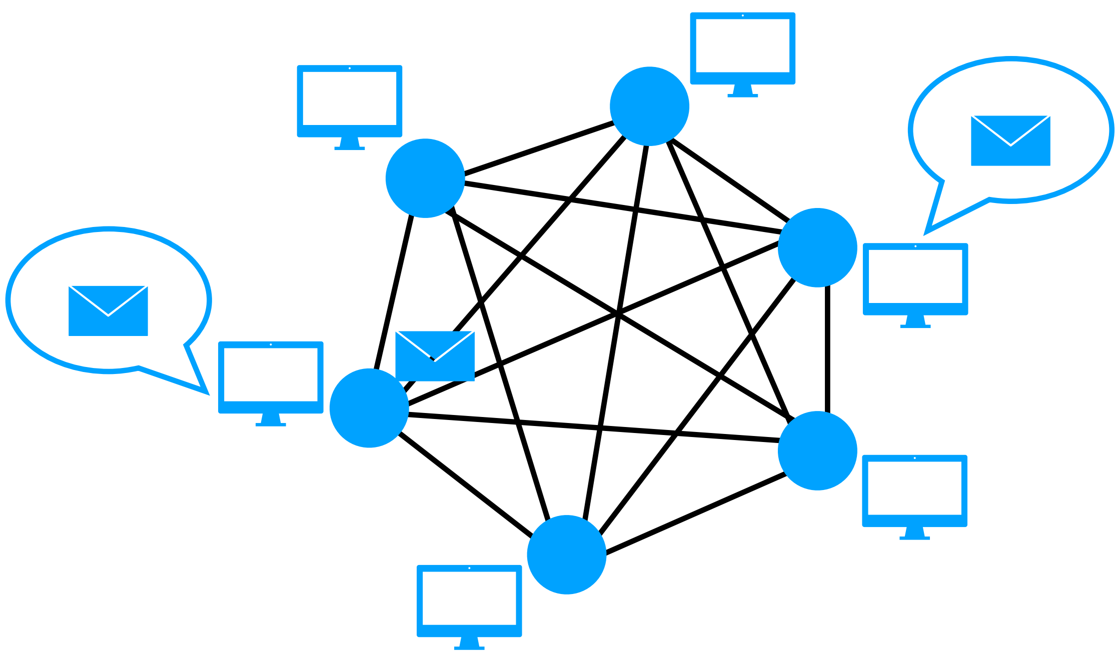
Many computation units communicate with each other

- Data centers, internet

- Operating System Scheduling

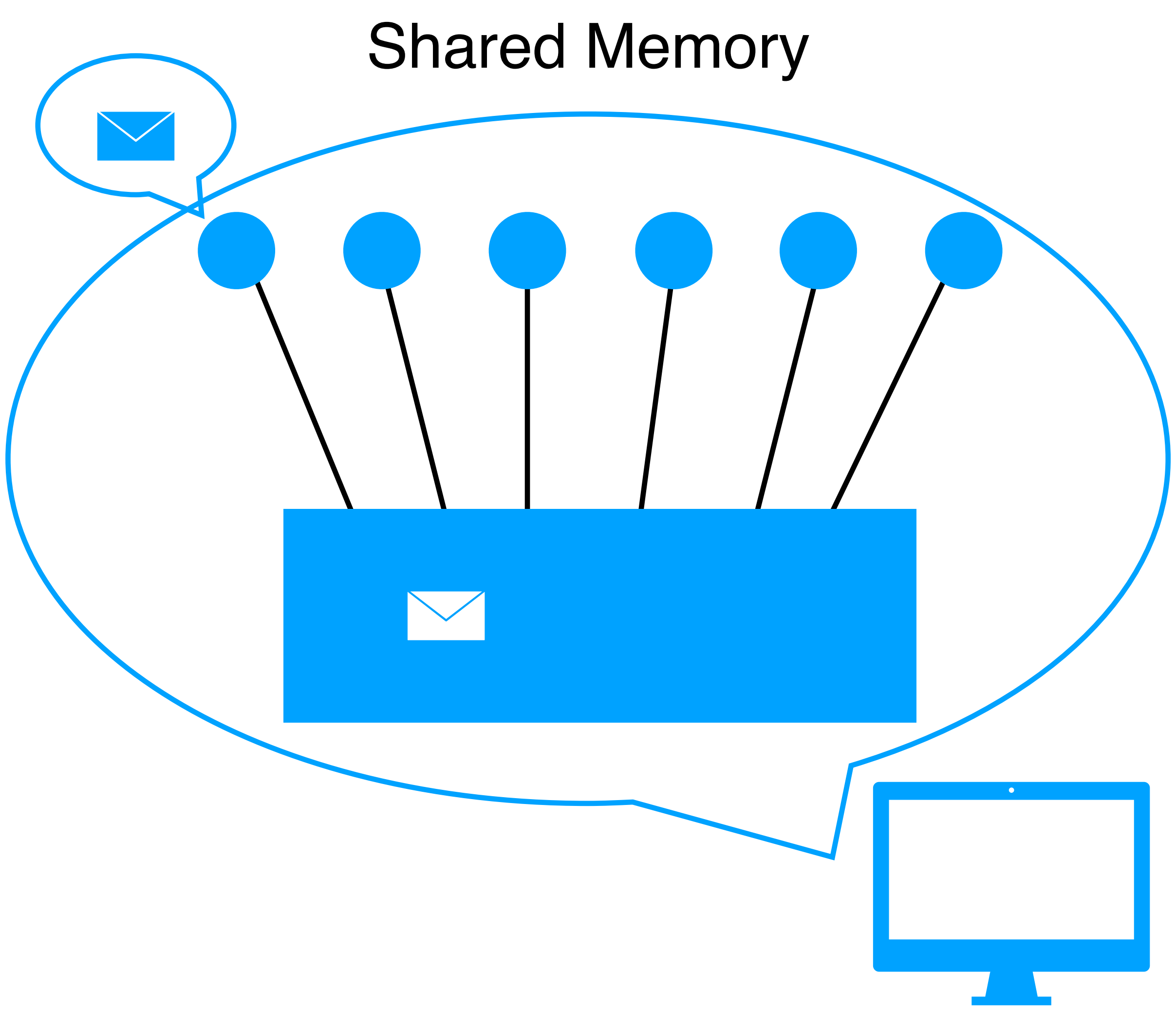
Message Passing

Message Passing



- Application: Data centers, internet
- Point-to-point messages over links

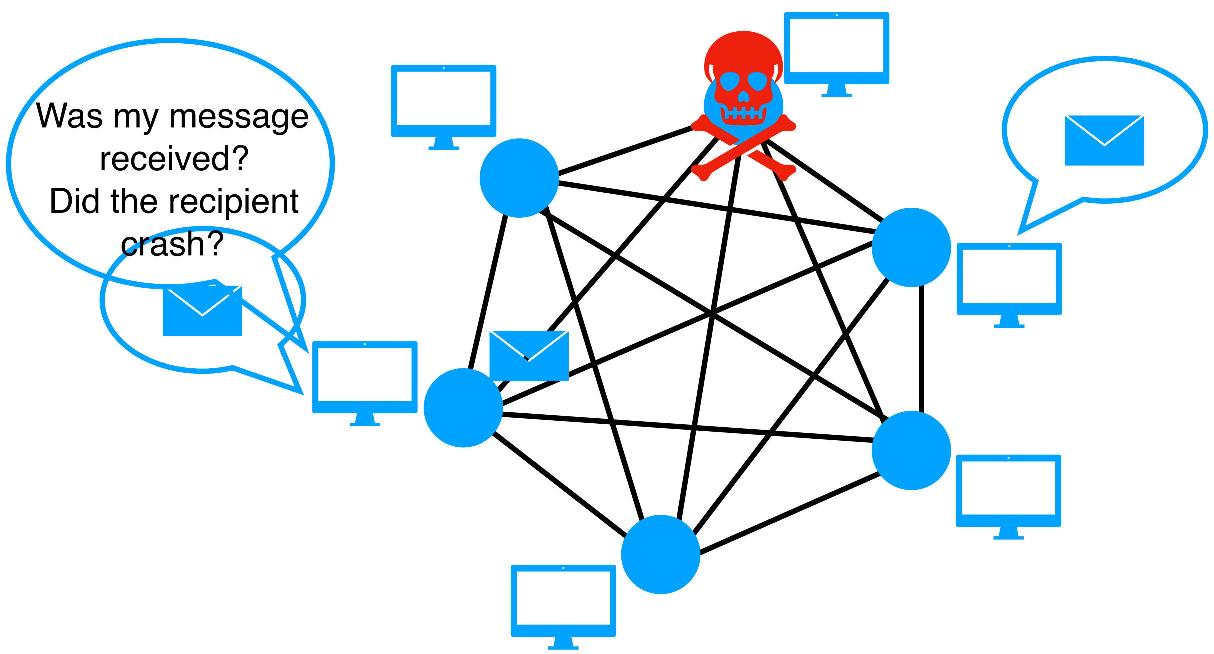
Shared Memory



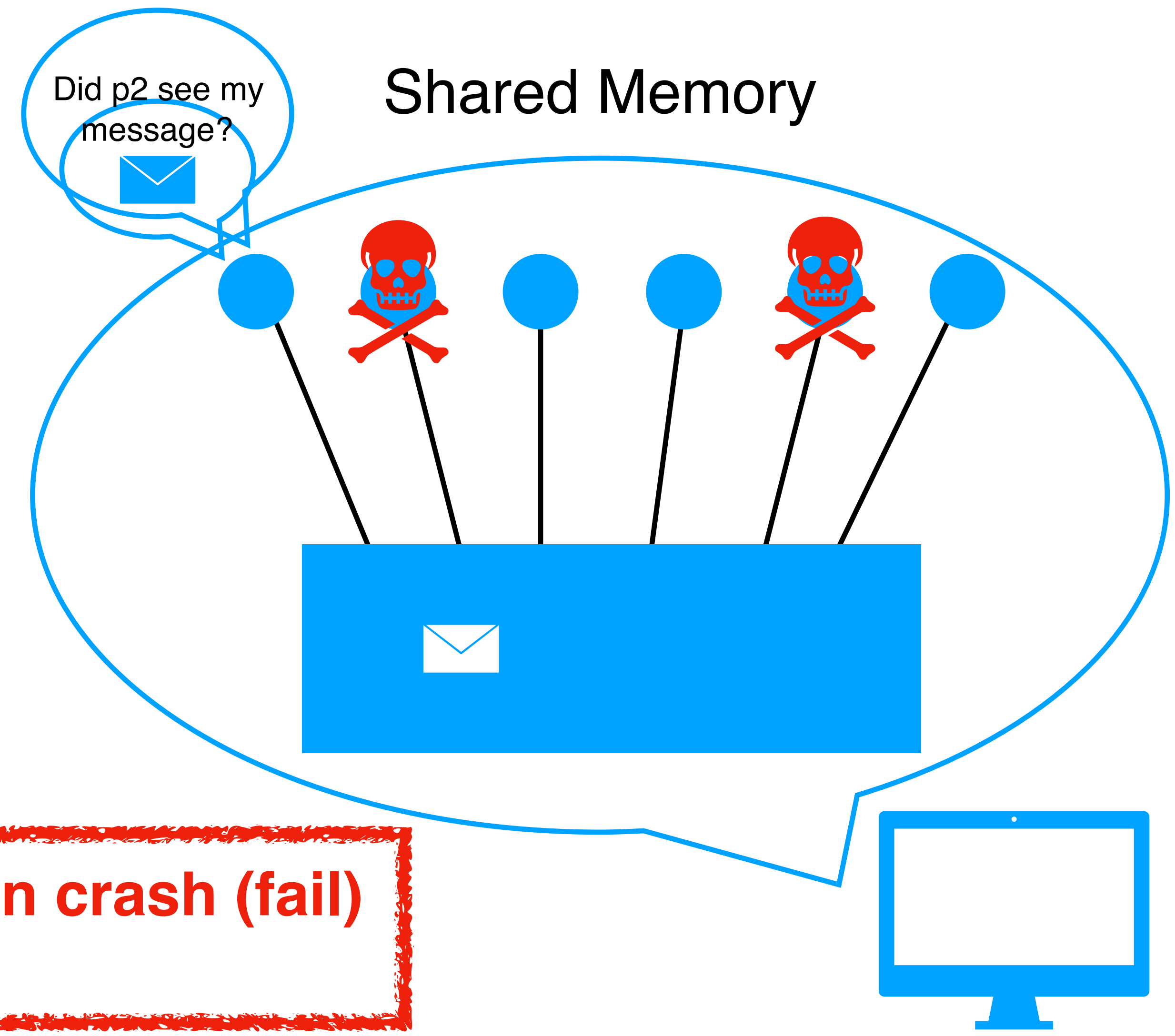
- Application: Multiprocessor computers
- Write and read common memory

Two Models

Message Passing



Shared Memory

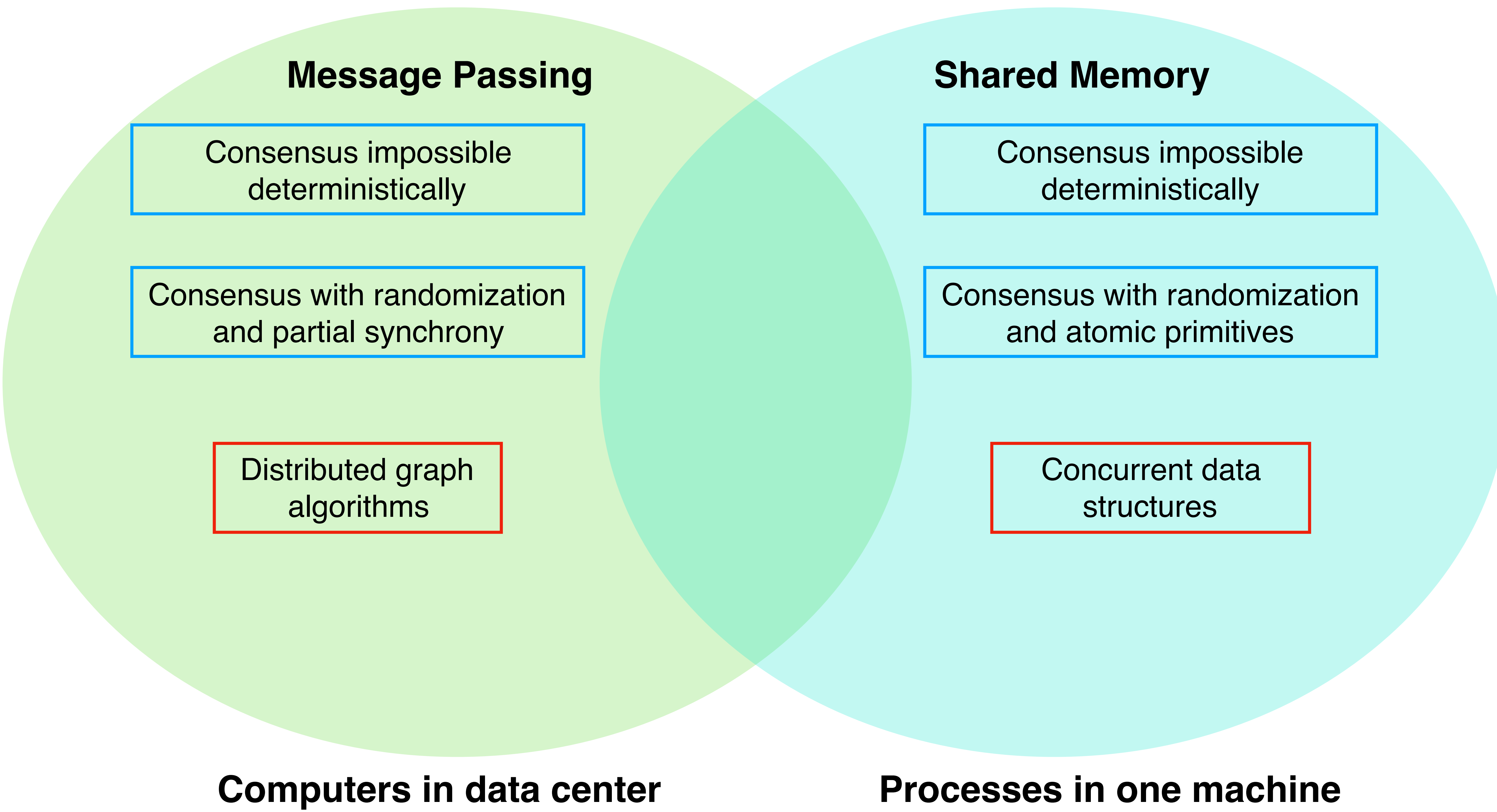


- **Processes can crash (fail)**
- **Asynchrony**

- Application: Data centers, internet
- Point-to-point messages over links

- Application: Multiprocessor computers
- Write and read common memory

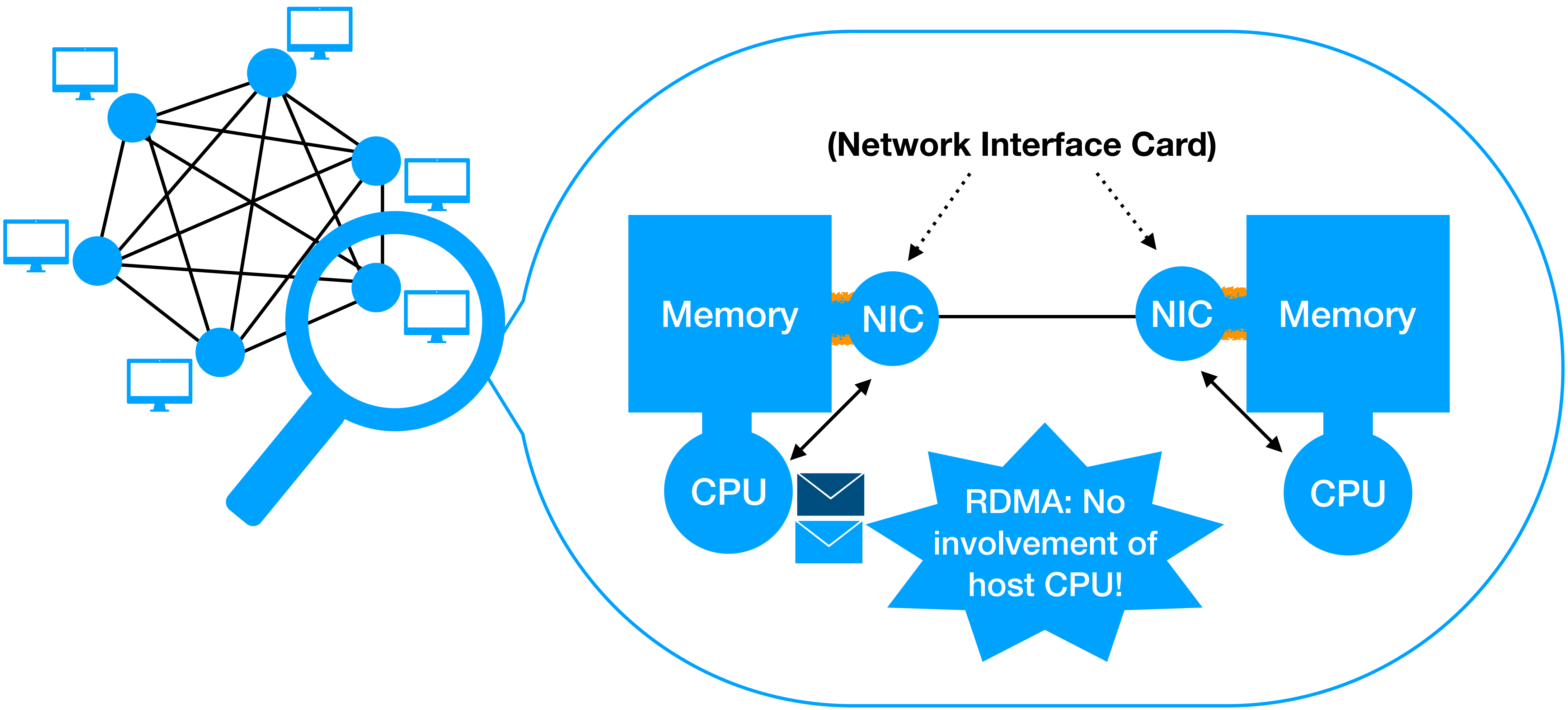
Two Models



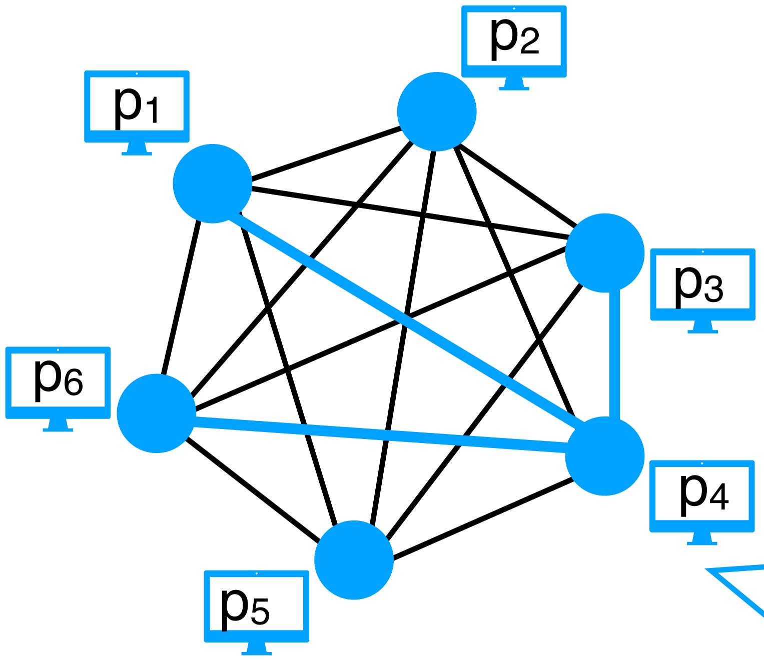
Message Passing
Consensus impossible deterministically
Consensus with randomization and partial synchrony
Distributed graph algorithms
Shared Memory
Consensus impossible deterministically
Consensus with randomization and atomic primitives
Concurrent data structures
Computers in data center
Processes in one machine

New Technology: RDMA

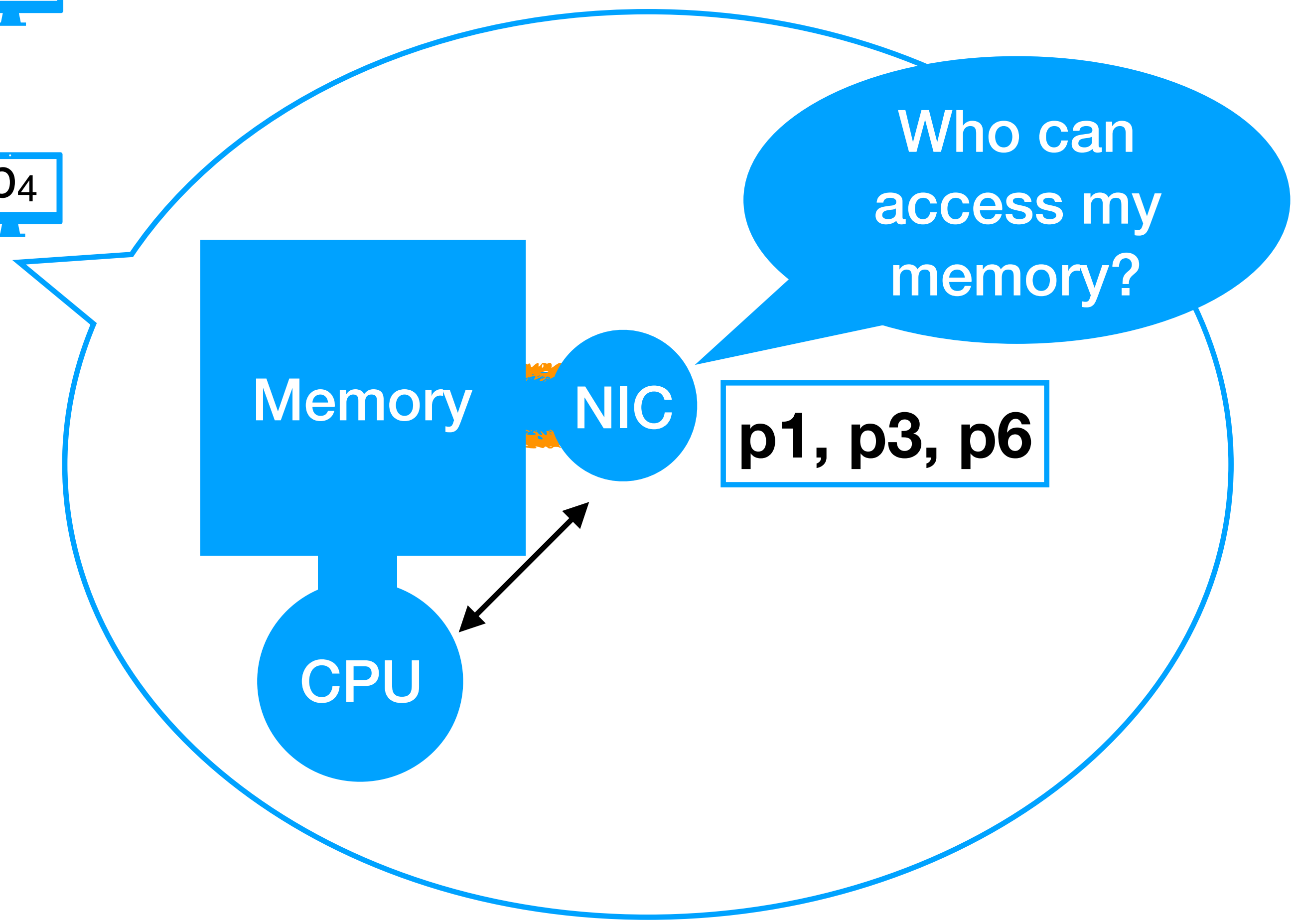
Remote Direct Memory Access



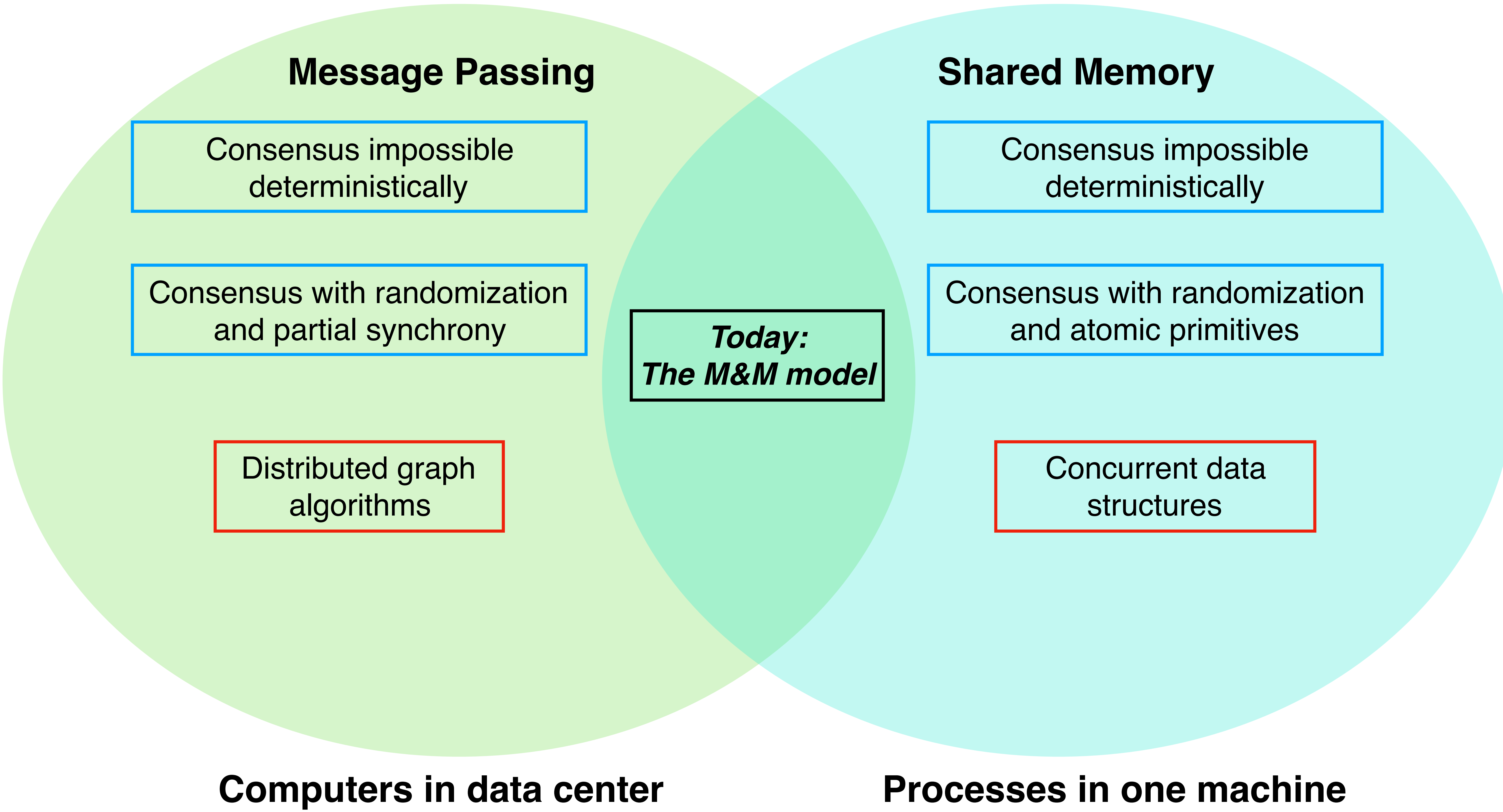
New Technology: RDMA



- Can choose RDMA connections
- Must maintain information about open connections in NIC's cache



Two Models


Message Passing
Shared Memory
Consensus impossible deterministically
Consensus with randomization and partial synchrony
Distributed graph algorithms
Today: The M&M model
Consensus impossible deterministically
Consensus with randomization and atomic primitives
Concurrent data structures
Computers in data center
Processes in one machine

What do we gain by combining the two models?

Equivalence

ABD'95:

"~~Message passing and shared memory are equivalent!~~"

computationally

"The models can solve the same set of problems"

What about tolerance to process failures?

What about synchrony requirements?

What about efficient algorithms?

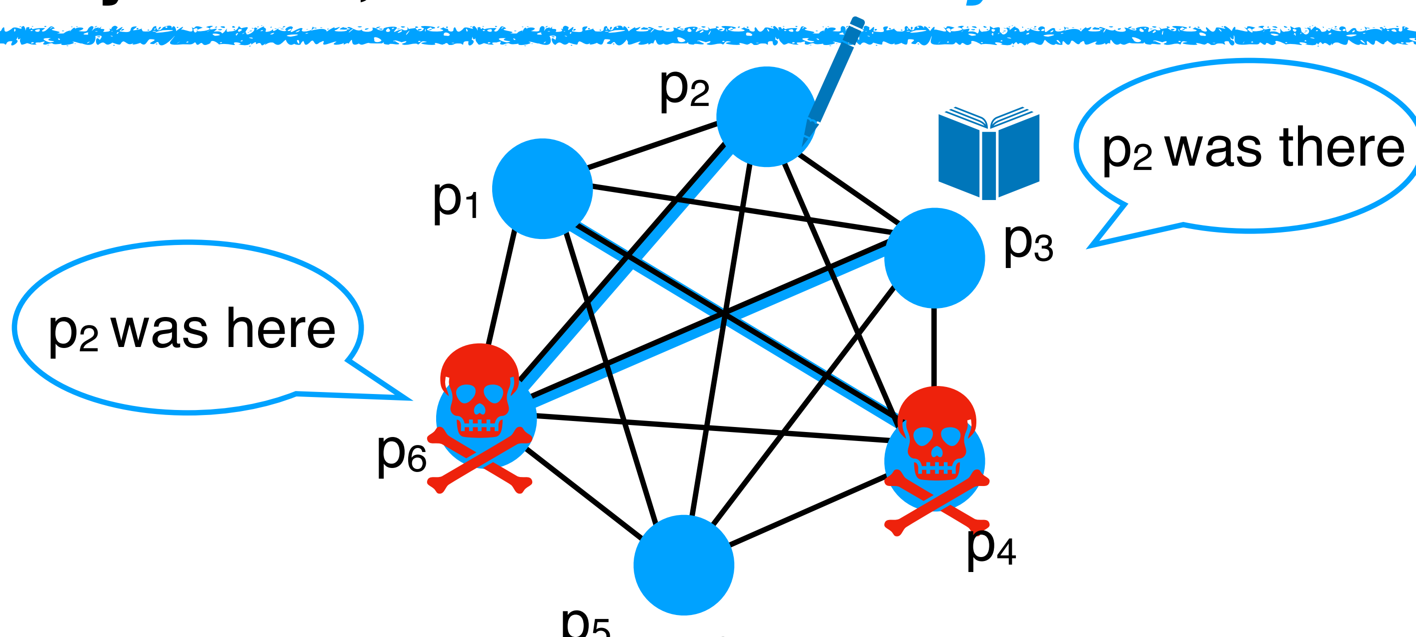
Outline

- Unifying Model: ***message-and-memory (M&M)*** model
- Consensus
    - Part 1: Process Crashes
        - ***Simulation Algorithm***
        - ***Tolerance lower bound***
    - Part 2: Memory Crashes
        - ***Definition and Intuition***
        - ***Disk Paxos and Disk Permissions***
- ***Leader election*** requires less synchrony in the M&M model

The M&M model

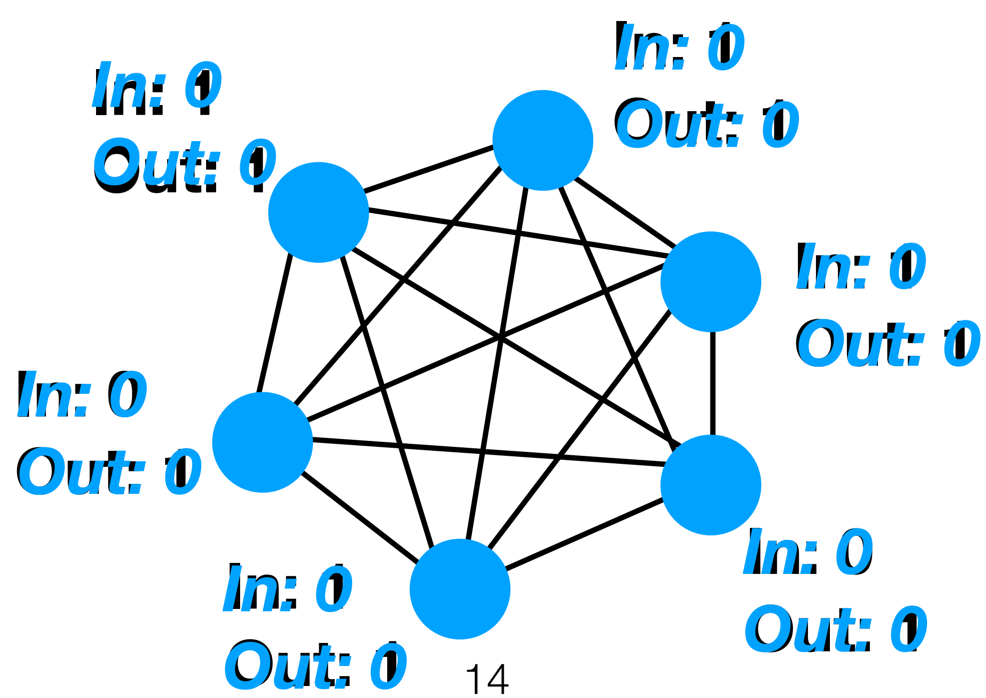
- ***Asynchronous*** network of *n* processes with up to *f* ***crash failures***
- Fully-connected message passing network: ***nodes=procs, edges=links***

- ***Each node owns a piece of memory***
- **Shared memory graph, *G_{SM} = (V, E)***
- **Nodes *u* and *v* can access each other's memory iff (*u,v*) ϵ *E***
- **Processes may crash, but their *memory remains accessible***



Consensus: Definition

- **Input**: every process gets either 0 or 1 as input
- **Output**: Every process outputs either 0 or 1
    - Agreement: All live processes output the same value
    - Validity: output value must be input of some process
    - Termination: must terminate



Outline

- Unifying Model: ***message-and-memory (M&M)*** model ✓
- Consensus ✓
  - Part 1: Process Crashes
    - ***Simulation Algorithm***
    - ***Tolerance lower bound***
  - Part 2: Memory Crashes
    - ***Definition and Intuition***
    - ***Disk Paxos and Disk Permissions***
- ***Leader election*** requires less synchrony in the M&M model

Part 1:
Process Crashes

Published in PODC’18

Consensus: Fault Tolerance

All processes must ***agree*** on the same value

Message Passing: Cannot solve consensus with less than
n/2 +1 live processes

Shared Memory: Can solve consensus even with
1 live process

Goal: Tolerate ***f > n/2*** failures when solving consensus in M&M network

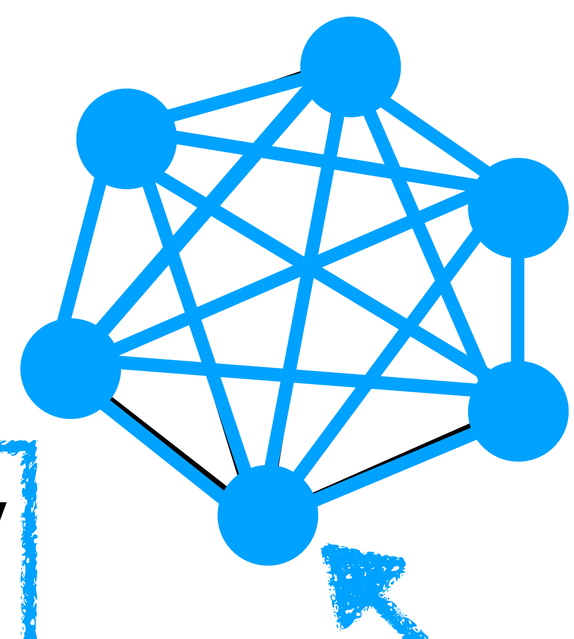
Fault Tolerance: Take 1

Idea: Connect all nodes over shared memory!

Now we can run any shared memory algorithm on this network

Require **only 1** process alive instead of **n/2 + 1**

… max degree is n-1



Goal: Keep max degree of G_{SM} low

Can we do better?

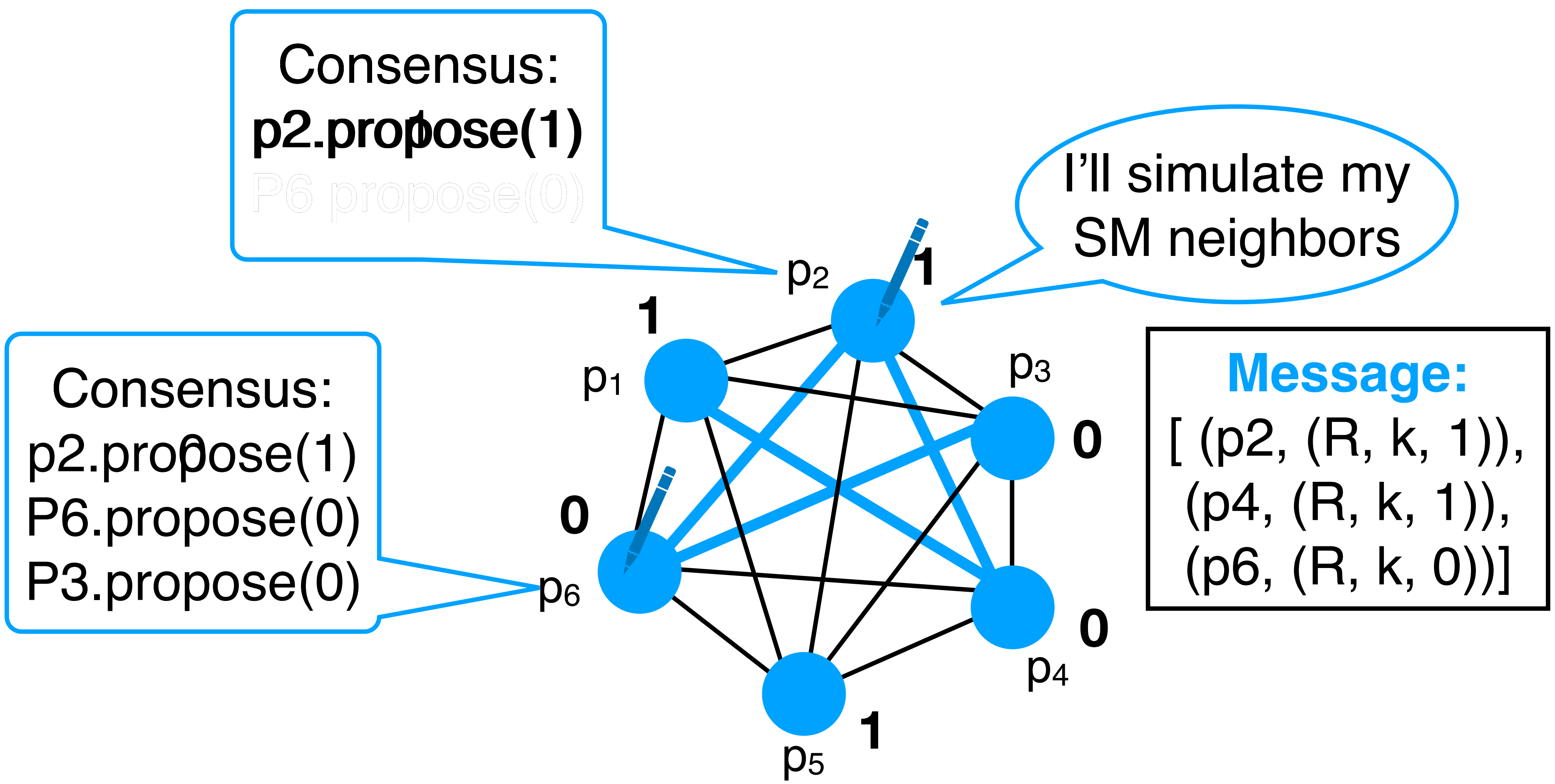
Infeasible to share memory with many processes

Everyone uses this memory location

M&M Consensus

Idea: Use shared memory to speak for your neighbors in a black-box message passing algorithm

Instead of sending just your message, agree with each neighbor using ***shared memory consensus***, then send a ***list of messages***



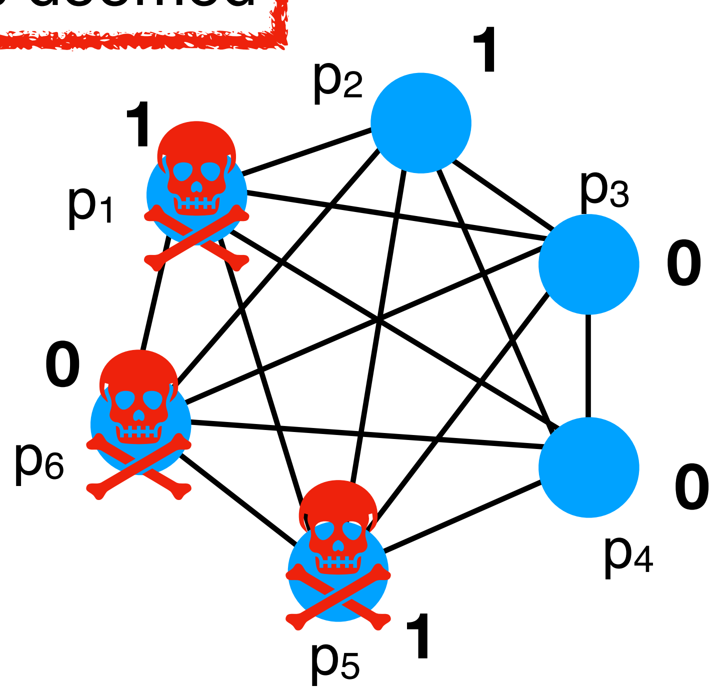
M&M Consensus

Idea: Use shared memory to speak for your neighbors in a black-box message passing algorithm

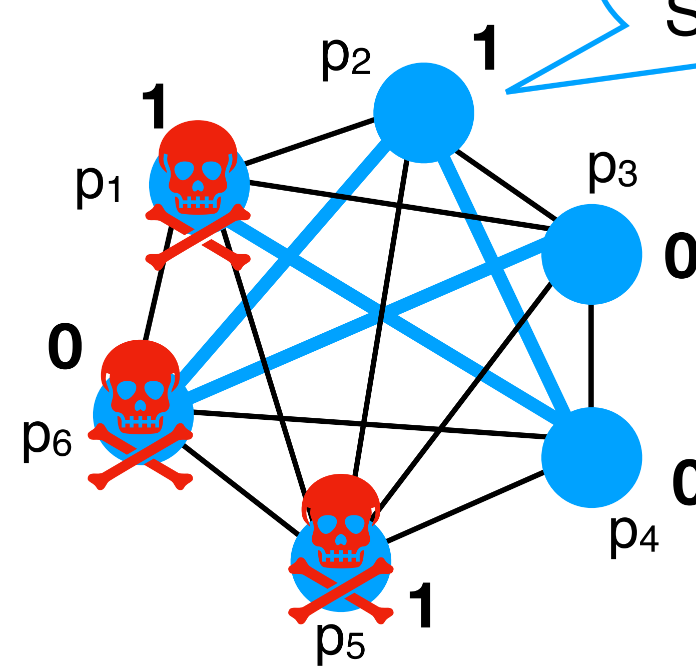
Instead of sending just your message, agree with each neighbor using ***shared memory consensus***, then send a ***list of messages***

Original Algorithm

Algorithm is doomed



M&M Algorithm



I'll simulate my SM neighbors

Message:
[(me, (R, k, 1)),
(p4, (R, k, 1)),
(p6, (R, k, 0))]

How Much Did We Gain?

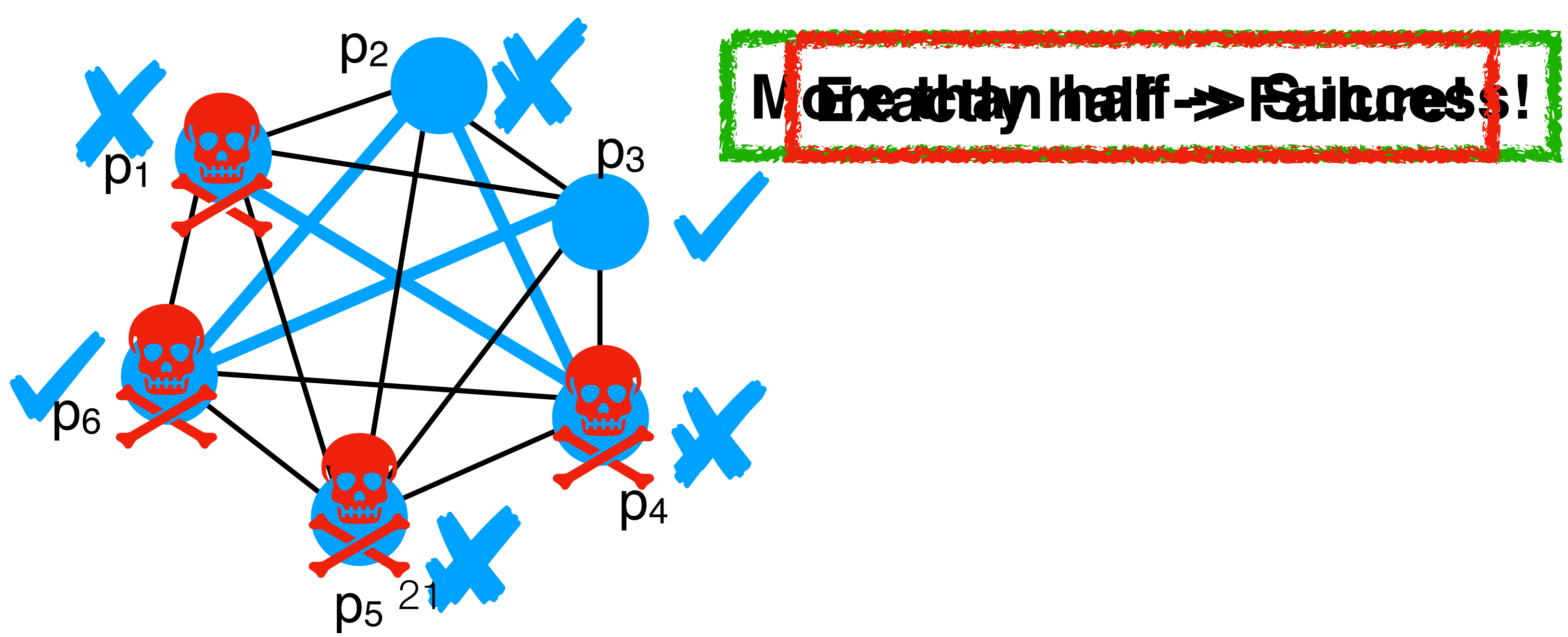
Depends on the shared memory graph G_{SM}

- More specifically, the number of ***neighbors of correct*** processes

Adversary chooses the set of correct processes

Want graphs with the following property:

All sets of at least n-f processes have many neighbors



Detour: Expander Graphs

Extremely well studied class of graphs

Let G=(V, E) be an undirected graph.

1. The ***vertex boundary*** of a set $S \subset V$ is $\delta S = \{ u \in V \mid \{u,v\} \in E, v \in S\} \setminus S$.
2. The ***vertex expansion ratio*** of G, denoted h(G), is defined as:

$$h(G) = \min_{S \text{ s.t. } |S| \leq |V|/2} |\delta S| / |S|$$

Detour: Expander Graphs

Neighbors of set S, not including S itself

Extremely well studied class of graphs

Let G=(V, E) be an undirected graph.

1. The ***vertex boundary*** of a set $S \subset V$ is $\delta S = \{ u \in V \mid \{u,v\} \in E, v \in S\} \setminus S$.
2. The ***vertex expansion ratio*** of G, denoted h(G), is defined as:

$$h(G) = \min_{S \text{ s.t. } |S| \leq |V|/2} |\delta S| / |S|$$

Detour: Expander Graphs

Extremely well studied class of graphs

Neighbors of set S, not including S itself

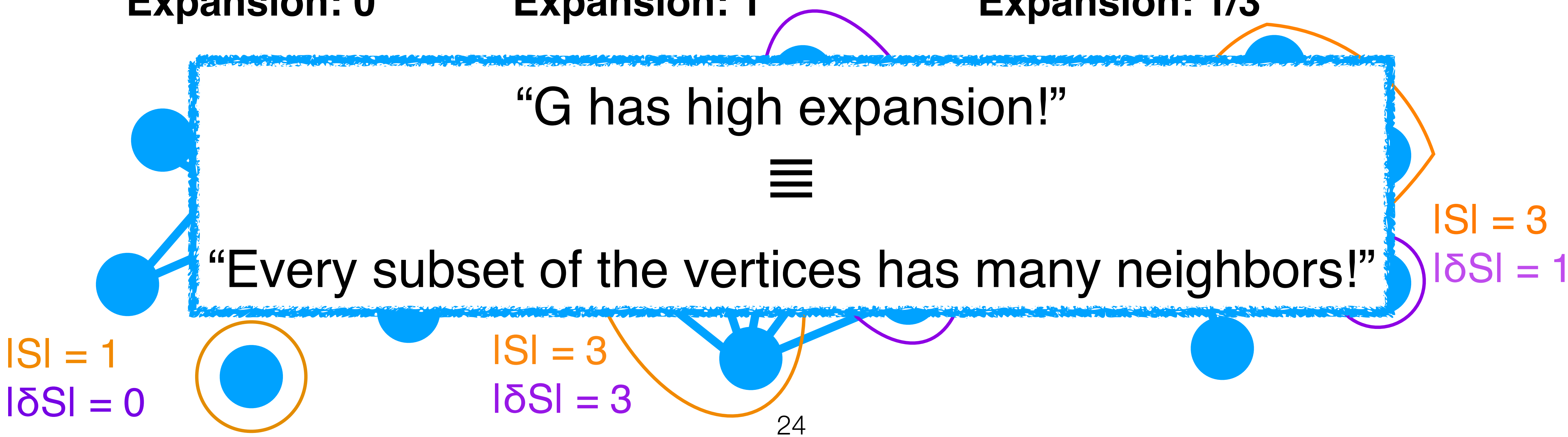
Let G=(V, E) be an undirected graph.

1. The ***vertex boundary*** of a set $S \subset V$ is $\delta S = \{ u \in V \mid \{u,v\} \in E, v \in S\} \setminus S$.
2. The ***vertex expansion ratio*** of G, denoted h(G), is defined as:

$$h(G)=\min_{S \text{ s.t. } |S| \leq |V|/2} |\delta S|/|S|$$

Min ratio of vertex boundary of S and the set S itself

Expansion: 0 **Expansion: 1** **Expansion: 1/3**



"G has high expansion!"

≡

"Every subset of the vertices has many neighbors!"

Putting it Together

- Think of ***set of live processes*** as S
- Adversary will pick S to be the set with the least expansion

G_{SM} with high expansion can tolerate more failures

Theorem: If G_{SM} has vertex expansion ratio h, then we can tolerate $f < \left(1 - \frac{1}{2 \cdot (1+h)}\right) \cdot n$ failures

Proof: The set of live processes, S, is of size ***|S| ≥ n-f.***
The original algorithm tolerates up to ***n/2*** failures.
We simulate that algorithm with ***|S| + |δS|*** live processes.
So, we can solve consensus if:

≥ (n-f)*h

simul procs = |S| + |δS|
≥ n-f + (n-f)*h > n/2
f < (1-1/(2(1+h)))*n

Outline

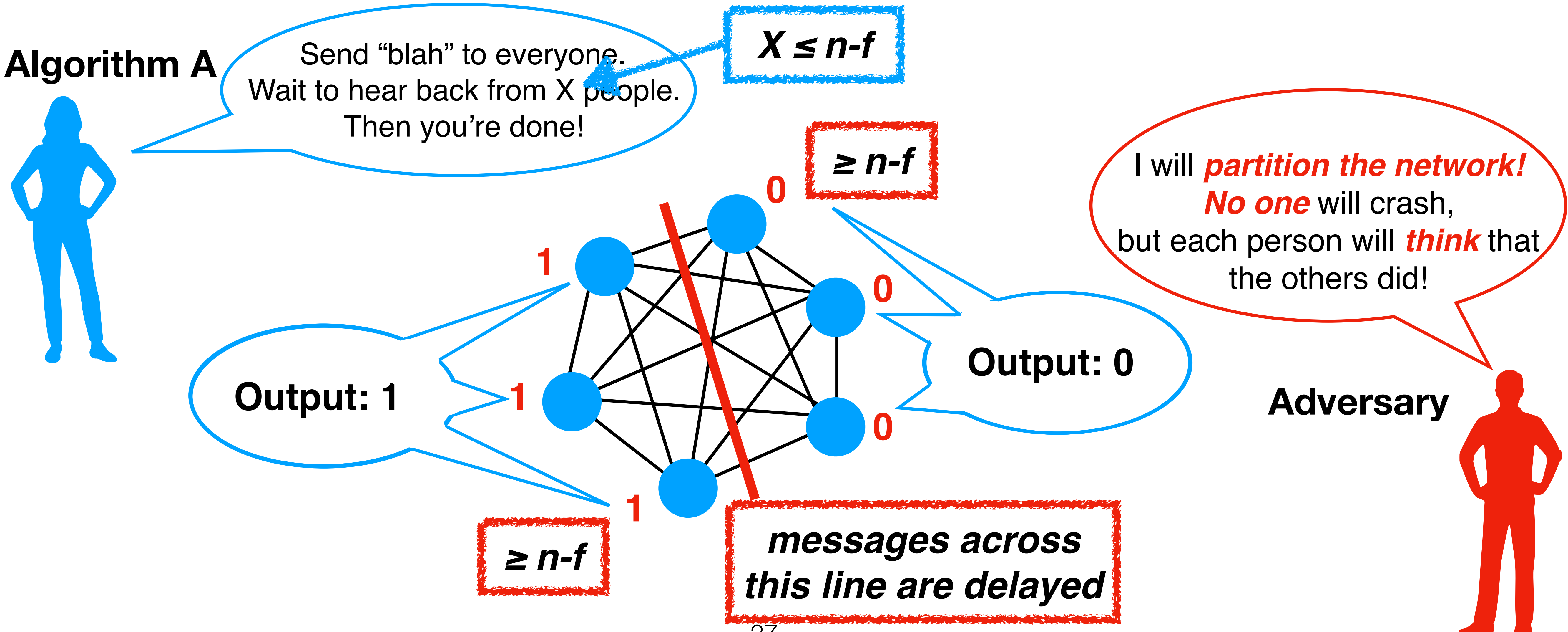
- Unifying Model: ***message-and-memory (M&M)*** model ✓
- Consensus ✓
  - Part 1: Process Crashes
    - ***Simulation Algorithm*** ✓
    - ***Tolerance lower bound***
  - Part 2: Memory Crashes
    - ***Definition and Intuition***
    - ***Disk Paxos and Disk Permissions***
- ***Leader election*** requires less synchrony in the M&M model

Message Passing: Partition

Majority requirement is ***inherent***.
[Ben-Or'83]

- **send M to $S \subseteq \{p_1, \ldots, p_n\}$**
- **wait to hear back from S'**

Assume by contradiction that algorithm ***A*** implements ***consensus*** in a system where ***f ≥ n/2***

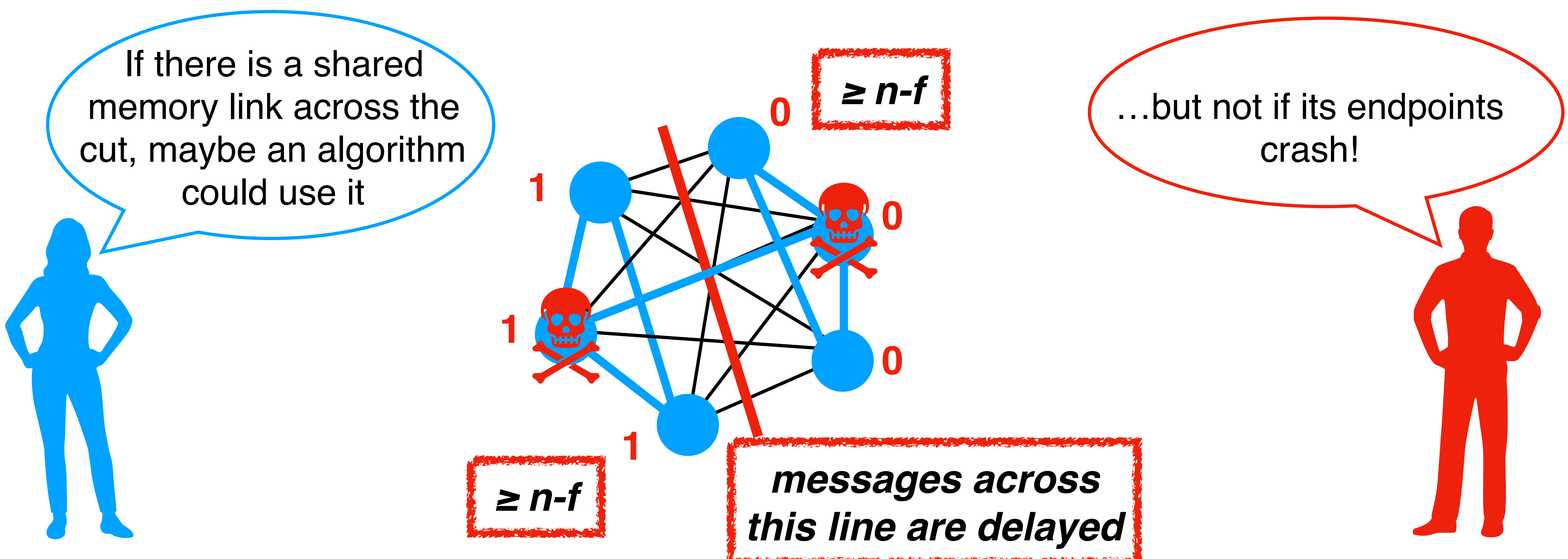


M&M Lower Bound

Where does partitioning fail in M&M?

- Shared memory links are stronger

Partitioning still works for a cut with no shared memory links



M&M Lower Bound

Define ***Shared-Memory Cut C=(B, S, T):***

Partitions graph into ***3 parts***: S, T, and B (boundary), such that
1. There are no edges between S and T, and
2. B can be partitioned into B1 and B2 where there are no edges {s, b2} and no edges {t, b1}

Theorem: In an M&M network with shared memory graph G = (V, E), consensus cannot be solved if $f > \min_{(B,S,T) \text{ in } Cuts(G)} n-|S|$

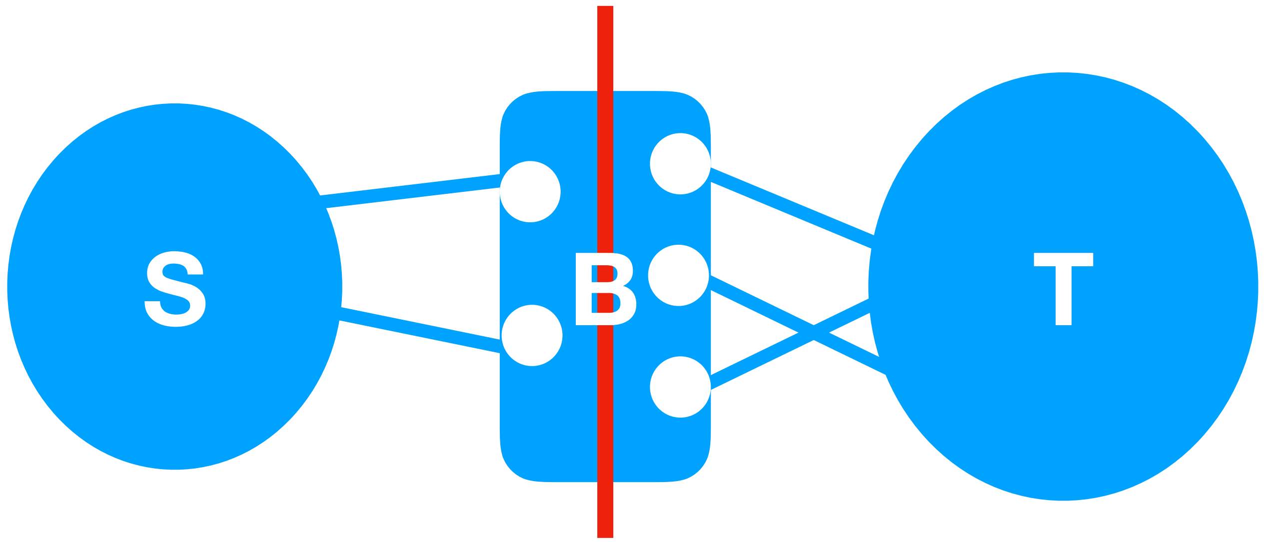
Intuition: **Adversary cuts in the middle of B, and crashes all nodes in B. Then S and T cannot communicate.**

Note: It must hold that $|S| \geq n-f$ and $|T| \geq n-f$

M&M Bound vs Expansion

To tolerate many failures, need to have **large SM-cuts**

i.e., every set S where **|S| ≥ n-f** must have **many neighbors**



Recall: Expansion ration considers all sets S where **|S| < |V|/2** and requires all such sets to have **many neighbors**

To tolerate many failures, relatively large sets must have a large vertex boundary

Outline

- Unifying Model: ***message-and-memory (M&M)*** model
- Consensus
  - Part 1: Process Crashes
    - ***Simulation Algorithm***
    - ***Tolerance lower bound***
  - Part 2: Memory Crashes
    - ***Definition and Intuition***
    - ***Disk Paxos and Disk Permissions***
- ***Leader election*** requires less synchrony in the M&M model

Part 2:
Memory Failures

Disclaimer: Ongoing research.

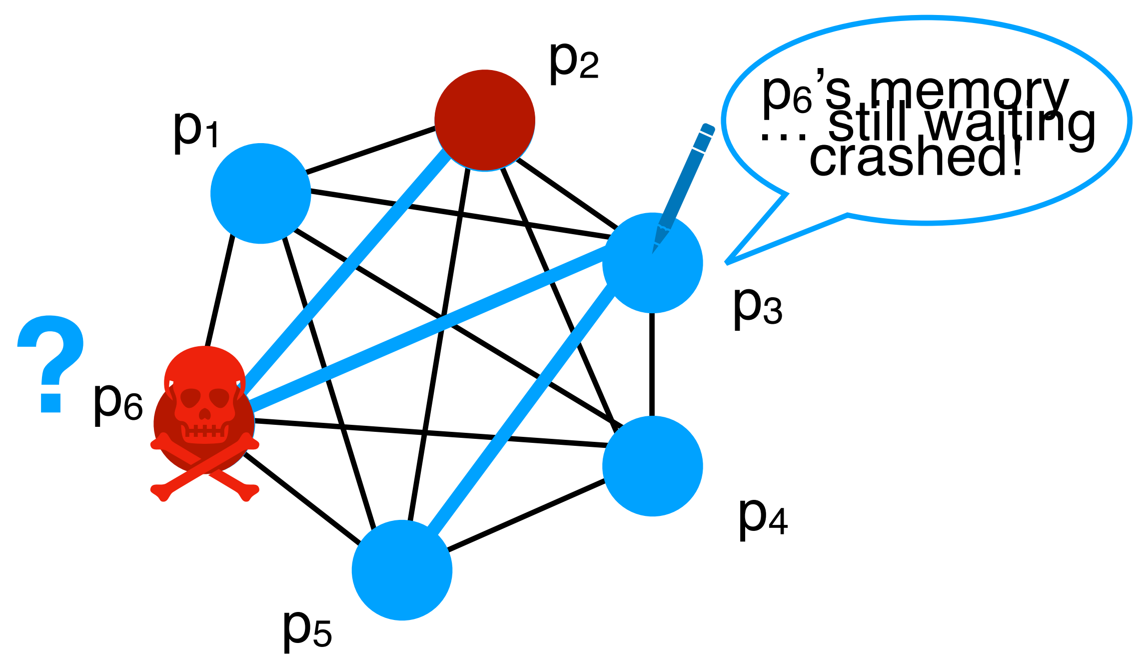
Memory Failures

What happens if memory crashes too?

How do we define memory failures?

Tougher to deal with, but requires less synchrony

- Responsive: failed memory returns NACK
- Unresponsive: failed memory hangs forever

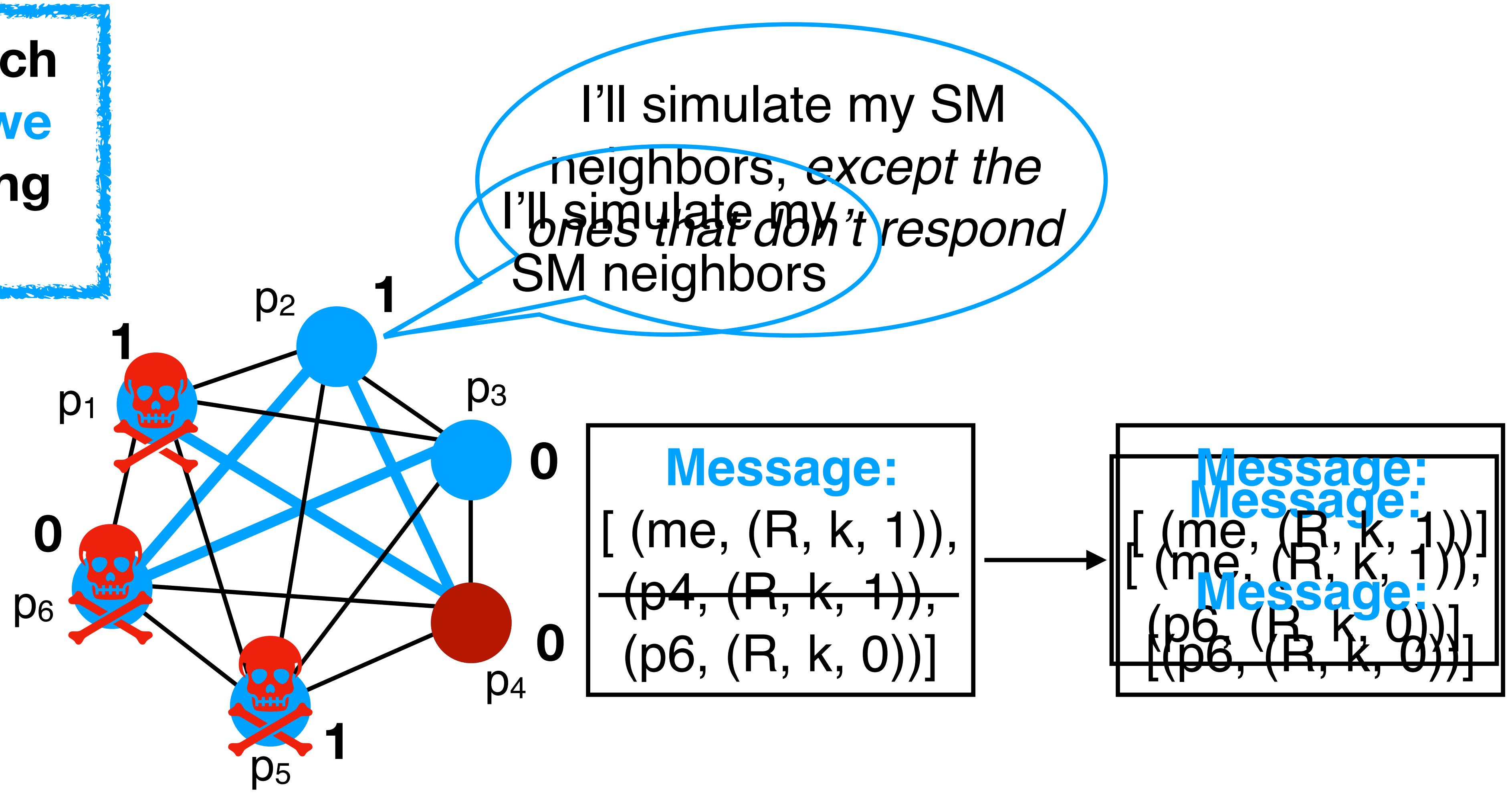


Memory Failures in Simulation

How do we deal with memory failures in our simulation?

- Do not simulate memory-and-process crashed nodes

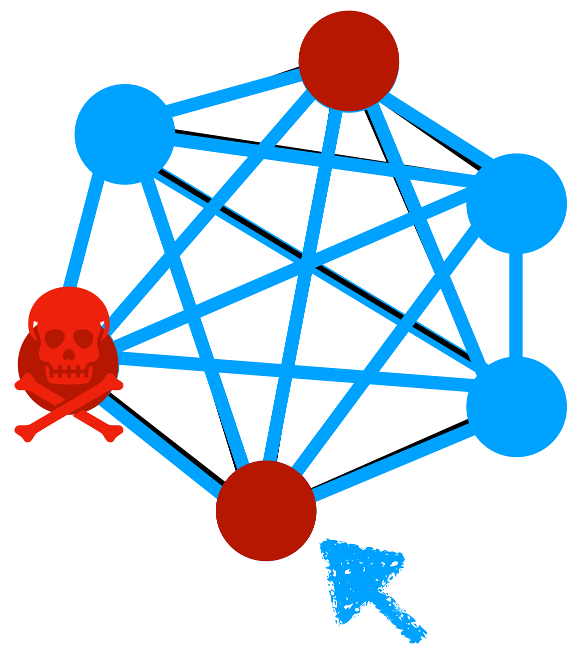
Must send value of each neighbor as soon as we know it, without waiting for all of the others



Fully Connected Graph

Not clear what to do even when graph is **fully connected**

Can no longer run a shared memory algorithm unchanged



Everyone uses this memory location

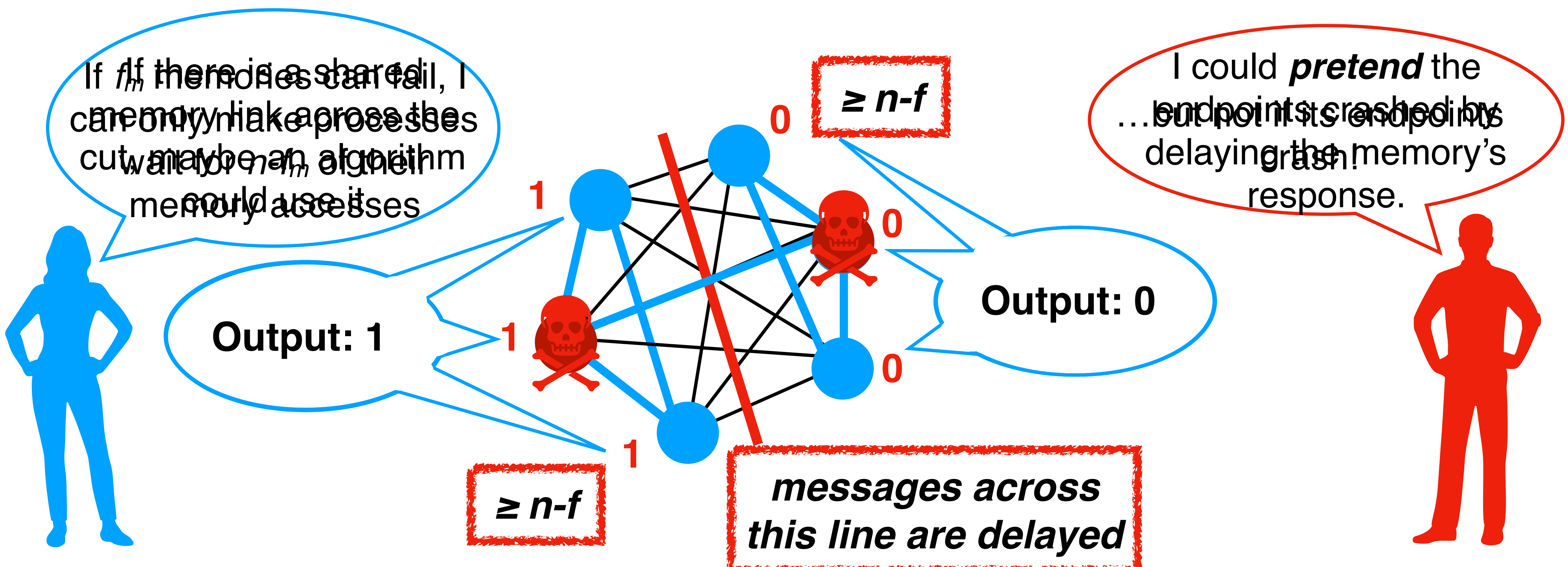
M&M Partitioning

Where does partitioning fail in M&M?

With memory failures, partition can cut through shared memory links

- Shared memory links are stronger **only if memory can't fail!**

Partitioning still works for a cut with no shared memory links



Quorums

How can we prevent a partition from occurring (in any model)?

If the set of processes I **sent information** to **overlaps** with the set of processes others **receive information** from

If X agents can fail, I will send my message to X+1 of them

Everyone can access p_5 so I will only write there

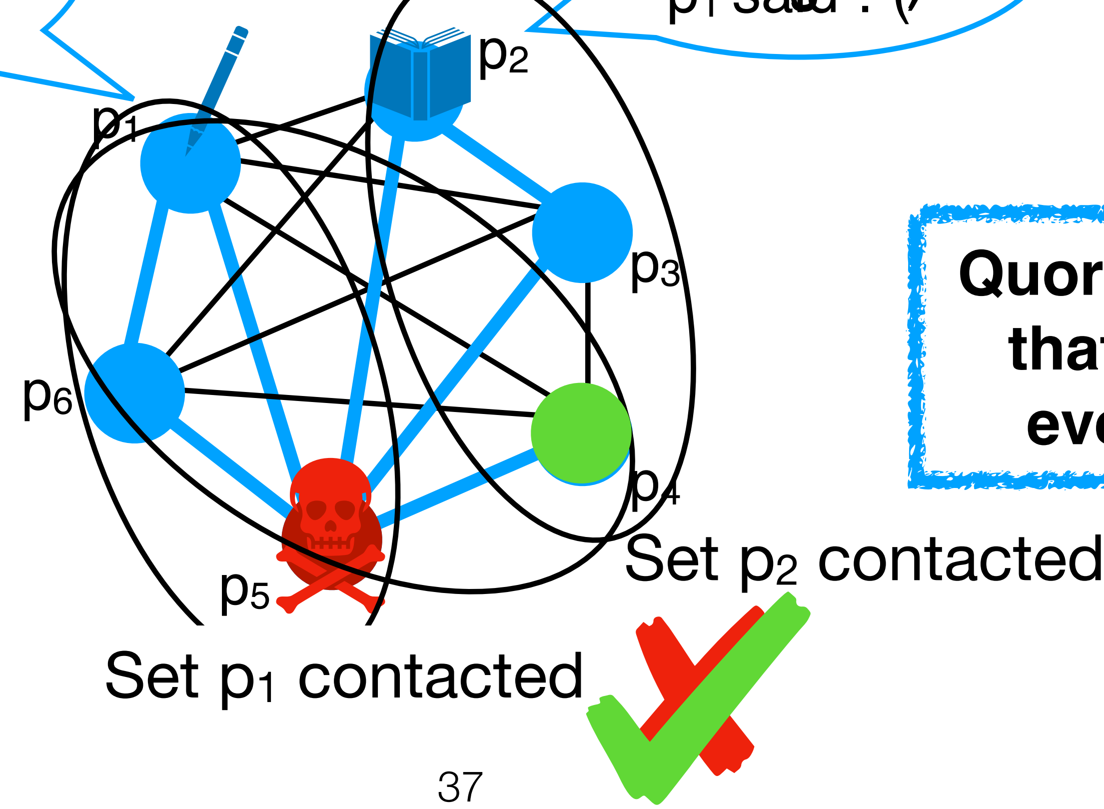
I don't know what p_1 said :-(

I got p_1's message :-)

Memory, processes, etc

If agents are reliable, accessing one is enough

If agents may fail, must contact enough to ensure at least one remains alive



Quorum: A set of agents that has overlap with every other quorum

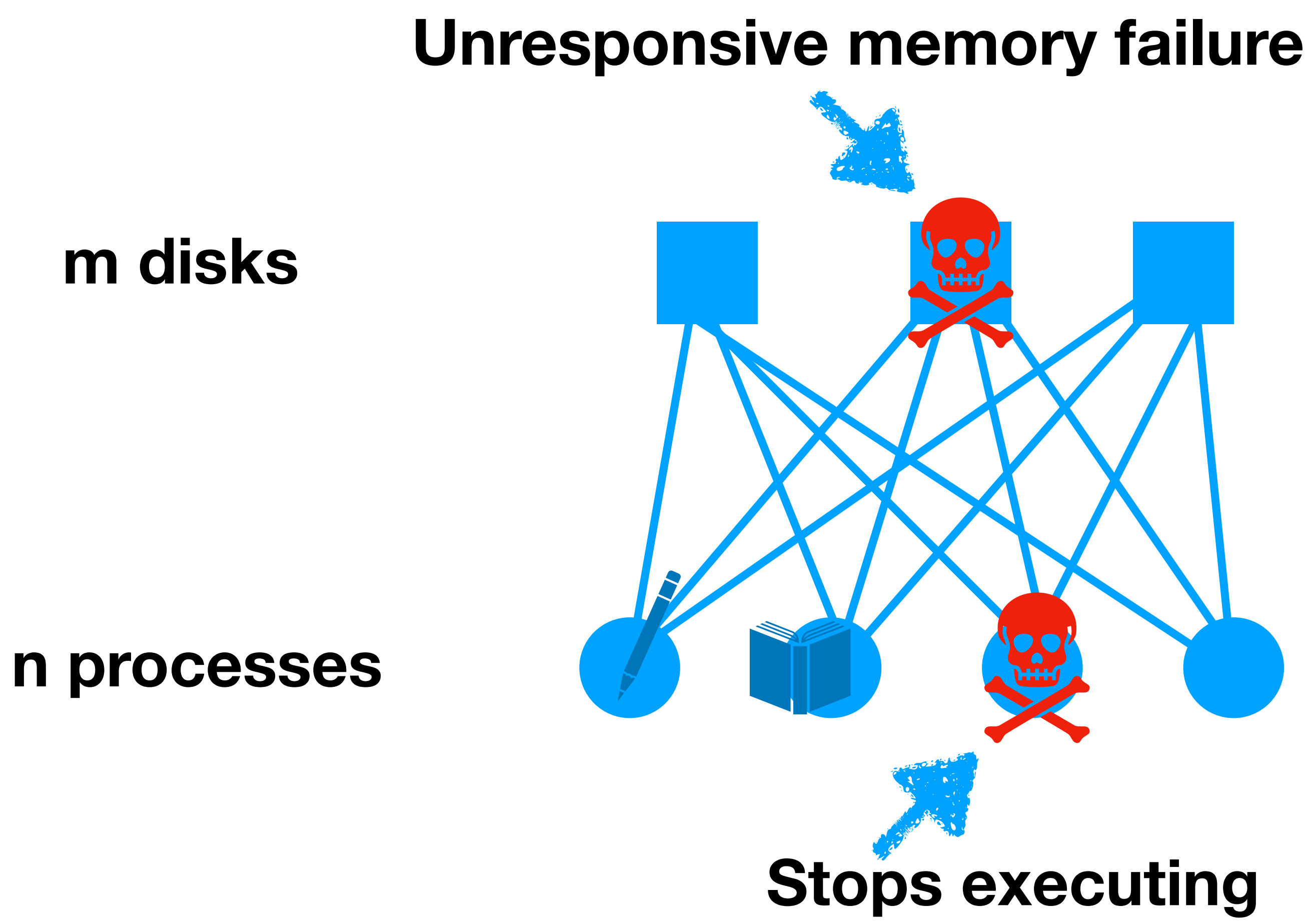
Outline

- Unifying Model: ***message-and-memory (M&M)*** model ✓
- Consensus ✓
    - Part 1: Process Crashes ✓
        - ***Simulation Algorithm*** ✓
        - ***Tolerance lower bound*** ✓
    - Part 2: Memory Crashes
        - ***Definition and Intuition*** ✓
        - ***Disk Paxos and Disk Permissions***
- ***Leader election*** requires less synchrony in the M&M model

Disk Paxos

Consensus using **disks** and **processes** [GafniLamport'02]

In disk model, consensus can be solved with 1 process and m/2+1 disks alive



Disk Paxos

[GafniLamport'02]

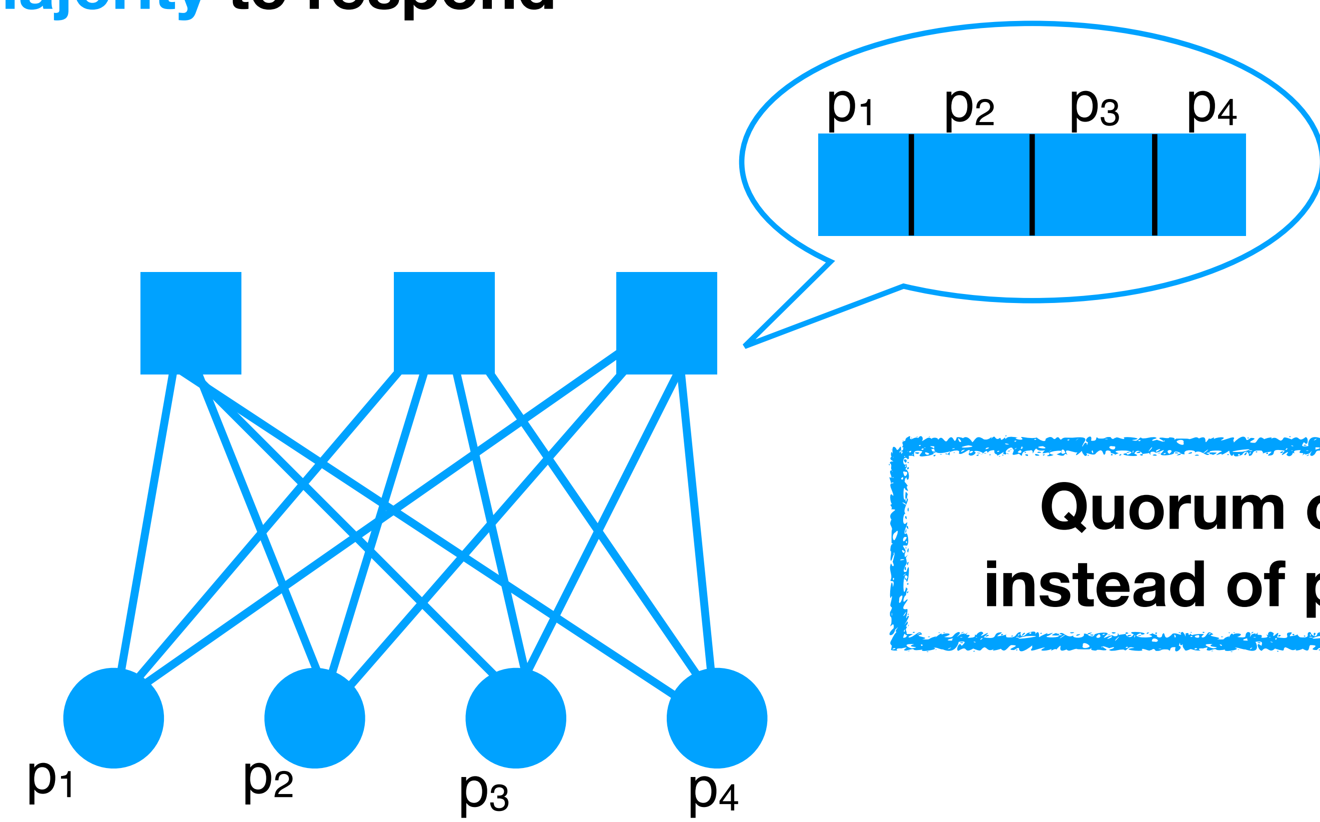
Idea: run classic message passing algorithm, but replace sends and receives with reads and writes

To send: **write your message in your slot in all disks;**
wait for majority to respond

To receive: read others' slots in all disks;
wait for majority to respond

m disks

n processes



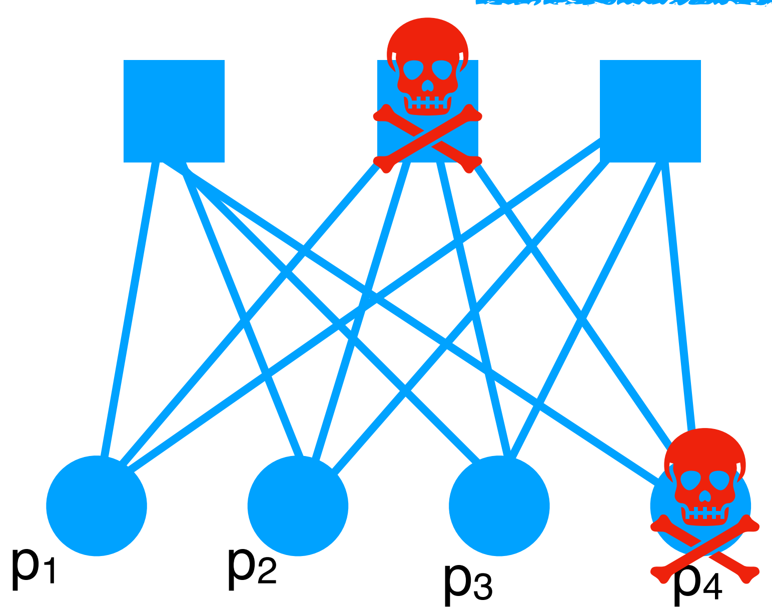
Quorum on disks instead of processes

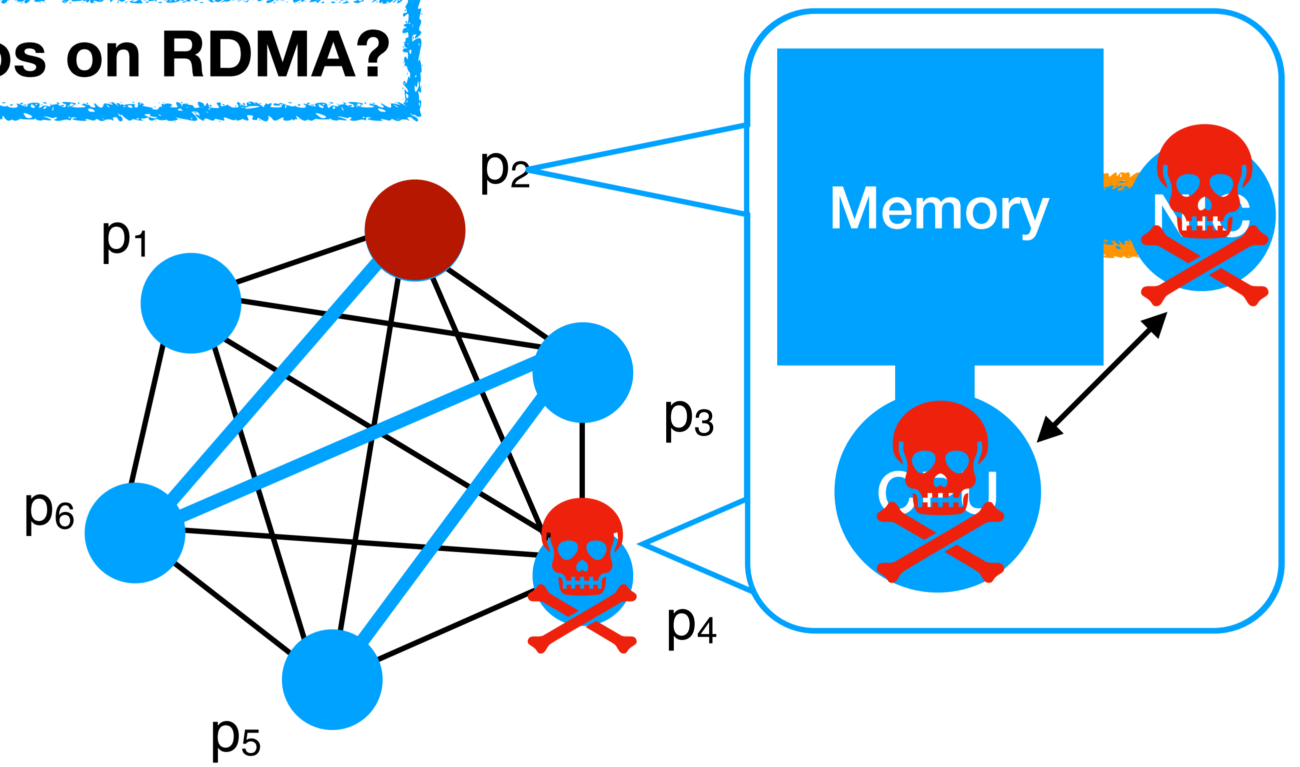
Disks vs RDMA

How similar is the disk model to RDMA?

- In RDMA, memory is associated with a specific process
- Process-only failures make sense; CPU error
- Memory-only failures make less sense, but interesting to study
- RDMA can also send messages!

Can we run disk paxos on RDMA?





Disk Paxos in RDMA

Can solve consensus in RDMA with **1 process** and **n/2+1 "disks"** alive

Can we do better?

Idea: use messages to expand quorum to include processes.

Algorithm for each step of Paxos:

- **Do one step of paxos on processes and one step of disk paxos on disks.**
- **Wait until a majority of (Process ∪ Disks) respond.**

Take away: if there are too few disks, processes can help, and vice versa

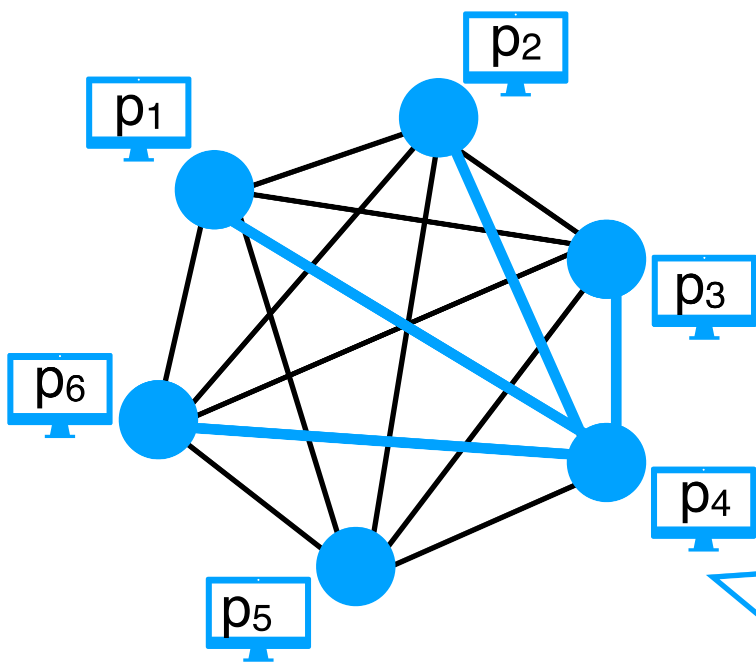
Partition argument shows that this is optimal.

RDMA: More details

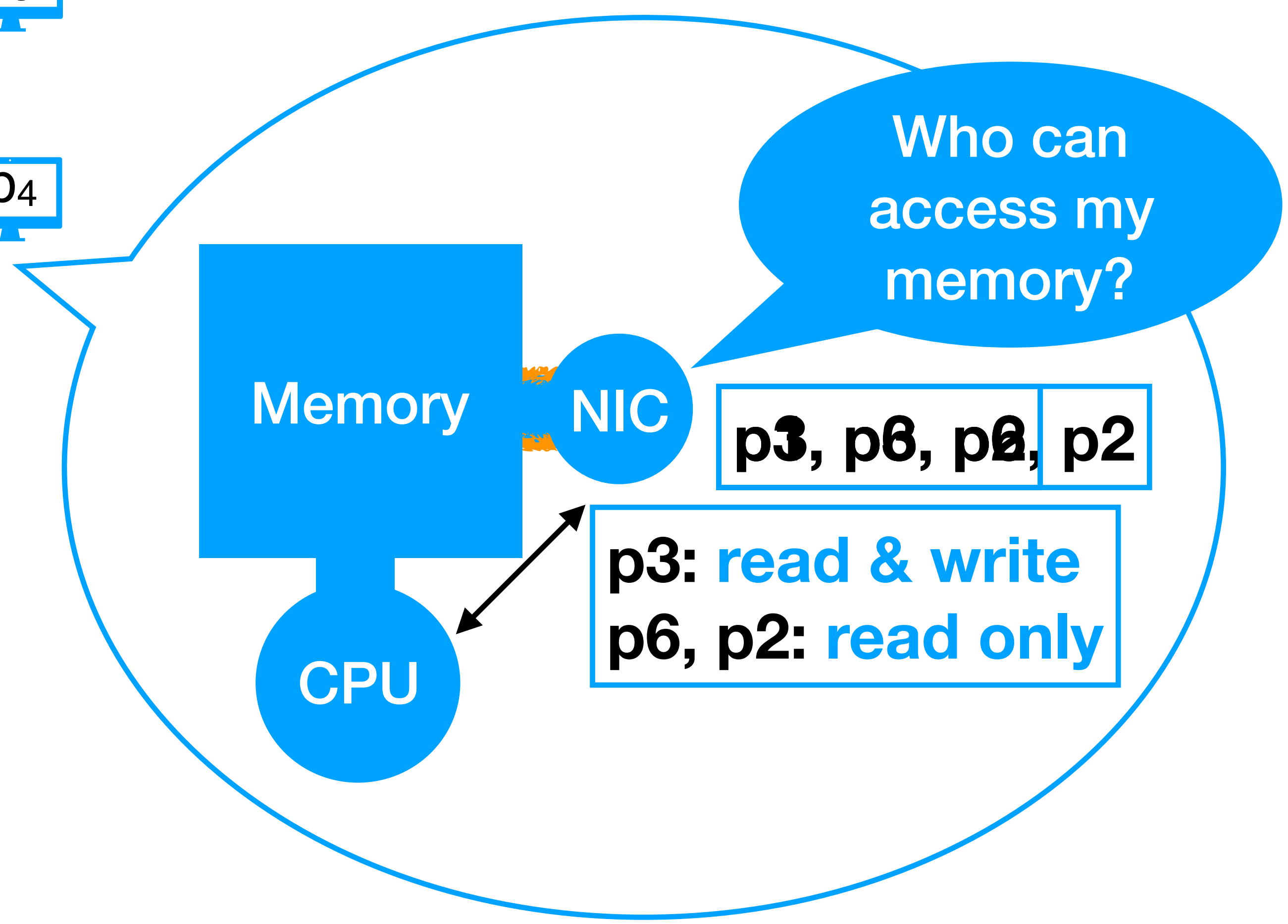
- Can *dynamically* choose RDMA connections

i.e., open and close RDMA connections

- Can specify read and write permissions



Can we gain something over the disk model using dynamic connections?



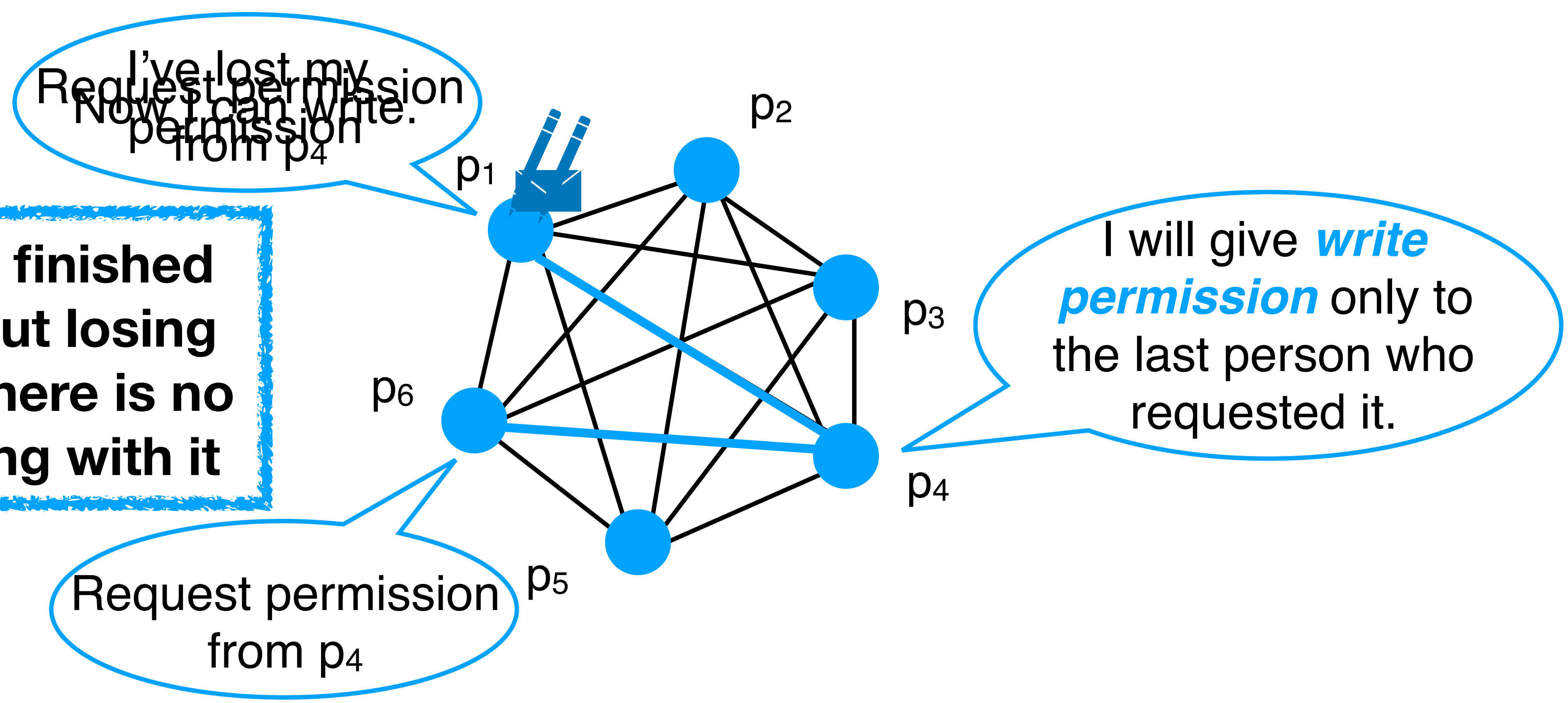
Disk Paxos with Permissions

In the Paxos algorithm, a proposer waits to hear back from others to **know whether there is someone competing with it**.

In Disk Paxos, this means **reading every value from every disk**.

Idea: leverage RDMA dynamic connections to get rid of this step.

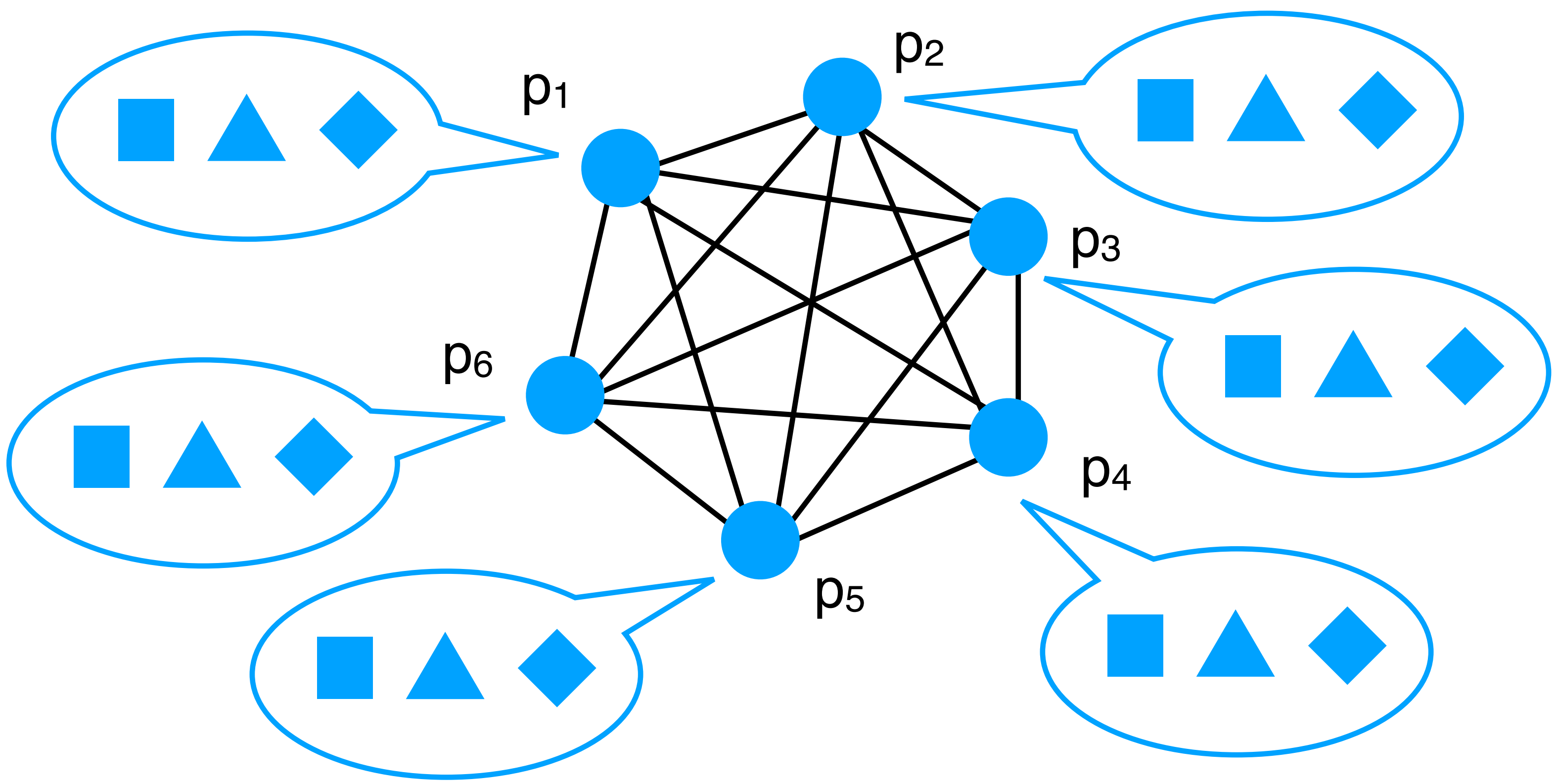
If a proposer finished writing without losing permission, there is no one competing with it



Replicated State Machine

Got rid of **one operation** on each disk, but only **when there is only one proposer**

But we might run consensus **many times!**



In practice, the system is **well behaved** most of the time

i.e., one designated leader proposes values

Byzantine Faults in RDMA

A **byzantine fault** is when a faulty process becomes **evil** instead of crashing

Message Passing | **RDMA** | **Shared Memory**

Message Passing

A LOT of research

- Hackers, software bugs
- Blockchains

Cannot solve consensus with n/3 byzantine processes

RDMA

Well motivated

Can use permissions to block byzantine process

Might be able to tolerate more failures by preventing lies

Shared Memory

Barely studied

- Byzantine faults unlikely within one machine
- A Byzantine process could completely corrupt the memory!

Outline

- Unifying Model: ***message-and-memory (M&M)*** model
- Consensus
  - Part 1: Process Crashes
    - ***Simulation Algorithm***
    - ***Tolerance lower bound***
  - Part 2: Memory Crashes
    - ***Definition and Intuition***
    - ***Disk Paxos and Disk Permissions***
- ***Leader election*** requires less synchrony in the M&M model

Summary

- ***Message-and-memory (M&M)*** model
- ***Consensus:***
    - Expanders tolerate many process failures
    - Disk model & permissions with memory failures

New exciting model, many new questions!

Thank you!