Lecture 11: Intro to Distributed Systems

Fall 2022
CS 484

Distributed System Examples

The Internet

Bitcoin

BitTorrent

TOR

Supercomputers

Web applications

Web Servers

Distributed Systems

Multi-Agent - made up of independent machines or programs working together

Transparency - it acts like a single system

Heterogeneity - it can be made up of different types of agents

Fault Tolerance - parts of it can fail, but the whole system will still work

Concurrency - multiple agents are running at the same time

Solving an impossible problem:

Sending a message does not imply the message was received

It is theoretically impossible to be 100% certain that two computers agree on anything

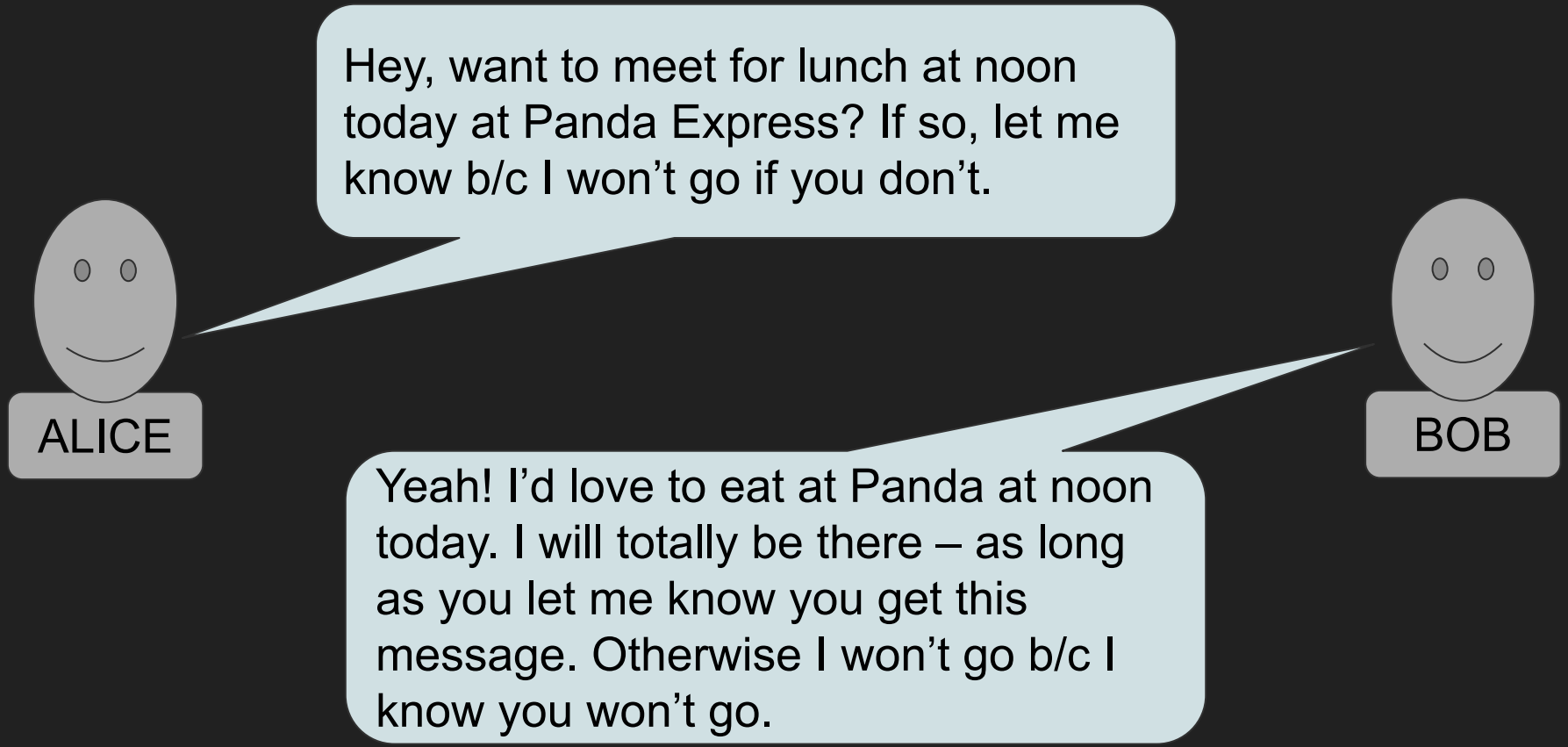
Dating is impossible - Consensus Impossibility

Alice wants to ask Bob out to lunch over text messages.

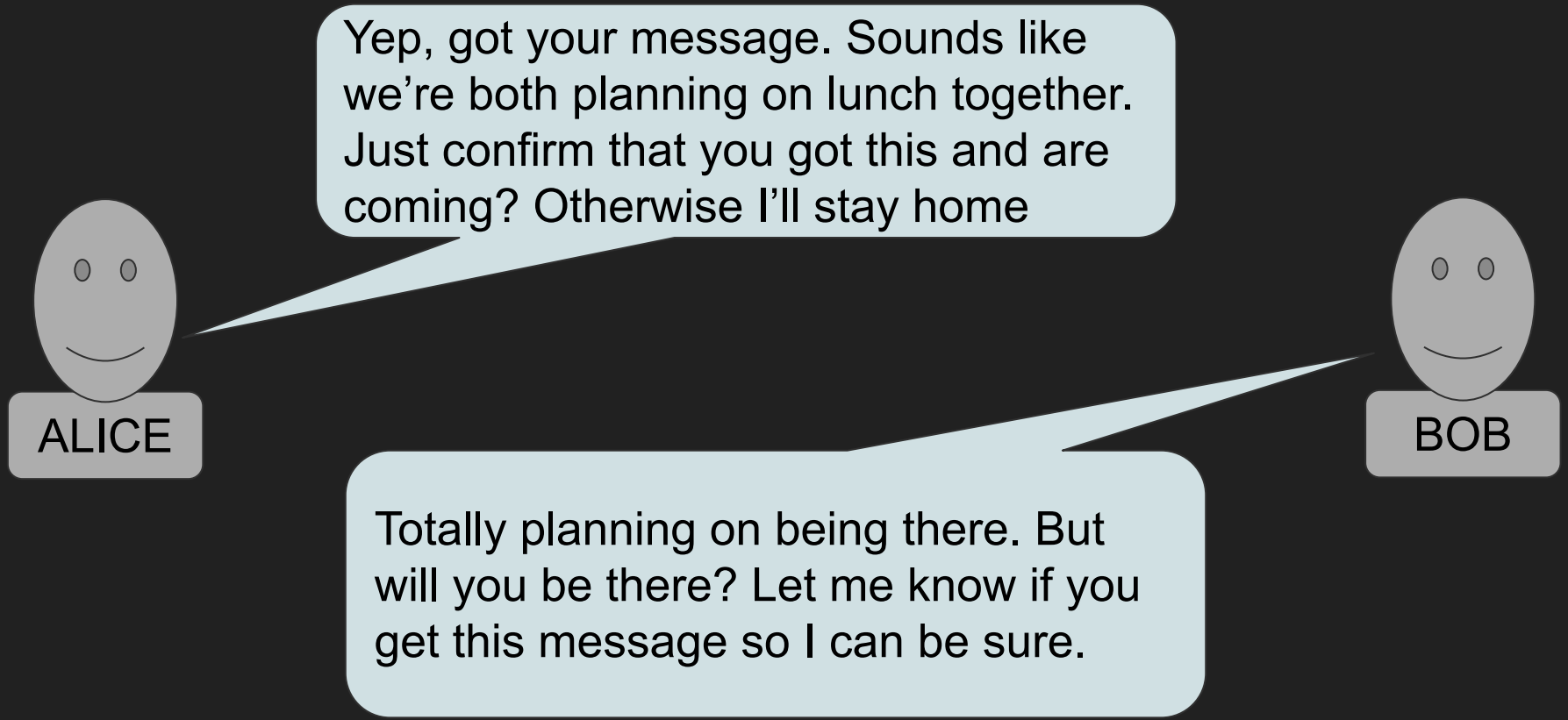
Neither Bob nor Alice is going to show up if they are not sure the other is coming – because that would be very awkward.

How can they ensure that they are both agreed on a time?

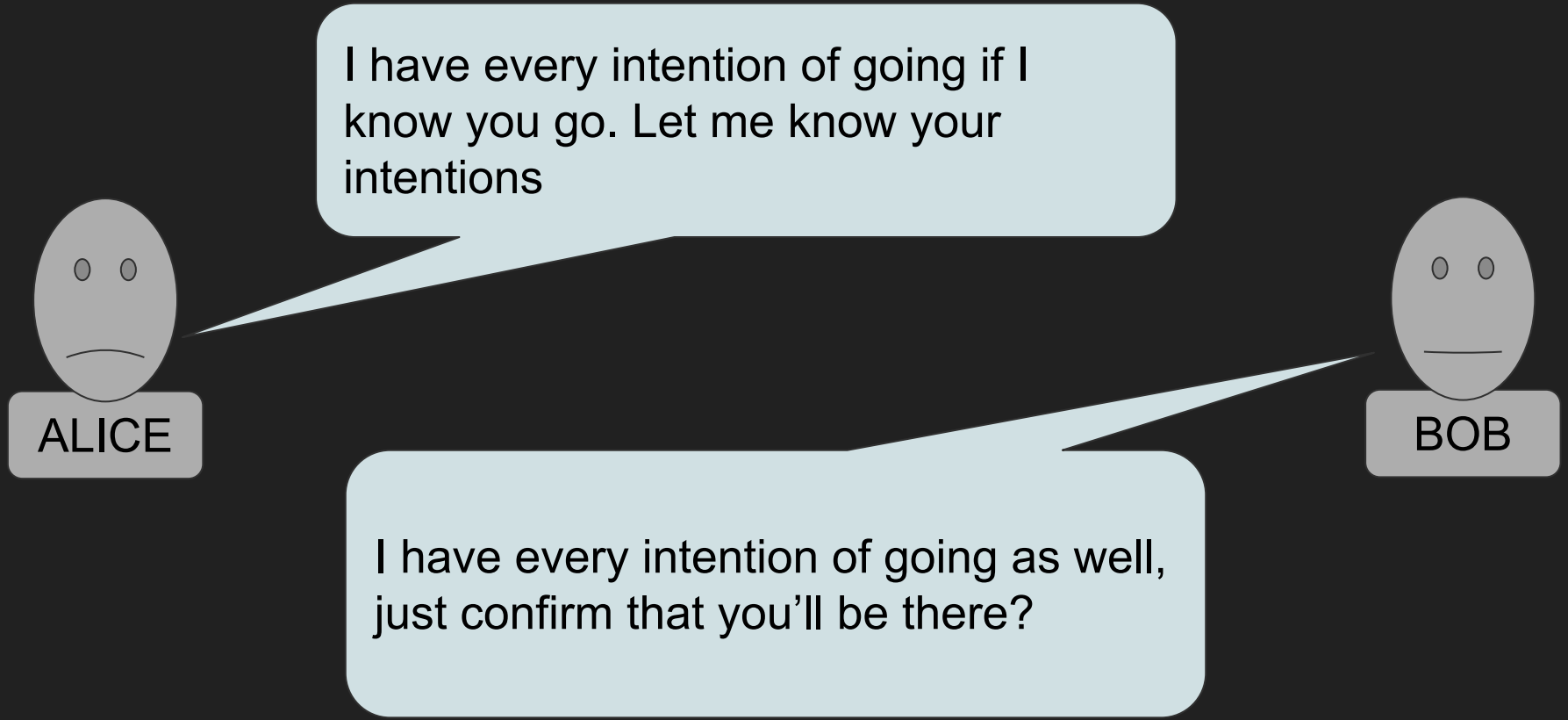
Dating is impossible - Consensus Impossibility



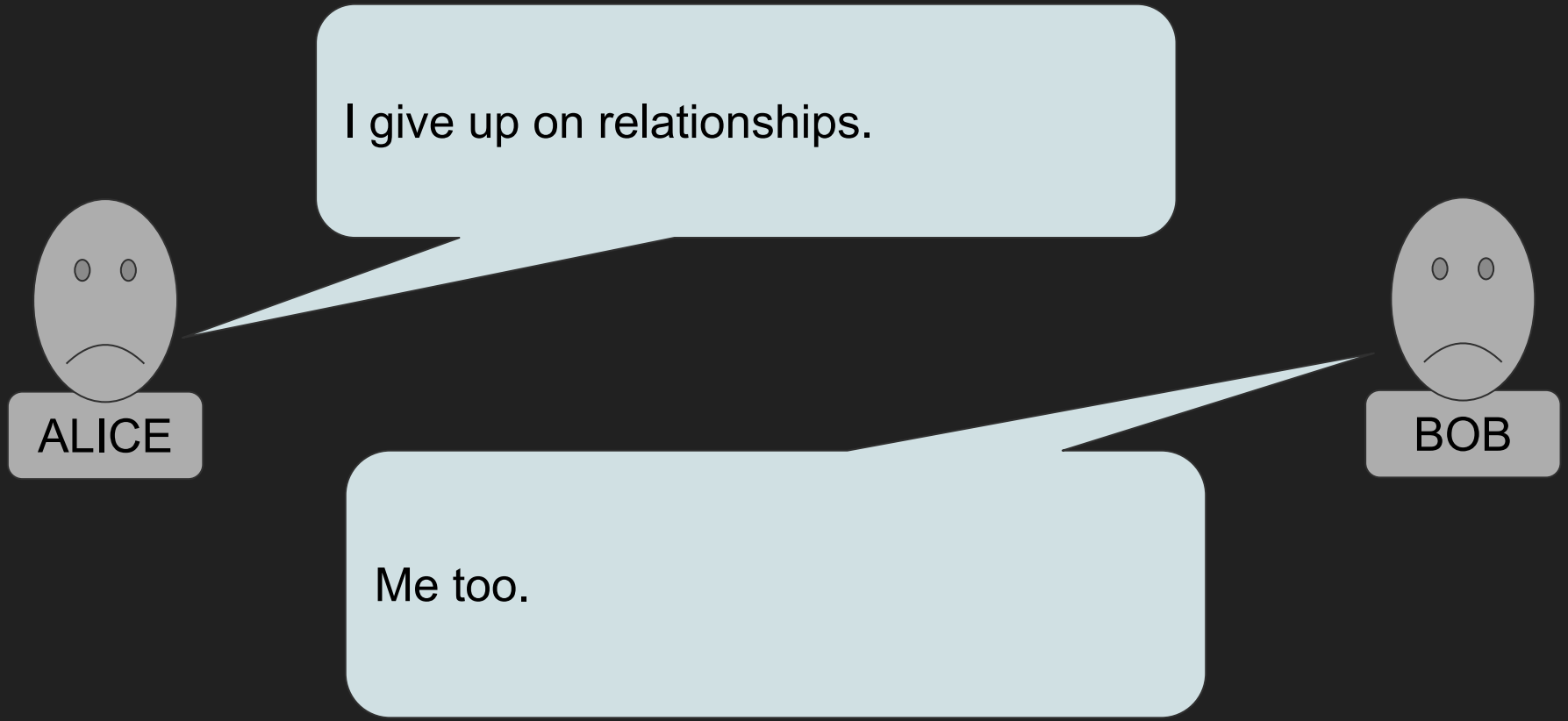
Dating is impossible - Consensus Impossibility



Dating is impossible - Consensus Impossibility



Dating is impossible - Consensus Impossibility

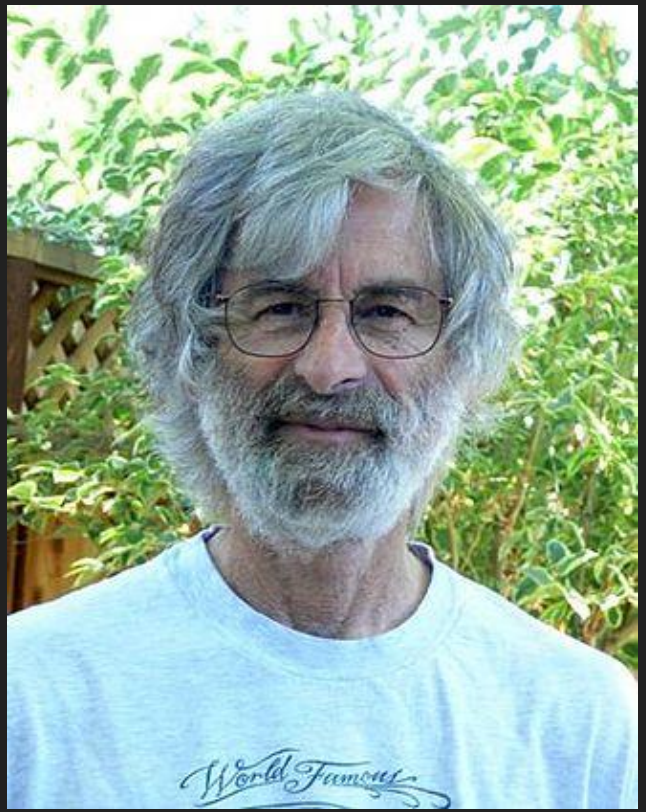


Leslie Lamport

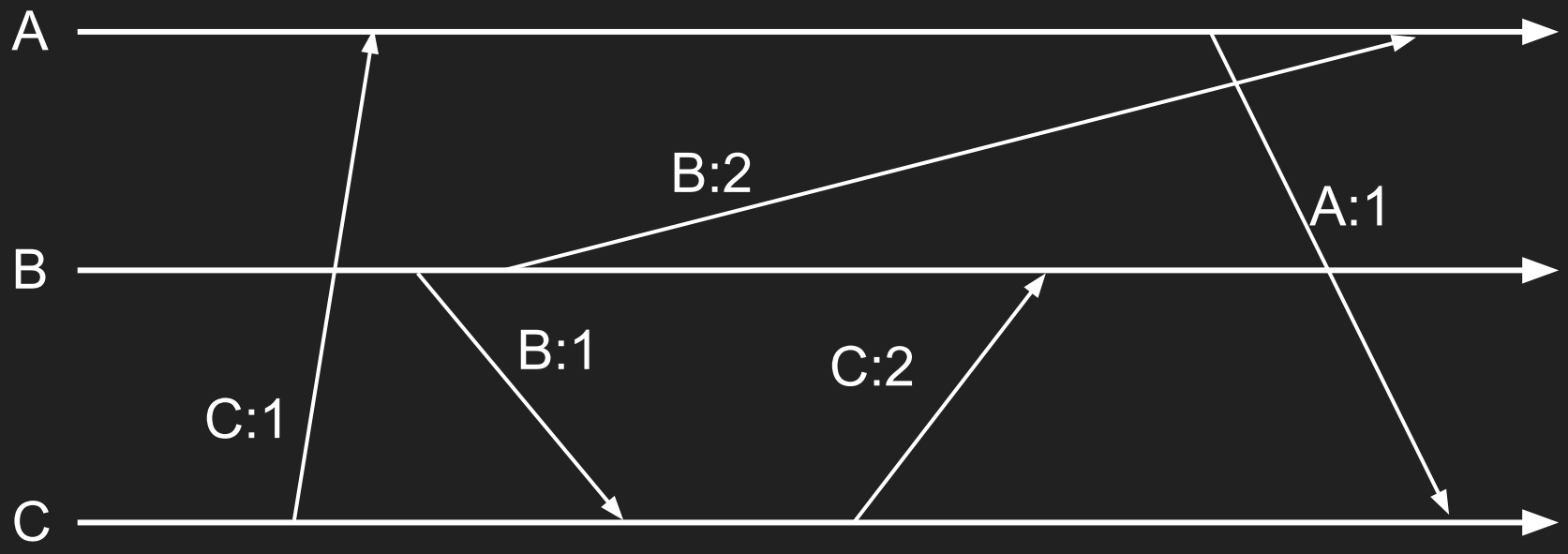
Inventor of theoretical methods to handle this impossible scenario.

-Lamport Clocks

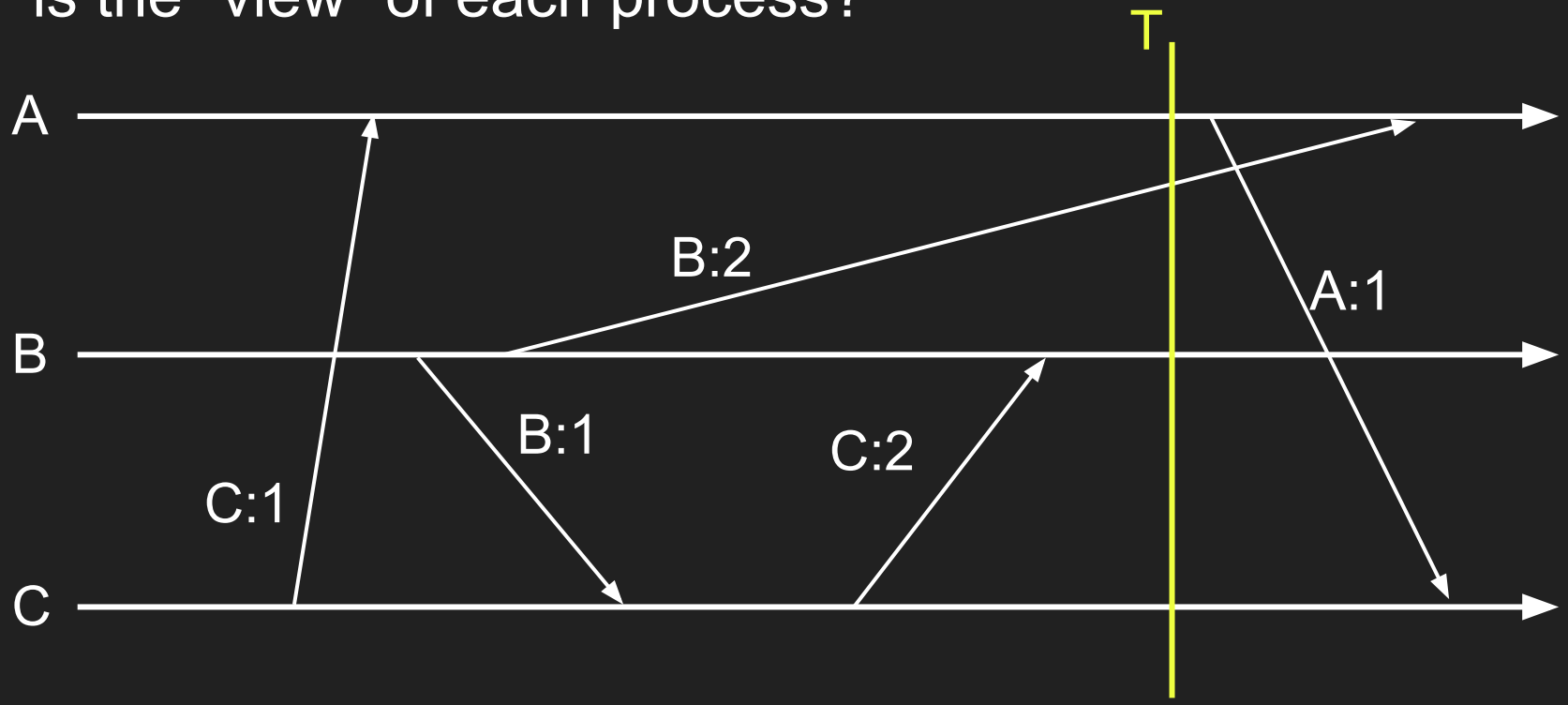
-Byzantine Fault Tolerance



Lamport Clocks:

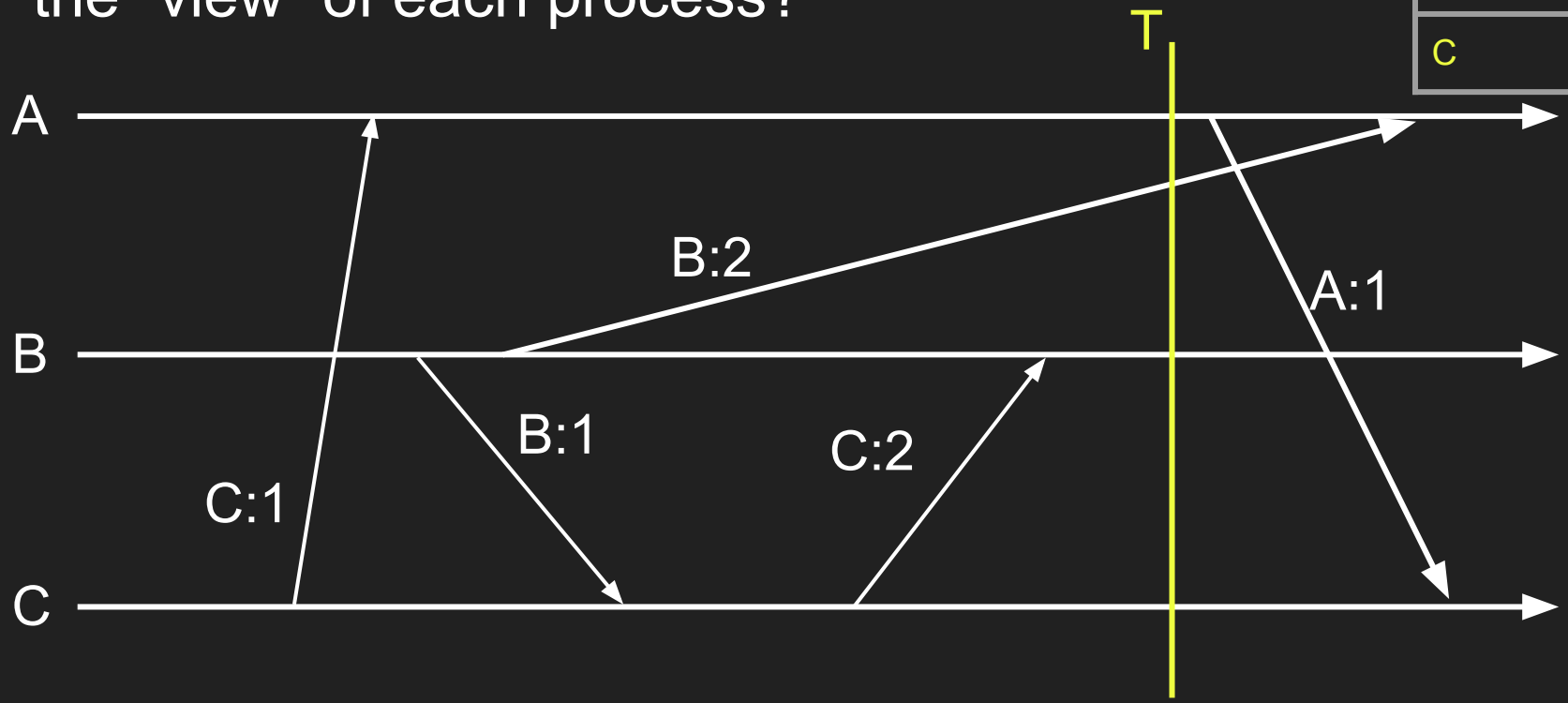


Lamport Clocks: At time “T”, what is the “view” of each process?



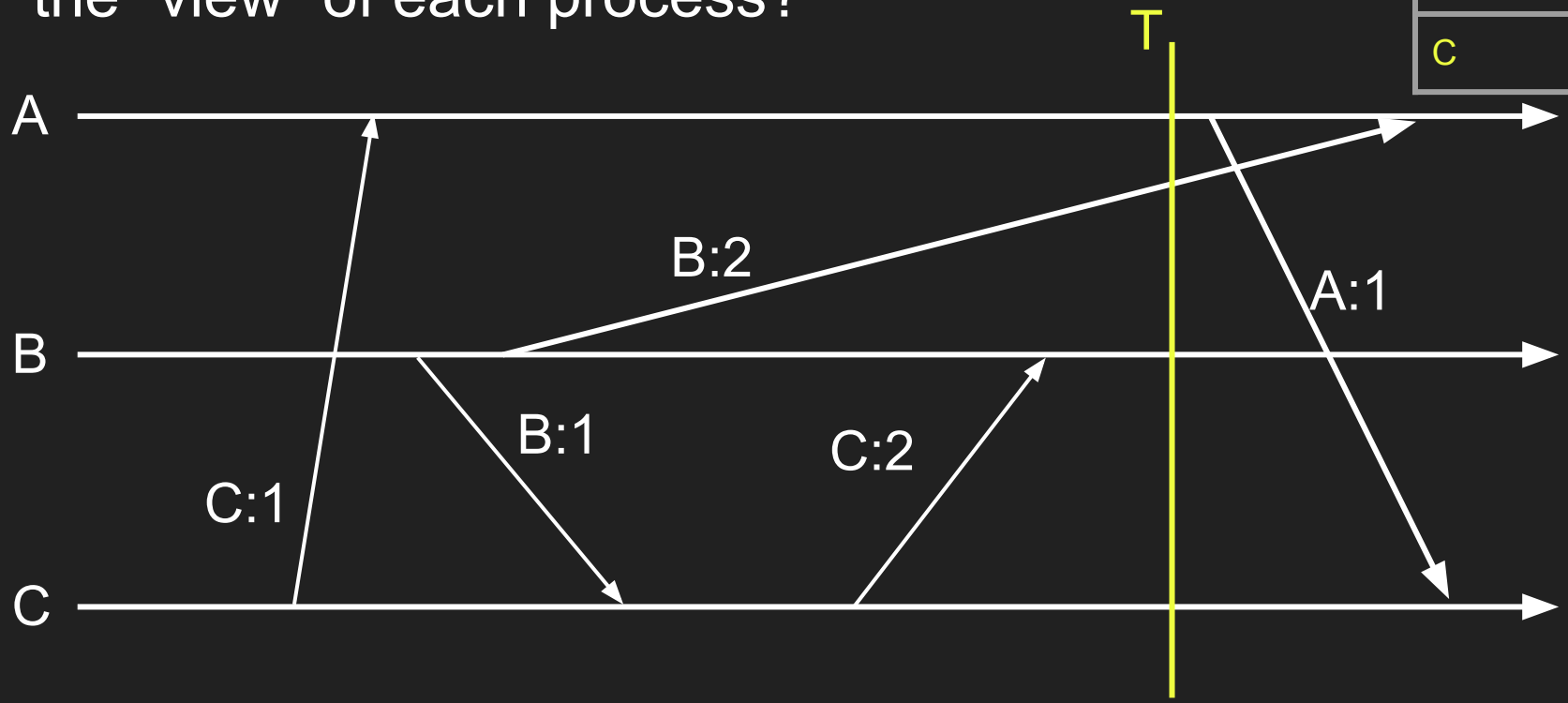
Lamport Clocks: At time “T”, what is the “view” of each process?

Process A	
A	0 events
B	0 events
C	1 event



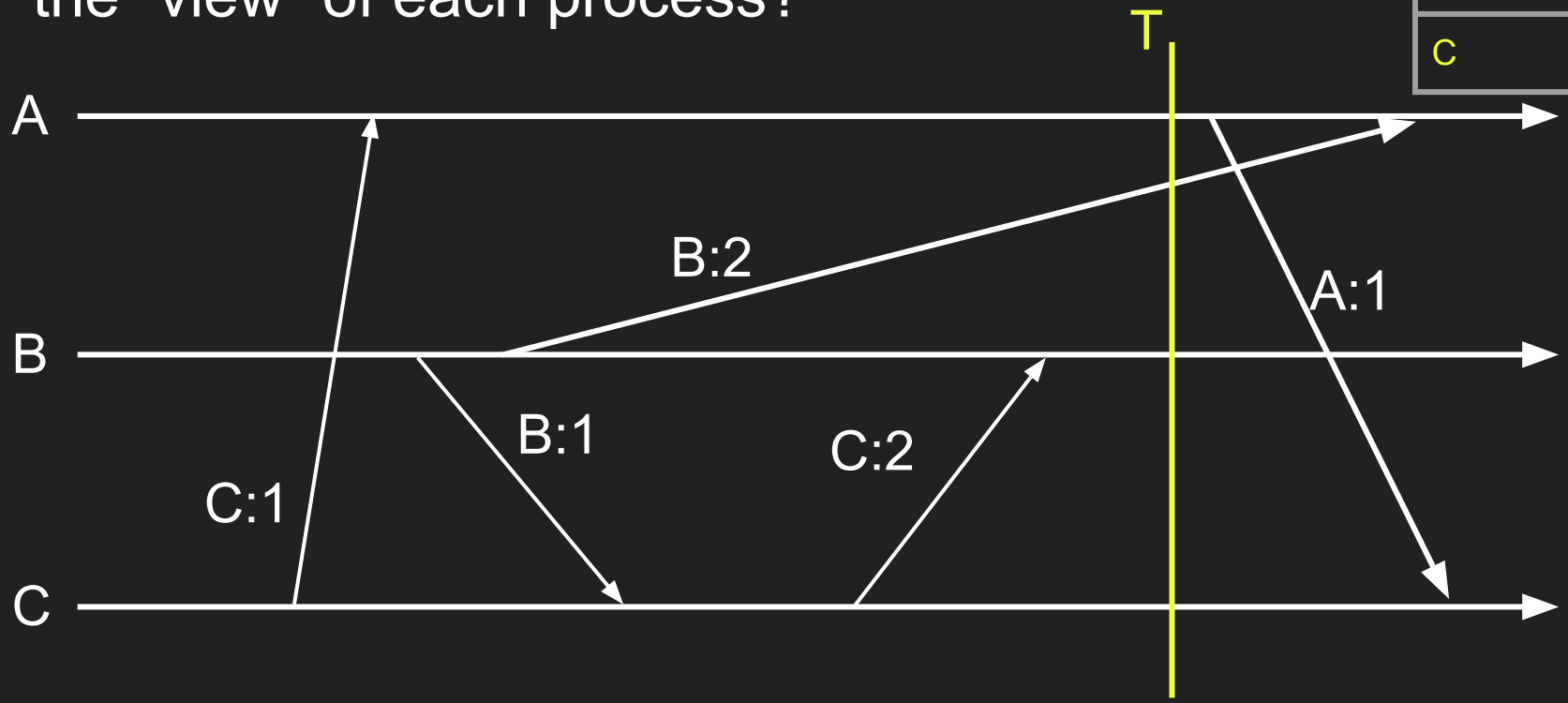
Lamport Clocks: At time “T”, what is the “view” of each process?

Process B	
A	0 events
B	2 events
C	2 events

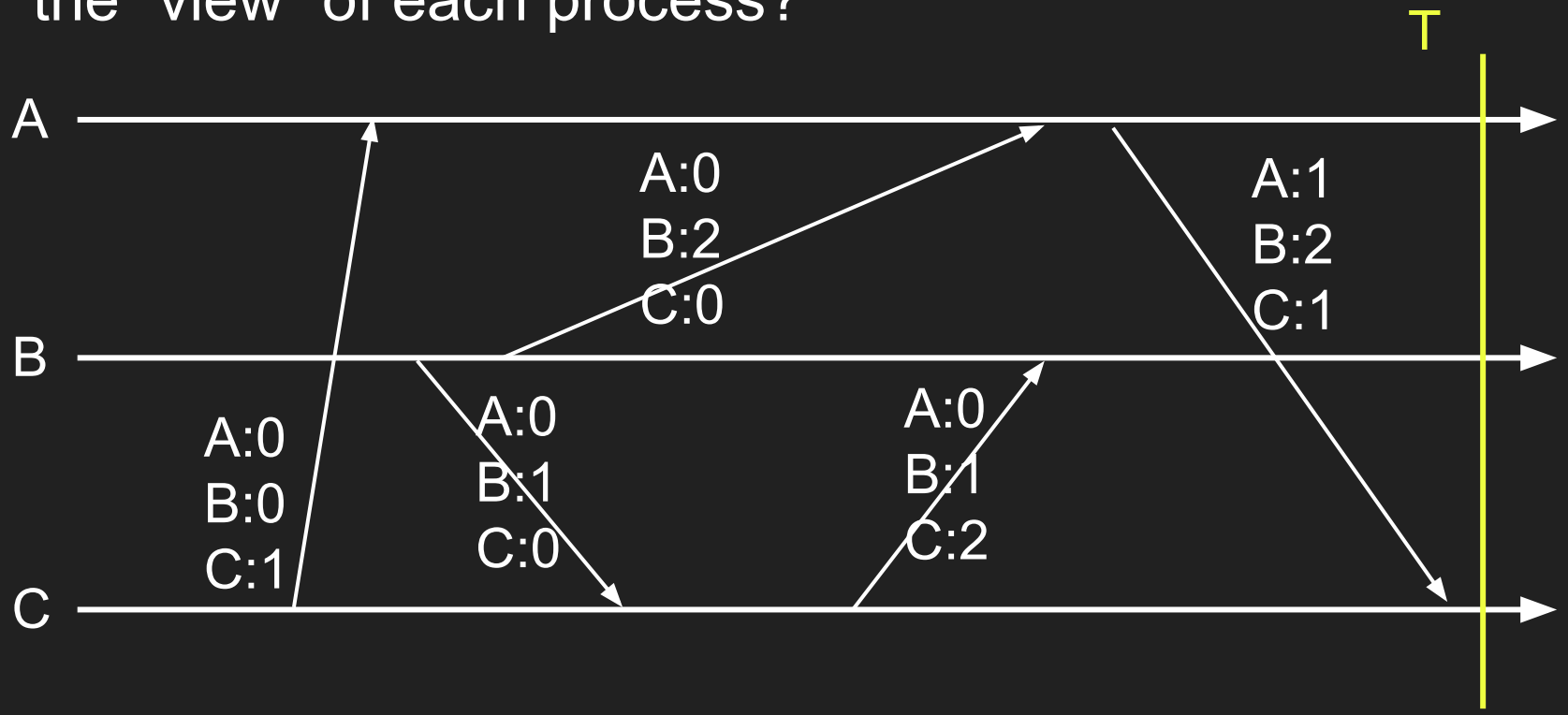


Lamport Clocks: At time “T”, what is the “view” of each process?

Process C	
A	0 events
B	1 event
C	2 events

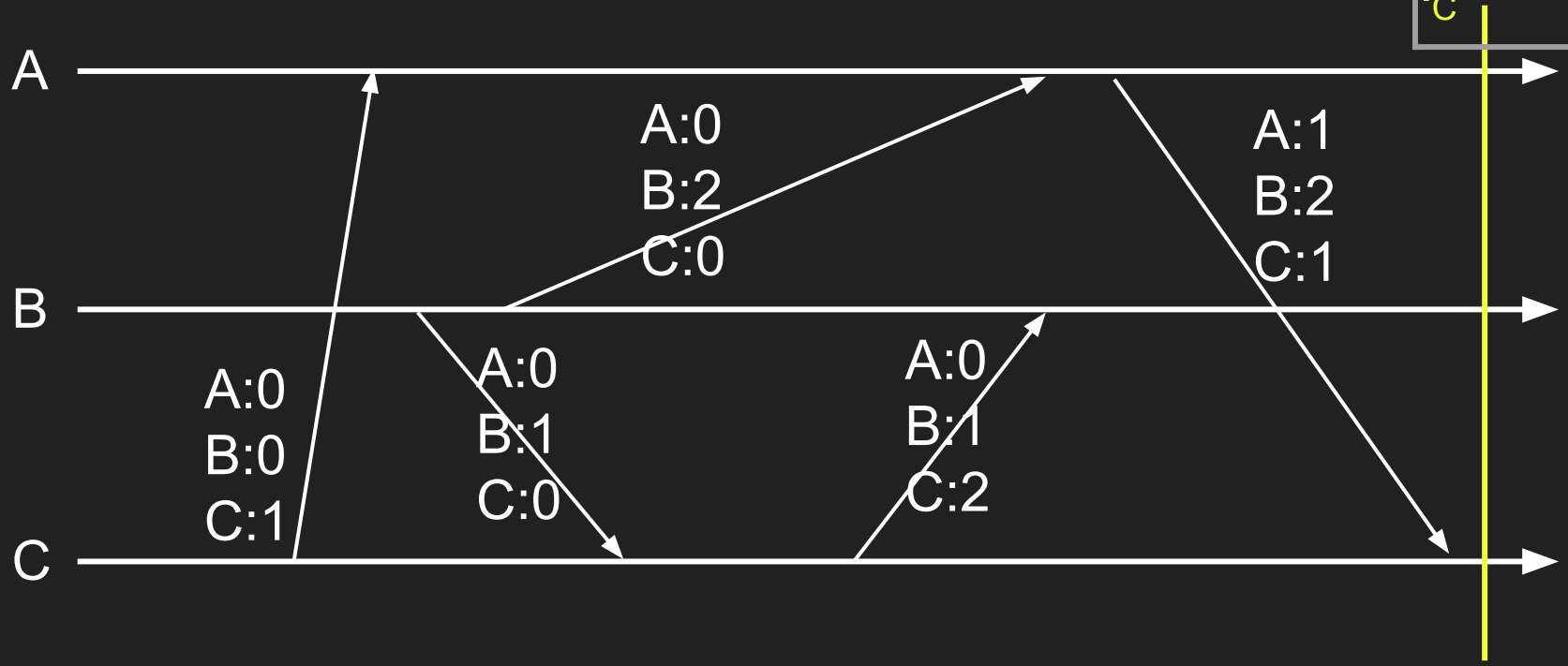


Vector Clocks: At time “T”, what is the “view” of each process?



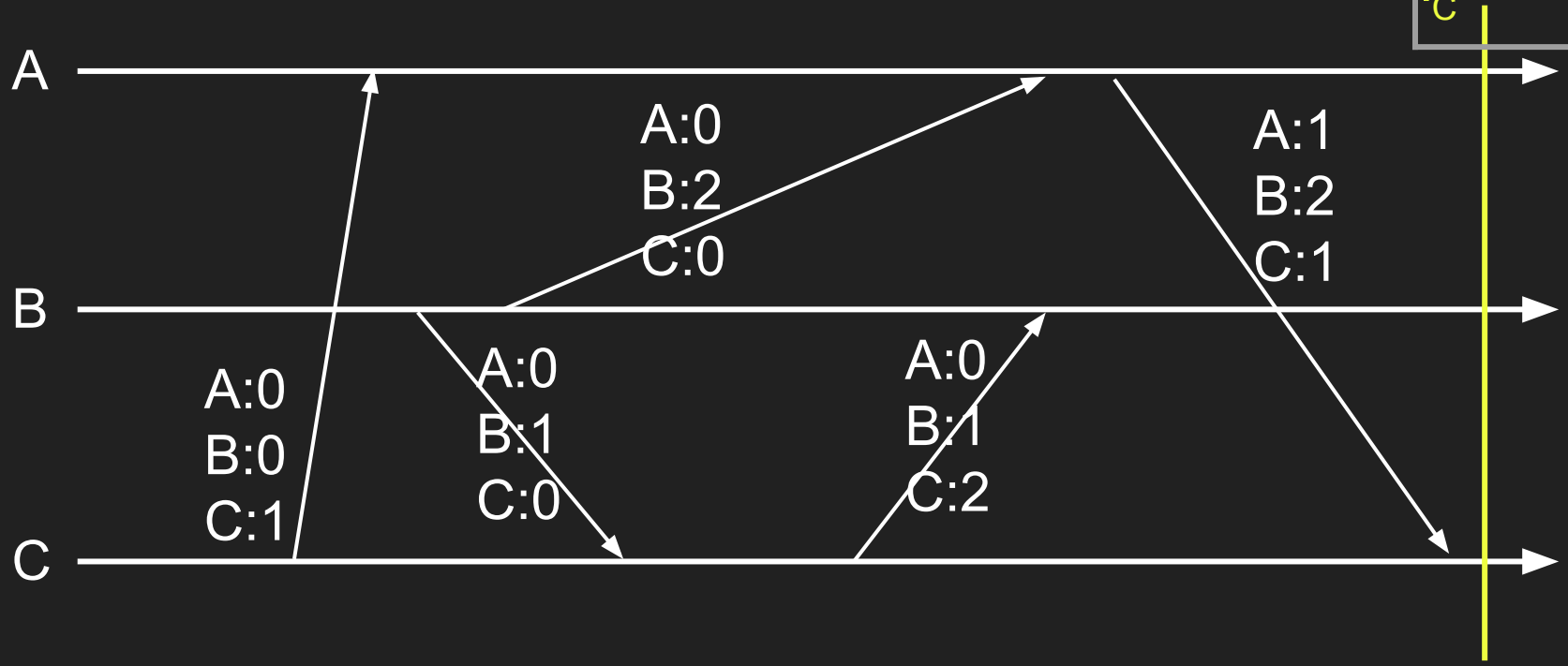
Vector Clocks: At time “T”, what is the “view” of each process?

Process A	
A	1 event
B	2 events
C	1 event



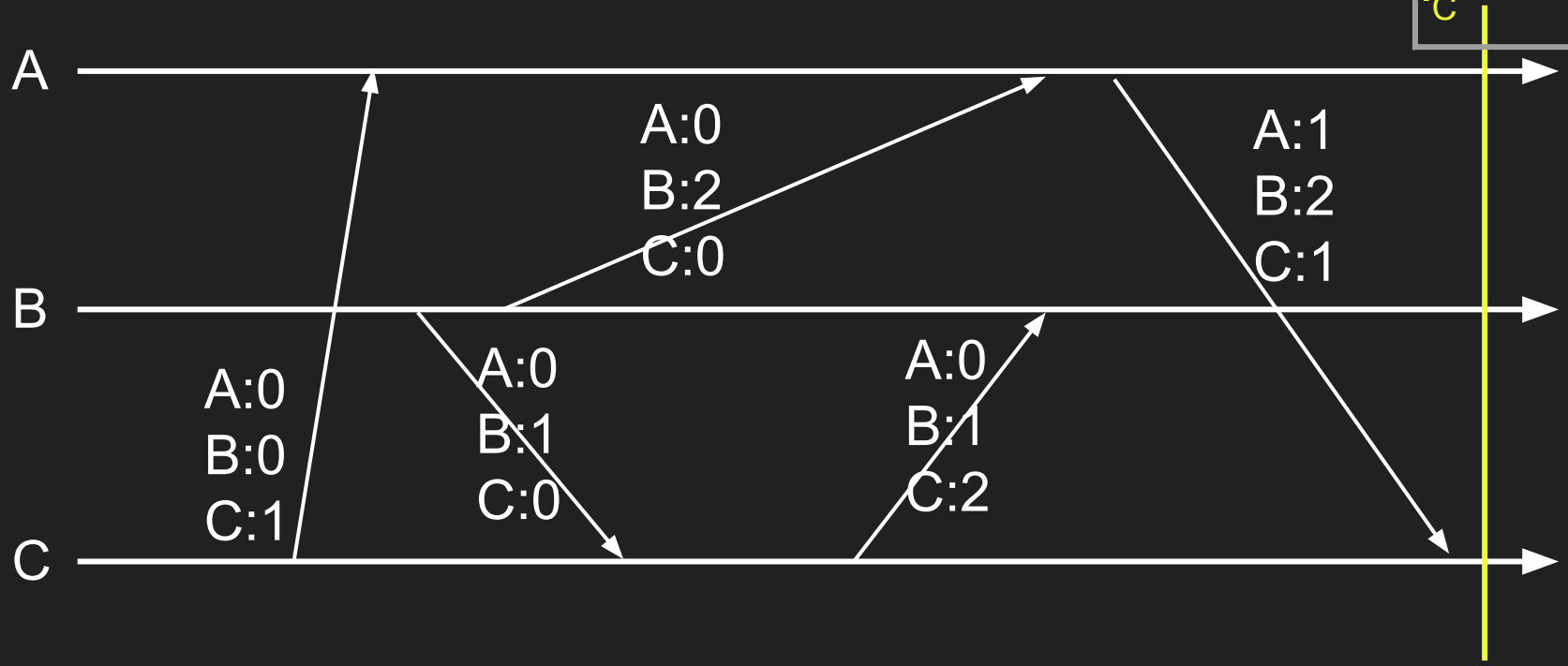
Vector Clocks: At time “T”, what is the “view” of each process?

Process B	
A	0 events
B	2 events
C	2 event

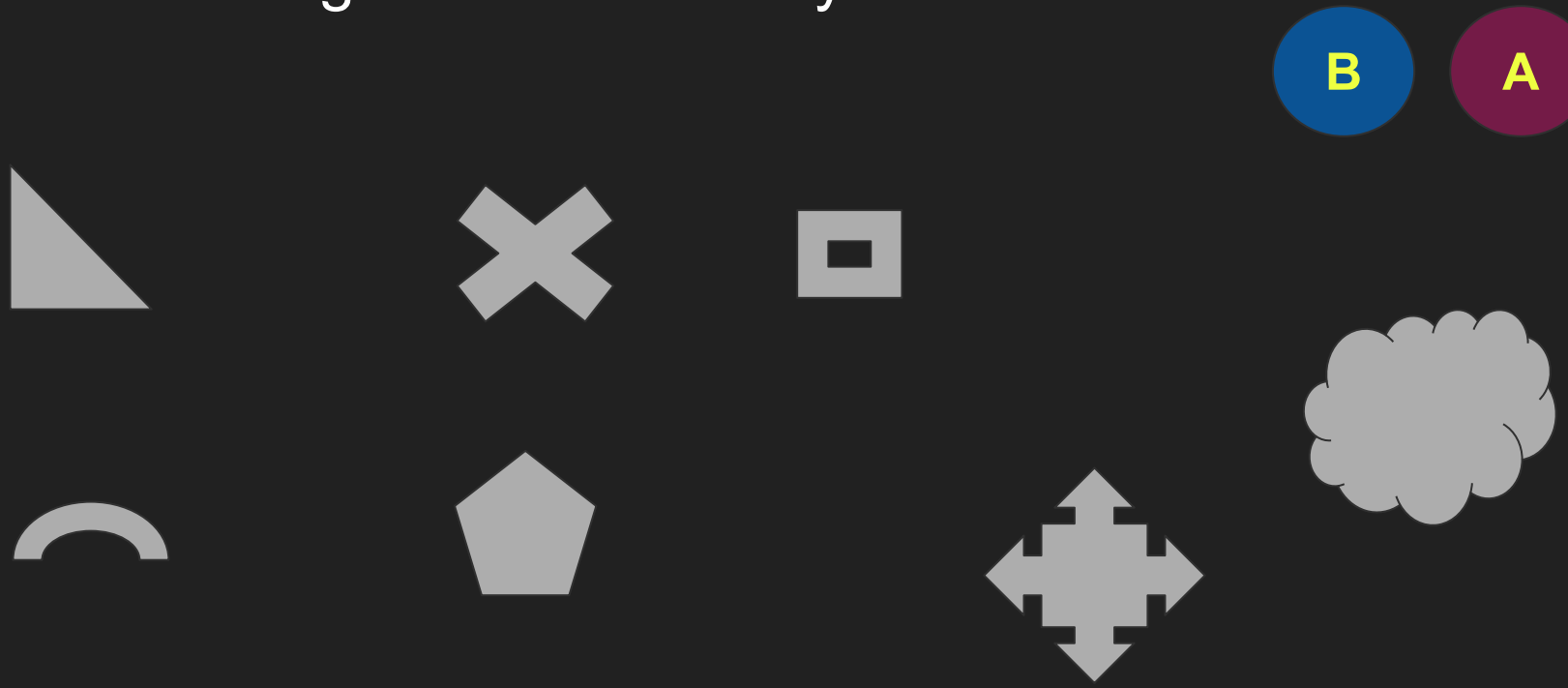


Vector Clocks: At time “T”, what is the “view” of each process?

Process C	
A	1 events
B	2 events
C	2 event



Decision Making in Distributed Systems: Fault Tolerance



Decision Making in Distributed Systems: Fault Tolerance

Fail-Stop Fault Tolerance

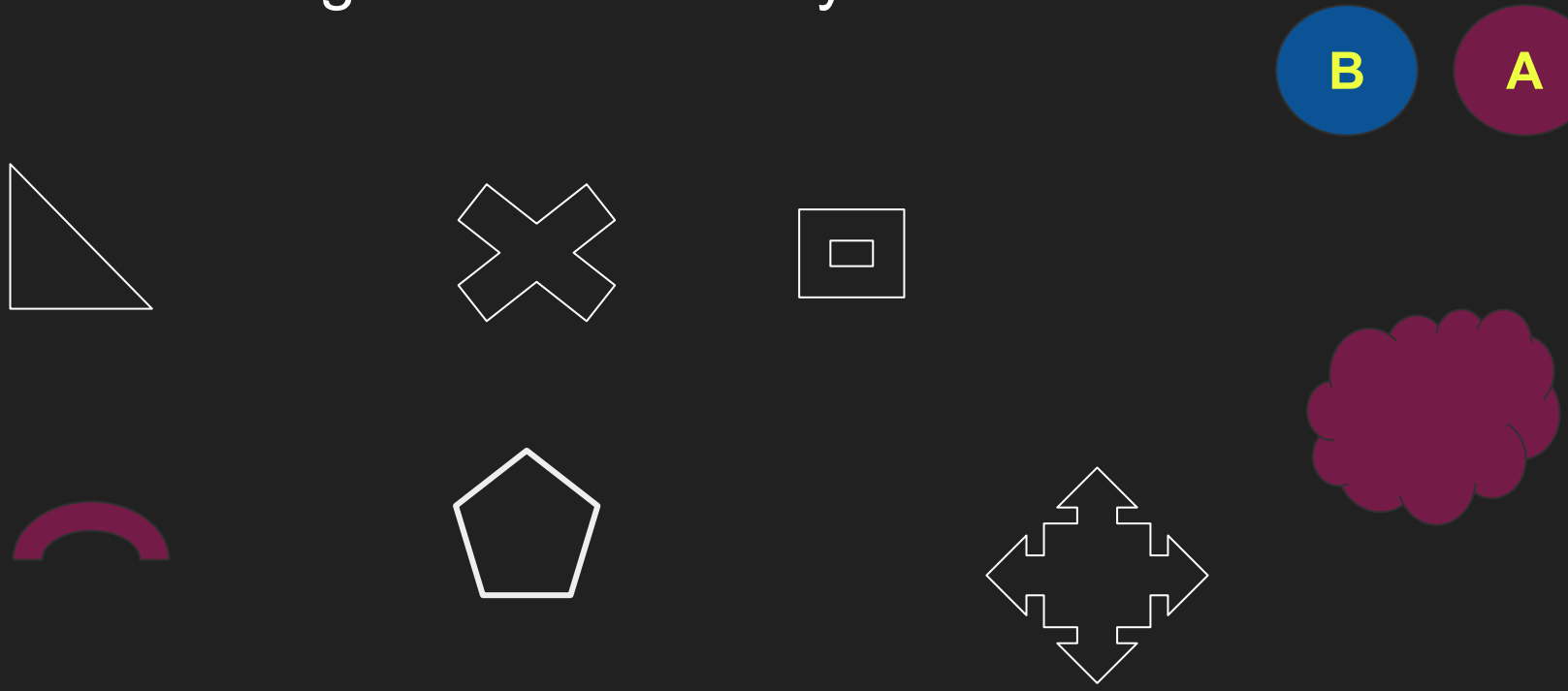
Some nodes will crash, and never send any decision – or at least not soon enough to be useful.

These failing nodes effectively fail and stop.

We can model a system of ‘**N**’ nodes and say it can tolerate up to ‘**f**’ failures

We assume all non-broken nodes produce the same answer.

Decision Making in Distributed Systems: Fault Tolerance



N = 7

F = ??? How many fail-stop failures can we tolerate?

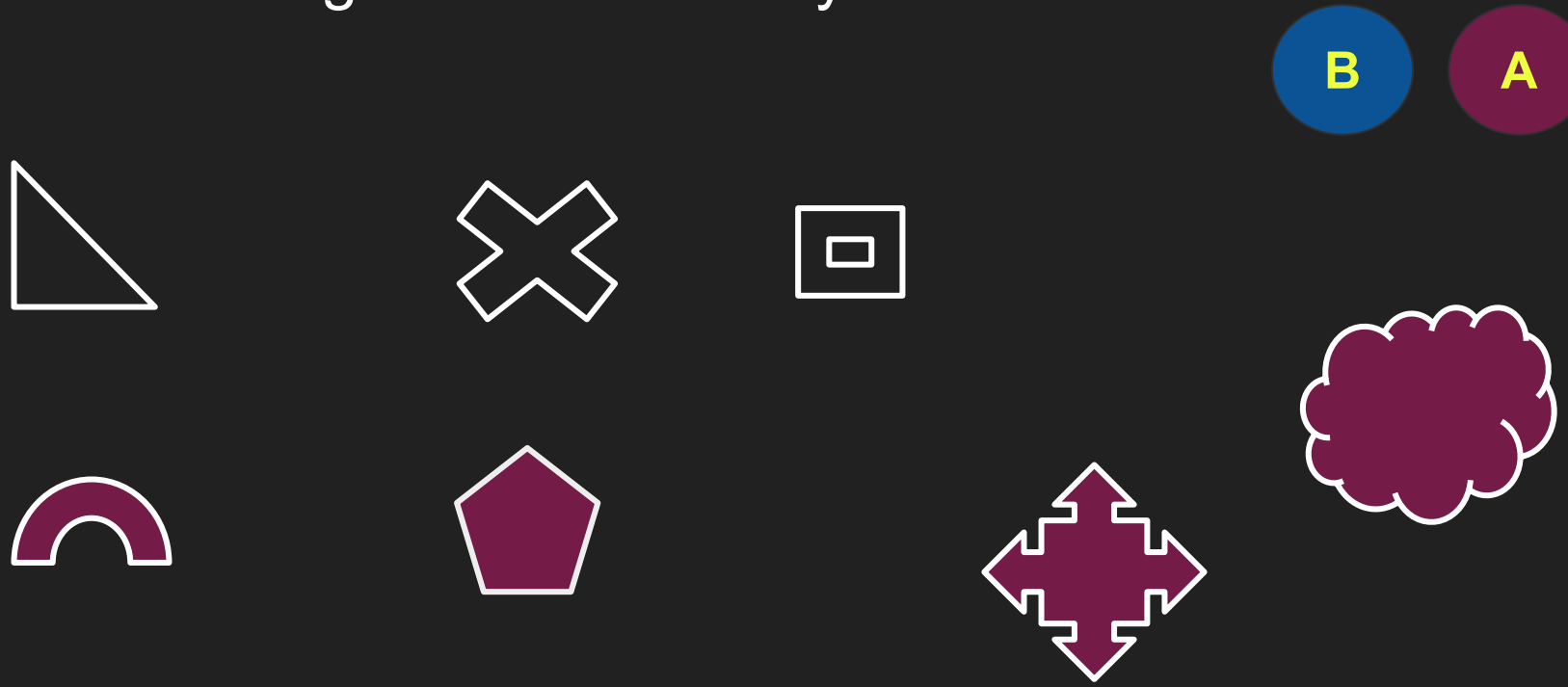
Solving Fail-Stop Fault Tolerance (N > 2f)

If we have f fail-stop failures, we will have (N - f) working nodes who will all agree.

If (N - f) = 0, then there is nothing we can do

If (N - f) > f, then we have a majority so we can move on even if f nodes never participate

Decision Making in Distributed Systems: Fault Tolerance



We can still get a majority and not halt forever as long as 4 nodes keep running

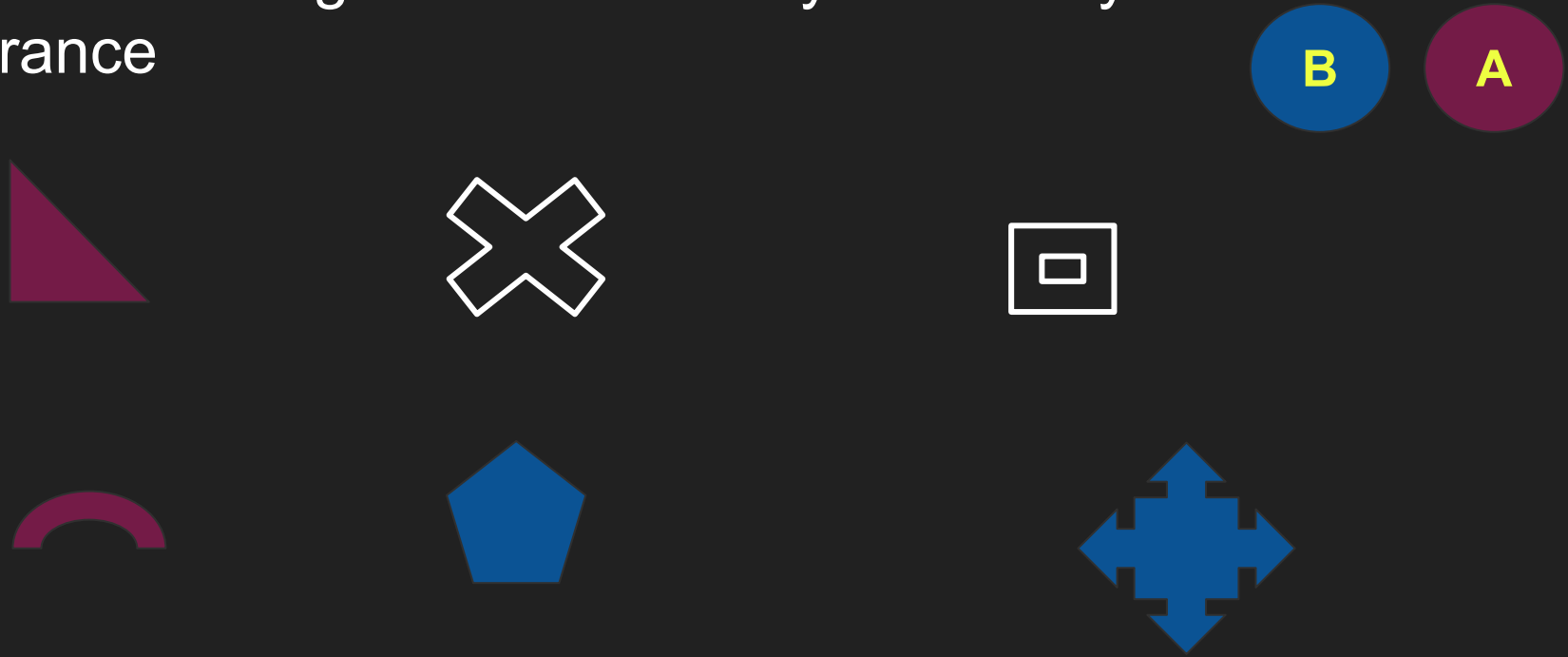
If N = 7, max(f) = 3

Byzantine Fault Tolerance

Evil nodes work together to try to get the wrong outcome, or block any outcome

Byzantine Generals Story

Decision Making in Distributed Systems: Byzantine Fault Tolerance



Byzantine nodes can lie, or stall on their answer
If 2 nodes are byzantine failures, this could turn out to choose A or B

Solving Byzantine Fault Tolerance (N must be > 3f)

Again, the system has N nodes, up to f of which are Byzantine faults.

Again (N - f) is the number of honest, working nodes

Once we hear from (N - f) nodes, we have to make a decision to avoid the Byzantine nodes stalling forever.

But, the honest nodes might also be all the slow ones, so among our group of (N - f) nodes we make a decision on, we still have to consider that up to f of them could be the Byzantine liars.

So, (N - 2f) is the number of honest nodes we can be sure of.

Until we hear from 2f + 1 nodes, the honest nodes could still come outvote the current favorite

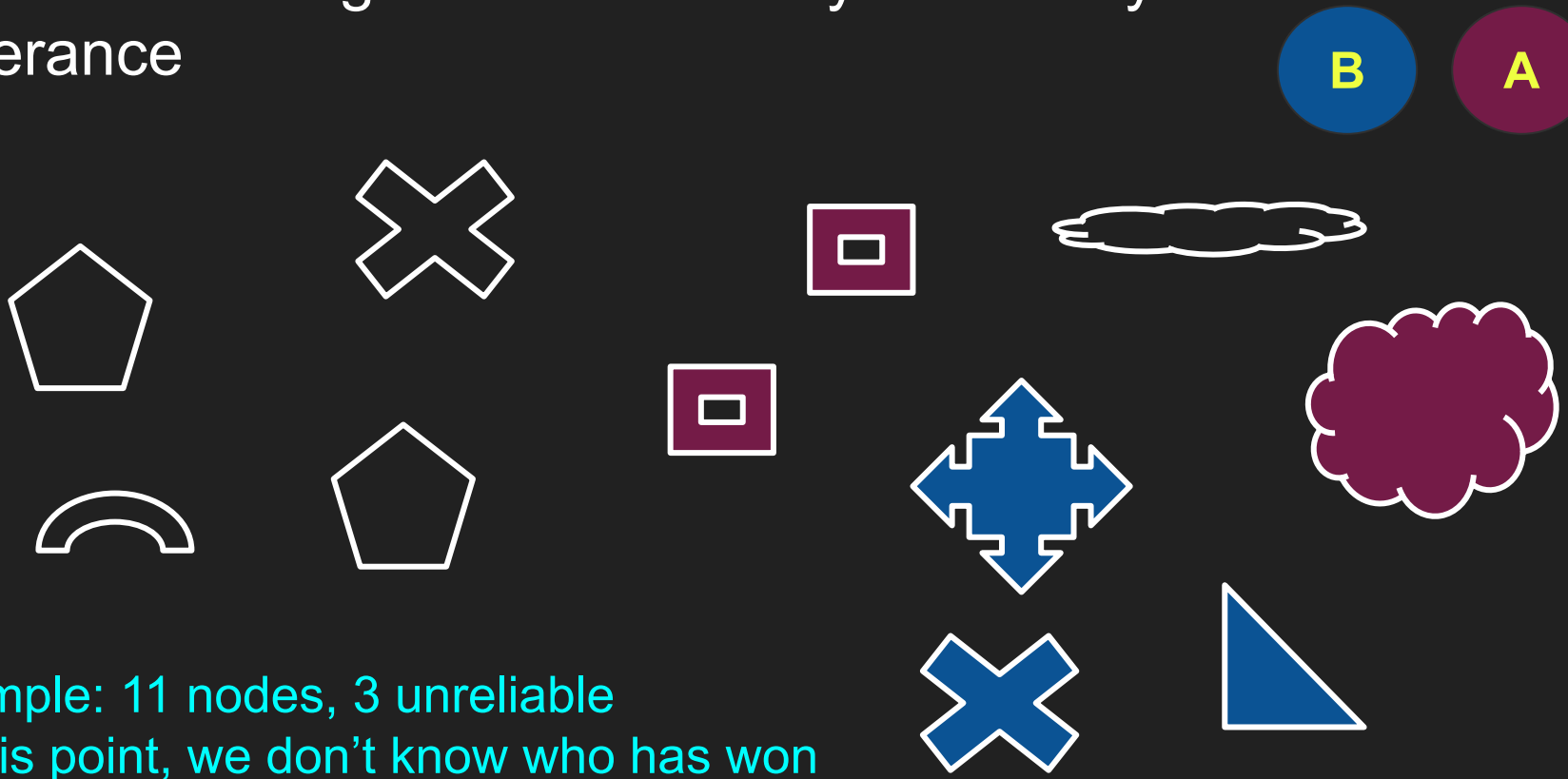
Decision Making in Distributed Systems: Byzantine Fault Tolerance

Example: 11 nodes, 3 unreliable

Decision Making in Distributed Systems: Byzantine Fault Tolerance

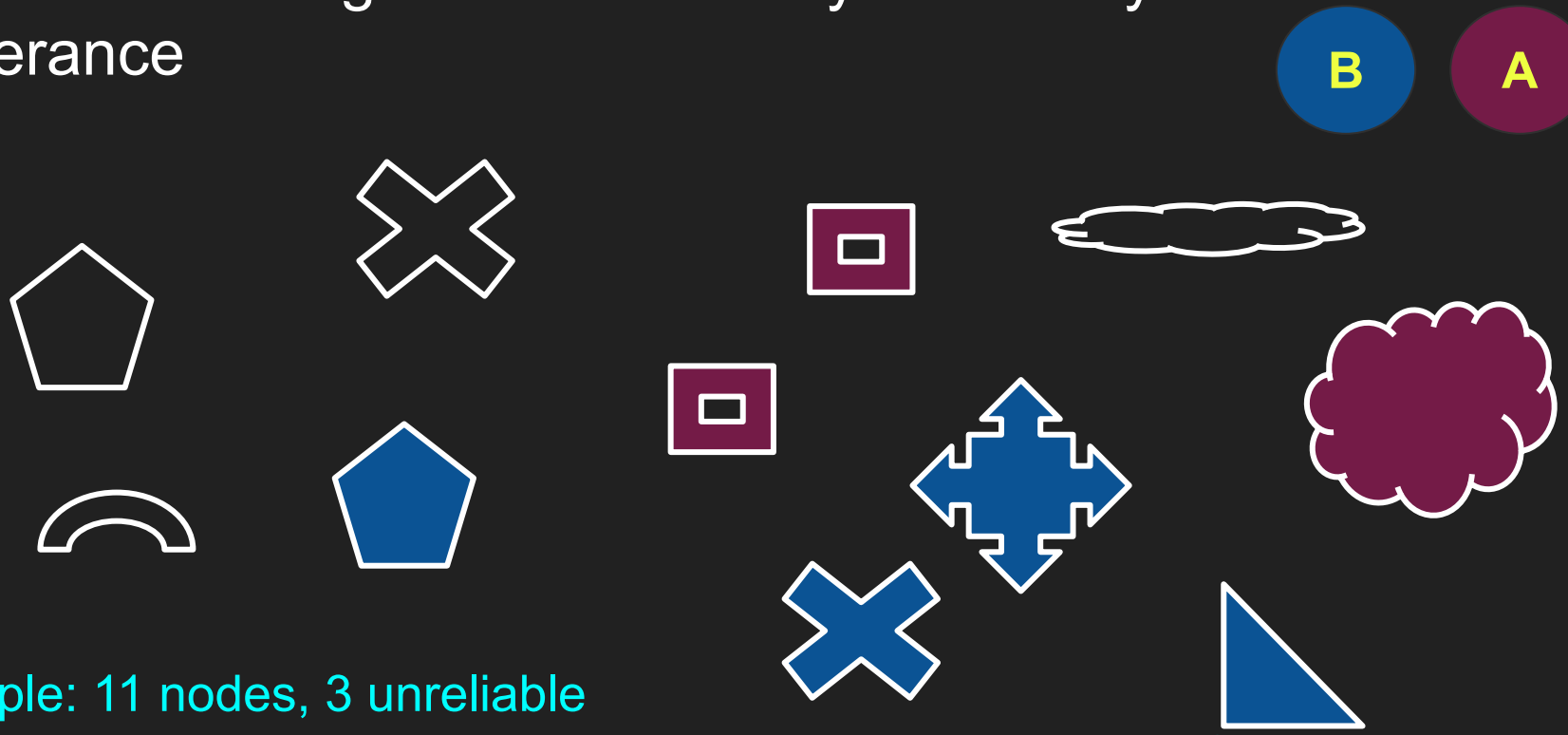
Example: 11 nodes, 3 unreliable
3 of these may be Byzantine and never reply,
So we have to choose by the time we hear from 8 nodes max (11 - 3)

Decision Making in Distributed Systems: Byzantine Fault Tolerance



Example: 11 nodes, 3 unreliable
At this point, we don't know who has won

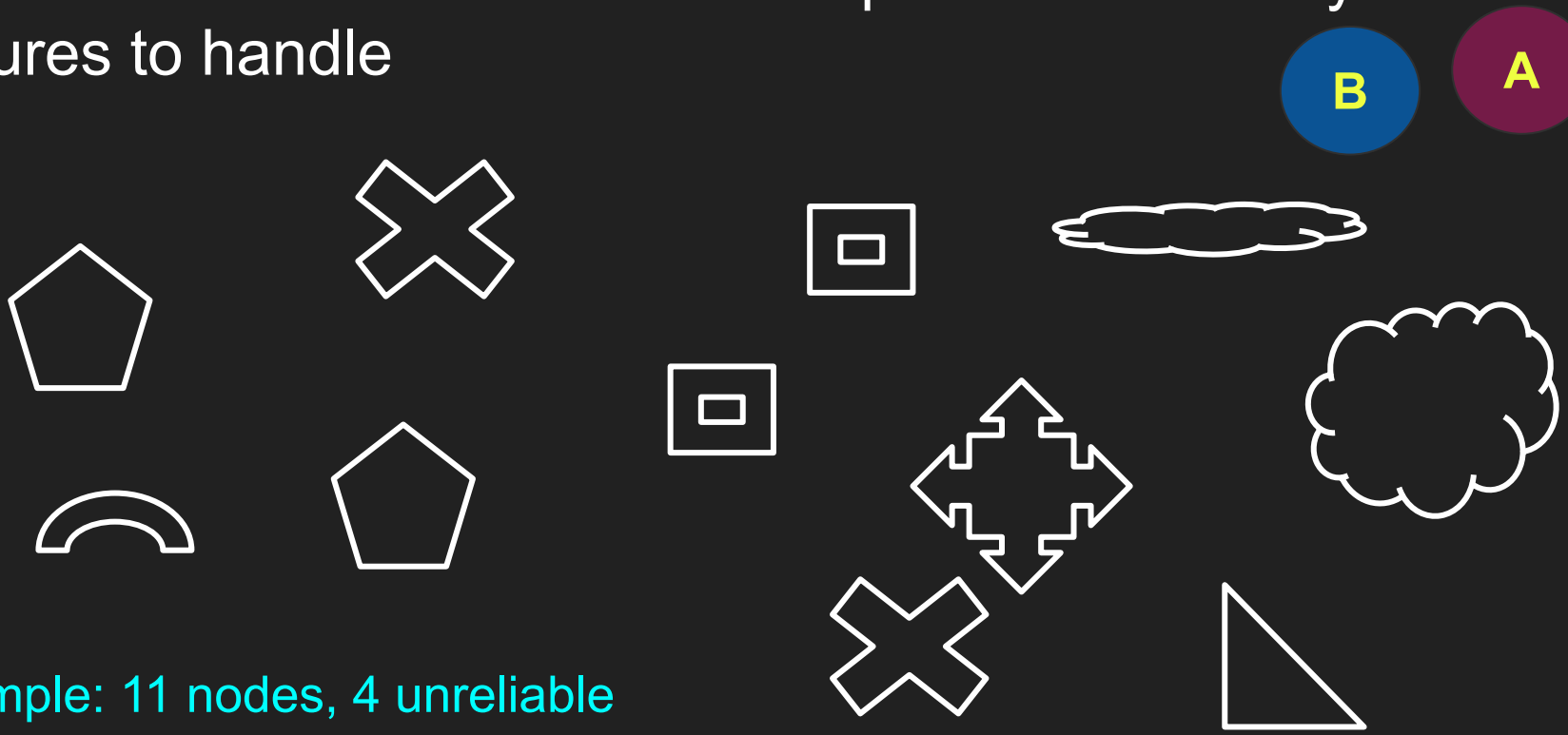
Decision Making in Distributed Systems: Byzantine Fault Tolerance



Example: 11 nodes, 3 unreliable

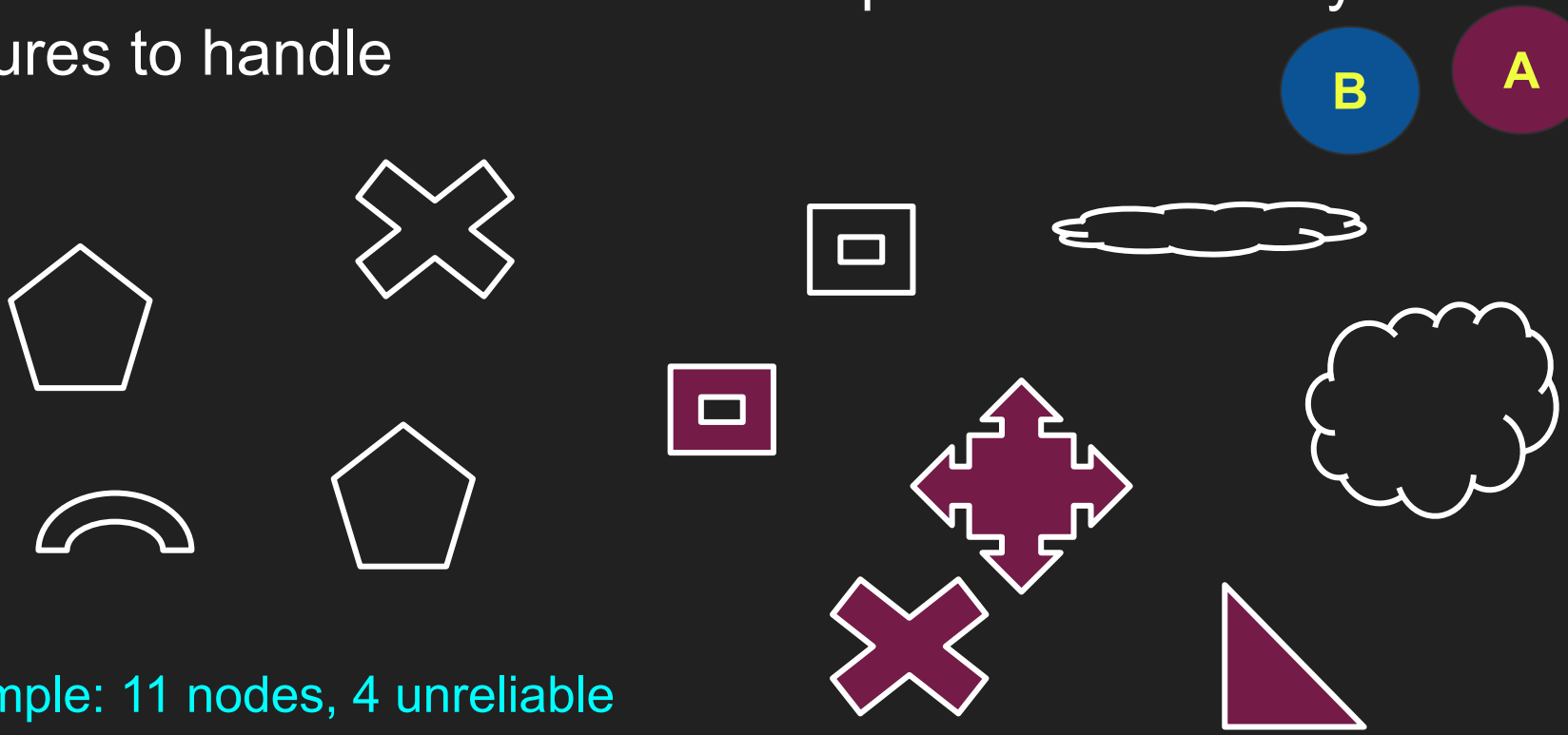
At this point, we know blue has won, because at least one of those blues is honest. If we wait long enough, the rest of the honest nodes must come in blue

Byzantine Fault Tolerance - Example with too many failures to handle



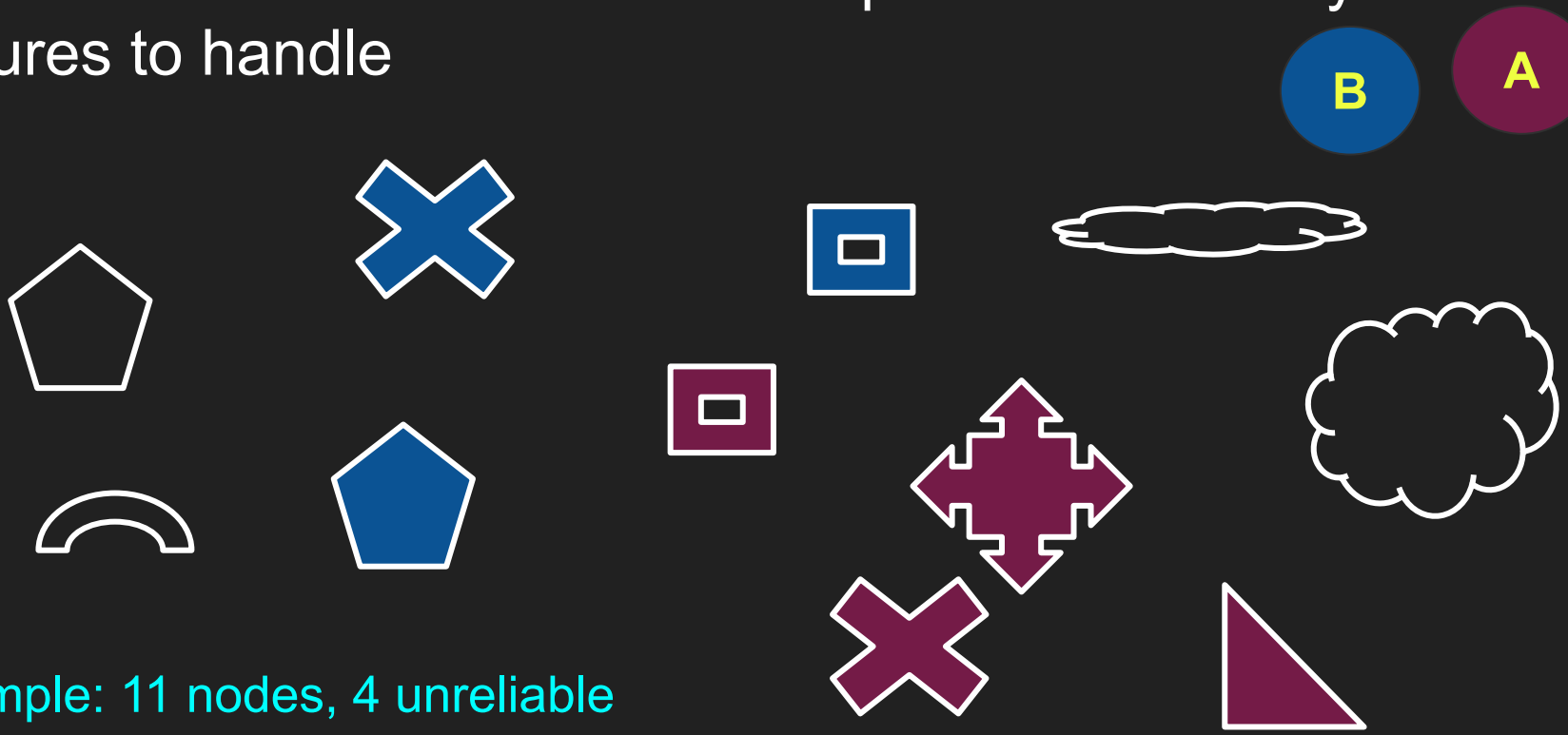
Example: 11 nodes, 4 unreliable
We have to assume we must make a decision by the time we hear from 7 nodes (11 - 4)

Byzantine Fault Tolerance - Example with too many failures to handle



Example: 11 nodes, 4 unreliable
We still don’t know the true answer, because all 4 of these nodes could be the impostor byzantine nodes

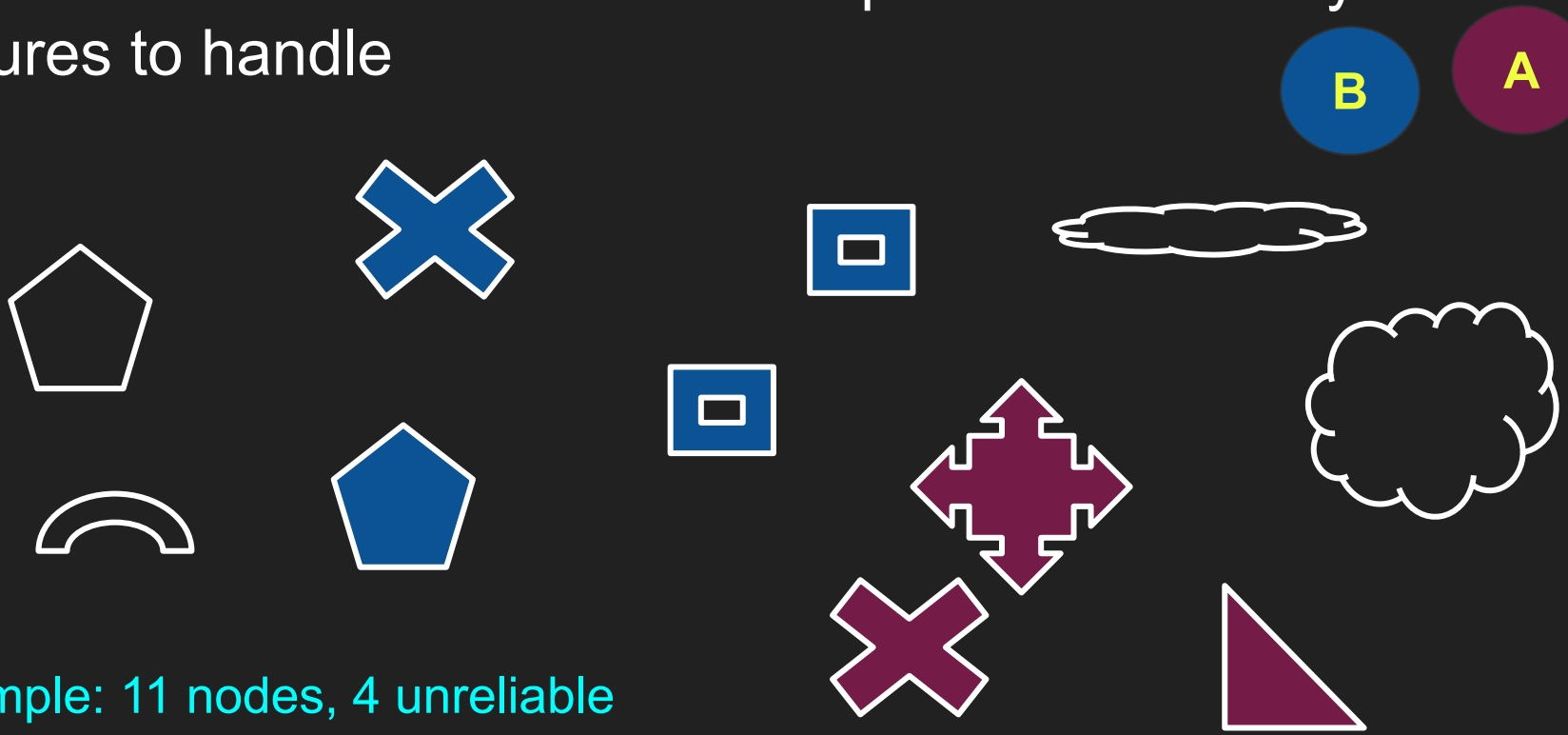
Byzantine Fault Tolerance - Example with too many failures to handle



Example: 11 nodes, 4 unreliable

Now, we have seen 7 nodes. We have to decide now, because otherwise the 4 unresponsive nodes might stall us forever

Byzantine Fault Tolerance - Example with too many failures to handle



Example: 11 nodes, 4 unreliable

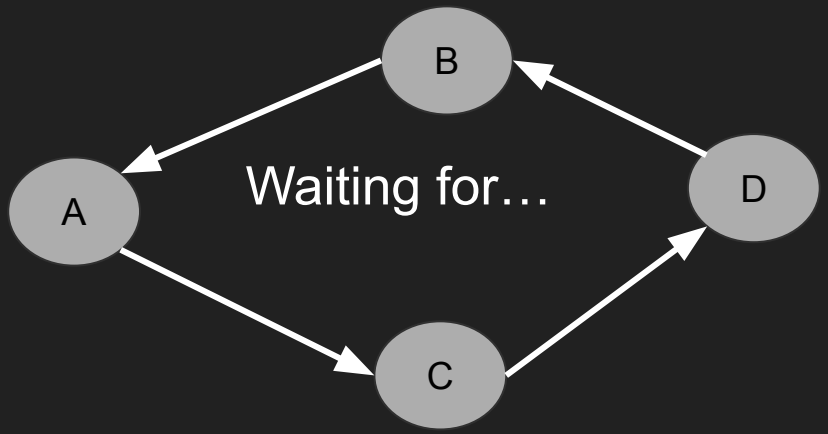
Imagine 1 of the nodes told different votes to some of the other nodes!

Practice Questions

In a system of 34 nodes, can it be Fail-Stop tolerant of 17 nodes?

Deadlock Prevention Options:

1. Don’t use any mutex locks
2. Never wait for a mutex while holding another mutex lock
3. Allow processes to forcibly acquire mutex locks from other threads
4. Set up the threads so that there is never a “circular wait”



Product Evaluation Activity

https://www.synergisticresearch.com/accessories/ethernet-switch-uef/

What network layer does this product operate at?

What does this product claim to do?

Can this product do what it claims to do?

What improvement at this network layer could affect an application like audio streaming?