MANY FACES OF CONSENSUS
OVERVIEW

▸ What is consensus?

▸ Consensus in crash failure model
  ▸ Scenarios
  ▸ What is impossible?
  ▸ What is possible?

▸ Consensus in byzantine model
  ▸ What is impossible?
FORMALISING CONSENSUS

- Given a set of parties with their inputs
- They all must agree on a decision based on their inputs
- Binary consensus
  - 0 or 1

Can be extended to any number of decisions
FORMALISING CONSENSUS

- Given a set of parties with their inputs
- They all must agree on a decision based on their inputs
- Binary consensus
  - 0 or 1

Consistency

All parties agree on same value

Validity

Agreed-upon value is some party’s input

Termination

Each party decided in finite number of steps
HESITATE AND YOU'RE LOST

Crash Failures
INTUITION DEVELOPMENT

CASE I

▸ Previous communication is allowed
▸ Unbounded but reliable communication
▸ No one dies

To be or not to be: Pizza or Pasta
INTUITION DEVELOPMENT

CASE I

- Previous communication is allowed
- Unbounded but reliable communication
- No one dies

To be or not to be: Pizza or Pasta

YES
To be or not to be: Pizza or Pasta

- Previous communication is allowed
- Unbounded but unreliable communication
- No one dies
### INTUITION DEVELOPMENT

#### CASE II

- Previous communication is allowed
- Unbounded but **unreliable** communication
- No one dies

---

**To be or not to be : Pizza or Pasta**
INTUITION DEVELOPMENT

CASE II

- Previous communication is allowed
- Unbounded but unreliable communication
- No one dies

To be or not to be: Pizza or Pasta
INTUITION DEVELOPMENT

CASE II

- Previous communication is allowed
- Unbounded but unreliable communication
- No one dies

To be or not to be: Pizza or Pasta
Case II

- Previous communication is allowed
- Unbounded but unreliable communication
- No one dies

Let me ACK the ACK!

I SHOULD ACK the ACK of the ACK

QUACK ... QUACK ... QUACK ... QUACK ... QUACK ... QUACK ... QUACK

Maybe not
To be or not to be: Pizza or Pasta

- Previous communication is allowed
- Unbounded but reliable communication
- One can die
- Others cannot differentiate between whether the crown is dead or the network is slow
CASE IV

- Previous communication is allowed
- **Bounded** but reliable communication
- One can die

To be or not to be: Pizza or Pasta

Use timeouts
CONSENSUS DEPENDS HEAVILY ON SYSTEM MODEL
CONSENSUS* IS IMPOSSIBLE IN A COMPLETELY ASYNCHRONOUS MESSAGE PASSING SYSTEMS WHERE EVEN A SINGLE PROCESSOR FAILS

[Fischer, Lynch, Paterson '85]
MODELS OF COMMUNICATION

MESSAGE PASSING

Hi

Hello

SHARED MEMORY

Hello

Hello

Registers

OUR PROOF

FLP
- Proof in asynchronous shared-memory
- More powerful than message-passing
  - Inherent broadcasting capabilities
  - On crashes, value still in memory

\[ \text{MP} = \text{SM} \ (t < 1/2) \]

[Herlihy '88]

[Attiya, Noy, Dolev '90]
Assume otherwise …

Reason about the properties of any such protocol

Derive a contradiction

Quod

Erat

Demonstrandum

Enough to consider binary consensus and \( n = 2 \)
Either A or B “moves”

Moving means

- Register read
- Register write
THE TWO-MOVE TREE

Initial state

Final states
BIVALENT: BOTH POSSIBLE

bivalent

1 1 1
0 0 0

1 1 1
univalent
DECISION VALUES

1-valent

x-valent: x is the only decision value
Any consensus protocol is a tree

Bivalent system states
- outcome not fixed

Univalent states
- Outcome is fixed
- May not be “known” yet

1-Valent and 0-Valent states
THERE EXISTS AN INITIAL BIVALENT STATE

What if inputs differ?

0

1
THERE EXISTS AN INITIAL BIVALENT STATE

Must decide on 0

In this solo execution by A
THERE EXISTS AN INITIAL BIVALENT STATE

Must decide on 1

In this solo execution by B
Mixed initial state bivalent

Solo execution by A must decide 0

Solo execution by B must decide 1
CRITICAL STATES

0-valent       critical       1-valent
FROM A CRITICAL STATE

If A goes first protocol decides 0

If B goes first protocol decides 1
WHY MUST THERE BE A CRITICAL STATE?

CA

CB

initially bivalent

univalent

univalent

univalent

0-Valent

1-Valent
CRITICAL STATES

- Starting from a bivalent state
- The protocol can reach a critical state
  - otherwise we can stay bivalent forever
- And the protocol does not terminate in finite steps
Starting from a critical state

Each processor fixes the outcome by

- Reading or writing
- Same or different registers

Leading to a 0 or 1 decision

And a contradiction
<table>
<thead>
<tr>
<th></th>
<th>A reads x</th>
<th>A reads y</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x.read</strong></td>
<td>[x.read]</td>
<td>[y.read]</td>
</tr>
<tr>
<td><strong>y.read</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>x.write</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>y.write</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A runs solo, eventually decides 0

B reads x

C

States look same to A

CONTRADICTION

eventually decides 1
### POSSIBLE INTERACTIONS

<table>
<thead>
<tr>
<th></th>
<th>x.read</th>
<th>y.read</th>
<th>x.write</th>
<th>y.write</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.read</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>y.read</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>x.write</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>y.write</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Writing Distinct Registers

A writes y
B writes x
C

B writes x
A writes y

Essentially the same states

Contradiction
## Possible Interactions

<table>
<thead>
<tr>
<th></th>
<th>x.read</th>
<th>y.read</th>
<th>x.write</th>
<th>y.write</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.read</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>y.read</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>x.write</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>no</td>
</tr>
<tr>
<td>y.write</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
</tbody>
</table>
WRITING SAME REGISTERS

A writes x
A runs solo, eventually decides 0

A writes x
0
State looks the same to A

C
B writes x

A writes x
A runs solo, eventually decides 1

CONTRADICTION
### Possible Interactions

<table>
<thead>
<tr>
<th></th>
<th>x.read</th>
<th>y.read</th>
<th>x.write</th>
<th>y.write</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.read</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>y.read</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>x.write</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>y.write</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Q.E.D.
CONSSENSUS* IS IMPOSSIBLE IN A COMPLETELY ASYNCHRONOUS MESSAGE PASSING SYSTEMS WHERE EVEN A SINGLE PROCESSOR FAILS

[Fischer, Lynch, Paterson ’85]
CONSENSUS* IS IMPOSSIBLE IN A COMPLETELY ASYNCHRONOUS MESSAGE PASSING SYSTEMS WHERE EVEN A SINGLE PROCESSOR FAILS

[Fischer, Lynch, Paterson ’85]
Are all three types of asynchrony needed simultaneously to obtain impossibility result?

[Dolev, Dwork, Stockmeyer ’87]
Minimal synchronism needed for distributed consensus
Processors can be synchronous or asynchronous.

Communication delay can be bounded or unbounded.

Messages can be ordered or unordered.

Transmission mechanism can be point-to-point or broadcast.
## World of (Im)Possibilities

<table>
<thead>
<tr>
<th>Processors</th>
<th>Message Order</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>Unordered</td>
<td>Unbounded</td>
</tr>
<tr>
<td></td>
<td>FLP</td>
<td>Bounded</td>
</tr>
<tr>
<td>Synchronous</td>
<td>Ordered</td>
<td>Unbounded</td>
</tr>
<tr>
<td></td>
<td>FLP</td>
<td>Unbounded</td>
</tr>
<tr>
<td></td>
<td>Broadcast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmission</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Point-to-point</td>
<td></td>
</tr>
</tbody>
</table>

**16 possibilities**
3 cases where N-resilient protocols exist

**Case 1**
Processors are synchronous and communication is bounded

**Case 2**
Messages are ordered and transmission mechanism is broadcast

**Case 3**
Processors are synchronous and messages are ordered
## Processors are synchronous and communication is bounded

<table>
<thead>
<tr>
<th>Processors</th>
<th>Unordered</th>
<th>Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>FLP</td>
<td>FLP</td>
</tr>
<tr>
<td>Synchronous</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

**Communication**

- Point-to-point: Unbounded
- Broadcast: Bounded
- Point-to-point: Unbounded

**Message Order**

- Unordered: FLP, FLP
- Ordered: YES, YES, YES, YES

**Use Timeout**
**Messages are ordered and transmission mechanism is broadcast**

<table>
<thead>
<tr>
<th>Communication</th>
<th>Process</th>
<th>Message Order</th>
<th>Transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unbounded</td>
<td>Unordered</td>
<td>FLP</td>
<td>Point-to-point</td>
</tr>
<tr>
<td>Bounded</td>
<td>Unordered</td>
<td>FLP</td>
<td>Broadcast</td>
</tr>
<tr>
<td>Unbounded</td>
<td>Ordered</td>
<td>YES</td>
<td>Point-to-point</td>
</tr>
<tr>
<td>Bounded</td>
<td>Ordered</td>
<td>YES</td>
<td>Broadcast</td>
</tr>
</tbody>
</table>
Processors are synchronous and messages are ordered
MORE RESULTS: [DOLEV, DWORK, STOCKMEYER '87]

- 3 cases where N-resilient protocols exist
- No t-resilient protocol if system weakened for t = 1, 2
  - Favourable -> Unfavourable

Processors: synchronous / asynchronous
Communication: bounded / unbounded
Messages: ordered / unordered
Transmission: broadcast / point-to-point
## World of (Im)Possibilities

<table>
<thead>
<tr>
<th>Processors</th>
<th>Unordered</th>
<th>Ordered</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>FLP</td>
<td>FLP</td>
<td>YES</td>
</tr>
<tr>
<td>Synchronous</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

*Transmission: B -> P2P, sync -> async, bdd -> u-bdd*
Probabilistic consensus possible in presence of faults
  
  do not terminate wlp
  
  if terminate, right result
  
  decisions not consistent wlp
PLOTTING A BYZANTINE AGREEMENT
BYZANTINE GENERALS PROBLEM

[Lamport, Shostak, Pease ’82]
Choose the majority vote
Reach consensus
BYZANTINE GENERALS PROBLEM

Do not care what bz decides

- **Attack** [A, R, A, R, A]
- **Retreat** [A, R, A, R, A]
- **Attack** [A, R, A, R, A]
- **Retreat** [A, R, A, R, A]
- **Attack** [A, R, A, R, A]
BYZANTINE GENERALS PROBLEM


Consensus violated
Is there a protocol such that

- **CONSISTENCY**: All honest parties agree on same value
- **VALIDITY**: Agreed-upon value is input to some honest party
- **TERMINATION**: Each honest party decides in finite number of steps
## World of (IM)Possibilities

<table>
<thead>
<tr>
<th>Processors</th>
<th>Unordered</th>
<th>Ordered</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>FLP</td>
<td>FLP</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Synchronous</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

**Broadcast Transmission**

**Point-to-point Transmission**

Impossible in CF model => Impossible in Byzantine model
## WORLD OF (IM)POSSIBILITIES

<table>
<thead>
<tr>
<th>Processors</th>
<th>Unordered</th>
<th>Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>FLP</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Synchronous</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transmission</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point-to-point</td>
<td>Unbounded</td>
</tr>
<tr>
<td>Broadcast</td>
<td>Bounded</td>
</tr>
<tr>
<td>Point-to-point</td>
<td>Unbounded</td>
</tr>
</tbody>
</table>
Is Byzantine adversary more powerful?

Broadcasts don’t lie!

Byzantine = Crash failure
<table>
<thead>
<tr>
<th>Processors</th>
<th>Unordered</th>
<th>Ordered</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>FLP</td>
<td>FLP</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Synchronous</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
| Point-to-point | Broadcast | Point-to-point | }
BYZANTINE CONSENSUS IS IMPOSSIBLE WHEN NO AUTHENTICATION IS POSSIBLE IF $T \geq \frac{1}{3} N$

[Lamport, Shostak, Pease '82]
Assume otherwise ...

Reason about the properties of any such protocol

Derive a contradiction

Quod
Erat
Demonstrandum

Show for $n = 3$ and $t = 1$
IMPOSSIBILITY PROOF (HIGH LEVEL)

validity: Decide 0
IMPOSSIBILITY PROOF (HIGH LEVEL)

validity: Decide 0

Decide 0: View of C is indistinguishable from Case 1
IMPOSSIBILITY PROOF (HIGH LEVEL)

validity: Decide 0

Decide 0: View of C is indistinguishable from Case 1

validity: Decide 1

A decides 0: View indistinguishable from Case 2

CONTRADICTION
## World of (Im)Possibilities

<table>
<thead>
<tr>
<th>Processors</th>
<th>Unordered</th>
<th>Ordered</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLP</td>
<td>FLP</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>YES*</td>
<td>YES</td>
<td>YES</td>
<td>YES*</td>
</tr>
<tr>
<td>Synchronous</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point-to-point</td>
<td>Broadcast</td>
<td>Point-to-point</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES*</td>
</tr>
</tbody>
</table>

*FLP indicates Failure to Produce.*
Case $n = 4$, $t = 1$

Choose a leader

Listen to what he says

Talk to others

See what they say about the leader said

Decide on the majority value
ALGORITHM IDEA WHEN $T < \frac{1}{3} N$

Case 1

CONSISTENCY

VALIDITY
ALGORITHM IDEA WHEN $T < \frac{1}{3} N$
BYZANTINE CONSENSUS IS IMPOSSIBLE WHEN NO AUTHENTICATION IS POSSIBLE IF T \geq \frac{1}{3} N

[Lamport, Shostak, Pease '82]
Malicious can send conflicting messages to other two
Malicious can send conflicting messages to other two
Use authenticated messages

Give byzantine consensus protocol
  - Remember the model (sync processors + bdd delay)

Use cryptographic techniques to sign
The Advances in Consensus: Sands of Time

- **'82**: LSP
  - Bz consensus is impossible unless t < n/3 if no crypto
  - Protocol using crypto
  - Sync + p-to-p + bdd delay

- **'83**: Ben
  - CF consensus is impossible for t = 1
  - Complete asynchronous

- **'85**: FLP
  - Minimum synchronisation for CF consensus
  - Different models

- **'87**: DDS
  - CF consensus is impossible for t = 1 in RW SM
  - Asynchronous processors

- **'88**: Her
  - MP = SM for t < 1/2
  - Different models

- **'90**: AND
  - Proof of stake

- **'08**: Sat
  - Practical models
Crash failure consensus is hard

Byzantine is even harder

Crypto is cool

Distributed + crypto is even cooler

that's why we have been doing it

that's why we will be doing it
Questions?
REFERENCES

- Crash failure impossibility proof from lecture slides of Maurice Herlihy and Nir Shavit
- Many Faces of Consensus in Distributed Systems by John Turek and Dennis Shasha