Peeking Into the Future

MPC Resilient to Super-Rushing Adversaries

Gilad Asharov

Anirudh Chandramouli

Ran Cohen

Yuval Ishai


BACK TO THE FUTURE
He was never in time for his classes...
He wasn't in time for his dinner...
Then one day... he wasn't in his time at all.
erik

“Well, hey, Doc, what’s the harm in bringing back a little info on the future? You know, maybe we could place a couple bets”

Biff's Attack on the Timeline

Biff's Attack on the Timeline

2015

Biff steals the almanac

Biff's Attack on the Timeline

1955

2015

Gives it to his past self

Biff steals the almanac

Biff's Attack on the Timeline

1955

2015

Gives it to his past self

Biff gets rich!

Biff steals the almanac

Biff's Attack on the Timeline

1955

2015

Biff gets rich!

Biff steals the almanac

Biff's Attack on the Tim[illegible]ne

19[illegible]

[illegible]15

[illegible]anac

Biff gets rich!

Biff

The Weather

Hill Valley Telegraph

Index

Vol. XVII, No. 38

PUBLISHED DAILY

PRICE 10 CENTS

BIFF TANNEN

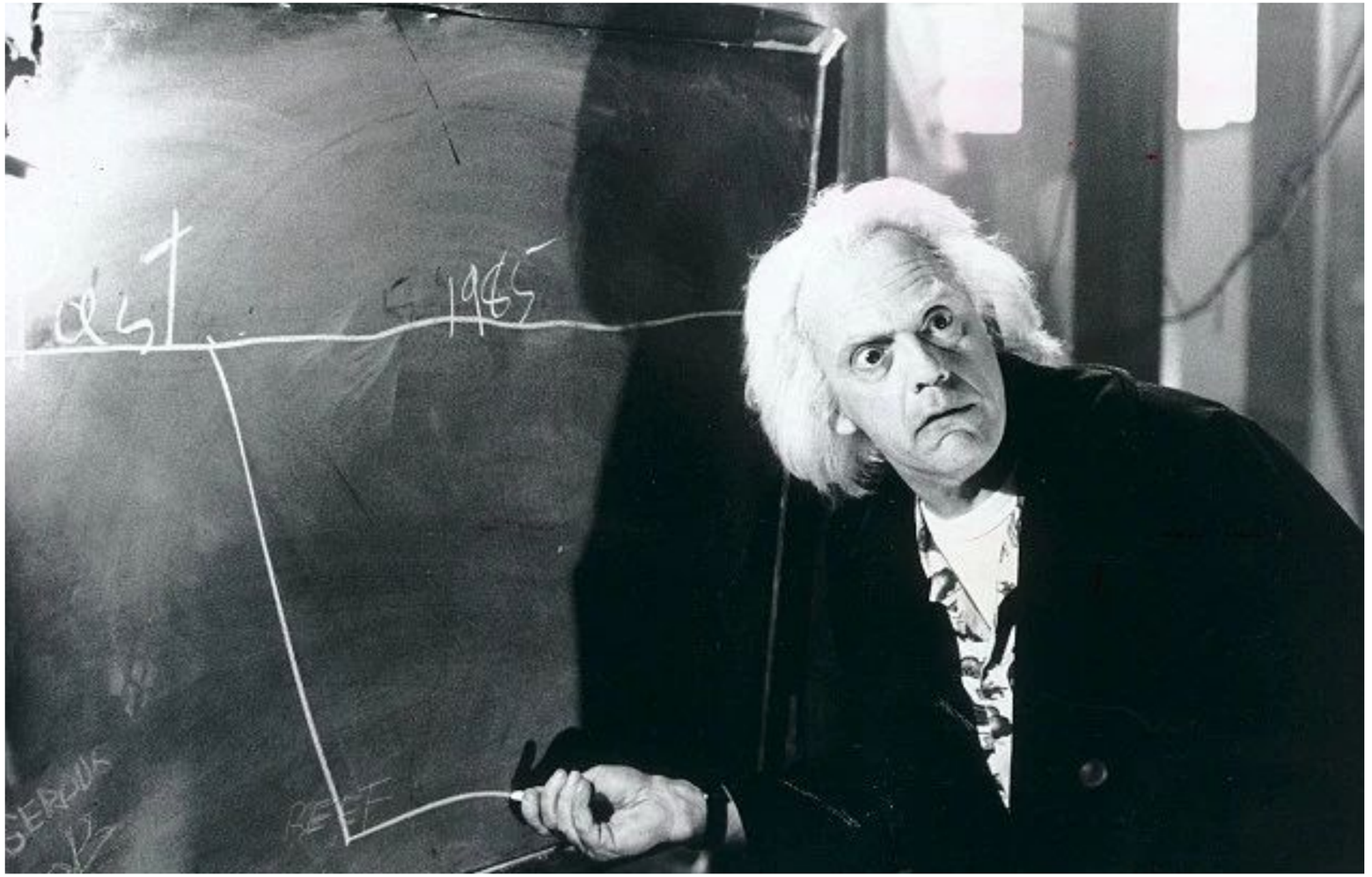
Luckiest Man On Earth

BIFF TANNEN

Nasser Accuses Reds of Plotting His Overthrow

State Likely to Start Payroll Tax in 1960

“No, Marty, we’ve already agreed that having information about the future could be extremely dangerous!”



Our Work

Our Work

1. The “Back to the Future” attack is real!

Our Work

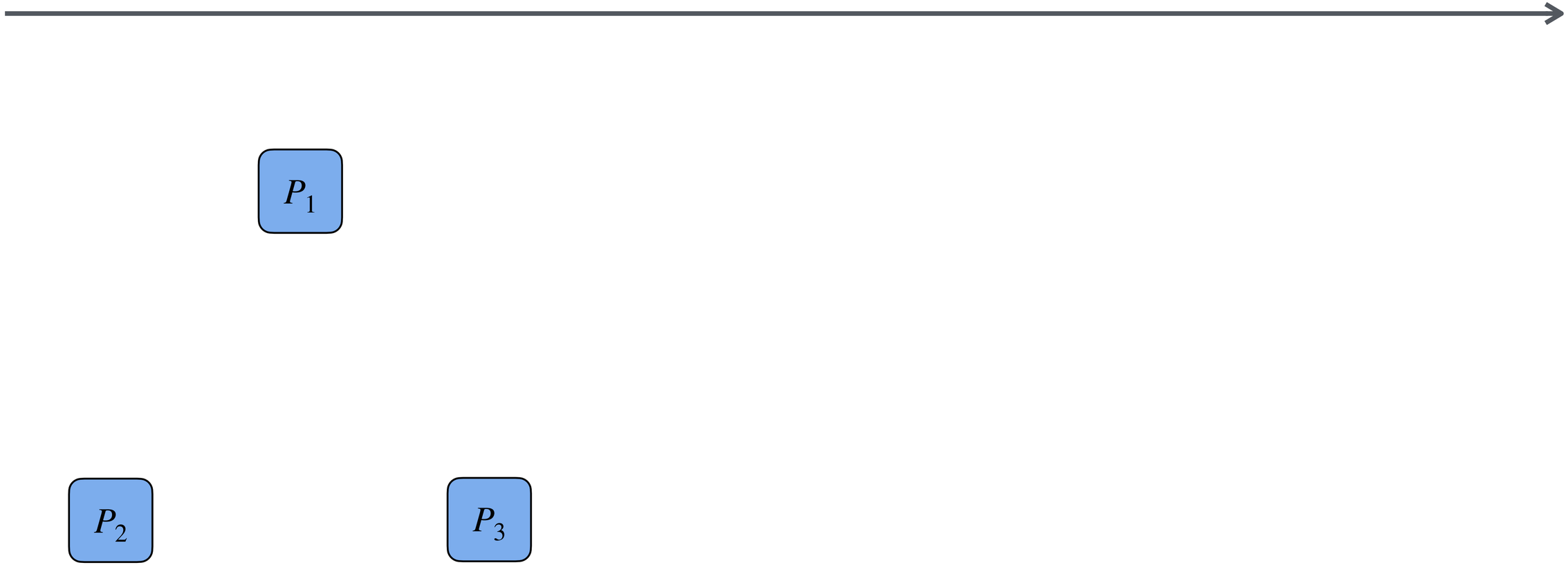
1. The “Back to the Future” attack is real!
   - Some implementations of synchronous protocols are vulnerable

Our Work

1. The “Back to the Future” attack is real!
   - Some implementations of synchronous protocols are vulnerable
2. **Our Aim:** To understand which protocols are vulnerable to such an attack

Synchronous Protocols

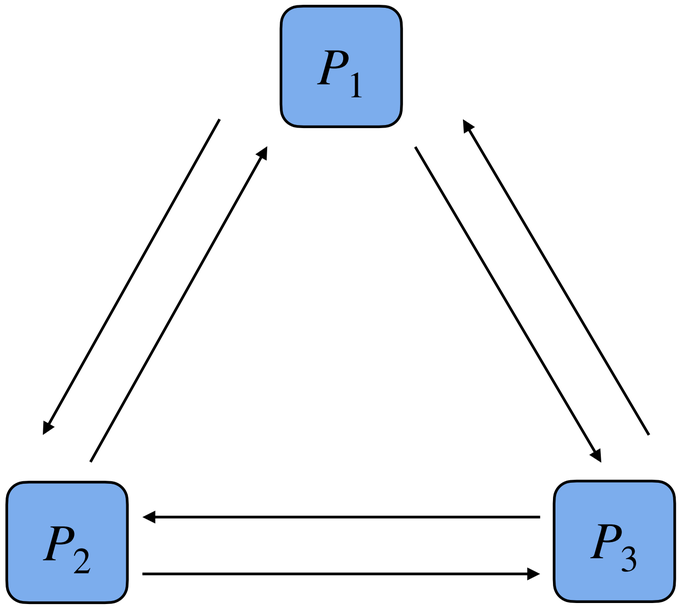
Let there be rounds!



Synchronous Protocols

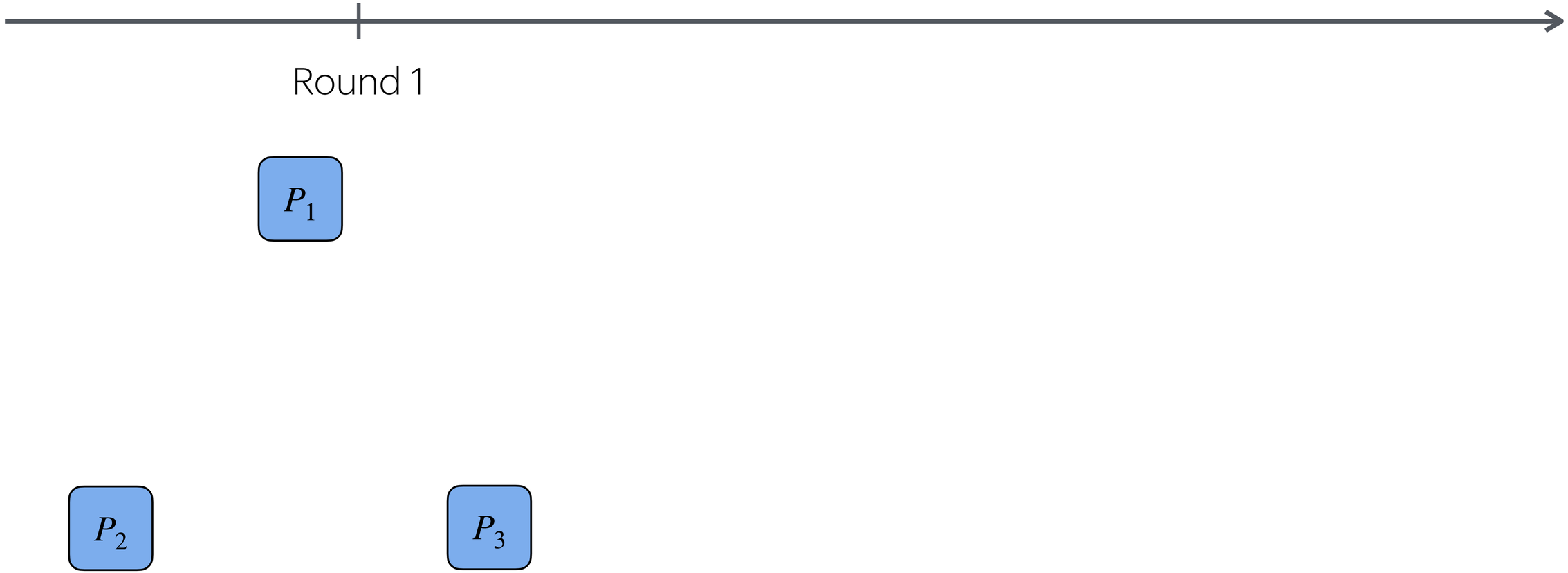
Let there be rounds!

Round 1



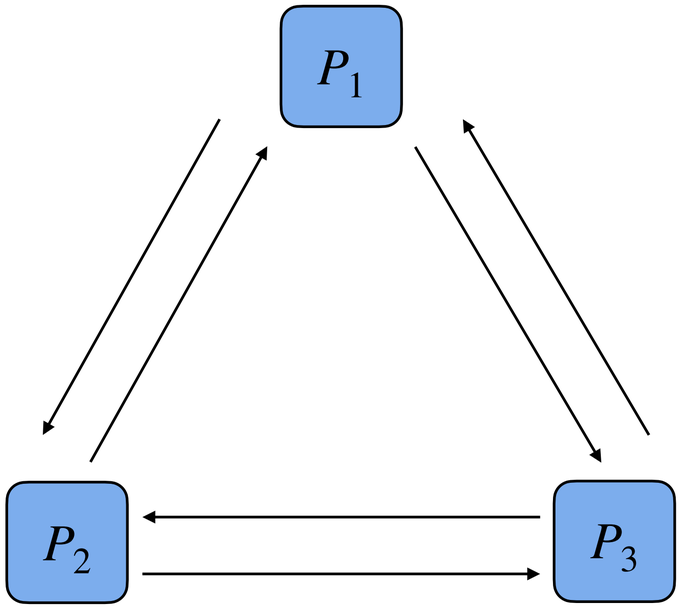
Synchronous Protocols

Let there be rounds!



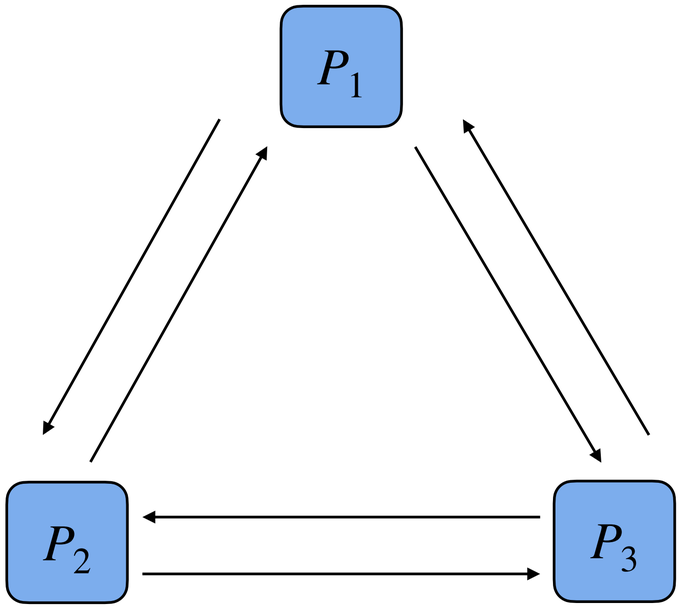
Synchronous Protocols

Let there be rounds!



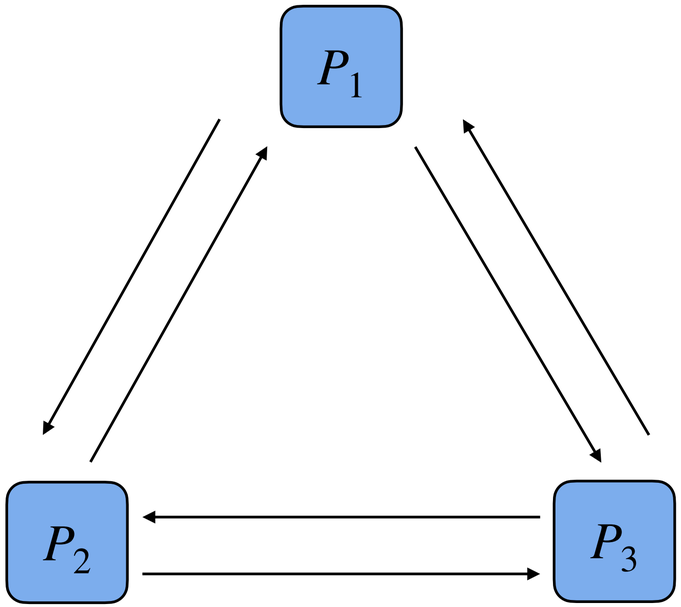
Synchronous Protocols

Let there be rounds!



Synchronous Protocols

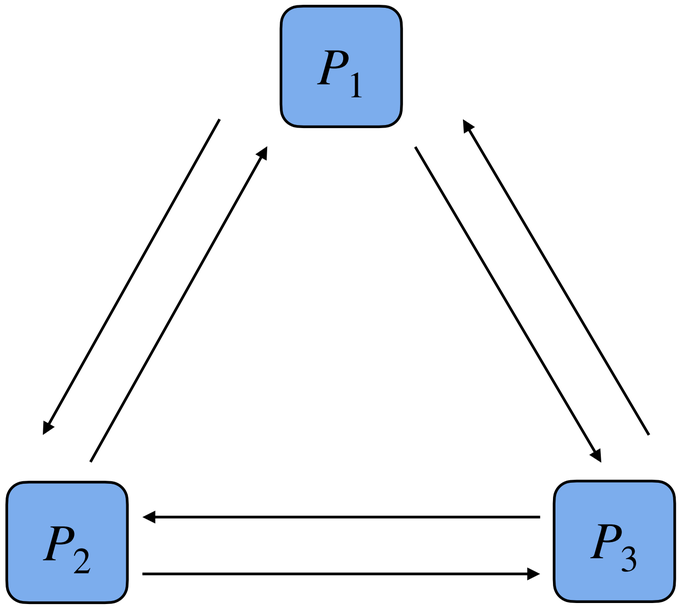
Let there be rounds!



- Idealized assumption: all round r messages are delivered before round r+1

Synchronous Protocols

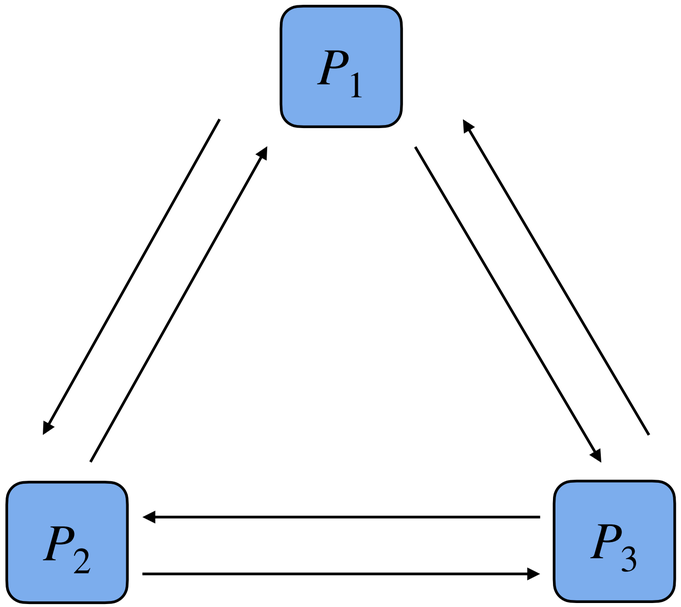
Let there be rounds!



- Idealized assumption: all round r messages are delivered before round r+1
- Majority of the literature

Synchronous Protocols

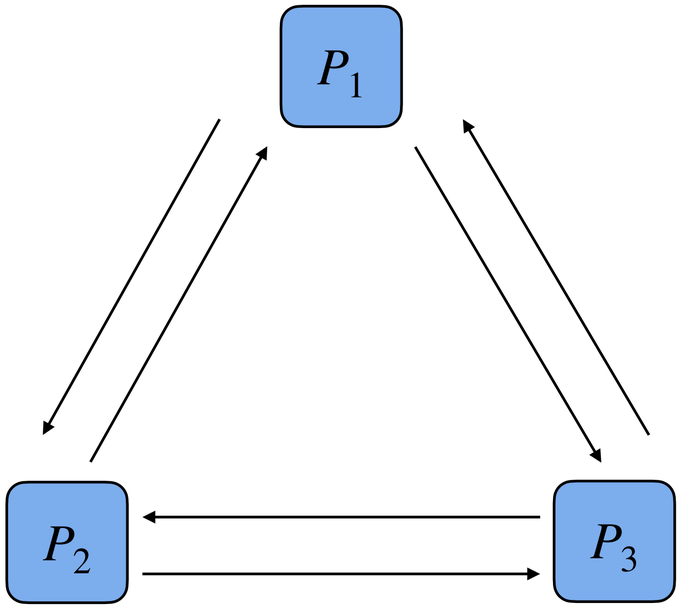
Let there be rounds!



- Idealized assumption: all round r messages are delivered before round r+1
- Majority of the literature
- How to decide the delay?

Synchronous Protocols

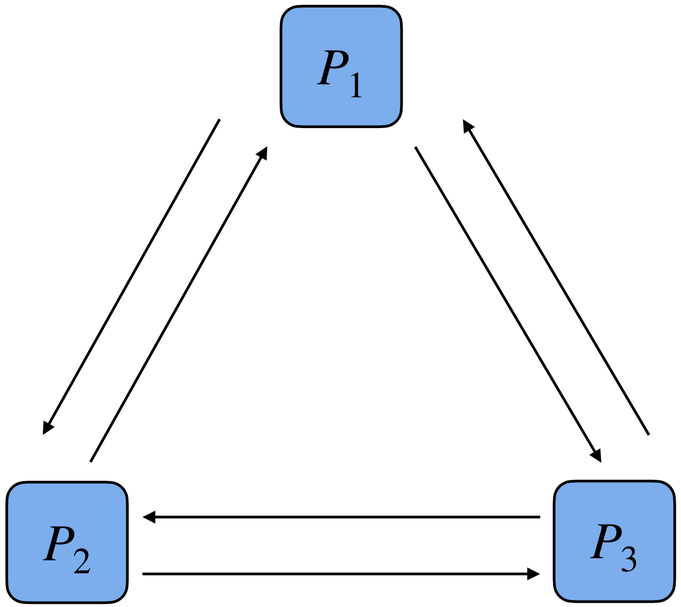
Let there be rounds!



- Idealized assumption: all round r messages are delivered before round r+1
- Majority of the literature
- How to decide the delay?

Synchronous Protocols

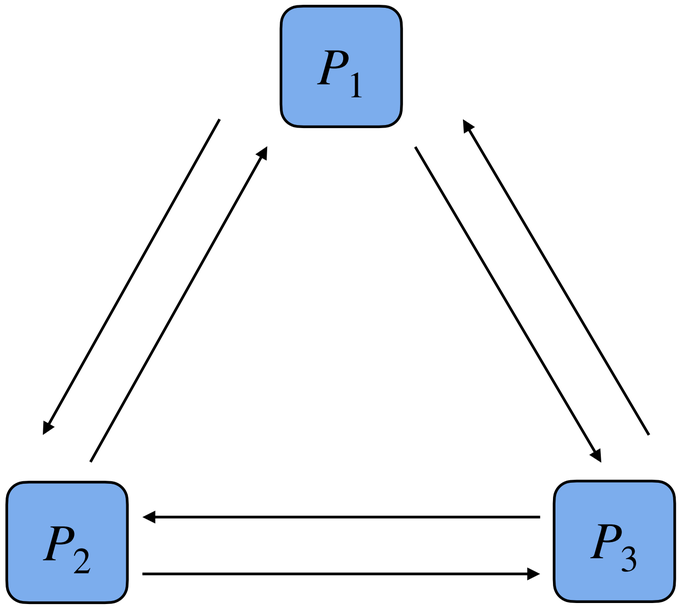
Let there be rounds!



- Idealized assumption: all round r messages are delivered before round r+1
- Majority of the literature
- How to decide the delay?

Synchronous Protocols

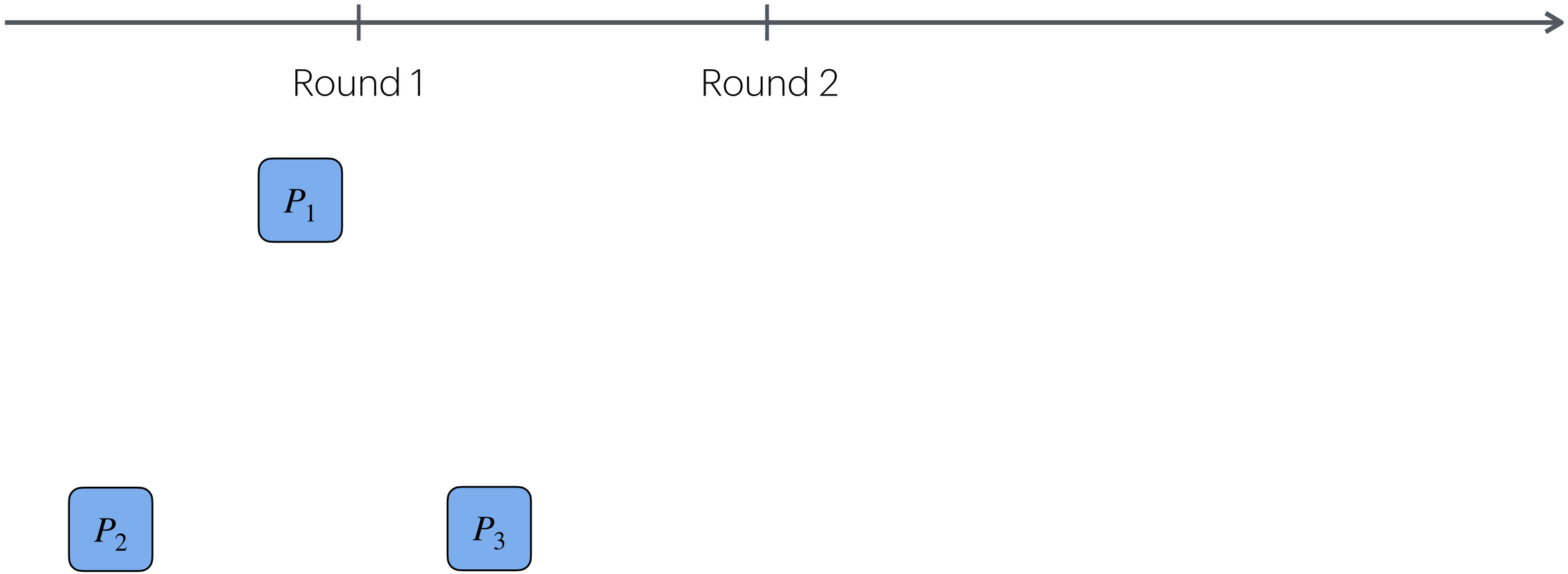
Let there be rounds!



- Idealized assumption: all round r messages are delivered before round r+1
- Majority of the literature
- How to decide the delay?

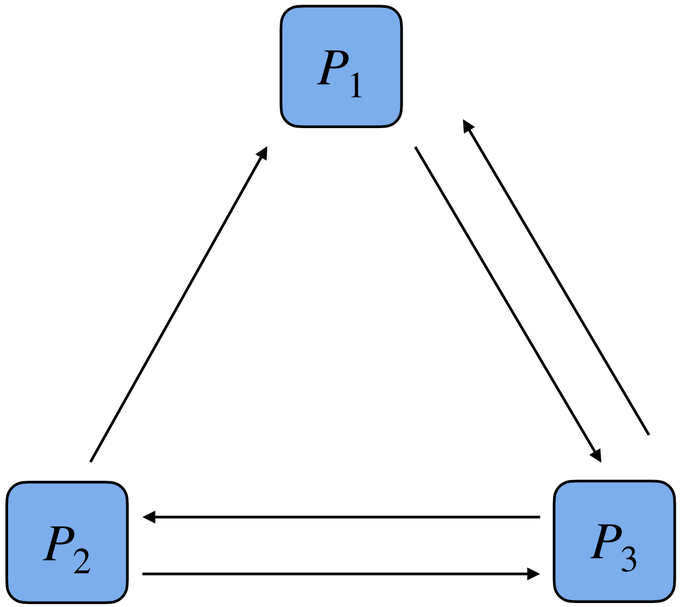
Protocol Implementations

Optimistic: Proceed to next round if all messages received



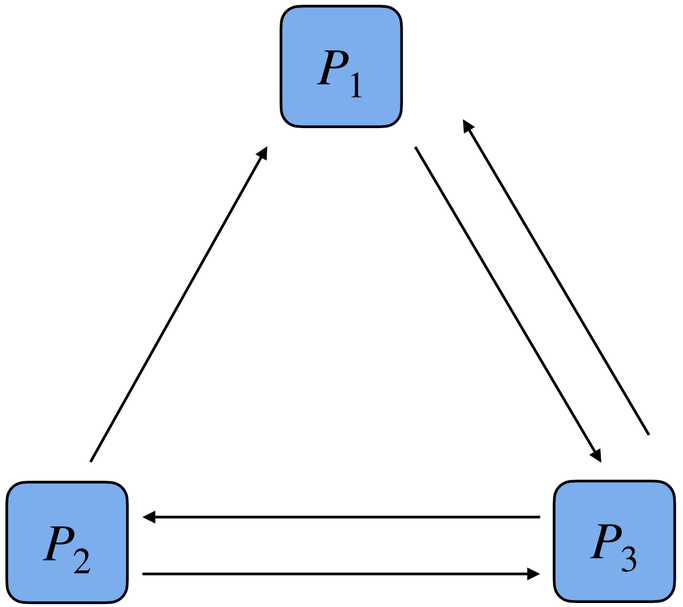
Protocol Implementations

Optimistic: Proceed to next round if all messages received



Protocol Implementations

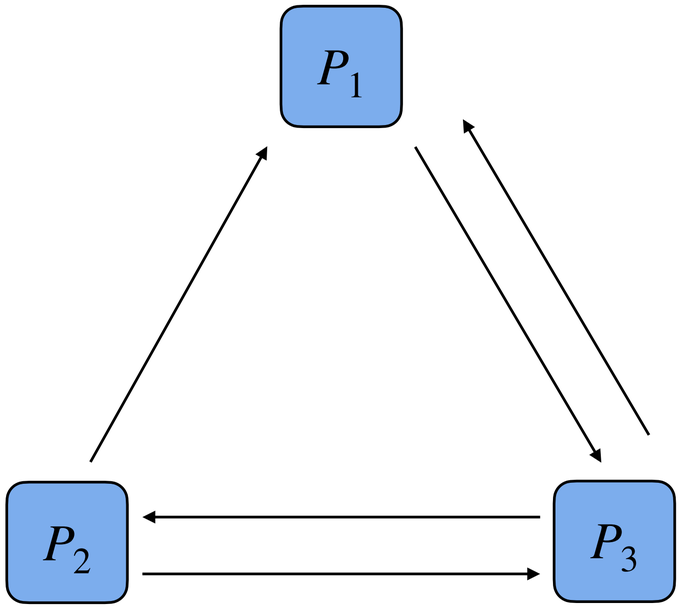
Optimistic: Proceed to next round if all messages received



P_1 has not yet sent round-1 message to P_2

Protocol Implementations

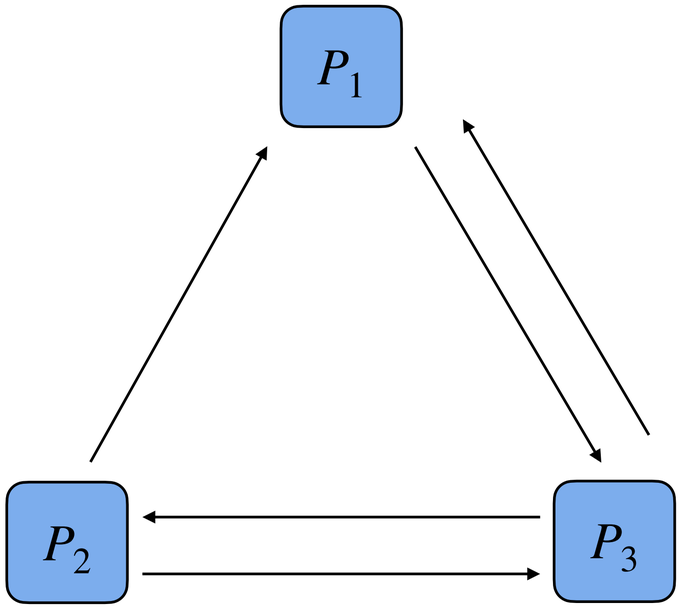
Optimistic: Proceed to next round if all messages received



P_1 has not yet sent round-1 message to P_2

Protocol Implementations

Optimistic: Proceed to next round if all messages received

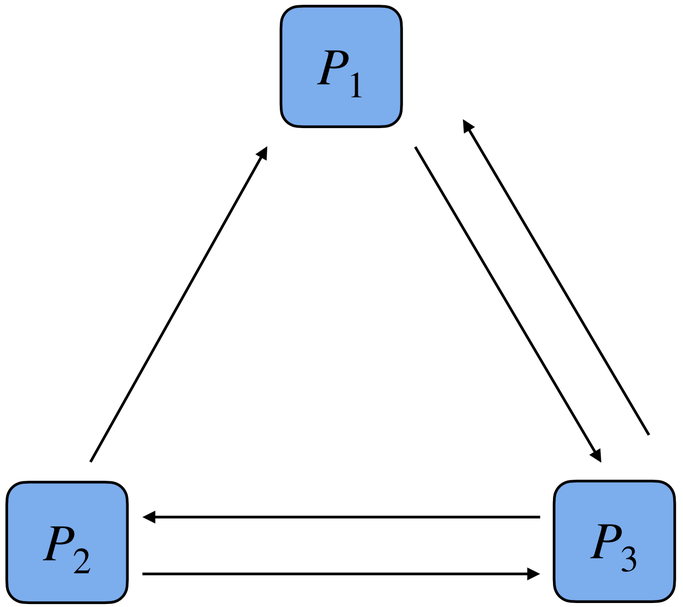


P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

Protocol Implementations

Optimistic: Proceed to next round if all messages received

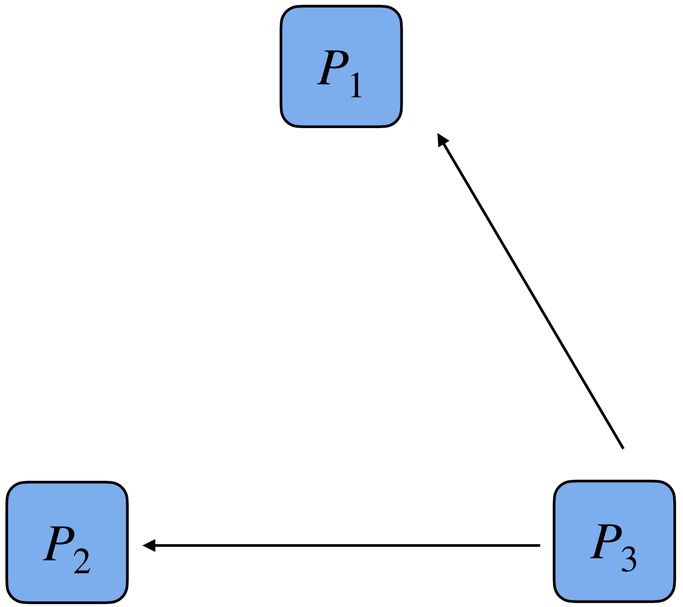


P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

Protocol Implementations

Optimistic: Proceed to next round if all messages received



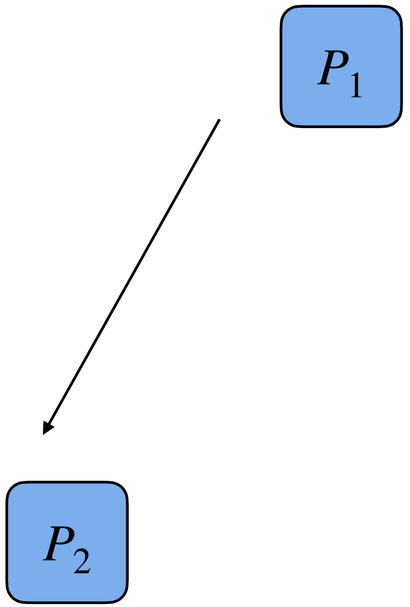
P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

P_3 sends round-2 messages

Protocol Implementations

Optimistic: Proceed to next round if all messages received



P_1 has not yet sent round-1 message to P_2

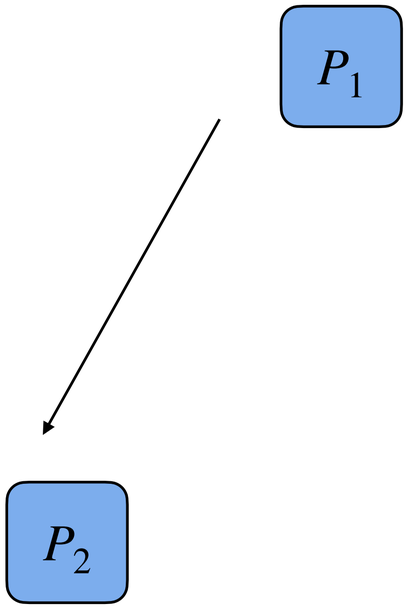
P_3 has finished round-1

P_3 sends round-2 messages

P_1 sends round-1 message to P_2

Protocol Implementations

Optimistic: Proceed to next round if all messages received



P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

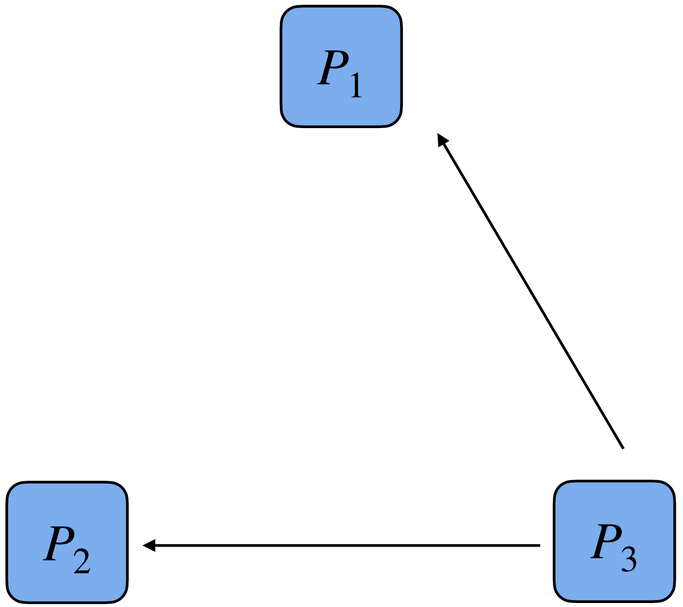
P_3 sends round-2 messages

P_1 sends round-1 message to P_2

P_1 gets a message from the future (from P_3)

“Wait a minute. Wait a minute Doc, uh, are you tellin’ me you built a time machine ... out of a synchronous network?”

No!



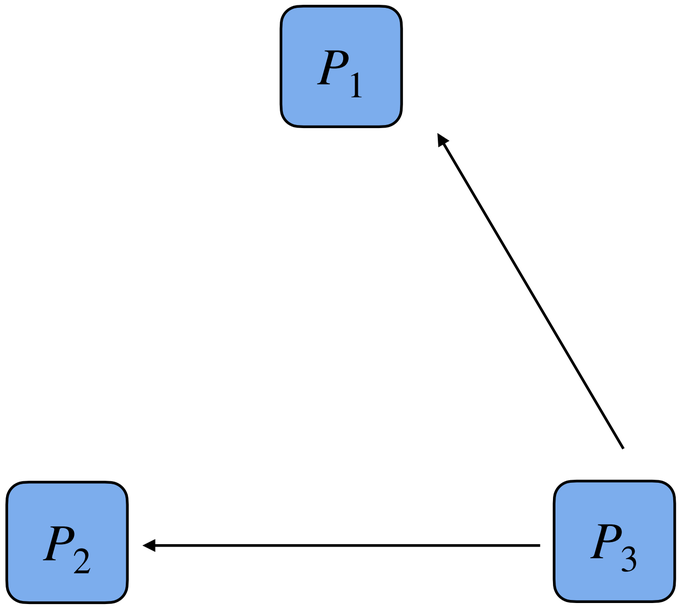
P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

P_3 sends round-2 messages

No!

P_1 can "peek" into the future one round



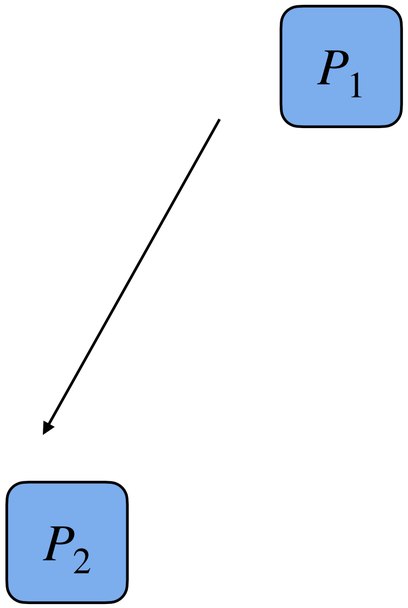
P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

P_3 sends round-2 messages

No!

P_1 can “peek” into the future one round



P_1 has not yet sent round-1 message to P_2

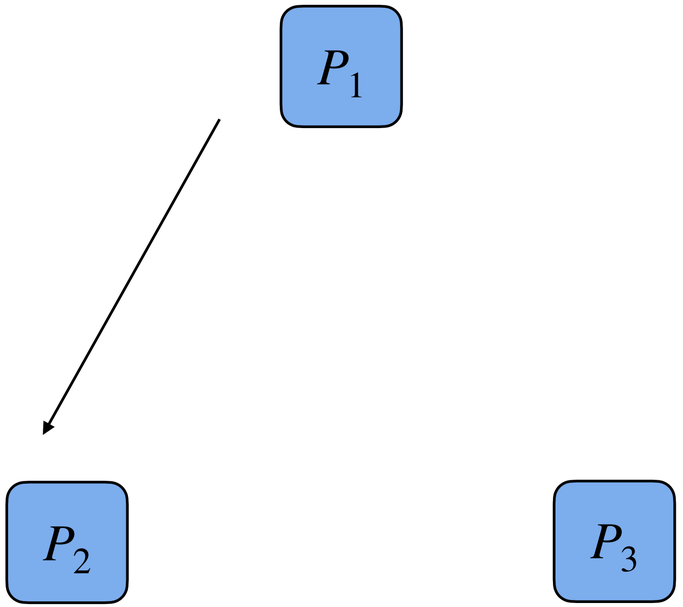
P_3 has finished round-1

P_3 sends round-2 messages

P_1 sends round-1 message to P_2

No!

P_1 can "peek" into the future one round



P_1 has not yet sent round-1 message to P_2

P_3 has finished round-1

P_3 sends round-2 messages

P_1 sends round-1 message to P_2

Note: All-to-all communication $\Longrightarrow$ Peeking at most 1 round

Peeking → Super-Rushing

Non-Rushing

Adversary sends round-r messages **before** receiving the honest parties' round-r messages

Non-Rushing

Adversary sends round-r messages **before** receiving the honest parties' round-r messages

Rushing

Adversary can send round-r messages **after** receiving the honest parties' round-r messages

Non-Rushing

Adversary sends round-r messages **before** receiving the honest parties' round-r messages

Rushing

Adversary can send round-r messages **after** receiving the honest parties' round-r messages

Super-Rushing

Adversary can send round-r messages **after** receiving the honest parties' round-$r' > r$ messages

A Gap in the Security Analysis

Theory

Practice

A Gap in the Security Analysis

Theory

Practice

Rushing

A Gap in the Security Analysis

Theory

Rushing

Practice

Super-Rushing

A Gap in the Security Analysis

Theory

Rushing

Practice

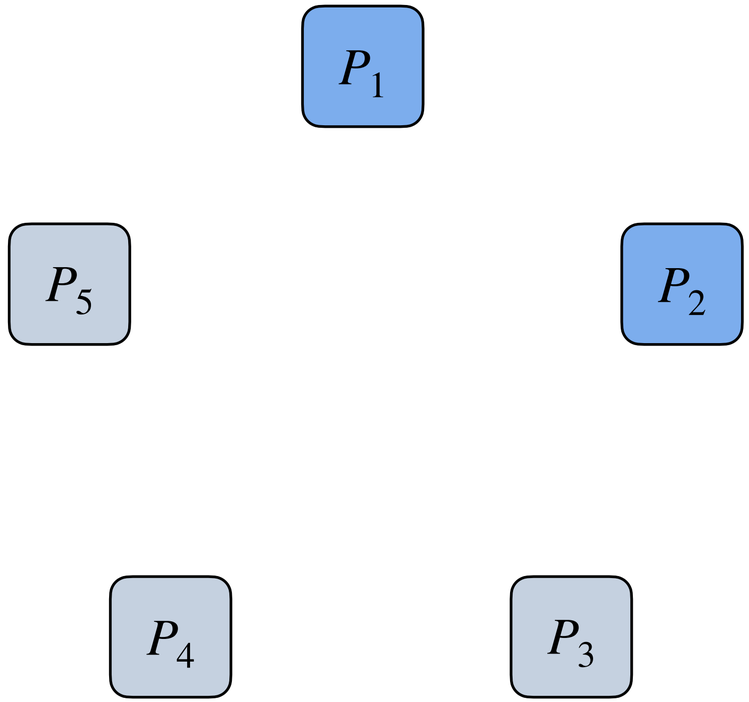
Super-Rushing

Question: Are existing synchronous protocols vulnerable to super-rushing attacks?

Yes! Some protocols are insecure against super-rushing adversaries

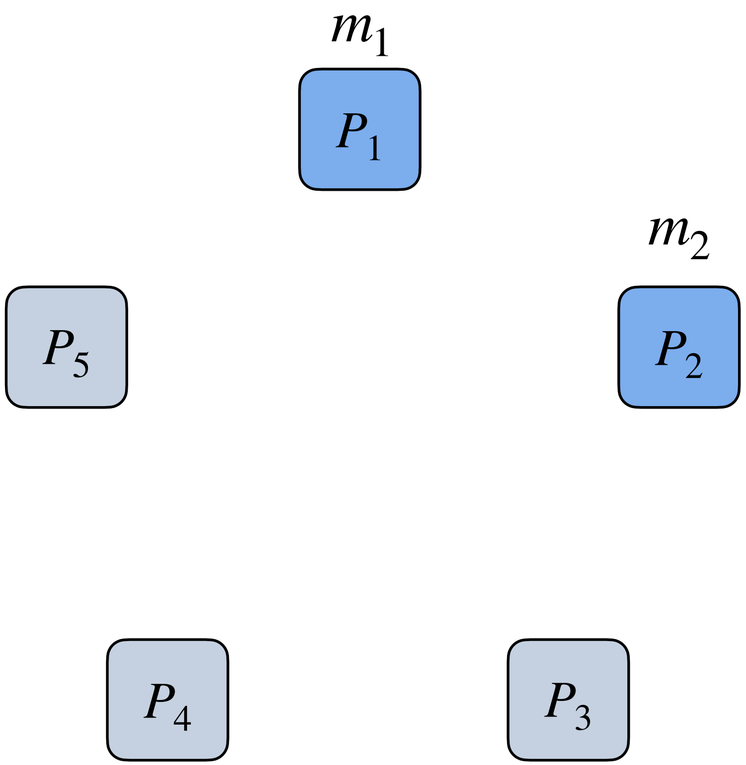
Simultaneous Broadcast [CGMA85]

1 corruption, 2 senders, 5 parties



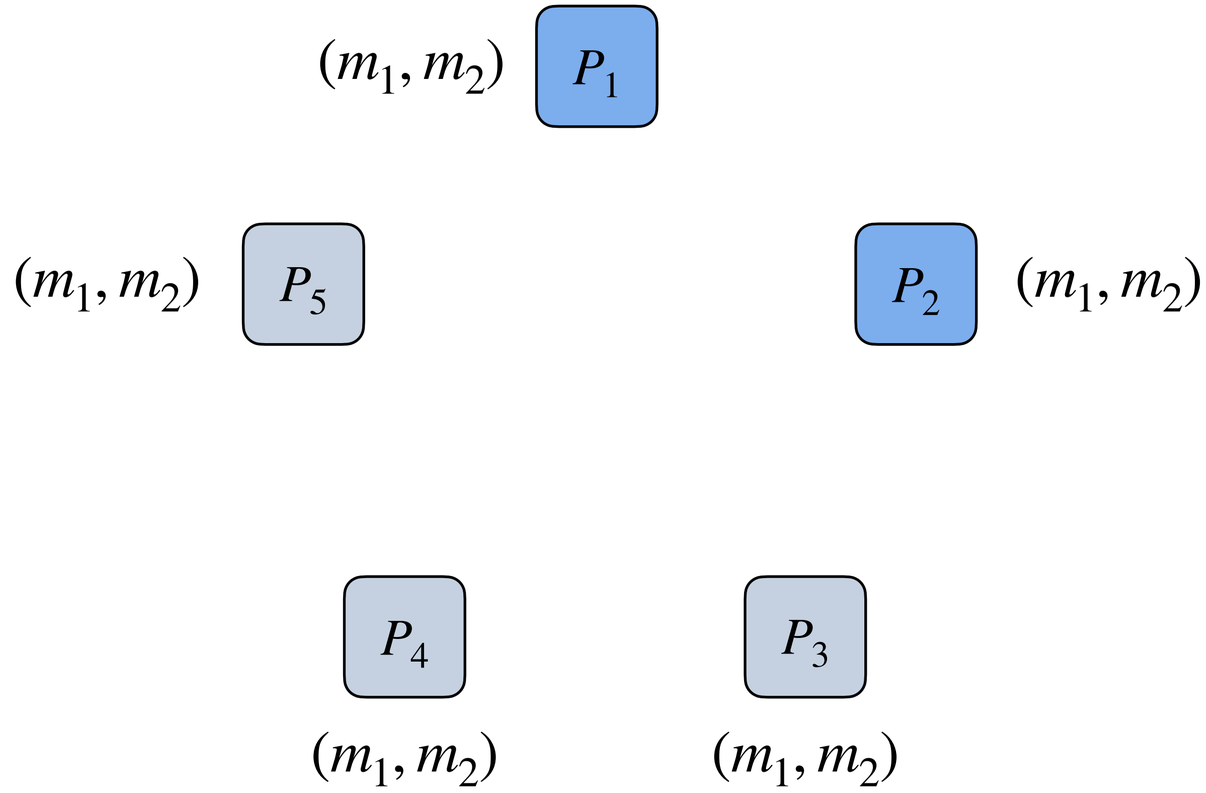
Simultaneous Broadcast [CGMA85]

1 corruption, 2 senders, 5 parties



Simultaneous Broadcast [CGMA85]

1 corruption, 2 senders, 5 parties

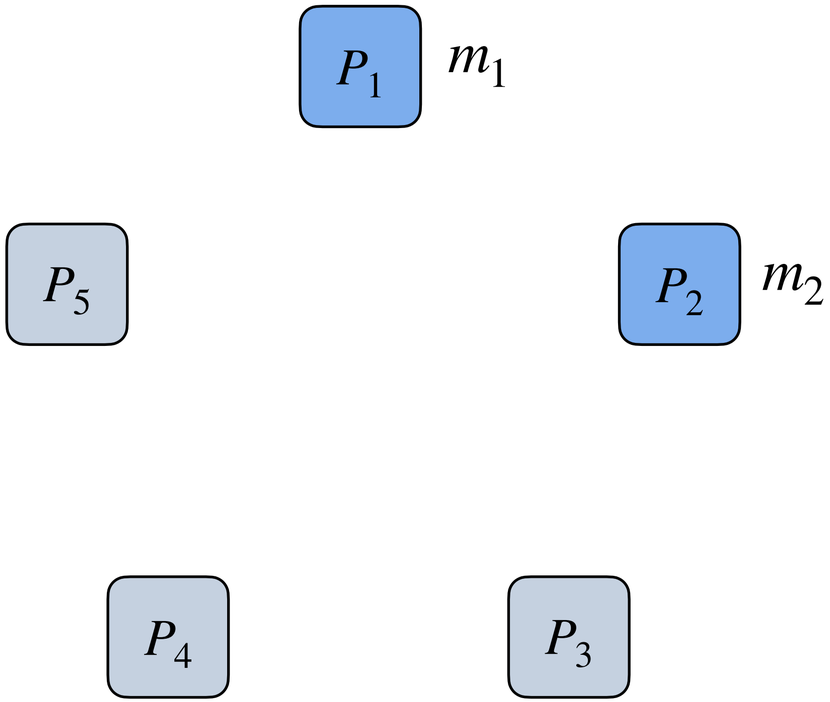


A Simple Simultaneous Broadcast Protocol

1 corruption, 2 senders, 5 parties

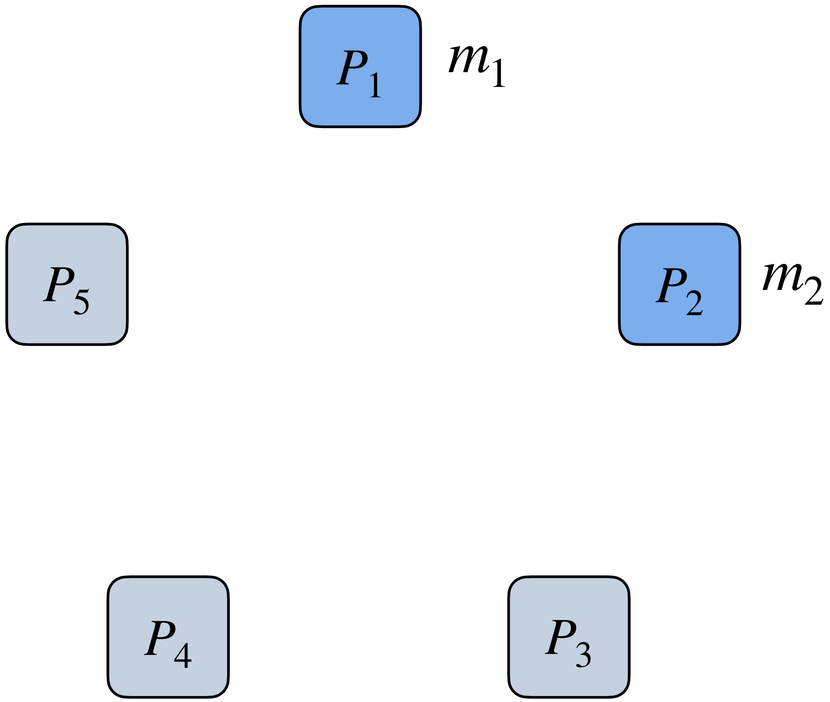
A Simple Simultaneous Broadcast Protocol

1 corruption, 2 senders, 5 parties



A Simple Simultaneous Broadcast Protocol

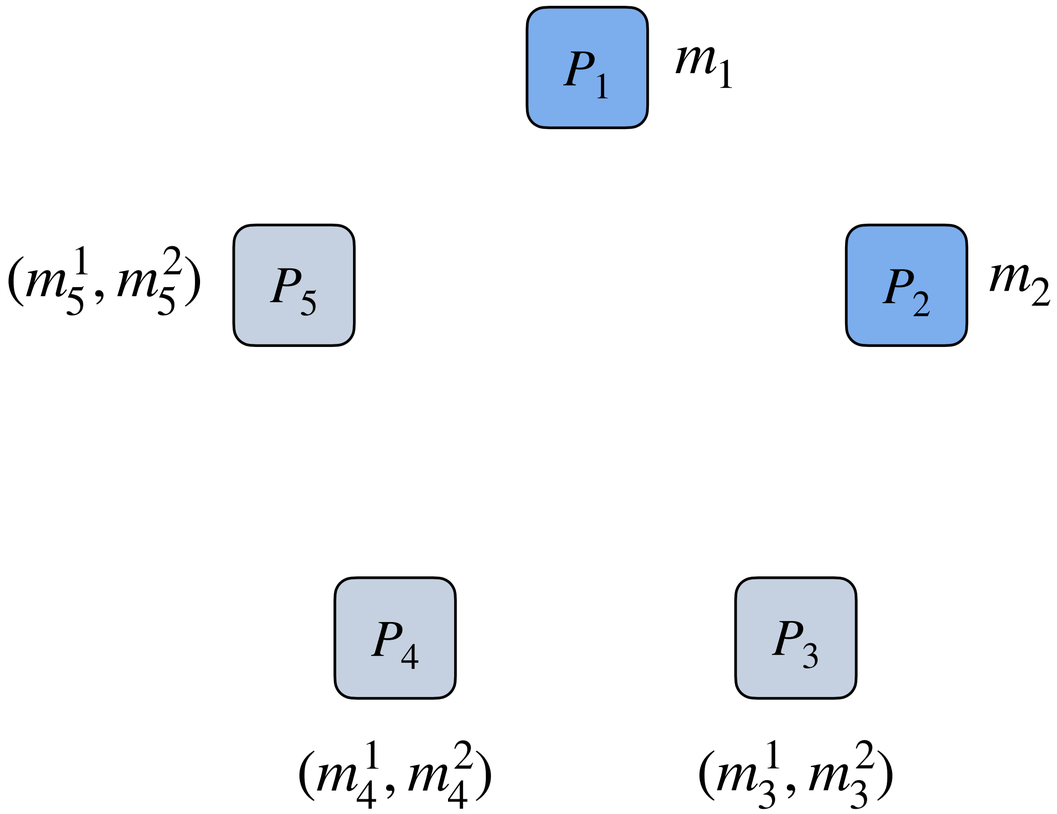
1 corruption, 2 senders, 5 parties



- Round 1: P_1, P_2 send input message to P_3, P_4, P_5

A Simple Simultaneous Broadcast Protocol

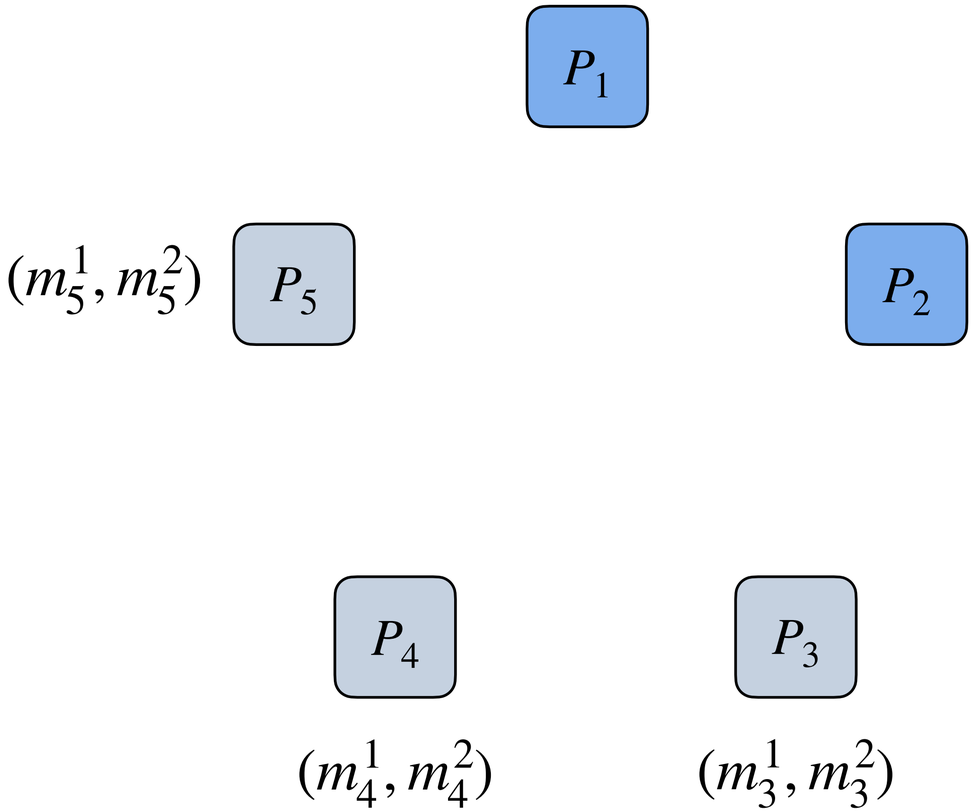
1 corruption, 2 senders, 5 parties



- Round 1: P_1, P_2 send input message to P_3, P_4, P_5

A Simple Simultaneous Broadcast Protocol

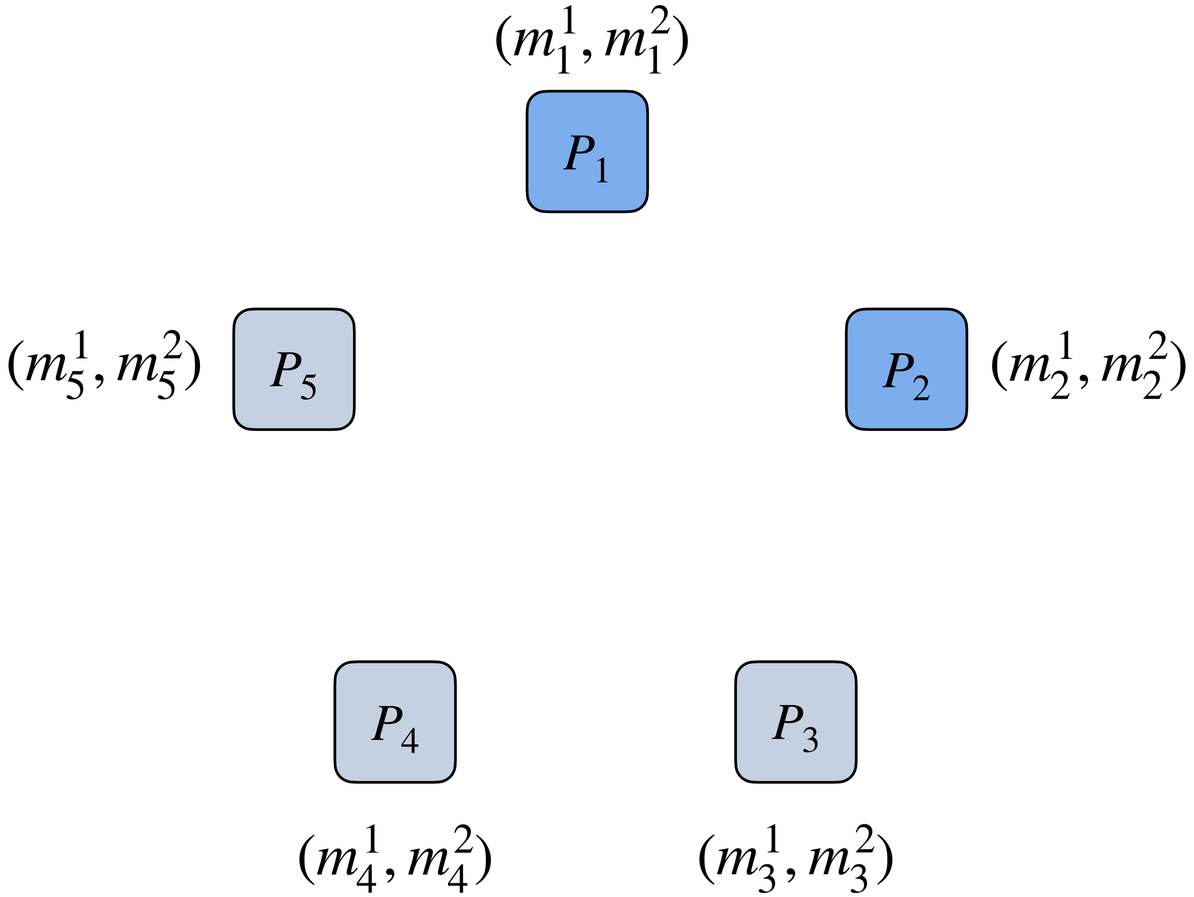
1 corruption, 2 senders, 5 parties



- Round 1: P_1, P_2 send input message to P_3, P_4, P_5
- Round 2: P_3, P_4, P_5 echo message to all parties

A Simple Simultaneous Broadcast Protocol

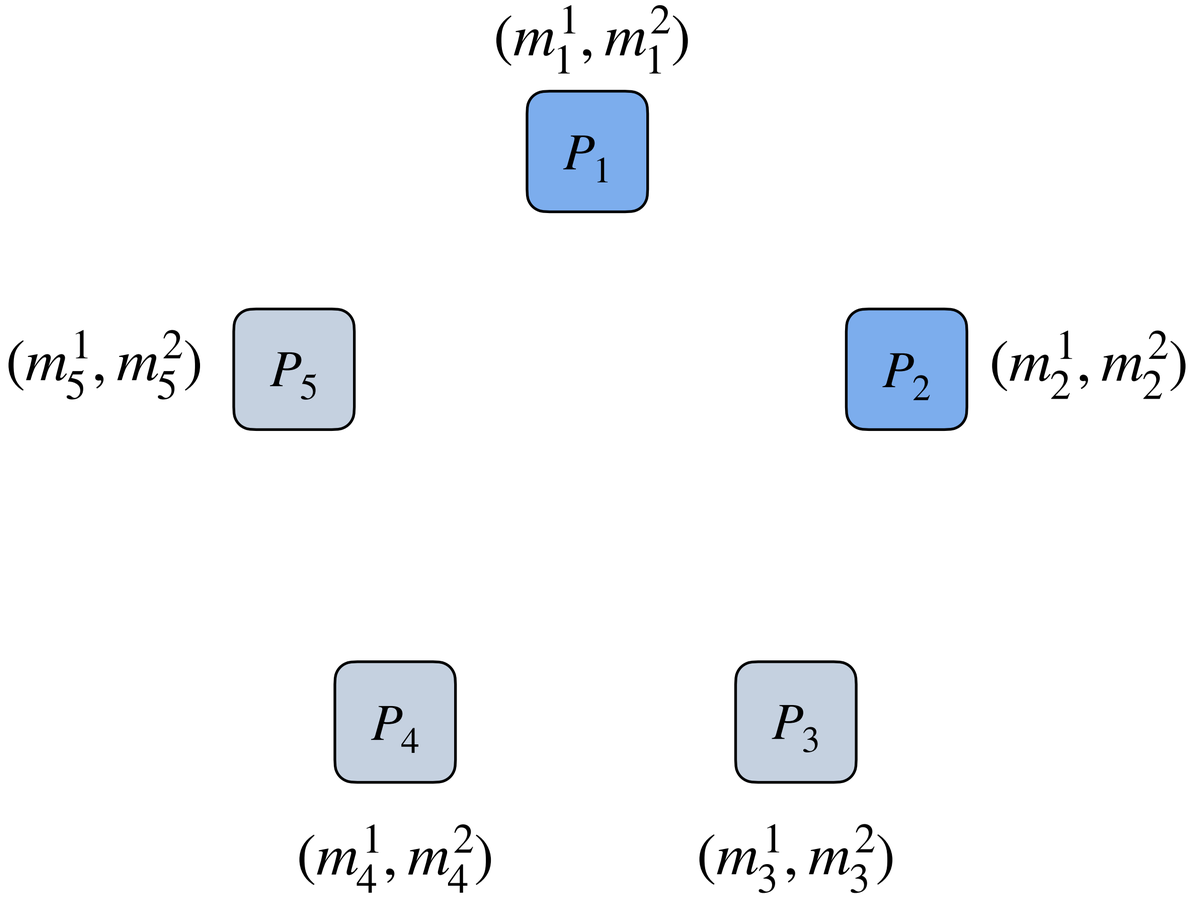
1 corruption, 2 senders, 5 parties



- Round 1: P_1, P_2 send input message to P_3, P_4, P_5
- Round 2: P_3, P_4, P_5 echo message to all parties

A Simple Simultaneous Broadcast Protocol

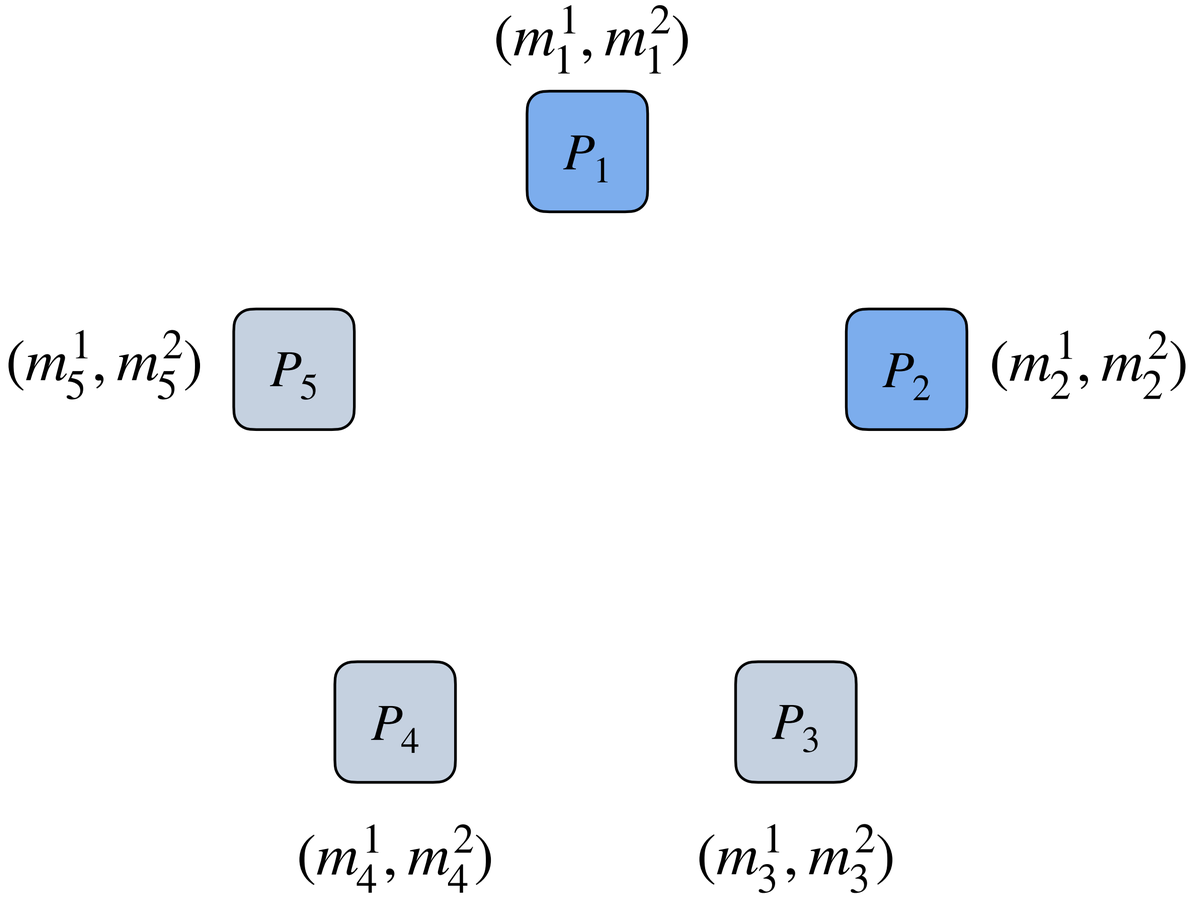
1 corruption, 2 senders, 5 parties



- Round 1: P_1, P_2 send input message to P_3, P_4, P_5
- Round 2: P_3, P_4, P_5 echo message to all parties
- Output: (m_1', m_2') such that at least 2 parties echoed (m_1', m_2')

A Simple Simultaneous Broadcast Protocol

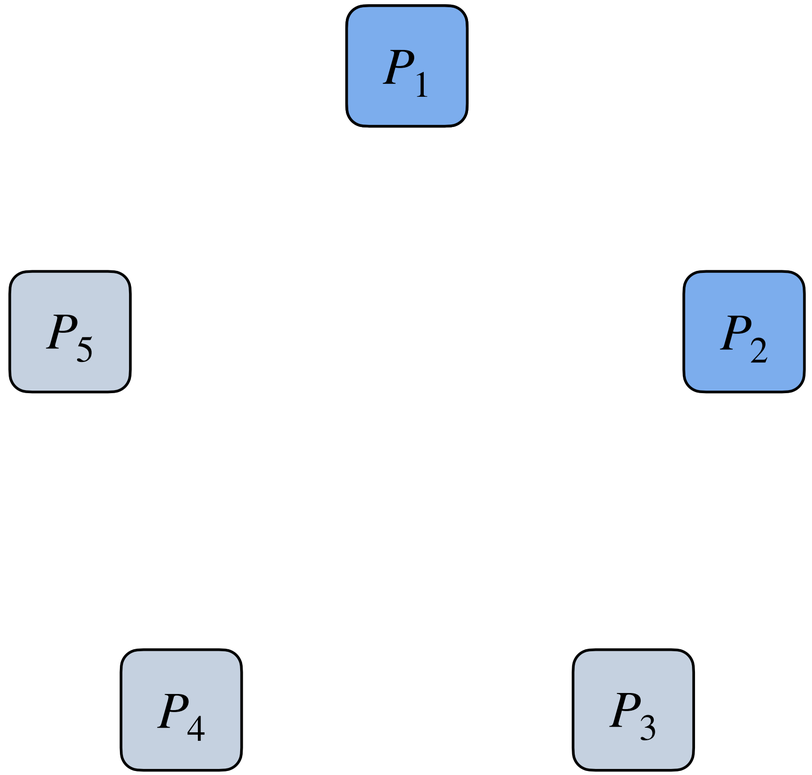
1 corruption, 2 senders, 5 parties



- Round 1: P_1, P_2 send input message to P_3, P_4, P_5
- Round 2: P_3, P_4, P_5 echo message to all parties
- Output: (m_1', m_2') such that at least 2 parties echoed (m_1', m_2')

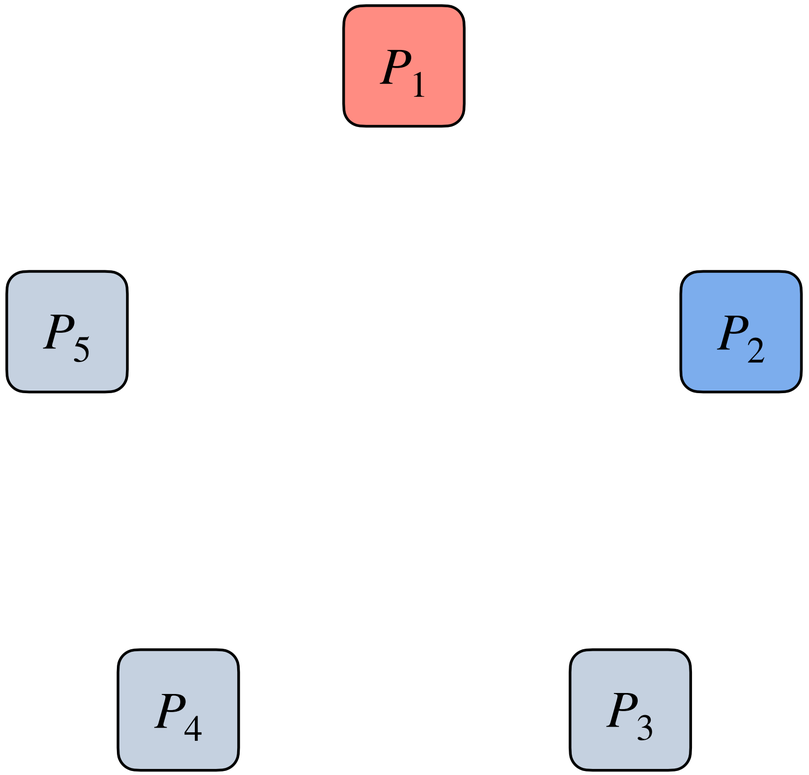
Rushing adversary cannot affect majority decision **and** cannot bias output

A Super-Rushing Attack

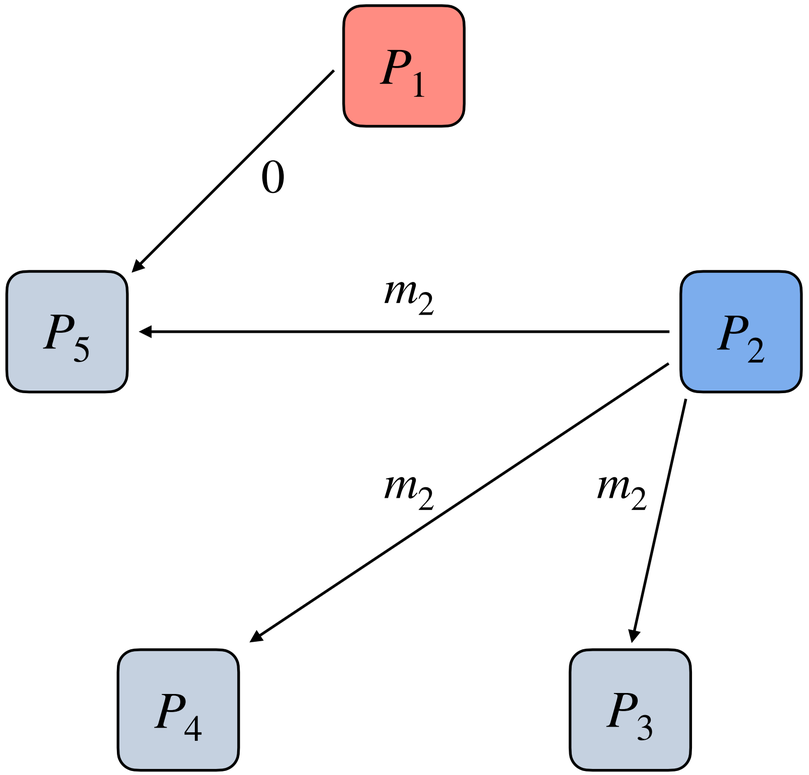


A Super-Rushing Attack

Attack: Corrupted P_1



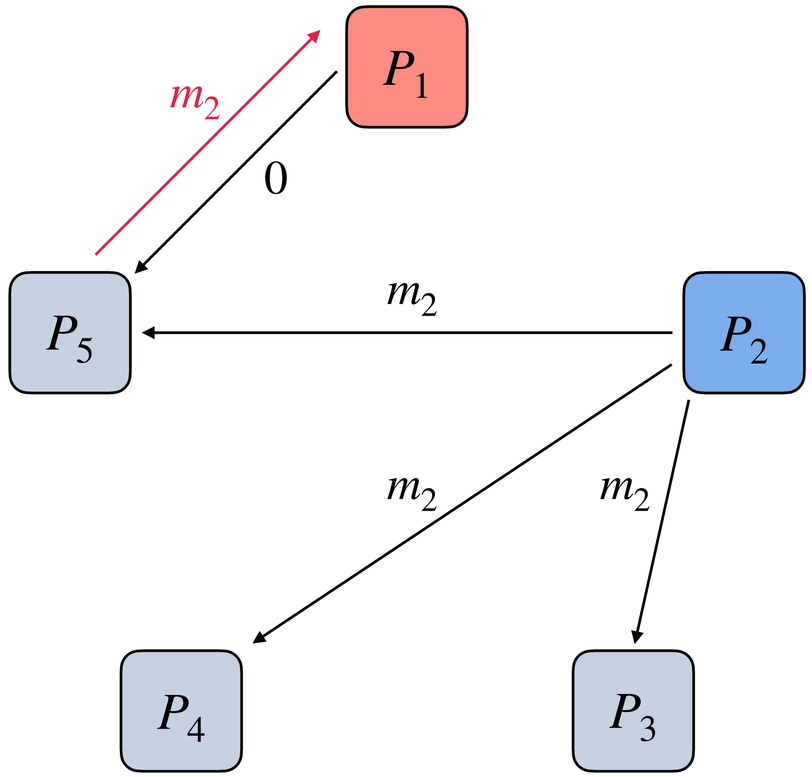
A Super-Rushing Attack



Attack: Corrupted P_1

- Round 1: P_1 sends 0 to only P_5

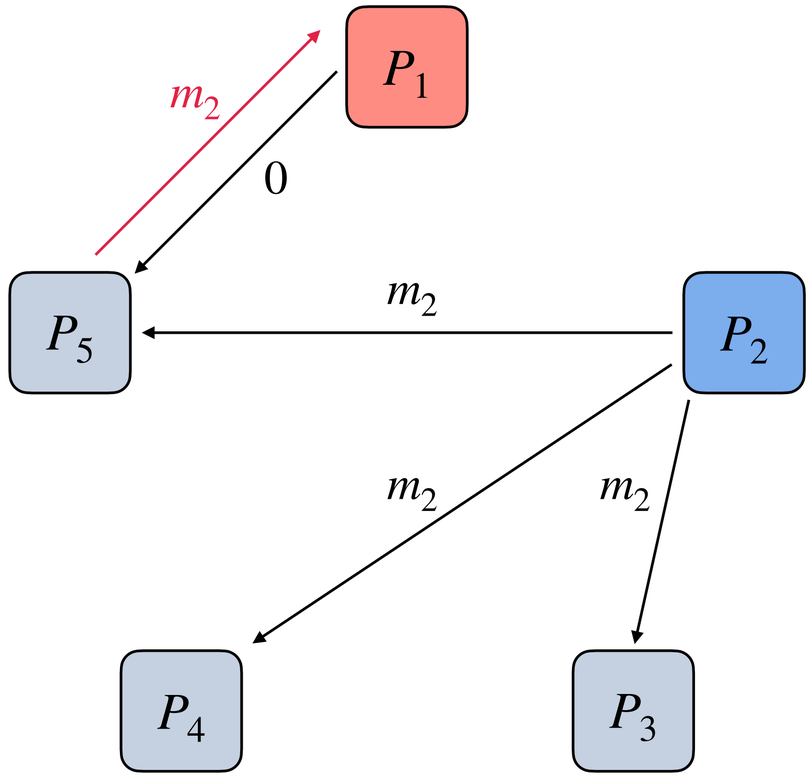
A Super-Rushing Attack



Attack: Corrupted P_1

- Round 1: P_1 sends 0 to only P_5
- Round 2: (only for P_5) Sends $(0, m_2)$ to P_1

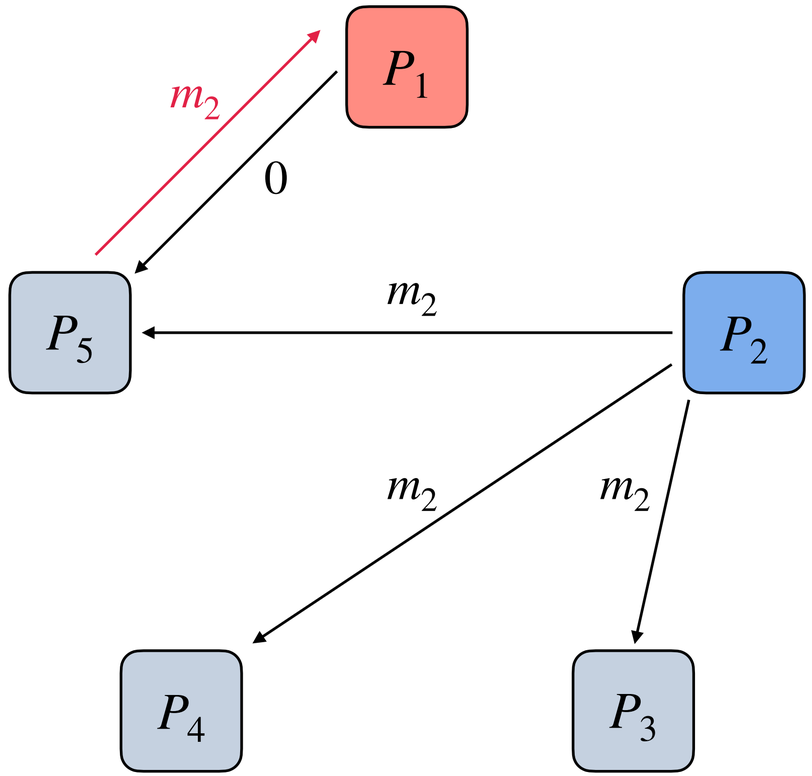
A Super-Rushing Attack



Attack: Corrupted P_1

- Round 1: P_1 sends 0 to only P_5
- Round 2: (only for P_5) Sends $(0, m_2)$ to P_1
- Round 1: P_1 sends $m_1 = m_2$ to P_4, P_5
- Round 2: (everyone) echos the remaining messages
- Output: Everyone outputs (m_2, m_2)

A Super-Rushing Attack



Attack: Corrupted P_1

- Round 1: P_1 sends 0 to only P_5
- Round 2: (only for P_5) Sends $(0, m_2)$ to P_1
- Round 1: P_1 sends $m_1 = m_2$ to P_4, P_5
- Round 2: (everyone) echos the remaining messages
- Output: Everyone outputs (m_2, m_2)

P_3, P_4: Looks like P_5 is cheating

Our Results #1

[CGMA85] (2,5) Simultaneous Broadcast

Theorem: There exists a protocol (with two input providers) that is secure against rushing adversaries but is insecure against super-rushing adversaries

Which synchronous protocols remain secure against super-rushing adversaries?

Modeling Super-Rushing

Extension of Rushing

Modeling Super-Rushing

Extension of Rushing

Rushing

Adversary can schedule messages within a round

Modeling Super-Rushing

Extension of Rushing

Rushing

Adversary can schedule messages within a round

Super-Rushing

+ Can “pull” future messages from honest parties (these parties must have finished all previous rounds!)

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties

P_1, P_2 provide inputs

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties

P_1, P_2 provide inputs

Parties learn the outputs

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties

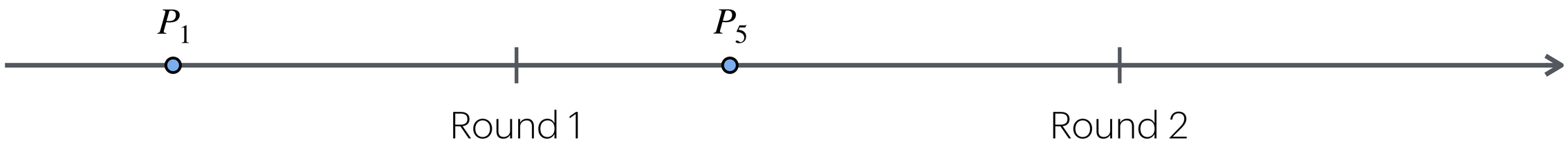
P_1, P_2 provide inputs

Parties learn the outputs

Adversary peeks into round-2 (P_5's round-2 message), learns partial output (P_2's input message)

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties



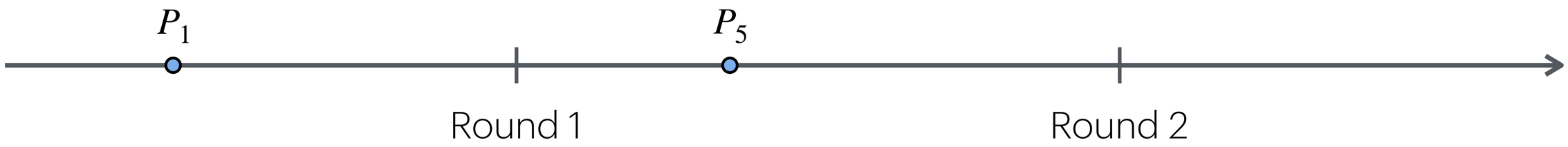
P_1, P_2 provide inputs

Parties learn the outputs

Adversary peeks into round-2 (P_5's round-2 message), learns partial output (P_2's input message)

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties



P_1, P_2 provide inputs

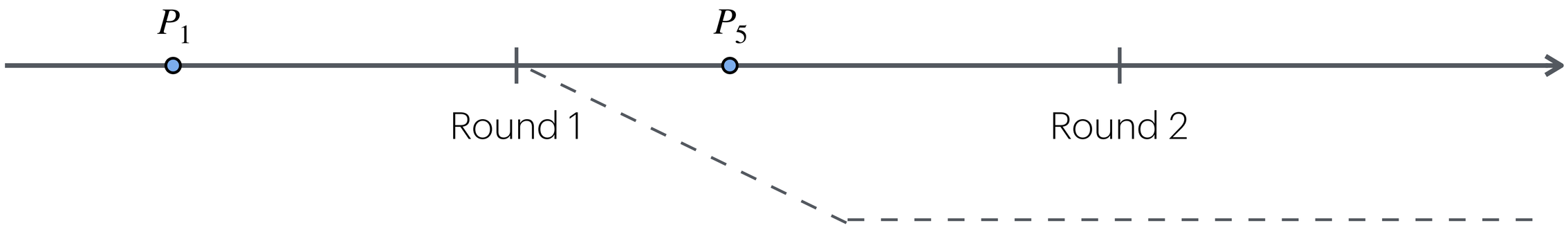
P_1 picks input message as a function of honest party's input (P_2's input message)

Parties learn the outputs

Adversary peeks into round-2 (P_5's round-2 message), learns partial output (P_2's input message)

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties



P_1, P_2 provide inputs

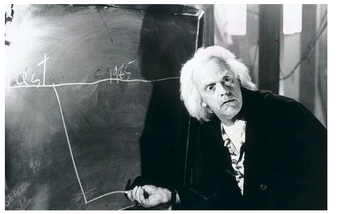
P_1 picks input message as a function of honest party's input (P_2's input message)

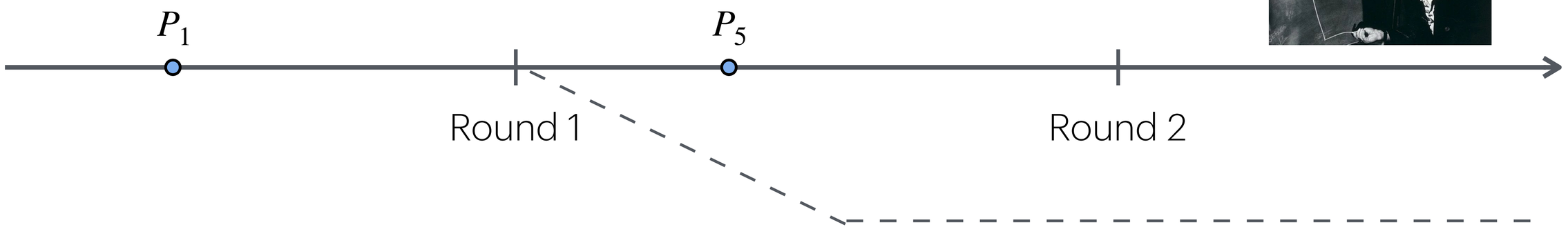
Parties learn the outputs

Adversary peeks into round-2 (P_5's round-2 message), learns partial output (P_2's input message)

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties





P_1, P_2 provide inputs

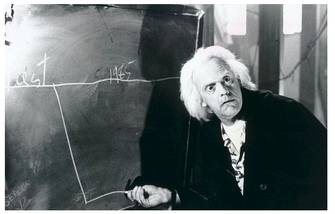
P_1 picks input message as a function of honest party's input (P_2's input message)

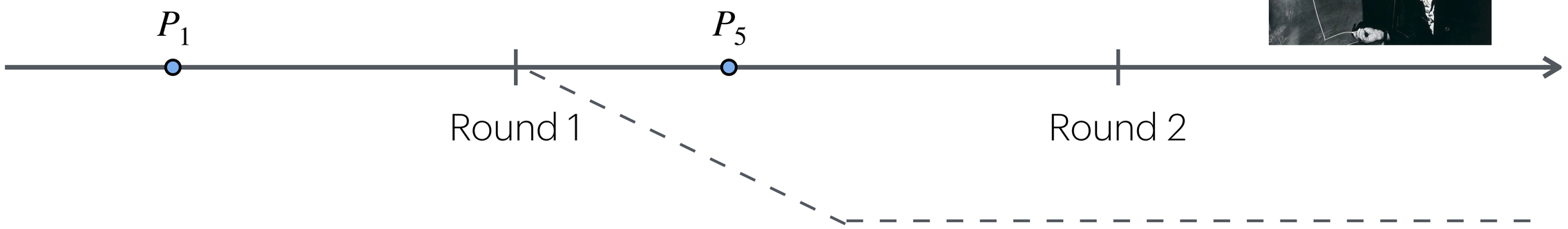
Parties learn the outputs

Adversary peeks into round-2 (P_5's round-2 message), learns partial output (P_2's input message)

Back to the Future: Simultaneous Broadcast

1 corruption, 2 senders, 5 parties





P_1, P_2 provide inputs

P_1 picks input message as a function of honest party's input (P_2's input message)

Parties learn the outputs

Adversary peeks into round-2 (P_5's round-2 message), learns partial output (P_2's input message)

Super-rushing breaks input independence

What if only one party provides input?

VSS, Broadcast, ...

Our Results #2

Super-Rushing ≡ Non-Rushing with a single input provider

Theorem: Every protocol for a single input provider that is secure against non-rushing adversaries is also secure against super-rushing adversaries

Our Results #2

Super-Rushing ≡ Non-Rushing with a single input provider

Theorem: Every protocol for a single input provider that is secure against non-rushing adversaries is also secure against super-rushing adversaries

We worked too hard to show too little!

The Story So Far

Security of protocols against super-rushing

The Story So Far

Security of protocols against super-rushing

✔ Single Input: Super-Rushing ≡ Non-Rushing

The Story So Far

Security of protocols against super-rushing

✔ Single Input: Super-Rushing ≡ Non-Rushing

✘ Two Input Providers: A protocol that is secure against rushing but not against super-rushing

The Story So Far

Security of protocols against super-rushing

✔ Single Input: Super-Rushing ≡ Non-Rushing

✘ Two Input Providers: A protocol that is secure against rushing but not against super-rushing

Does not look like secure computation protocols → no privacy!

The Story So Far

Security of protocols against super-rushing

✔ Single Input: Super-Rushing ≡ Non-Rushing

✘ Two Input Providers: A protocol that is secure against rushing but not against super-rushing

Does not look like secure computation protocols → no privacy!

The Story So Far

Security of protocols against super-rushing

✔ Single Input: Super-Rushing ≡ Non-Rushing

✘ Two Input Providers: A protocol that is secure against rushing but not against super-rushing

Does not look like secure computation protocols → no privacy!

The Story So Far

Security of protocols against super-rushing

✔ Single Input: Super-Rushing ≡ Non-Rushing

✘ Two Input Providers: A protocol that is secure against rushing but not against super-rushing

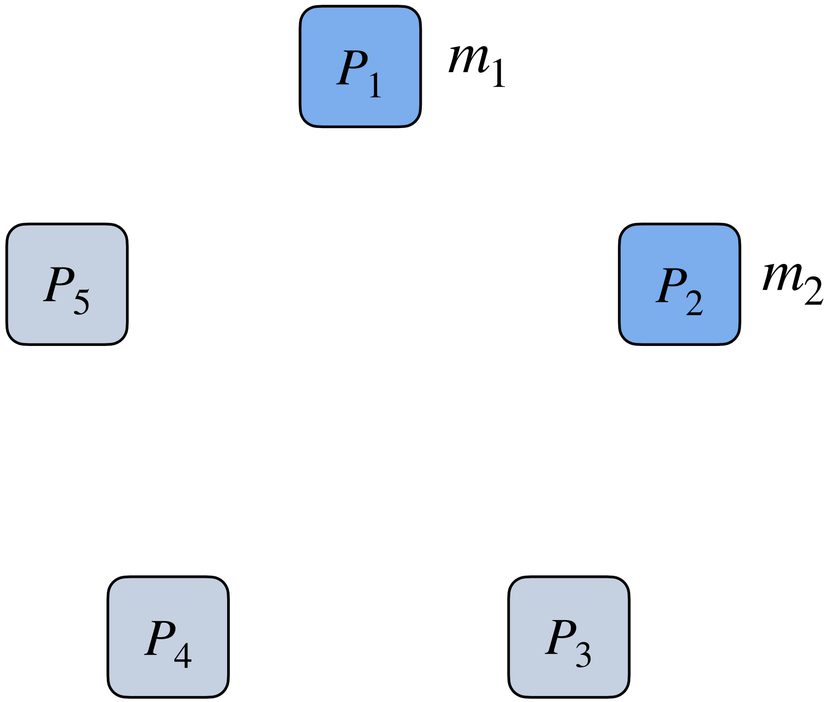
Does not look like secure computation protocols → no privacy!

Simultaneous Broadcast [GIKR02]

1 corruption, 2 senders, 5 parties + Committal round

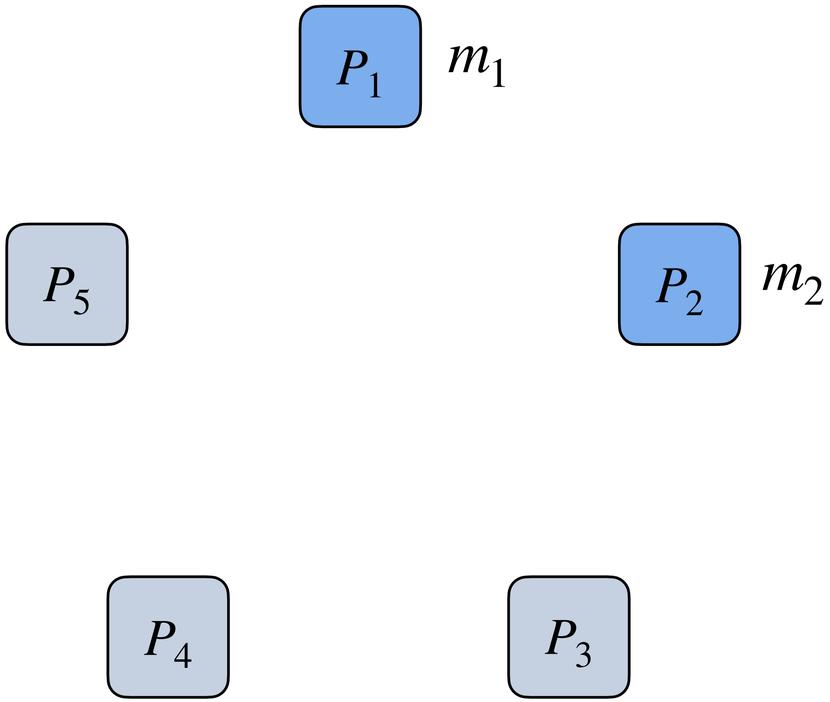
Simultaneous Broadcast [GIKR02]

1 corruption, 2 senders, 5 parties + Committal round



Simultaneous Broadcast [GIKR02]

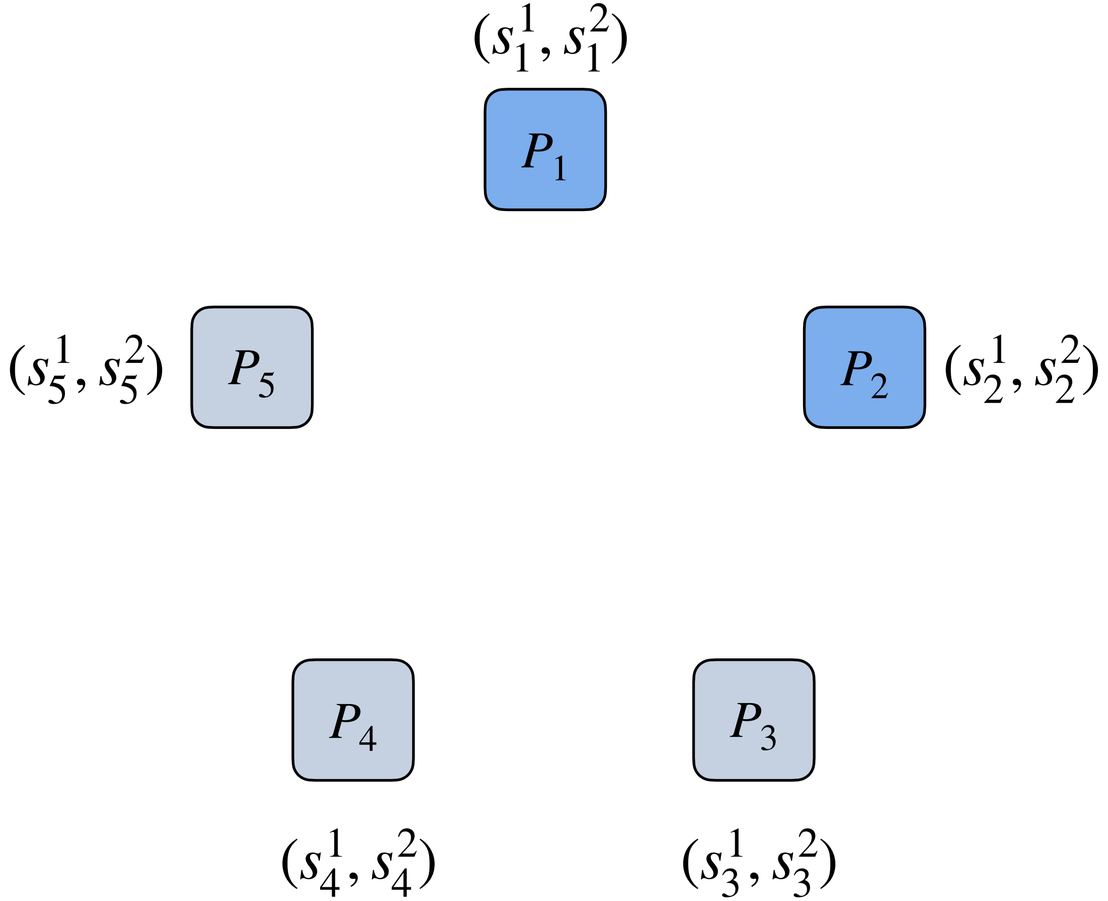
1 corruption, 2 senders, 5 parties + Committal round



- Round 1: P_1, P_2 perform 1-round VSS of their inputs

Simultaneous Broadcast [GIKR02]

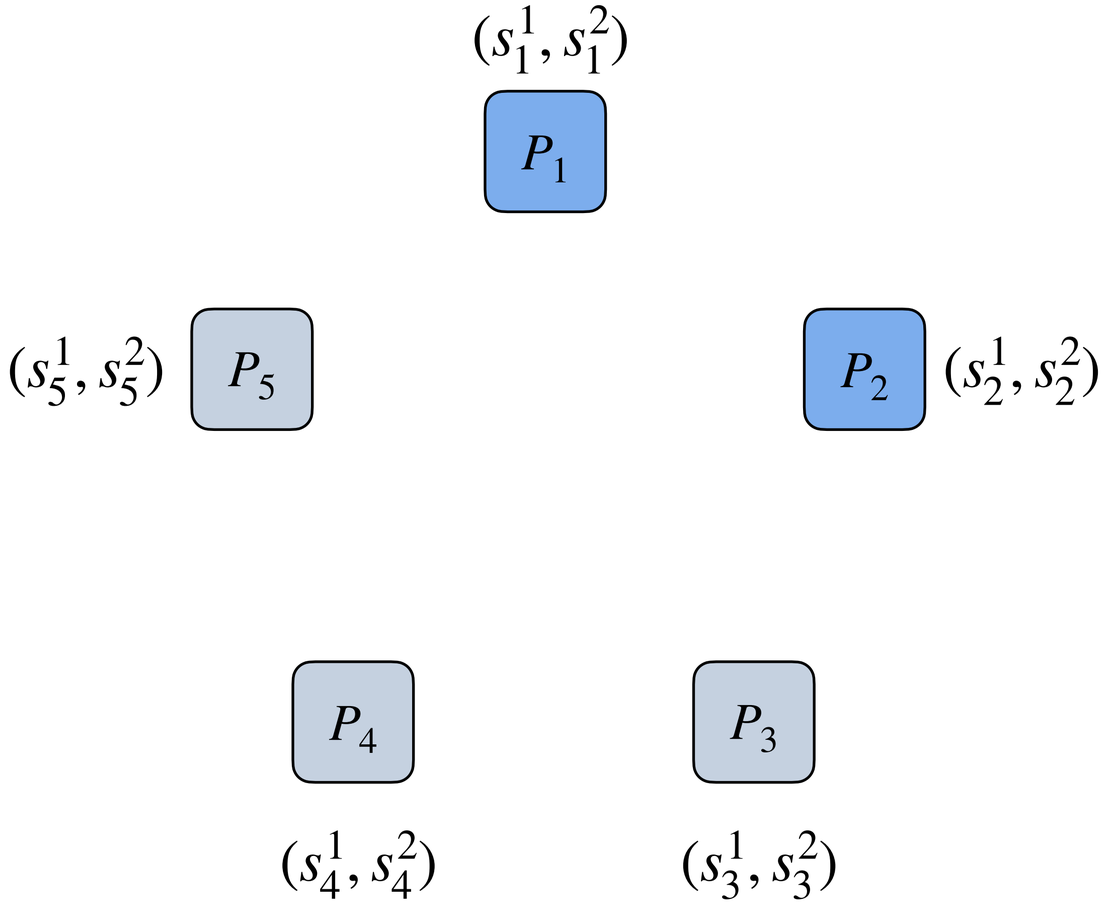
1 corruption, 2 senders, 5 parties + Committal round



- Round 1: P_1, P_2 perform 1-round VSS of their inputs

Simultaneous Broadcast [GIKR02]

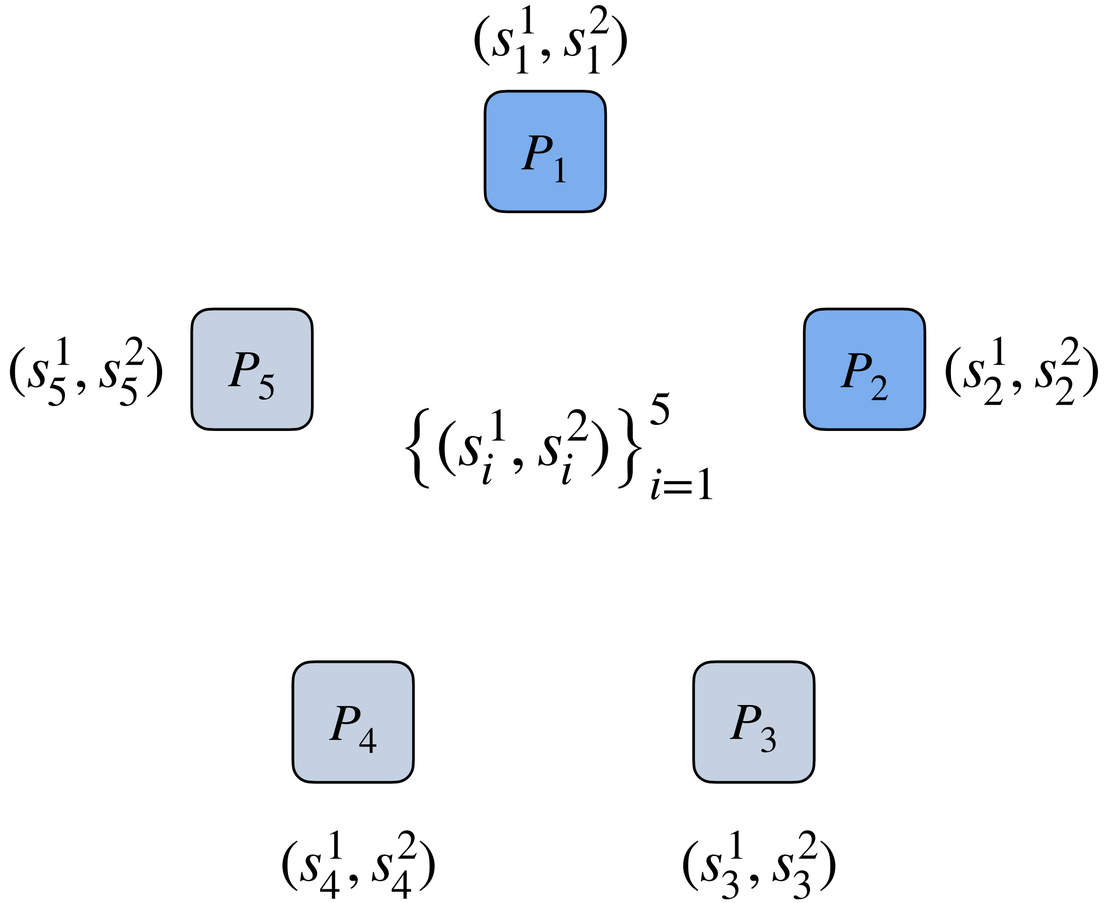
1 corruption, 2 senders, 5 parties + Committal round



- Round 1: P_1, P_2 perform 1-round VSS of their inputs
- Round 2: Parties perform VSS reconstruction

Simultaneous Broadcast [GIKR02]

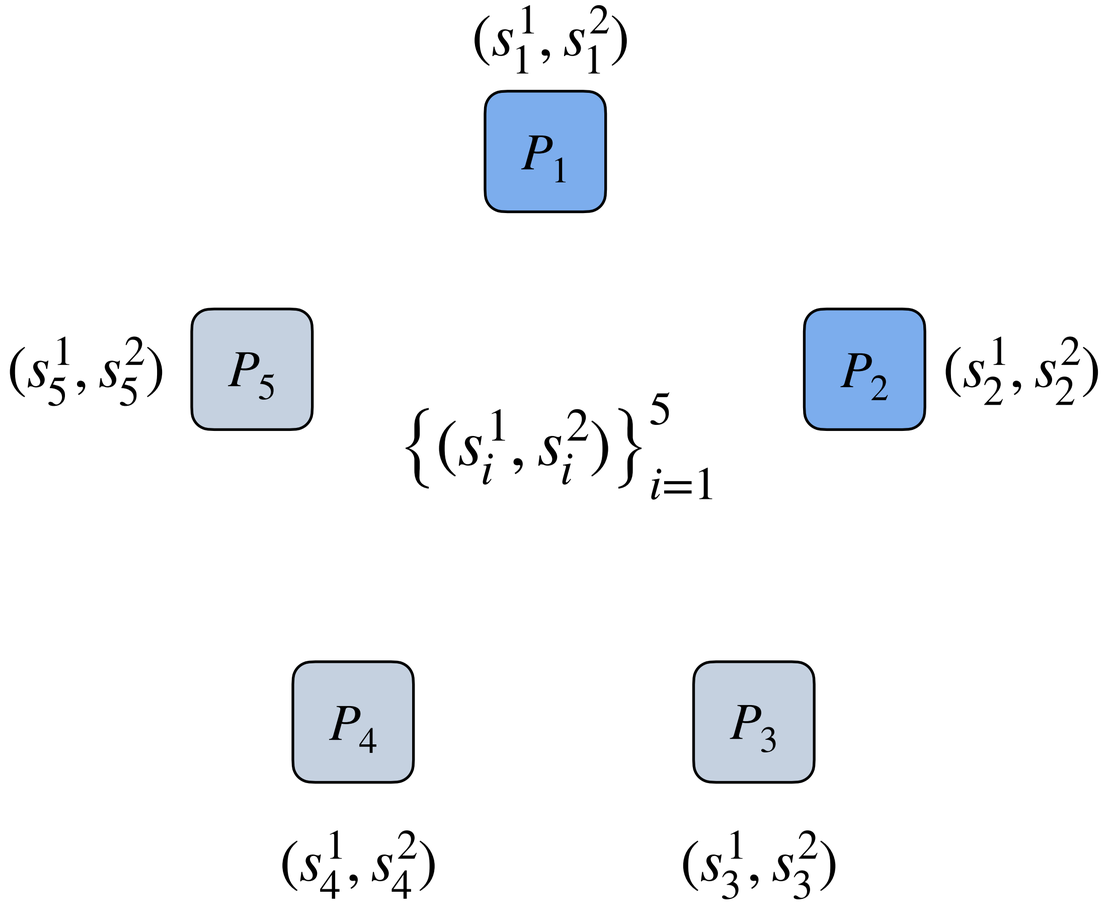
1 corruption, 2 senders, 5 parties + Committal round



- Round 1: P_1, P_2 perform 1-round VSS of their inputs
- Round 2: Parties perform VSS reconstruction

Simultaneous Broadcast [GIKR02]

1 corruption, 2 senders, 5 parties + Committal round



- Round 1: P_1, P_2 perform 1-round VSS of their inputs
- Round 2: Parties perform VSS reconstruction
- Output: (m_1', m_2') from VSS reconstruction.

Super-rushing still breaks independence of inputs

What are the sufficient conditions for security against super-rushing adversaries?

Our Sufficient Conditions

All-to-all communication $\Longrightarrow$ Peeking up to 1 round

Our Sufficient Conditions

All-to-all communication $\Longrightarrow$ Peeking up to 1 round

Our Sufficient Conditions

All-to-all communication $\Longrightarrow$ Peeking up to 1 round

Round CR

Inputs are committed

Our Sufficient Conditions

All-to-all communication $\Longrightarrow$ Peeking up to 1 round

Inputs are committed

Our Sufficient Conditions

All-to-all communication $\Longrightarrow$ Peeking up to 1 round

Our Sufficient Conditions

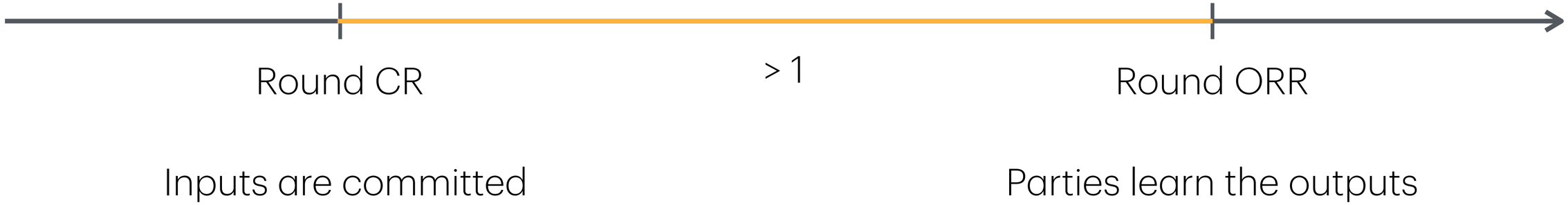
All-to-all communication $\Longrightarrow$ Peeking up to 1 round

Inputs are committed

Parties learn the outputs

Our Sufficient Conditions

All-to-all communication $\Longrightarrow$ Peeking up to 1 round



Mitigates Super-Rushing

Peeking into round ORR only possible after all parties complete round CR

Our Results #3

General Sufficient Conditions

Theorem: Every protocol that is

1. secure against rushing adversaries*
2. has all-to-all communication
3. ORR > CR+1

is also secure against super-rushing adversaries

Our Results #3

General Sufficient Conditions

Theorem: Every protocol that is

1. secure against rushing adversaries*
2. has all-to-all communication
3. ORR > CR+1

is also secure against super-rushing adversaries

* security is assumed via compatible simulation (see our paper)

Extensions

What if ORR = CR + 1

Theorem: Every protocol that is

1. secure against rushing adversaries*
2. has all-to-all communication
3. CR is over broadcast or is a synchronization round

is also secure against super-rushing adversaries

* security is assumed via compatible simulation (see our paper)

Our Results #4

Sequential Composition

Theorem: There exists a protocol (with one input provider) that is secure against rushing adversaries but is insecure against super-rushing adversaries when sequentially composed even once

Our Results #4

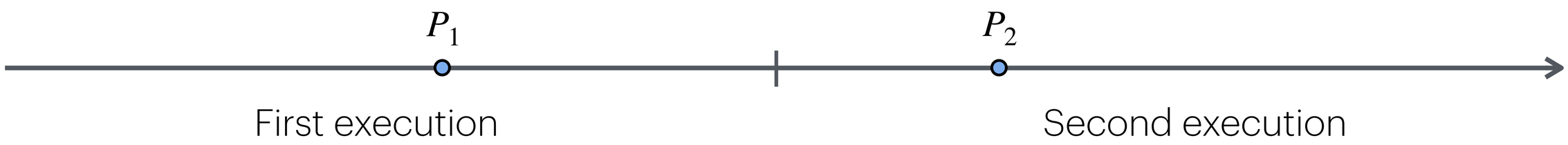
Sequential Composition

Theorem: There exists a protocol (with one input provider) that is secure against rushing adversaries but is insecure against super-rushing adversaries when sequentially composed even once

Our Results #4

Sequential Composition

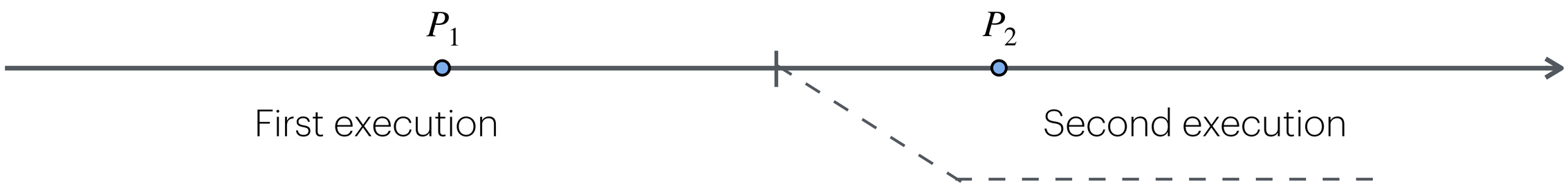
Theorem: There exists a protocol (with one input provider) that is secure against rushing adversaries but is insecure against super-rushing adversaries when sequentially composed even once



Our Results #4

Sequential Composition

Theorem: There exists a protocol (with one input provider) that is secure against rushing adversaries but is insecure against super-rushing adversaries when sequentially composed even once



Our Results

Perfect Security

✔ Single Input: Super-Rushing ≡ Non-Rushing

- Two Input Providers:
  - ✘ secure against rushing but not against super-rushing
  - ✘ committal round does not help

✘ Sequential composition of single input protocols is not secure

✔ Main Result: Rushing $\Longrightarrow$ Super-Rushing for protocols with all-to-all communication and ORR > CR + 1

✔ Extensions when ORR = CR + 1

BGW is secure against super-rushing attacks

No need to reprove security

An Alternate Strategy

An Alternate Strategy

Asynchrony admits Super-Rushing Attacks

An Alternate Strategy

Asynchrony admits Super-Rushing Attacks

- Asynchronous MPC with guaranteed output delivery inevitably ignores some honest parties' inputs

An Alternate Strategy

Asynchrony admits Super-Rushing Attacks

- Asynchronous MPC with guaranteed output delivery inevitably ignores some honest parties' inputs
- Universally composable protocols [Canetti01] typically settle for security with abort

Alternate Sufficient Conditions

Kushilevitz, Lindell, Rabin, STOC '06

Theorem: Every protocol that is

1. secure against rushing adversaries; and
2. has synchronization steps,

is also UC-secure [Canetti01]

Alternate Sufficient Conditions

Kushilevitz, Lindell, Rabin, STOC '06

Theorem: Every protocol that is

1. secure against rushing adversaries; and
2. has synchronization steps,

is also UC-secure [Canetti01]

$\times 2$ round complexity and $+O(n^2)$ communication complexity

Statistical Security

We require different sufficient conditions

Theorem: There exists a protocol that is

1. secure against rushing adversaries (with statistical security)
2. has all-to-all communication
3. ORR > CR+1

but is insecure against super-rushing adversaries

Conclusion

Conclusion

- Back to the Future attack is real on some implementations

Conclusion

- Back to the Future attack is real on some implementations

Conclusion

- Back to the Future attack is real on some implementations

2015

Biff steals the almanac

Conclusion

- Back to the Future attack is real on some implementations

1955

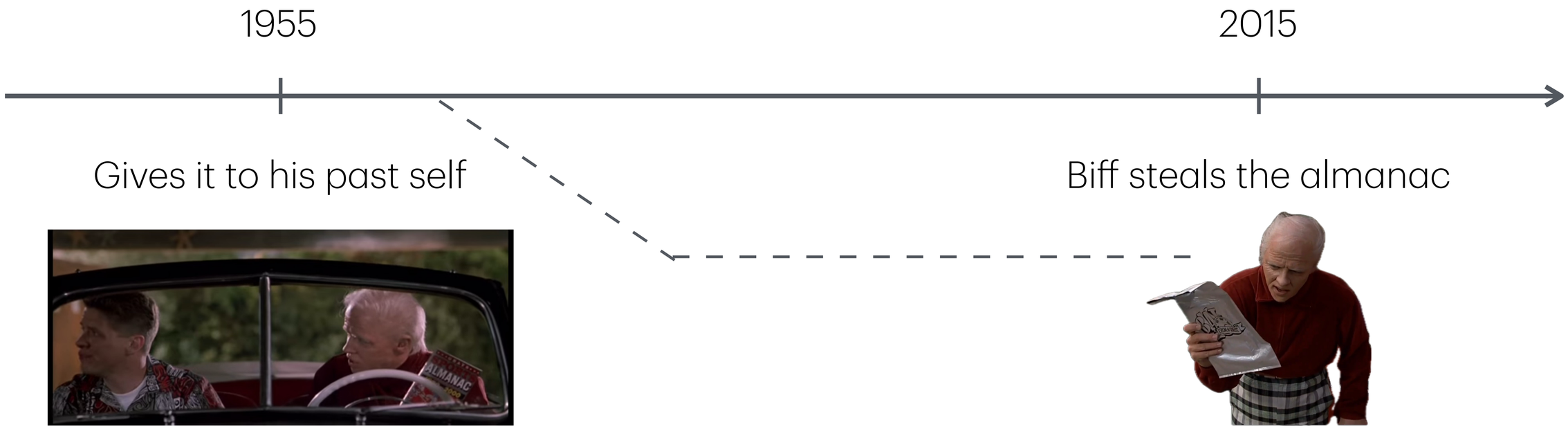
Gives it to his past self

2015

Biff steals the almanac

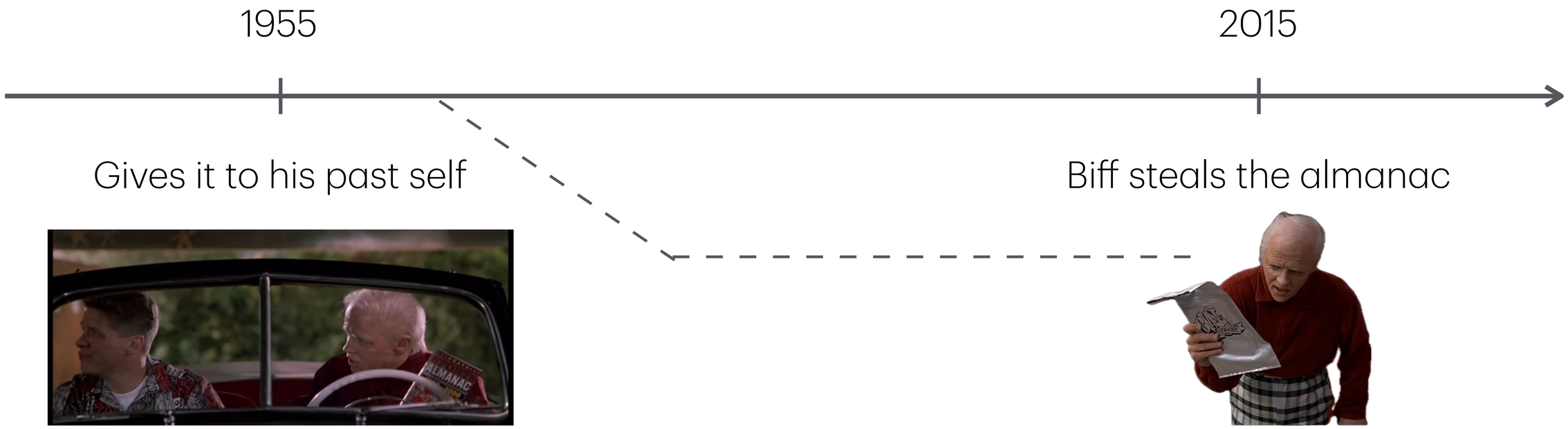
Conclusion

- Back to the Future attack is real on some implementations



Conclusion

- Back to the Future attack is real on some implementations
- ORR > CR + 1 sufficient for Rushing $\Longrightarrow$ Super-Rushing



Conclusion

- Back to the Future attack is real on some implementations
- ORR > CR + 1 sufficient for Rushing $\Longrightarrow$ Super-Rushing

1955

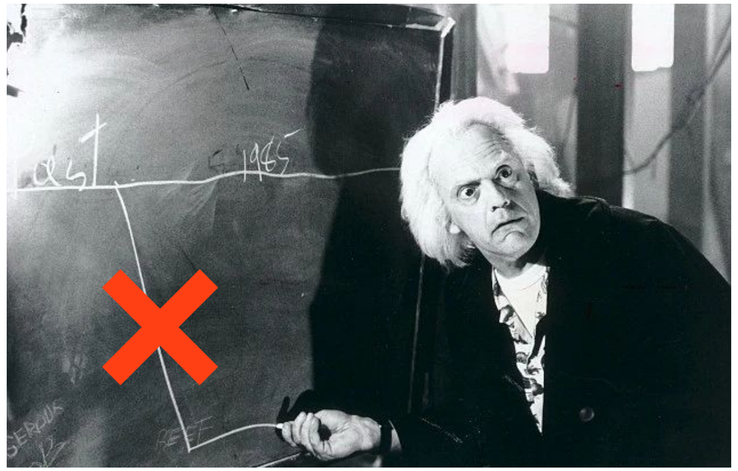
Gives it to his past self

2015

Biff steals the almanac

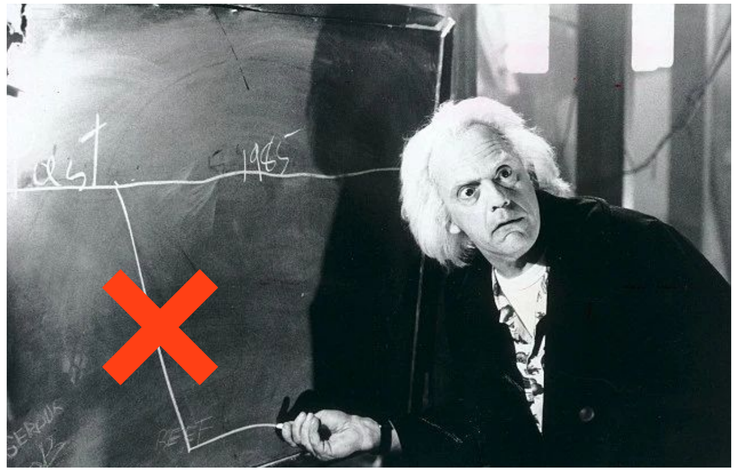
Conclusion

- Back to the Future attack is real on some implementations
- ORR > CR + 1 sufficient for Rushing $\Longrightarrow$ Super-Rushing



Conclusion

- Back to the Future attack is real on some implementations
- ORR > CR + 1 sufficient for Rushing $\Longrightarrow$ Super-Rushing



Thank you