
On the Message Complexity of Indulgent Consensus

Seth Gilbert, Rachid Guerraoui, Dariusz Kowalski

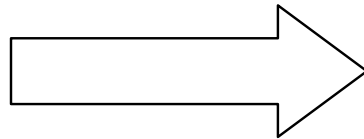
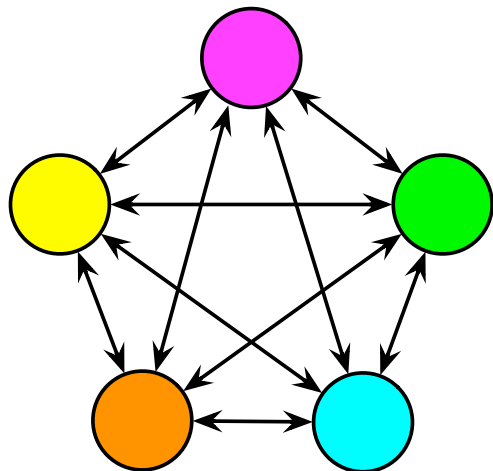
September 24
DISC 2007

Consensus

Fundamental Agreement Problem

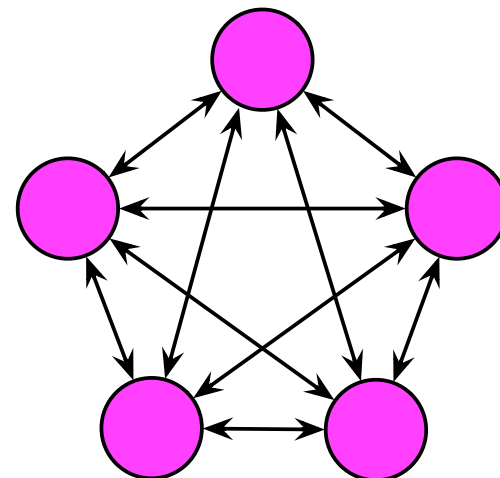
Basic Situation:

- n nodes $\{p_1, \dots, p_n\}$
- Initial values $\{v_1, \dots, v_n\}$



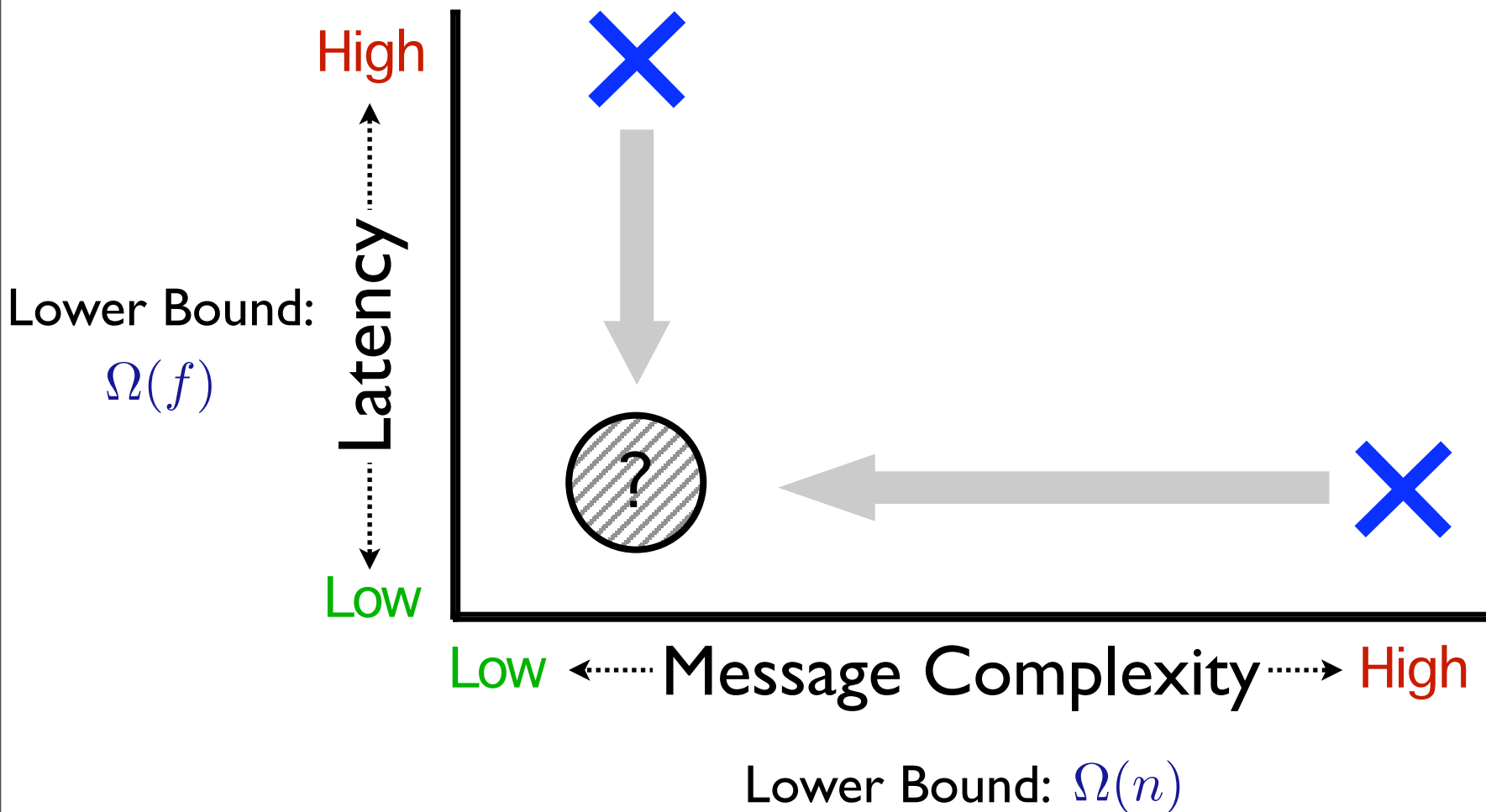
Three properties:

1. Agreement
2. Validity
3. Termination



Trade-Offs

Latency vs. Message Complexity



Trade-Offs

Synchrony vs. Asynchrony

Most real systems are synchronous...
...except when they are not.

Indulgent Algorithms:

- * Efficient when **synchronous**
- * Correct when **asynchronous**

Results

Message-Efficient Indulgent Consensus

	Round Complexity	Message Complexity
Algorithm 1 (Message Optimal)	$O(n^{1+\epsilon})$	$O(n)$
Algorithm 2 (Round Optimal)	$O(f)$	$O(n \log^6 n)$

** In any *synchronous* execution.

Basic Strategy

Message-Efficient Transformation

Synchronous



Eventually
Synchronous

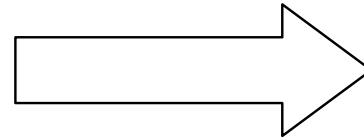
Basic Strategy

Message-Efficient Transformation

Synchronous

I. [Galil, Mayer, Yung]

- Messages: $O(n^{1+\epsilon})$
- Rounds: $O(n)$



Message-
Efficient

Eventually
Synchronous

Algorithm I

- Message Optimal

Basic Strategy

Message-Efficient Transformation

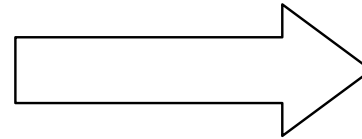
Synchronous

1. [Galil, Mayer, Yung]

- Messages: $O(n^{1+\epsilon})$
- Rounds: $O(n)$

2. [Chlebus, Kowalski]

- Adaptive
- Messages: $O(n \log^6 n)$
- Rounds: $O(f)$



Eventually Synchronous

Algorithm 1

- Message Optimal

Algorithm 2

- Round Optimal

Message-
Efficient

Message-
Efficient

Preserves
Early-Decision
Property

Basic Strategy

Outline

Basic Strategy

Outline

- I. Simulate synchronous rounds.

Basic Strategy

Outline

1. Simulate synchronous rounds.
2. In each round, simulate one round of synchronous consensus.

Basic Strategy

Outline

1. Simulate synchronous rounds.
2. In each round, simulate one round of synchronous consensus.
3. When the consensus protocol is complete, check if it succeeded.

Basic Strategy

Outline

1. Simulate synchronous rounds.
2. In each round, simulate one round of synchronous consensus.
3. When the consensus protocol is complete, check if it succeeded.
 - a. If it succeeded, then decide and terminate.

Basic Strategy

Outline

1. Simulate synchronous rounds.
2. In each round, simulate one round of synchronous consensus.
3. When the consensus protocol is complete, check if it succeeded.
 - a. If it succeeded, then decide and terminate.
 - b. Otherwise, run a (less efficient) fall-back consensus protocol.

Model

Eventually Synchronous System

- n processes.
- $< n/2$ failures.
- Point-to-point reliable communication.
- Eventual bound on message delay: d
 - Eventually, every message delivered with d time.
- Eventual bound on clock skew: δ
 - In synchronous executions, at time t , every clock is within:
 $[(1 - \delta)t, (1 + \delta)t]$

Building Blocks

Summary

- Message-efficient Synchronous Protocols
 - Synchronous Consensus
 - Synchronous Conditional Gossip
 - Synchronous Conditional Wake-Up
- Inefficient Partially Synchronous Protocols
 - Indulgent Consensus

Building Blocks

Consensus

- Properties:
 - Agreement
 - (Unconditional) Validity
 - Termination
- 1. Synchronous [Galil, Mayer, Yung],[Chlebus, Kowalski]
 - Message efficient
 - Adaptive (early deciding)
- 2. Eventually Synchronous (e.g., Paxos [Lamport])
 - Less efficient

Building Blocks

Synchronous Protocols

- Conditional Gossip(f)
 - Every process begins with a rumor.
 - If there are $\leq f$ failures, then every correct process learns every rumor.
 - If there are $> f$ failures, then no guarantee.
 - Message efficient.
 - Adaptive: round complexity depends on f .

Building Blocks

Synchronous Protocols

- Conditional WakeUp(f)
 - Some processes begin awake.
 - Some processes begin asleep.
 - If there are $\leq f$ failures and some process is awake, then every correct process wakes up.
 - If every process is asleep, then every process remains asleep and no messages are sent.
 - Message efficient.
 - Adaptive: round complexity depends on f .

Building Blocks

Summary

- Message-efficient Synchronous Protocols
 - Consensus
 - Conditional Gossip(f)
 - Conditional Wake-Up(f)
- Inefficient Partially Synchronous Protocols
 - Consensus

Building Blocks

Summary

- Message-efficient Synchronous Protocols

- Consensus

- Conditional Gossip(f)

- Conditional Wake-Up(f)

	Messages	Rounds
1.	$O(n)$	$O(n^{1+\epsilon})$
2.	$O(n \log^4 n)$	$O(f)$

- Inefficient Partially Synchronous Protocols

- Consensus

Algorithm I

Outline

Four phases:

1. Agreement
2. Locking
3. Decision
4. Fall-back to inefficient consensus.

Algorithm I

Outline

Four phases:

1. **Agreement**

- Simulate synchronous consensus.
- If simulation decides, store output as a *candidate*.

2. **Locking**

3. **Decision**

4. **Fall-back to inefficient consensus.**

Algorithm I

Outline

Four phases:

1. Agreement
2. Locking
 - Simulate synchronous gossip.
 - If a majority have the same *candidate*, then lock that value.
3. Decision
4. Fall-back to inefficient consensus.

Claim I: At most one value is locked.

Algorithm I

Outline

Four phases:

1. Agreement
2. Locking
3. Decision
 - Simulate synchronous gossip (again).
 - If a majority locked the same candidate, then decide that value and terminate.
4. Fall-back to inefficient consensus.

Algorithm I

Outline

Four phases:

1. Agreement
2. Locking
3. Decision
4. Fall-back to inefficient consensus.
 - Send *fall-back* message to every process.
 - Collect all locked values from each process.
 - If any value is locked, then adopt that value.
 - Run fall-back consensus.

Claim 2: At most one value is decided.

Algorithm I

Outline

Four phases:

1. Agreement
2. Locking
3. Decision
4. Fall-back to inefficient consensus.

Claim 3: In every synchronous execution, every correct process decides by the end of phase 3.

Algorithm I

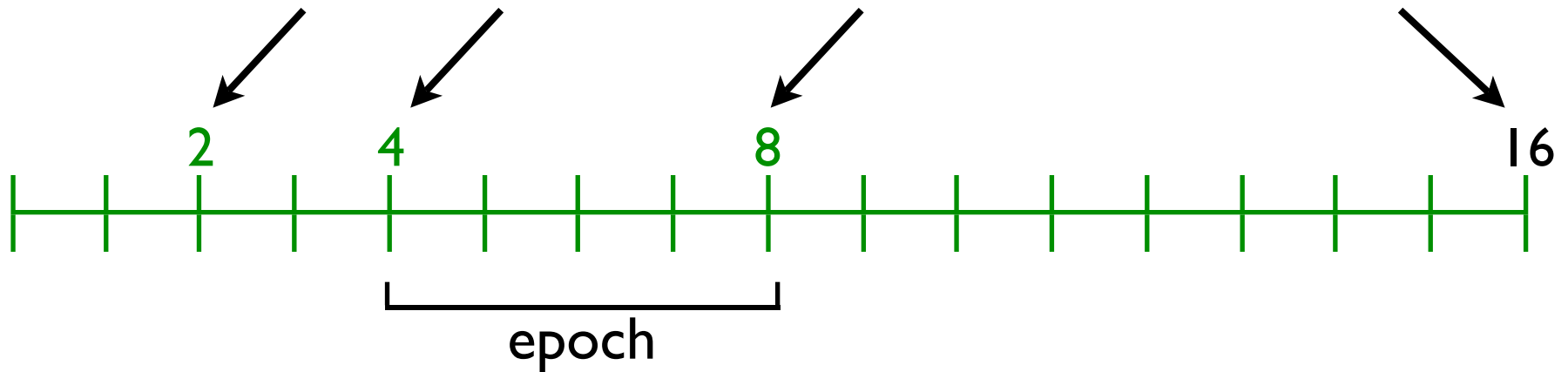
Summary

- Satisfies agreement, validity, termination.
- Efficient in synchronous executions:
 - Phase 1: Consensus
 - Phase 2: Gossip
 - Phase 3: Gossip
- Not adaptive:
 - Does not detect early decision.

Algorithm 2

Summary

- Check for early decision of consensus:



- Not too often..
- In each epoch, simulate rounds of consensus.

Algorithm 2

Outline

Epoch i :

1. Wake-Up
2. **Agreement**: simulate $O(2^i)$ rounds of consensus.
3. **Locking**: simulate $O(2^i)$ rounds of gossip.
 - Conditional: tolerates 2^i failures.
4. **Decision**: simulate $O(2^i)$ rounds of gossip.
 - Conditional: tolerates 2^i failures.

If all epochs complete with no decision:

Fall-back to inefficient consensus.

Algorithm 2

Outline

Epoch i :

I. Wake-Up

At the beginning of each epoch:

- Some have decided and are asleep.
- Some have not decided and are awake.

Wake-Up Protocol ensures:

- If all have decided, no messages sent.
- If any have not decided, all wake up.

If all epochs complete with no decision:

Fall-back to inefficient consensus.

Algorithm 2

Outline

Epoch i :

1. **Wake-Up**
2. **Agreement**: simulate $O(2^i)$ rounds of consensus.
3. **Locking**: simulate $O(2^i)$ rounds of gossip.
 - Conditional: tolerates 2^i failures.
4. **Decision**: simulate $O(2^i)$ rounds of gossip.
 - Conditional: tolerates 2^i failures.

If all epochs complete with no decision:

Fall-back to inefficient consensus.

Omitted Details

In Brief

- Analysis of Algorithm 2.
 - Somewhat more involved, but essentially the same as Algorithm 1.
- Implementing synchronous building blocks
 - Derived from prior research, particularly:
 - [Chlebus, Kowalski]
 - [Galil, Mayer, Yung]

Results

Message-Efficient Indulgent Consensus

via efficient transformations from synchronous algorithms.

	Round Complexity	Message Complexity
Algorithm 1 (Message Optimal)	$O(n^{1+\epsilon})$	$O(n)$
Algorithm 2 (Round Optimal)	$O(f)$	$O(n \log^6 n)$

** In any synchronous execution.

