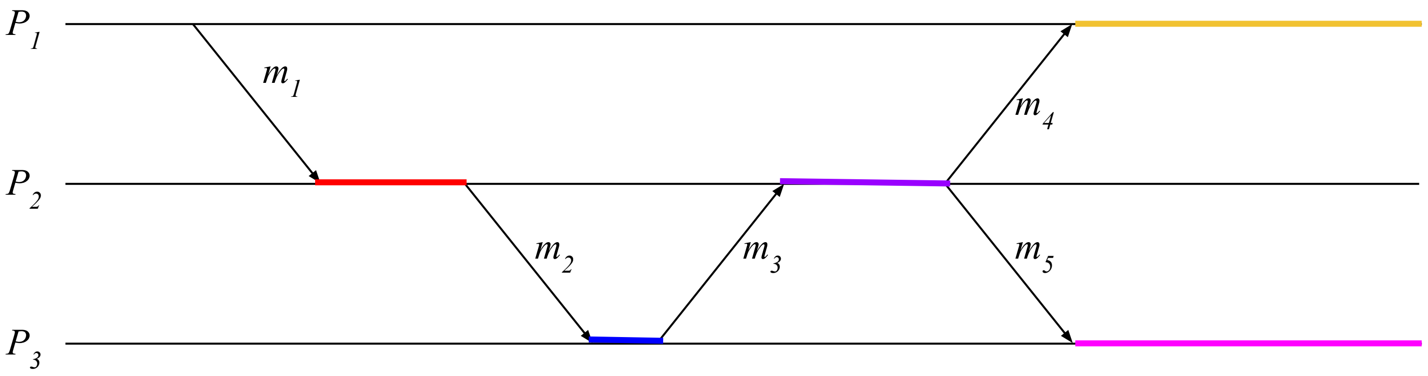
Specification and Verification of Multi-Paxos

Saksham Chand,
Stony Brook University
11/15/2019

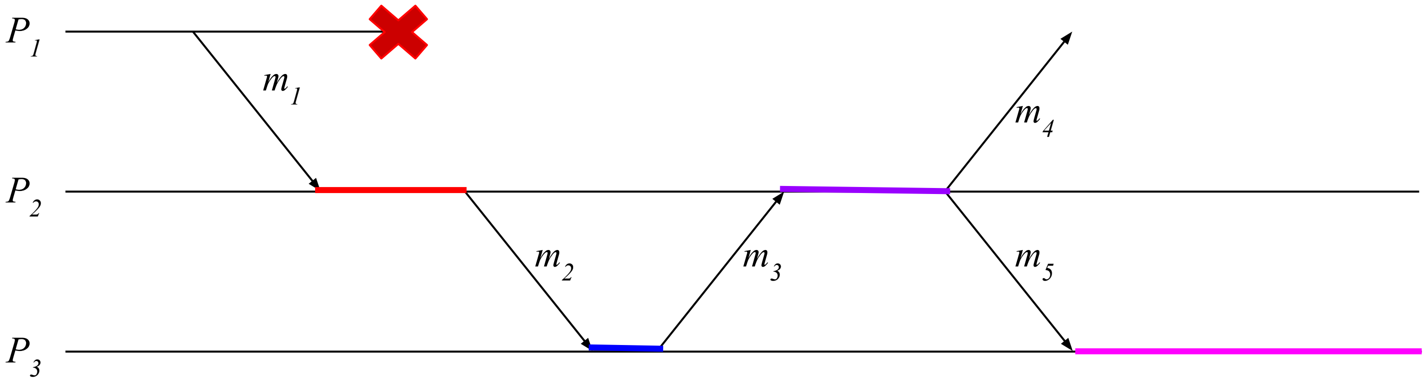
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*



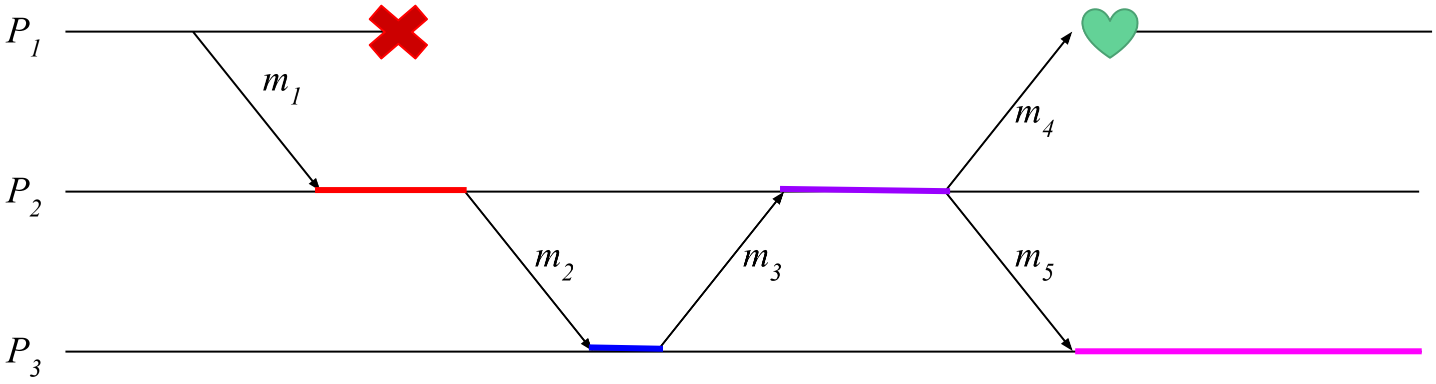
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*
- Processes may crash



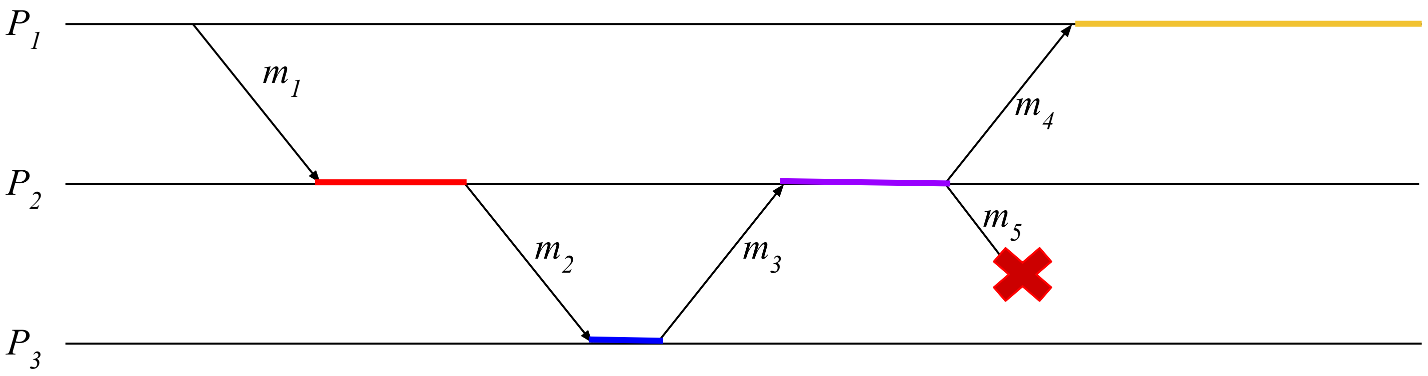
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*
- Processes may crash and later recover



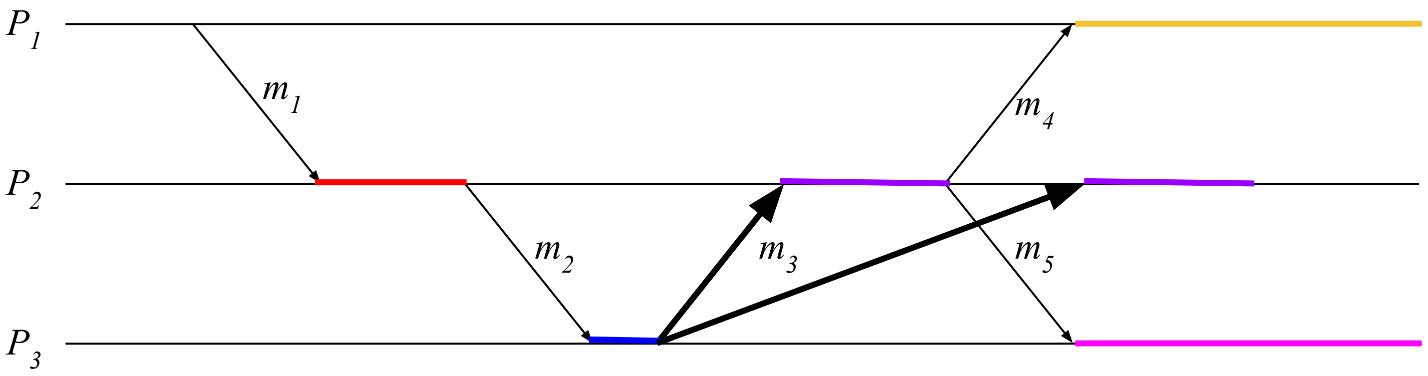
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*
- Processes may crash and later recover
- Messages may get lost



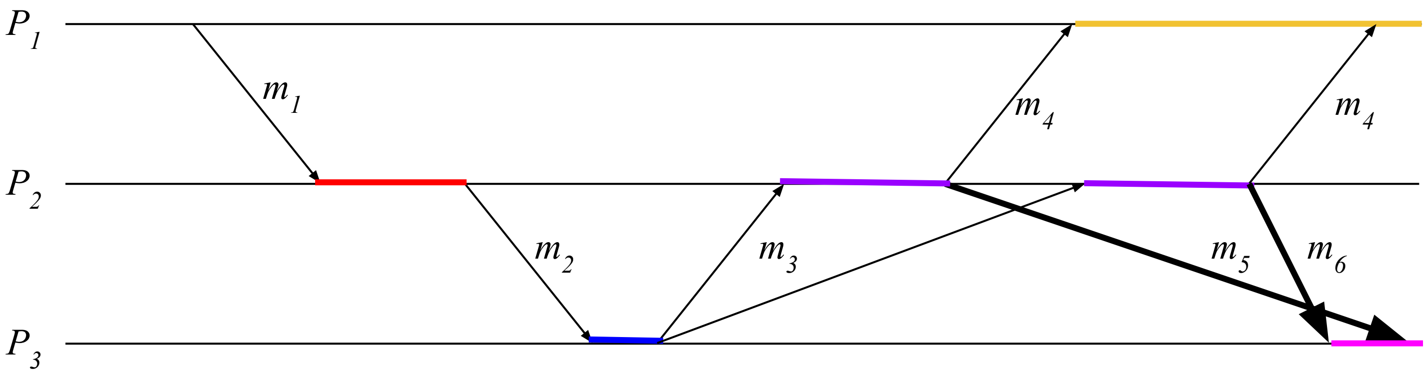
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*
- Processes may crash and later recover
- Messages may get lost, duplicated



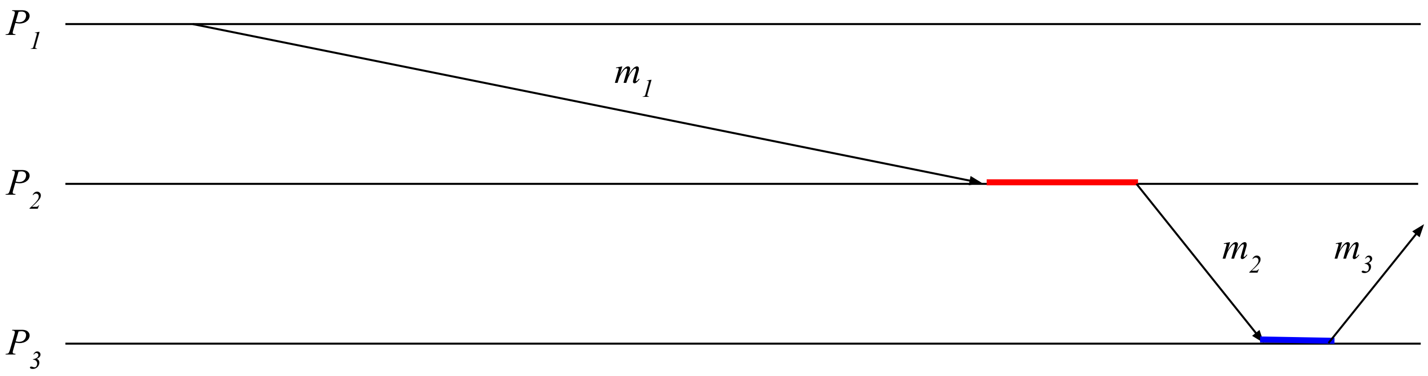
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*
- Processes may crash and later recover
- Messages may get lost, duplicated, received out-of-order



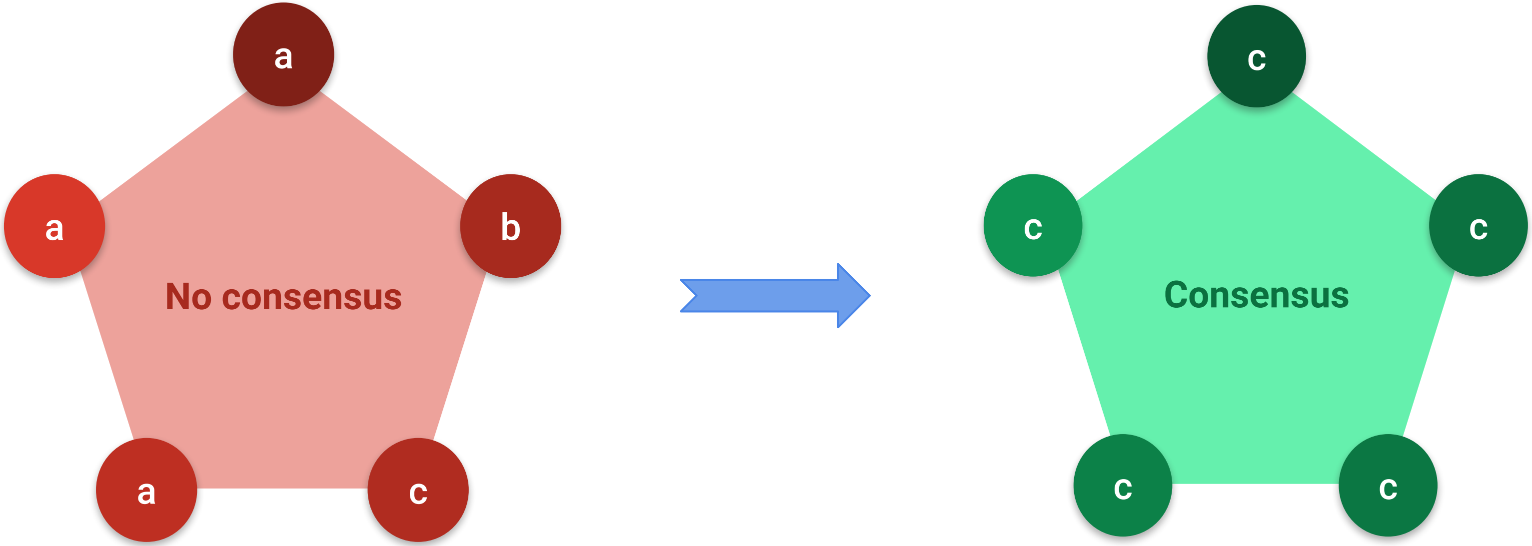
System

- **Distributed system** = Processes + Communication Channel
- Processes communicate by *message-passing*
- Processes may crash and later recover
- Messages may get lost, duplicated, received out-of-order, or delayed indefinitely



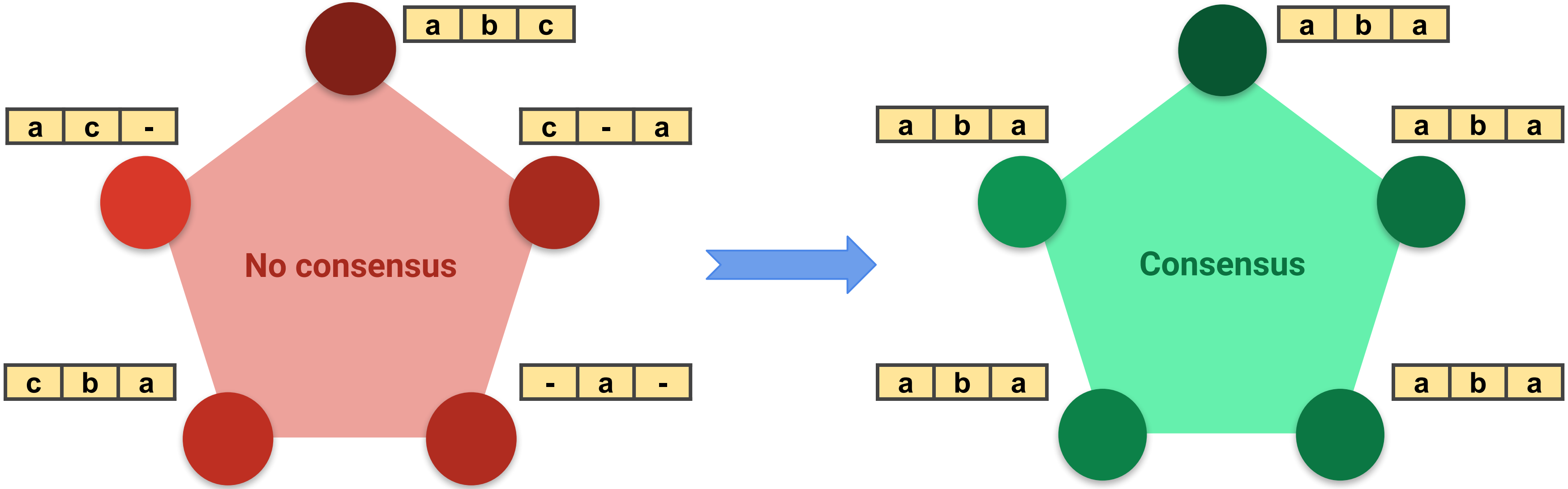
The Problem of Consensus

- **Single-value Consensus:** Processes need to agree on a single value



The Problem of Consensus

- **Multi-value Consensus:** Processes need to agree on a sequence of values



Properties for consensus algorithms†

- **Safety**: At most a single value is chosen

$$\forall v_1, v_2 \in Values: Chosen(v_1) \land Chosen(v_2) \Rightarrow v_1 = v_2$$

- **Liveness**: Under some stable conditions, a value is eventually chosen.
  - [FLP]* result states that in general consensus cannot be solved in an asynchronous system where even one process can fail. Thus, liveness is subject to some stable conditions.

† Presented for single-value consensus

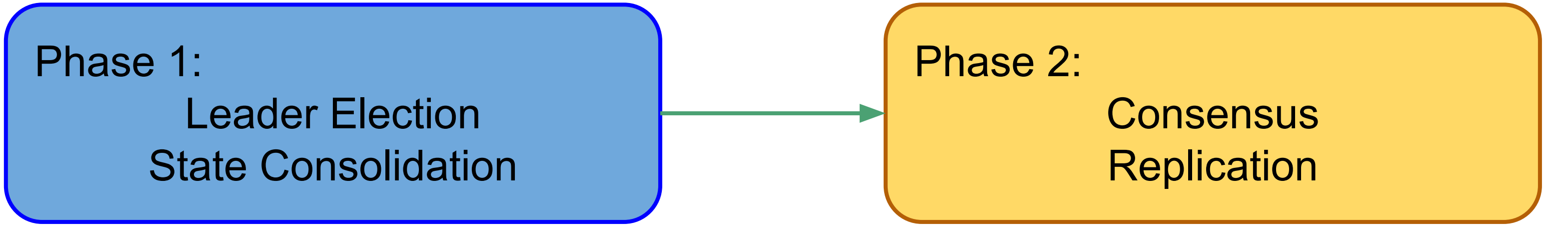
*[FLP]Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. J. ACM 32, 2 (April 1985), 374-382. DOI=http://dx.doi.org/10.1145/3149.214121

Famous consensus algorithms

		Name	Description	Language used
Group Membership	1	VS-ISIS	Reliable group communication, Birman-Joseph 1987	English (items)
	2	VS-ISIS2	Virtual synchrony, Birman-Joseph 1987	English
	3	EVS	Extended Virtual Synchrony for network partition, Amir et al 1995	pseudocode
	4	Paxos-VS	Virtually synchronous Paxos, Birman-Malkhi-van Reness 2012	pseudocode
Primary-Backup	5	VR	Viewstamped replication, Oki-Liskov 1988	pseudocode (coarse)
	6	VR-Revisit	VR revisited, Liskov 2012	English (items)
PAXOS	7	Paxos-Synod	Paxos in part-time parliament, Lamport 1998	TLA (single-value)
	8	Paxos-Basic	Single-value Paxos, Lamport 2001	English (items)
	9	Paxos-Fast	Single-value Paxos with replicas proposing, Lamport 2006	English (items), TLA+
	10	Paxos-Vertical	Single-value Paxos with external starting of leader election, Lamport-Malkhi-Zhou 2009	PlusCal
Failure Detectors	11	ACT	Single-value consenus with failure detection, Aguilera-Chen-Toueg 2000	pseudocode
	12	ACT-Store	ACT with stable storage, same reference as above	pseudocode

Paxos at a high-level

- Assigns 2 roles to processes:
  - Proposer, $\mathscr{P}$: Propose values that can be chosen
  - Acceptor, $\mathscr{A}$: Vote on proposed values
- Assumes a Quorum set, $\mathscr{Q} \subseteq \mathbf{P}(\mathscr{A})$ such that $\forall Q_1, Q_2 \in \mathscr{Q}: Q_1 \cap Q_2 \neq \varnothing$
- 2 phase voting based algorithm:



Lamport's English description of Paxos[†]

Putting the actions of the proposer and acceptor together, we see that the algorithm operates in the following two phases.

Phase 1. (a) A proposer selects a proposal number n and sends a *prepare* request with number n to a majority of acceptors.

(b) If an acceptor receives a *prepare* request with number n greater than that of any *prepare* request to which it has already responded, then it responds to the request with a promise not to accept any more proposals numbered less than n and with the highest-numbered proposal (if any) that it has accepted.

Phase 2. (a) If the proposer receives a response to its *prepare* requests (numbered n) from a majority of acceptors, then it sends an *accept* request to each of those acceptors for a proposal numbered n with a value v, where v is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.

(b) If an acceptor receives an *accept* request for a proposal numbered n, it accepts the proposal unless it has already responded to a *prepare* request having a number greater than n.

[†] Lamport, L. (2001). Paxos made simple. ACM Sigact News, 32(4), 18-25.

Paxos in TLA+ by Lamport et al.[†]

```
Phase1a(b) == /\ ~ \E m \in msgs : (m.type = "1a") /\ (m.bal = b)
              /\ Send([type |-> "1a", bal |-> b])
              /\ UNCHANGED <<maxVBal, maxBal, maxVal>>

Phase1b(a) ==
  \E m \in msgs :
     /\ m.type = "1a"
     /\ m.bal > maxBal[a]
     /\ Send([type |-> "1b", bal |-> m.bal, maxVBal |-> maxVBal[a],
              maxVal |-> maxVal[a], acc |-> a])
     /\ maxBal' = [maxBal EXCEPT ![a] = m.bal]
     /\ UNCHANGED <<maxVBal, maxVal>>
```

```
Phase2a(b) ==
  /\ ~ \E m \in msgs : (m.type = "2a") /\ (m.bal = b)
  /\ \E v \in Values :
       /\ \E Q \in Quorums :
            \E S \in SUBSET {m \in msgs : (m.type = "1b") /\ (m.bal = b)} :
               /\ \A a \in Q : \E m \in S : m.acc = a
               /\ \/ \A m \in S : m.maxVBal = -1
                  \/ \E c \in 0..(b-1) :
                        /\ \A m \in S : m.maxVBal =< c
                        /\ \E m \in S : /\ m.maxVBal = c
                                        /\ m.maxVal = v
       /\ Send([type |-> "2a", bal |-> b, val |-> v])
  /\ UNCHANGED <<maxBal, maxVBal, maxVal>>

Phase2b(a) ==
  \E m \in msgs :
    /\ m.type = "2a"
    /\ m.bal >= maxBal[a]
    /\ Send([type |-> "2b", bal |-> m.bal, val |-> m.val, acc |-> a])
    /\ maxVBal' = [maxVBal EXCEPT ![a] = m.bal]
    /\ maxBal' = [maxBal EXCEPT ![a] = m.bal]
    /\ maxVal' = [maxVal EXCEPT ![a] = m.val]
```

[†] Lamport, L., Merz, S., Doligez, D.: A TLA+ specification of the Paxos Consensus algorithm
https://github.com/tlaplus/v1-tlapm/blob/master/examples/paxos/Paxos.tla (Last modified Fri Nov 28 10:39:17 PST 2014 by Lamport. Accessed Feb 6, 2018)

Moving forward

- ***What's done:***

  Formal specification with proof of safety of the exact phases of *single*-value Paxos in a general, high-level direct language like TLA+

- ***What's lacking:***

  Formal specification with proof of safety of the exact phases of *multi*-value Paxos (also called Multi-Paxos) in a general, high-level direct language like TLA+

Formal Verification

of

Multi-Paxos

for

Distributed Consensus[†]

[†]Chand, S., Liu Y.A., Stoller, S.D.: Formal Verification of Multi-Paxos for Distributed Consensus. In: Proceedings of the 21st International Symposium on Formal Methods, pp. 119–136. Springer (2016)

Contributions

- Complete formal specification of Multi-Paxos in TLA+, minimally extending Lamport at al.'s specification for single-value Paxos.
- Machine-checked proof of safety, using TLA Proof System (TLAPS), with complete invariants, and a systematic proof method.
- Extending the specification and proof with Preemption, showing method for extending existing specification and proof.

Specification

- Minimally extends Lamport et al.'s specification for single-value consensus
- Adds ***slots*** to become multi. For example,
  - Before: Propose value v
  - After: Propose ⟨value, slot⟩ pairs: $\langle v_1, \mathbf{3}\rangle, \langle v_2, \mathbf{5}\rangle, \langle v_3, \mathbf{10}\rangle, \ldots$
- Changes
  - Scalars → Vectors
    - apples = 5 → apples = ⟨1, 2, 2⟩
  - Sets of scalars → Sets of vectors
    - apples = {fuji, gala, honeycrisp} → apples = {⟨fuji, 1⟩, ⟨gala, 2⟩, ⟨honeycrisp, 2⟩}
  - Records of scalars → Records of sets of vectors
    - [apples ↦ 5] → [apples ↦ {⟨fuji, 1⟩, ⟨gala, 2⟩, ⟨honeycrisp, 2⟩}]

Phase1a

Basic Paxos	Multi-Paxos
$Phase1a(b \in \mathcal{B}) \triangleq$	$Phase1a(p \in \mathcal{P}) \triangleq \exists\, b \in \mathcal{B}:$
$\land \nexists\, m \in msgs : \land m.type = \text{“1a”}$	$\land \nexists\, m \in msgs : \land m.type = \text{“1a”}$
$\land m.bal = b$	$\land m.bal = b$
$\land Send([type \mapsto \text{“1a”},$	$\land Send([type \mapsto \text{“1a”}, \underline{from \mapsto p},$
$bal \mapsto b])$	$bal \mapsto b])$
	$\underline{\land pBal' = [pBal \text{ EXCEPT } ![p] = b]}$
$\land \text{UNCHANGED } \langle maxVBal, maxBal, maxVal \rangle$	$\land \text{UNCHANGED } \langle aBal, aVoted \rangle$

Phase1b

Basic Paxos	Multi-Paxos
$Phase1b(a \in \mathcal{A}) \triangleq$	$Phase1b(a \in \mathcal{A}) \triangleq$
$\exists\, m \in msgs :$	$\exists\, m \in msgs :$
$\land m.type = \text{"1a"}$	$\land m.type = \text{"1a"}$
$\land m.bal > maxBal[a]$	$\land m.bal > aBal[a]$
$\land Send([type \mapsto \text{"1b"},$	$\land Send([type \mapsto \text{"1b"},$
$acc \mapsto a,$	$from \mapsto a,$
$bal \mapsto m.bal,$	$bal \mapsto m.bal,$
$maxVBal \mapsto maxVBal[a],$	$voted \mapsto aVoted[a]])$
$maxVal \mapsto maxVal[a]])$	
$\land maxBal' = [maxBal \text{ EXCEPT } ![a] = m.bal]$	$\land aBal' = [aBal \text{ EXCEPT } ![a] = m.bal]$
$\land \text{UNCHANGED } \langle maxVBal, maxVal \rangle$	$\land \text{UNCHANGED } \langle pBal, aVoted \rangle$

Phase2a

Basic Paxos	Multi-Paxos
$Phase2a(b \in \mathcal{B}) \triangleq$ $\land \nexists m \in msgs : \land m.type = \text{"2a"}$ $\qquad\qquad \land m.bal = b$ $\land \exists v \in \mathcal{V} :$ $\quad \land \exists Q \in \mathcal{Q}, S \subseteq^{\dagger} \{m \in msgs : m.type = \text{"1b"} \land$ $\qquad m.bal = b\} :$ $\qquad \land \forall a \in Q : \exists m \in S : m.acc = a$ $\qquad \land \lor \forall m \in S : m.maxVBal = -1$ $\qquad\quad \lor \exists c \in 0..(b-1) :$ $\qquad\qquad \land \forall m \in S : m.maxVBal \le c$ $\qquad\qquad \land \exists m \in S : (m.maxVBal = c)$ $\qquad\qquad\quad \land m.maxVal = v$ $\quad \land Send([type \mapsto \text{"2a"}, bal \mapsto b, val \mapsto v])$ $\land \text{UNCHANGED } \langle maxBal, maxVBal, maxVal \rangle$	$Phase2a(p \in \mathcal{P}) \triangleq$ $\land \nexists m \in msgs : \land m.type = \text{"2a"}$ $\qquad\qquad \land m.bal = pBal[p]$ $\land \exists Q \in \mathcal{Q}, S \subseteq \{m \in msgs : m.type = \text{"1b"} \land$ $\quad m.bal = pBal[p]\} :$ $\quad \land \forall a \in Q : \exists m \in S : m.from = a$ $\quad \land Send([type \mapsto \text{"2a"},$ $\qquad from \mapsto p,$ $\qquad bal \mapsto pBal[p],$ $\qquad propSV \mapsto PropSV(\text{UNION}$ $\qquad\quad \{m.voted : m \in S\})])$ $\land \text{UNCHANGED } \langle pBal, aBal, aVoted \rangle$

Phase2a

Basic Paxos	Multi-Paxos
$Phase2a(b \in \mathcal{B}) \triangleq$	$Phase2a(p \in \mathcal{P}) \triangleq$
$\land \nexists m \in msgs : \land m.type = \text{“2a”}$	$\land \nexists m \in msgs : \land m.type = \text{“2a”}$
$\land m.bal = b$	$\land m.bal = pBal[p]$
$\land \exists\, v \in \mathcal{V} :$	
$\land \exists\, Q \in \mathcal{Q}, S \subseteq^{\dagger} \{m \in msgs : m.type = \text{“1b”} \land$	$\land \exists\, Q \in \mathcal{Q}, S \subseteq \{m \in msgs : m.type = \text{“1b”} \land$
$m.bal = b\} :$	$m.bal = pBal[p]\} :$
$\land \forall\, a \in Q : \exists\, m \in S : m.acc = a$	$\land \forall\, a \in Q : \exists\, m \in S : m.from = a$
$\land \lor \forall\, m \in S : m.maxVBal = -1$	$\land Send([type \mapsto \text{“2a”},$
$\lor \exists\, c \in 0..(b-1) :$	$from \mapsto p,$
$\land \forall\, m \in S : m.maxVBal \leq c$	$bal \mapsto pBal[p],$
$\land \exists\, m \in S : (m.maxVBal = c)$	$propSV \mapsto PropSV(\text{UNION}$
$\land m.maxVal = v$	$\{m.voted : m \in S\})])$
$\land Send([type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v])$	
$\land \text{UNCHANGED } \langle maxBal, maxVBal, maxVal \rangle$	$\land \text{UNCHANGED } \langle pBal, aBal, aVoted \rangle$
	where,
	$PropSV(T) \triangleq MaxSV(T) \cup NewSV(T)$

Phase2a

Basic Paxos	Multi-Paxos
$Phase2a(b \in \mathcal{B}) \triangleq$ $\land \nexists m \in msgs : \land m.type = \text{"2a"}$ $\qquad \land m.bal = b$ $\land \exists\, v \in \mathcal{V} :$ $\quad \land \exists\, Q \in \mathcal{Q}, S \subseteq^{\dagger} \{m \in msgs : m.type = \text{"1b"} \land$ $\qquad m.bal = b\} :$ $\qquad \land \forall\, a \in Q : \exists\, m \in S : m.acc = a$ $\qquad \land \lor \forall\, m \in S : m.maxVBal = -1$ $\qquad\quad \lor \exists\, c \in 0..(b-1) :$ $\qquad\qquad \land \forall\, m \in S : m.maxVBal \leq c$ $\qquad\qquad \land \exists\, m \in S : (m.maxVBal = c)$ $\qquad\qquad\quad \land m.maxVal = v$ $\quad \land Send([type \mapsto \text{"2a"}, bal \mapsto b, val \mapsto v])$ $\land \text{UNCHANGED}\ \langle maxBal, maxVBal, maxVal \rangle$	$Phase2a(p \in \mathcal{P}) \triangleq$ $\land \nexists m \in msgs : \land m.type = \text{"2a"}$ $\qquad \land m.bal = pBal[p]$ $\land \exists\, Q \in \mathcal{Q}, S \subseteq \{m \in msgs : m.type = \text{"1b"} \land$ $\quad m.bal = pBal[p]\} :$ $\quad \land \forall\, a \in Q : \exists\, m \in S : m.from = a$ $\quad \land Send([type \mapsto \text{"2a"},$ $\qquad from \mapsto p,$ $\qquad bal \mapsto pBal[p],$ $\qquad propSV \mapsto PropSV(\text{UNION}$ $\qquad\quad \{m.voted : m \in S\})])$ $\land \text{UNCHANGED}\ \langle pBal, aBal, aVoted \rangle$ where, $MaxBSV(T) \triangleq \{t \in T : \forall\, t2 \in T :$ $\quad t2.slot = t.slot \Rightarrow t2.bal \leq t.bal\}$ $MaxSV(T) \triangleq \{[slot \mapsto t.slot,$ $\quad val \mapsto t.val] : t \in MaxBSV(T)\}$ $PropSV(T) \triangleq MaxSV(T) \cup NewSV(T)$

Phase2a

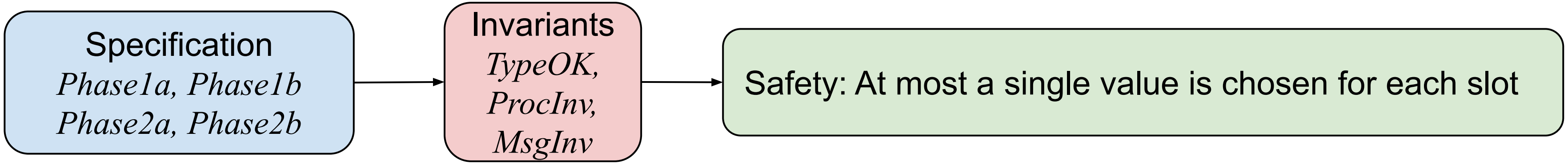
Basic Paxos	Multi-Paxos
$Phase2a(b \in \mathcal{B}) \triangleq$ $\land \nexists m \in msgs : \land m.type = \text{“2a”}$ $\qquad\qquad \land m.bal = b$ $\land \exists\, v \in \mathcal{V} :$ $\quad \land \exists\, Q \in \mathcal{Q}, S \subseteq^{\dagger} \{m \in msgs : m.type = \text{“1b”} \land$ $\qquad m.bal = b\} :$ $\qquad \land \forall\, a \in Q : \exists\, m \in S : m.acc = a$ $\qquad \land \lor \forall\, m \in S : m.maxVBal = -1$ $\qquad\quad \lor \exists\, c \in 0..(b-1) :$ $\qquad\qquad \land \forall\, m \in S : m.maxVBal \leq c$ $\qquad\qquad \land \exists\, m \in S : (m.maxVBal = c)$ $\qquad\qquad\quad \land m.maxVal = v$ $\quad \land Send([type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v])$ $\land \text{UNCHANGED } \langle maxBal, maxVBal, maxVal \rangle$	$Phase2a(p \in \mathcal{P}) \triangleq$ $\land \nexists m \in msgs : \land m.type = \text{“2a”}$ $\qquad\qquad \land m.bal = pBal[p]$ $\land \exists\, Q \in \mathcal{Q}, S \subseteq \{m \in msgs : m.type = \text{“1b”} \land$ $\quad m.bal = pBal[p]\} :$ $\quad \land \forall\, a \in Q : \exists\, m \in S : m.from = a$ $\quad \land Send([type \mapsto \text{“2a”},$ $\qquad from \mapsto p,$ $\qquad bal \mapsto pBal[p],$ $\qquad propSV \mapsto PropSV(\text{UNION}$ $\qquad\quad \{m.voted : m \in S\})])$ $\land \text{UNCHANGED } \langle pBal, aBal, aVoted \rangle$ where, $MaxBSV(T) \triangleq \{t \in T : \forall\, t2 \in T :$ $\quad t2.slot = t.slot \Rightarrow t2.bal \leq t.bal\}$ $MaxSV(T) \triangleq \{[slot \mapsto t.slot,$ $\quad val \mapsto t.val] : t \in MaxBSV(T)\}$ $UnusedS(T) \triangleq \{s \in \mathcal{S} : \nexists t \in T : t.slot = s\}$ $NewSV(T) \triangleq \text{CHOOSE } D \subseteq [slot :$ $\quad UnusedS(T), val : \mathcal{V}] : \forall\, d1, d2 \in D :$ $\qquad d1.slot = d2.slot \Rightarrow d1 = d2$ $PropSV(T) \triangleq MaxSV(T) \cup NewSV(T)$

Phase2b

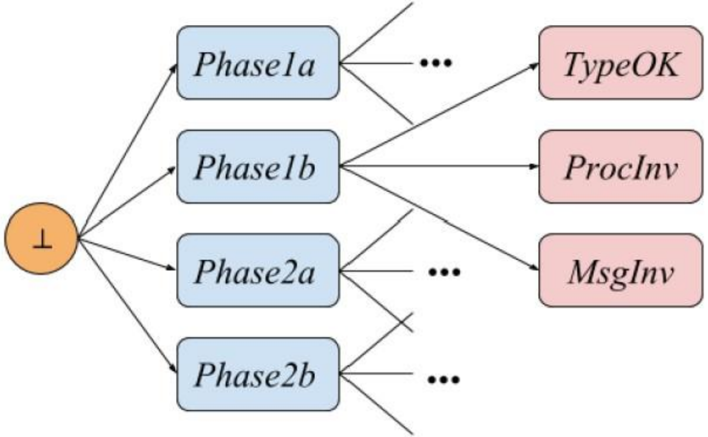
Basic Paxos	Multi-Paxos
$Phase2b(a \in \mathcal{A}) \triangleq$ $\exists\, m \in msgs :$ $\quad \land m.type = \text{"2a"}$ $\quad \land m.bal \geq maxBal[a]$ $\quad \land Send([type \mapsto \text{"2b"},$ $\qquad acc \mapsto a,$ $\qquad bal \mapsto m.bal,$ $\qquad val \mapsto m.val])$ $\quad \land maxBal' = [maxBal \text{ EXCEPT } ![a] = m.bal]$ $\quad \land maxVBal' =$ $\qquad [maxVBal \text{ EXCEPT } ![a] = m.bal]$ $\quad \land maxVal' =$ $\qquad [maxVal \text{ EXCEPT } ![a] = m.val]$	$Phase2b(a \in \mathcal{A}) \triangleq$ $\exists\, m \in msgs :$ $\quad \land m.type = \text{"2a"}$ $\quad \land m.bal \geq aBal[a]$ $\quad \land Send([type \mapsto \text{"2b"},$ $\qquad from \mapsto a,$ $\qquad bal \mapsto m.bal,$ $\qquad \underline{propSV \mapsto m.propSV}])$ $\quad \land aBal' = [aBal \text{ EXCEPT } ![a] = m.bal]$ $\quad \land aVoted' = [aVoted \text{ EXCEPT } ![a] =$ $\qquad \cup \{[bal \mapsto m.bal, slot \mapsto d.slot,$ $\qquad\quad val \mapsto d.val] : d \in m.propSV\}$ $\qquad \cup \{e \in aVoted[a] :$ $\qquad\quad \nexists\, r \in m.propSV : e.slot = r.slot\}]$ $\quad \land \text{UNCHANGED } \langle pBal \rangle$

Verification

- Similar proof strategy and skeleton to Lamport et al.’s



- Hierarchical proof by induction



Overview of Results

Metric	Single-value Paxos	Multi-Paxos		Multi-Paxos w/ Preemption	
	Value	Value	Increase	Value	Increase
Specification size	52	55	6%	74	35%
Proof size	310	787	154%	831	6%
No. of obligations	239	779	226%	825	6%
Proof check time(s)	16	58	263%	110	90%

- Increase in proof parameters for Multi-Paxos due to complications due to vectors
- Conflated increase in proof check time for Multi-Paxos w/Preemption due to a new library import

Can we do better?

- Can the Specification be even more high-level?

- Can the Invariants be made to more easily follow from the specification?

Can we do better?

- Can the Specification be even more high-level?

- Can the Invariants be made to more easily follow from the specification?

Use **History Variables!!!**

Simpler Specifications

and

Easier Proofs

using

History Variables[†]

[†]Chand, S., & Liu, Y. A. Simpler specifications and easier proofs of distributed algorithms using history variables. In NASA Formal Methods Symposium (pp. 70-86). Springer (2018).

Contributions

- Demonstrate a **Systematic Style** to write specifications of distributed algorithms using message history variables - *sent* and *received* (in TLA+).
- A **Method** to systematically derive important invariants of the algorithm.

- Case studies:
  - Single-value Paxos for single-value consensus by Lamport et al. [1]
  - Multi-Paxos for multi-value consensus by Chand at al. [2]
  - Multi-Paxos with preemption by Chand at al. [2]

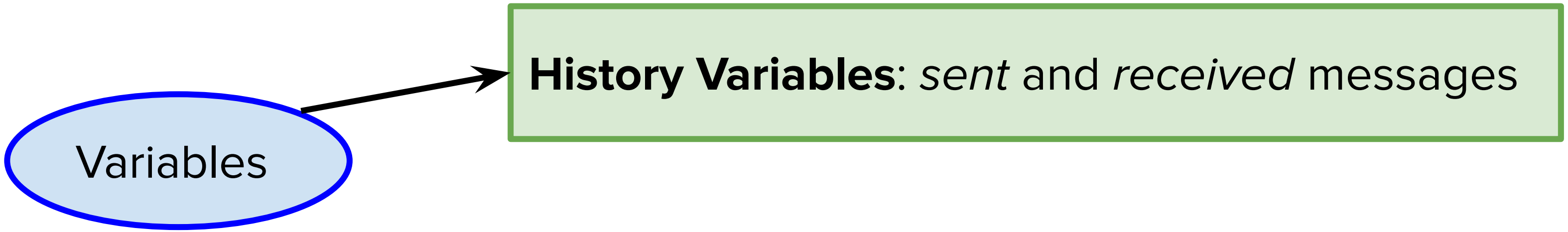
[1] Lamport, L., Merz, S., Doligez, D. Paxos.tla. github.com/tlaplus/v1-tlapm/blob/master/examples/paxos/Paxos.tla
[2] Chand, S., Liu, Y. A., and Stoller, S. D. Formal verification of multi-paxos for distributed consensus. In FM 2016: Formal Methods: 21st International Symposium, pages 119–136. Springer, 2016

Results

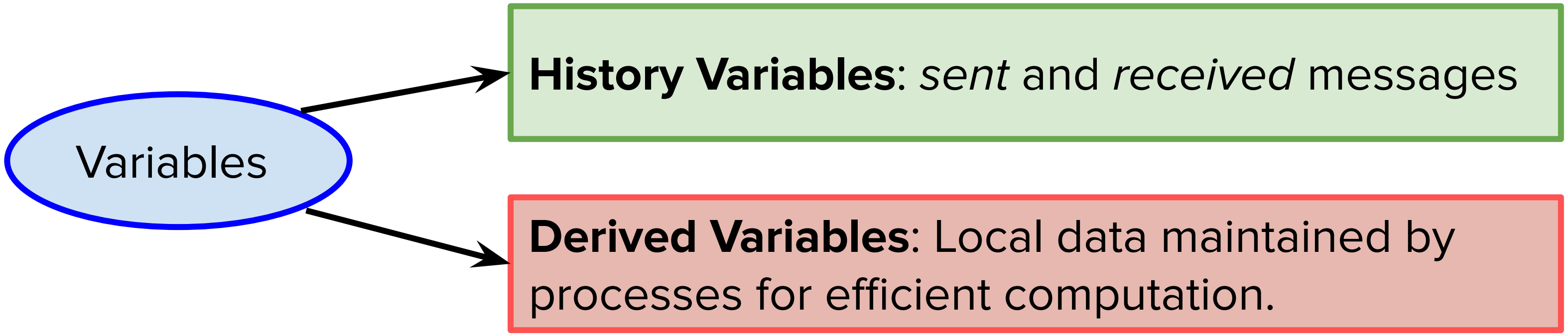
Metric	Basic Paxos			Multi-Paxos			Multi-Paxos w/ Preemption		
	Lam	Us	Decr	Cha	Us	Decr	Cha	Us	Decr
Specification size	52	39	**25%**	55	42	**24%**	74	52	**30%**
Proof size	310	227	**27%**	787	520	**34%**	831	538	**35%**
No. of invariants	15	5, 1^	**60%**	16	7, 1^	**50%**	17	7, 1^	**53%**
Proof check time(s)	16	12	**25%**	58	28	**52%**	110	30	**73%**

Lam: Lamport et al. [1], Cha: Chand et al. [2], Us: Using History Variables only, Decr: percentage of decrease.

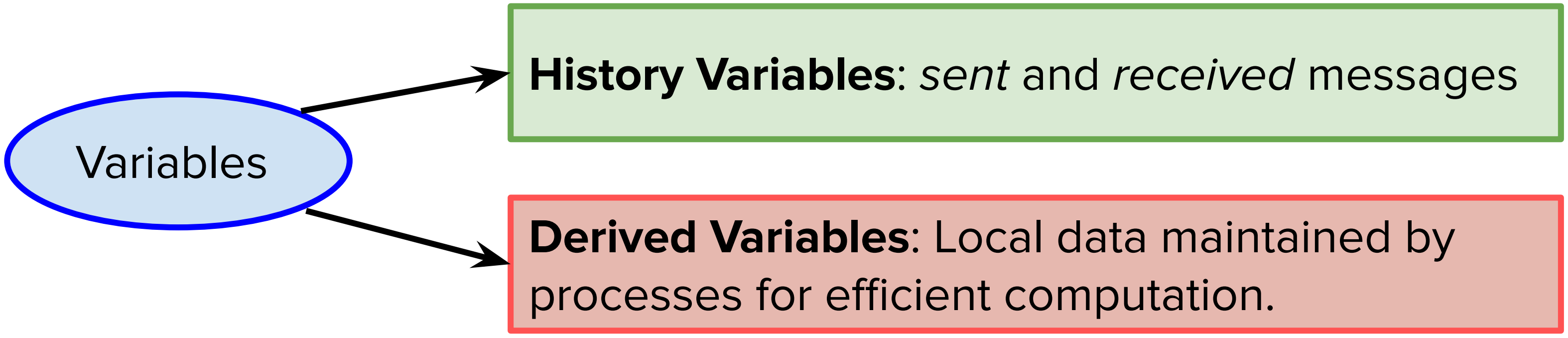
History and Derived Variables



History and Derived Variables



History and Derived Variables



- BUT, local data is updated upon receiving some set of messages
- Thus, derived variables are functions of history variables
- So, write specifications using history variables ***only***

Specification of Basic Paxos

<table>
<tr><td colspan="2">Phase 1a. A proposer selects a proposal number b and sends a 1a request with number b to a majority of acceptors.</td></tr>
<tr><td>Lamport et al.'s</td><td>Using sent only</td></tr>
<tr><td>$Phase1a(b \in \mathcal{B}) \triangleq$
$\land \nexists\, m \in sent : (m.type = \text{“1a”}) \land (m.bal = b)$
$\land Send([type \mapsto \text{“1a”}, bal \mapsto b])$
$\land \text{UNCHANGED}\ \langle maxVBal, maxBal, maxVal \rangle$</td><td>$Phase1a(b \in \mathcal{B}) \triangleq$
$Send([type \mapsto \text{“1a”}, bal \mapsto b])$</td></tr>
</table>

Phase 1b. If an acceptor receives a 1a request with number *bal* greater than that of any 1a request to which it has already responded, then it responds to the request with a promise not to accept any more proposals numbered less than *bal* and with the highest-numbered proposal (if any) that it has accepted.

Lamport et al.'s	Using *sent* only
$Phase1b(a \in \mathcal{A}) \triangleq$	$Phase1b(a \in \mathcal{A}) \triangleq$
$\exists\, m \in sent :$	$\exists\, m \in sent, r \in max\_prop(a) :$
$\wedge m.type = \text{“1a”}$	$\wedge m.type = \text{“1a”}$
$\wedge m.bal > maxBal[a]$	$\wedge \forall\, m2 \in sent : m2.type \in \{\text{“1b”}, \text{“2b”}\} \wedge$ $m2.acc = a \Rightarrow m.bal > m2.bal$
$\wedge Send([type \mapsto \text{“1b”},$ $acc \mapsto a, bal \mapsto m.bal,$ $maxVBal \mapsto maxVBal[a],$ $maxVal \mapsto maxVal[a]])$	$\wedge Send([type \mapsto \text{“1b”},$ $acc \mapsto a, bal \mapsto m.bal,$ $maxVBal \mapsto r.bal,$ $maxVal \mapsto r.val])$
$\wedge maxBal' =$ $[maxBal \text{ EXCEPT } ![a] = m.bal]$	$2bs(a) \triangleq \{m \in sent : m.type = \text{“2b”} \wedge m.acc = a\}$
$\wedge \text{UNCHANGED } \langle maxVBal, maxVal \rangle$	$max\_prop(a) \triangleq$ $\text{IF } 2bs(a) = \emptyset \text{ THEN } \{[bal \mapsto -1, val \mapsto \bot]\}$ $\text{ELSE } \{m \in 2bs(a) : \forall m2 \in 2bs(a) : m.bal \geq m2.bal\}$

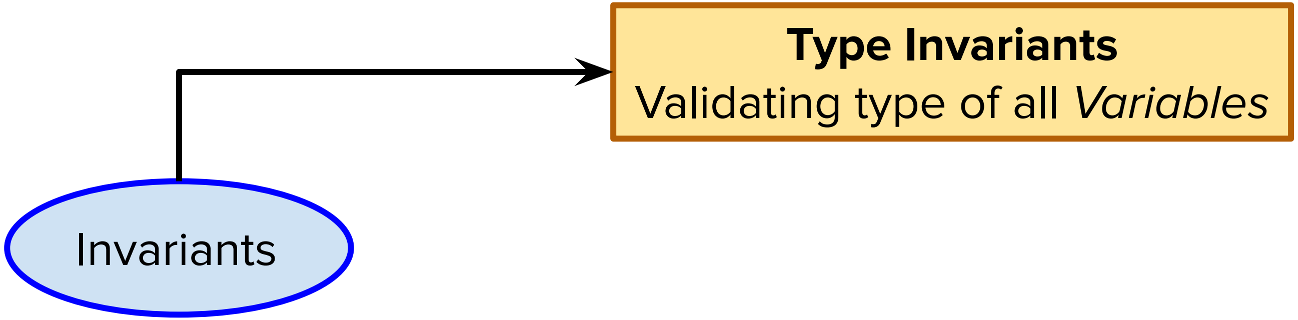
Phase 2a. If the proposer receives a response to its 1a requests (numbered b) from a majority of acceptors, then it sends a 2a request to each of those acceptors for a proposal numbered b with a value v, where v is the value of the highest-numbered proposal among the 1b responses, or is any value if the responses reported no proposals.

Lamport et al.'s	Using *sent* only
$Phase2a(b \in \mathcal{B}) \triangleq$ $\wedge \nexists m \in sent : m.type = \text{"2a"} \wedge m.bal = b$ $\wedge \exists\, v \in \mathcal{V}, Q \in \mathcal{Q}, S \subseteq \{m \in sent :$ $\quad m.type = \text{"1b"} \wedge m.bal = b\} :$ $\quad\quad \wedge \forall\, a \in Q : \exists\, m \in S : m.acc = a$ $\quad\quad \wedge \vee \forall\, m \in S : m.maxVBal = -1$ $\quad\quad\quad \vee \exists\, c \in 0..(b-1) :$ $\quad\quad\quad\quad \wedge \forall\, m \in S : m.maxVBal \leq c$ $\quad\quad\quad\quad \wedge \exists\, m \in S : \wedge m.maxVBal = c$ $\quad\quad\quad\quad\quad\quad \wedge m.maxVal = v$ $\quad \wedge Send([type \mapsto \text{"2a"}, bal \mapsto b, val \mapsto v])$ $\wedge \text{UNCHANGED } \langle maxBal, maxVBal,$ $\quad maxVal \rangle$	$Phase2a(b \in \mathcal{B}) \triangleq$ $\wedge \nexists m \in sent : m.type = \text{"2a"} \wedge m.bal = b$ $\wedge \exists\, v \in \mathcal{V}, Q \in \mathcal{Q}, S \subseteq \{m \in sent :$ $\quad m.type = \text{"1b"} \wedge m.bal = b\} :$ $\quad\quad \wedge \forall\, a \in Q : \exists\, m \in S : m.acc = a$ $\quad\quad \wedge \vee \forall\, m \in S : m.maxVBal = -1$ $\quad\quad\quad \vee \exists\, c \in 0..(b-1) :$ $\quad\quad\quad\quad \wedge \forall\, m \in S : m.maxVBal \leq c$ $\quad\quad\quad\quad \wedge \exists\, m \in S : \wedge m.maxVBal = c$ $\quad\quad\quad\quad\quad\quad \wedge m.maxVal = v$ $\quad \wedge Send([type \mapsto \text{"2a"}, bal \mapsto b, val \mapsto v])$

Phase 2b. If an acceptor receives a 2a request for a proposal numbered *bal*, it accepts the proposal unless it has already responded to a 1a request having a number greater than *bal*.

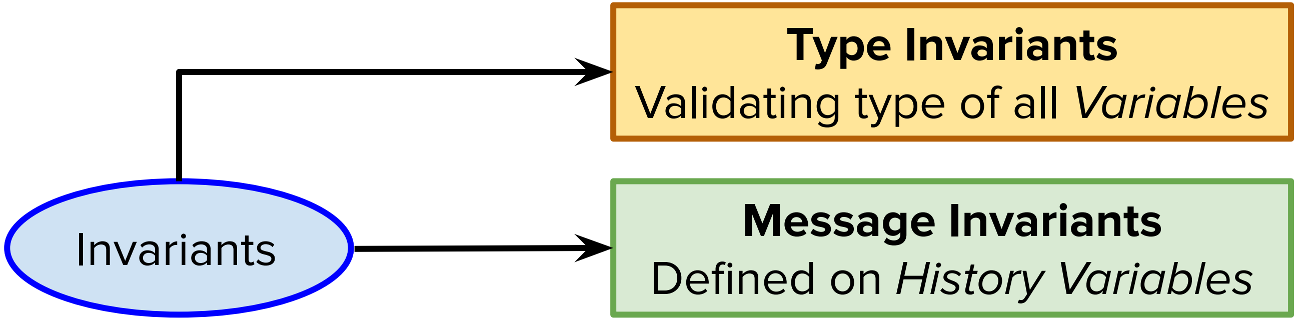
Lamport et al.'s	Using *sent* only
$Phase2b(a \in \mathcal{A}) \triangleq$	$Phase2b(a \in \mathcal{A}) \triangleq$
$\exists\, m \in sent :$	$\exists\, m \in sent :$
$\land m.type =$ "2a"	$\land m.type =$ "2a"
$\land m.bal \geq maxBal[a]$	$\land \forall\, m2 \in sent : m2.type \in \{\text{"1b"},\text{"2b"}\} \land$ $m2.acc = a \Rightarrow m.bal \geq m2.bal$
$\land Send([type \mapsto \text{"2b"}, acc \mapsto a, bal \mapsto m.bal, val \mapsto m.val])$	$\land Send([type \mapsto \text{"2b"}, acc \mapsto a, bal \mapsto m.bal, val \mapsto m.val])$
$\land maxBal' = [maxBal \text{ EXCEPT } ![a] = m.bal]$	
$\land maxVBal' = [maxVBal \text{ EXCEPT } ![a] = m.bal]$	
$\land maxVal' = [maxVal \text{ EXCEPT } ![a] = m.val]$	

Invariants



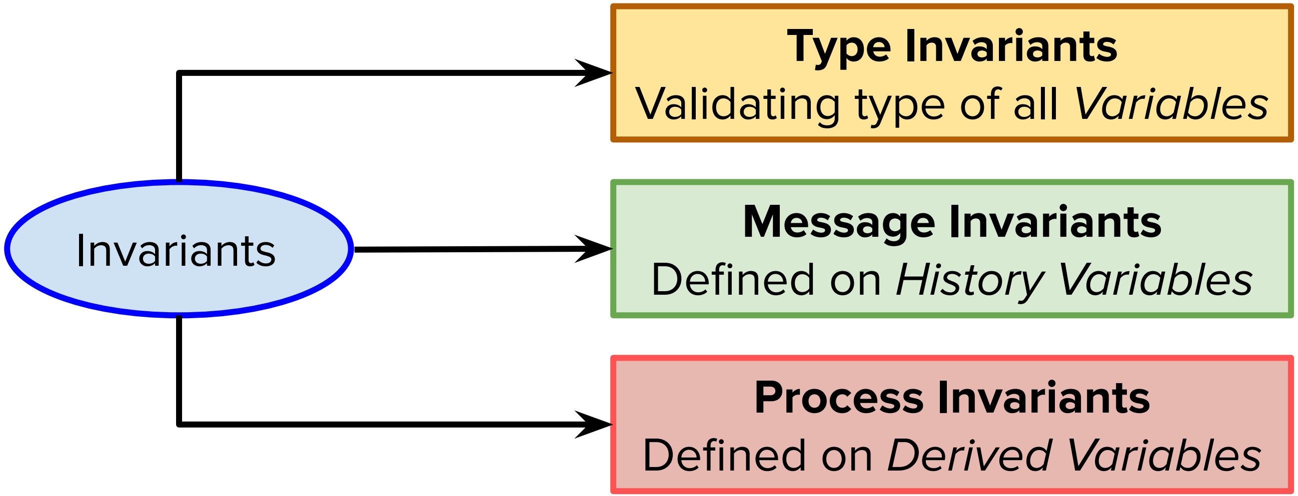
Type Invariants
Validating type of all *Variables*
Invariants

Invariants



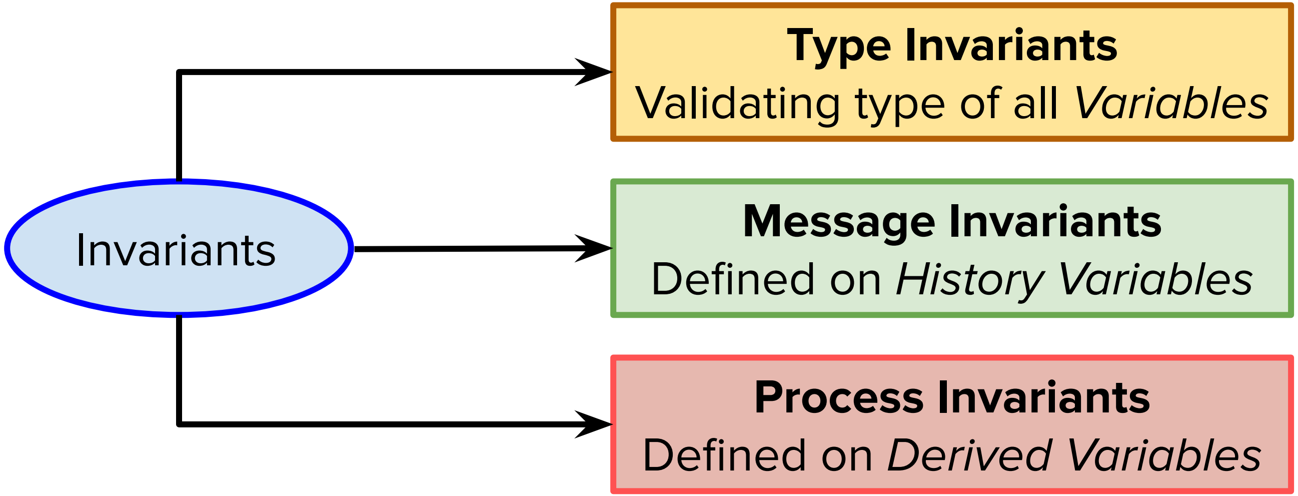
Invariants
Type Invariants
Validating type of all *Variables*
Message Invariants
Defined on *History Variables*

Invariants



Invariants
Type Invariants
Validating type of all *Variables*
Message Invariants
Defined on *History Variables*
Process Invariants
Defined on *Derived Variables*

Invariants



Advantages with History Variables:

1) No derived variables ⇒ Fewer invariants (~50% fewer in our case studies)
2) Systematically derive message invariants (Could derive all but 1 in our case studies)

3-Step Method for deriving Message Invariants

1. Increment
    a. Look at the increment made to *sent* by an action

Reply. Upon receiving a request numbered n, a process replies with an acknowledgement with the same number n.

$$
\begin{aligned}
& Reply(p \in \mathcal{P}) \triangleq \\
& \exists\, m \in sent : \\
& \quad \land Receive(m, p) \\
& \quad \land m.type = \text{"req"} \\
& \quad \land Send([type \mapsto \text{"ack"}, \\
& \qquad\qquad reqnum \mapsto m.reqnum, \\
& \qquad\qquad to \mapsto m.from])
\end{aligned}
$$

3-Step Method for Deriving Message Invariants

1. Increment
   - Look at the increment made to *sent* by an action
2. Analyze
   - Analyze and connect the contents of msg and the body of the action

$\phi(msg) = \exists\ m \in sent:$

$m.type = \text{“req”} \ \wedge$

$msg.reqnum = m.reqnum \ \wedge$

$msg.to = m.from$

Reply. Upon receiving a request numbered n, a process replies with an acknowledgement with the same number n.

$$
\begin{aligned}
&Reply(p \in \mathcal{P}) \triangleq \\
&\exists\, m \in sent : \\
&\quad \wedge Receive(m, p) \\
&\quad \wedge m.type = \text{“req”} \\
&\quad \wedge Send([type \mapsto \text{“ack”}, \\
&\qquad\qquad reqnum \mapsto m.reqnum, \\
&\qquad\qquad to \mapsto m.from])
\end{aligned}
$$

3-Step Method for Deriving Message Invariants

1. Increment
   - Look at the increment made to *sent* by an action
2. Analyze
   - Analyze and connect the contents of msg and the body of the action
3. Assimilate
   - Use the properties found in step 2 to derive an invariant:

$$MsgInvAcc \triangleq \forall msg \in sent: msg.type = \text{“ack”} \Rightarrow \phi(msg)$$

$\phi(msg) = \exists m \in sent: m.type = \text{“req”} \land msg.to = m.from \land msg.reqnum = m.reqnum$

2b Message Invariant

$$
\begin{aligned}
&Phase2b(a \in \mathcal{A}) \triangleq \\
&\exists\, m \in sent: \\
&\quad \wedge\, \underline{m.type = \text{“2a”}} \\
&\quad \wedge\, \forall\, m2 \in sent : m2.type \in \{\text{“1b”}, \text{“2b”}\} \wedge \\
&\qquad\qquad m2.acc = a \Rightarrow m.bal \geq m2.bal \\
&\quad \wedge\, Send([type \mapsto \text{“2b”}, acc \mapsto a, \\
&\qquad \underline{bal \mapsto m.bal, val \mapsto m.val}])
\end{aligned}
$$

$$
\begin{aligned}
&\forall\, msg \in sent: msg.type = \text{“2b”} \Rightarrow \\
&\quad \exists\, m \in sent: \\
&\qquad \wedge\, \underline{m.type = \text{“2a”}} \\
&\qquad \wedge\, \underline{msg.bal = m.bal} \\
&\qquad \wedge\, \underline{msg.val = m.val}
\end{aligned}
$$

2a Message Invariant

$$
\begin{aligned}
&Phase2a(b \in \mathcal{B}) \triangleq \\
&\land \forall\, m \in sent : m.type = \text{“2a”} \Rightarrow m.bal \neq b \\
&\land \exists\, v \in \mathcal{V}, Q \in \mathcal{Q}, S \subseteq \{m \in sent : \\
&\quad m.type = \text{“1b”} \land m.bal = b\} : \\
&\qquad \land \forall\, a \in Q : \exists\, m \in S : m.acc = a \\
&\qquad \land \lor \forall\, m \in S : m.maxVBal = -1 \\
&\qquad\quad \lor \exists\, c \in 0..(b-1) : \\
&\qquad\qquad \land \forall\, m \in S : m.maxVBal \leq c \\
&\qquad\qquad \land \exists\, m \in S : \land m.maxVBal = c \\
&\qquad\qquad\qquad\qquad \land m.maxVal = v \\
&\quad \land Send([type \mapsto \text{“2a”}, bal \mapsto b, val \mapsto v])
\end{aligned}
$$

$$
\begin{aligned}
&\forall\, msg \in sent: msg.type = \text{“2a”} \Rightarrow \\
&\quad \forall\, m \in sent: \\
&\qquad \land\ m.type = \text{“2a”} \\
&\qquad \land\ msg.bal = m.bal \\
&\qquad \Rightarrow msg = m
\end{aligned}
$$

1b Message Invariant

$$
\begin{aligned}
&Phase1b(a \in \mathcal{A}) \triangleq \\
&\exists\, m \in sent, r \in max\_prop(a): \\
&\quad \land m.type = \text{“1a”} \\
&\quad \land \forall\, m2 \in sent : m2.type \in \{\text{“1b”}, \text{“2b”}\} \land \\
&\qquad m2.acc = a \Rightarrow m.bal > m2.bal \\
&\quad \land Send([type \mapsto \text{“1b”}, \\
&\qquad acc \mapsto a, bal \mapsto m.bal, \\
&\qquad maxVBal \mapsto r.bal, \\
&\qquad maxVal \mapsto r.val])
\end{aligned}
$$

$$
\begin{aligned}
&2bs(a) \triangleq \{m \in sent : m.type = \text{“2b”} \land m.acc = a\} \\
&max\_prop(a) \triangleq \\
&\quad \text{IF } 2bs(a) = \emptyset \text{ THEN } \{[bal \mapsto -1, val \mapsto \bot]\} \\
&\quad \text{ELSE } \{m \in 2bs(a) : \forall m2 \in 2bs(a) : m.bal \geq m2.bal\}
\end{aligned}
$$

$\Rightarrow$

$$
\begin{aligned}
&\forall\, msg \in sent: msg.type = \text{“1b”} \Rightarrow \\
&\quad \lor\ msg.maxVBal = -1 \\
&\quad \lor\ \exists\, m \in sent: \\
&\qquad \land\ m.type = \text{“2b”} \\
&\qquad \land\ msg.acc = m.acc \\
&\qquad \land\ msg.maxVBal = m.bal \\
&\qquad \land\ msg.maxVal = m.val
\end{aligned}
$$

1b Message Invariant using max

$$Phase1b(a \in \mathcal{A}) \triangleq$$
$$\exists\, m \in sent, r \in max\_prop(a):$$
$$\land m.type = \text{“1a”}$$
$$\land \forall\, m2 \in sent : m2.type \in \{\text{“1b”}, \text{“2b”}\} \land$$
$$m2.acc = a \Rightarrow m.bal > m2.bal$$
$$\land Send([type \mapsto \text{“1b”},$$
$$acc \mapsto a, \underline{bal \mapsto m.bal},$$
$$\underline{maxVBal \mapsto r.bal},$$
$$maxVal \mapsto r.val])$$

$$2bs(a) \triangleq \{m \in sent : m.type = \text{“2b”} \land m.acc = a\}$$
$$\underline{max\_prop(a)} \triangleq$$
$$\text{IF } 2bs(a) = \emptyset \text{ THEN } \{[bal \mapsto -1, val \mapsto \bot]\}$$
$$\text{ELSE } \{m \in 2bs(a) : \forall m2 \in 2bs(a) : \underline{m.bal \geq m2.bal}\}$$

$\Rightarrow$

$\forall\, msg \in sent$: $msg.type = \text{“1b”} \Rightarrow$

$\forall\, b2 \in (\underline{msg.maxVBal}, \underline{msg.bal})$:

$\nexists m \in sent$:

$\land\ \underline{m.type = \text{“2b”}}$

$\land\ \underline{msg.acc = m.acc}$

$\land\ \underline{m.bal = b2}$

Summary

- Distributed systems are complex and difficult to reason about.
- Distributed consensus is a fundamental problem and Paxos is a well-known algorithm for it.
- We discussed a formal specification and safety proof of multi-value Paxos in TLA+ and TLAPS respectively.
- We discussed a systematic method that uses history variables to specify and verify distributed algorithms.

THANKS!

Q + A