Lattice Agreement in Message Passing Systems

Xiong Zheng, Changyong Hu, and Vijay Garg

Parallel and Distributed Systems Lab,
Department of Electrical and Computer Engineering,
The University of Texas at Austin,
Austin, TX 78712

Road Map

- System Model
- Motivation
- Lattice Agreement
  - Definition
  - Related Work
  - Synchronous Protocol
  - Asynchronous Protocol
- Generalized Lattice Agreement
  - Definition
  - Asynchronous Protocol
- Future Work

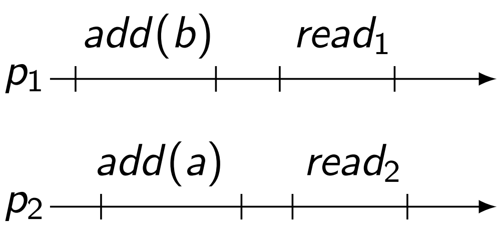
System Model

- A completely connected message passing system.
- Synchronous and asynchronous systems.
- Crash failures but no Byzantine failures.
- Reliable communication.

Motivation: Linearizable Replicated State Machine(RSM)

Lattice agreement can be applied to implement linearizable RSM [Faleiro et al, 2012, PODC]

- Lattice Agreement vs Consensus
  Synchronous: consensus needs at least $f + 1$ rounds. Lattice agreement can be solved in $\log f + 1$ rounds.
  Asynchronous: consensus is impossible. Lattice agreement can be solved in $O(f)$ rounds.



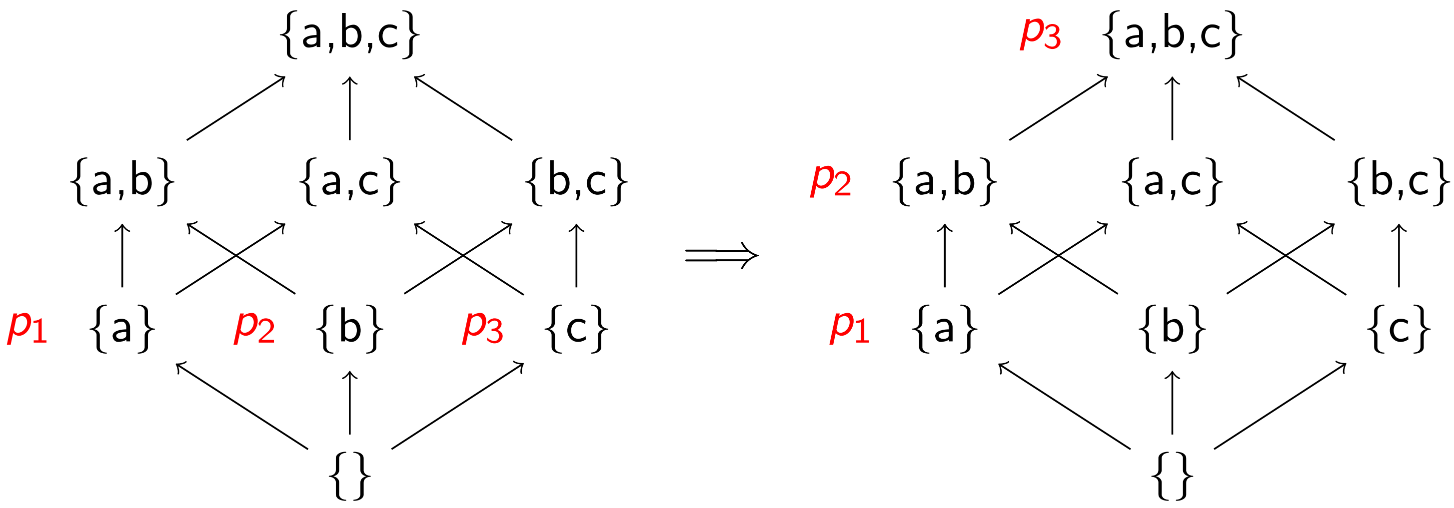
$read_1$	$read_2$	**Valid**
{b}	{a,b}	Yes
{a,b}	{a}	Yes
{a,b}	{a,b}	Yes
{b}	{a}	No

Road Map

- Motivation
- System Model
- Lattice Agreement
  - Definition
  - Related Work
  - Synchronous Protocol
  - Asynchronous Protocol
- Generalized Lattice Agreement
  - Definition
  - Asynchronous Protocol
- Future Work

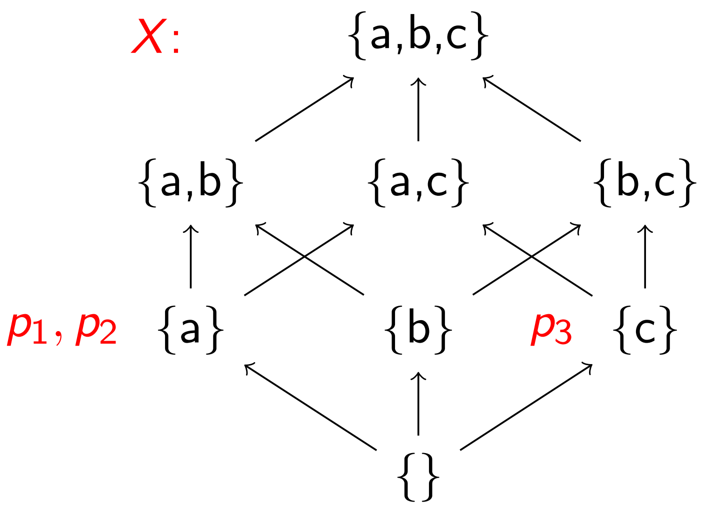
Problem Definition

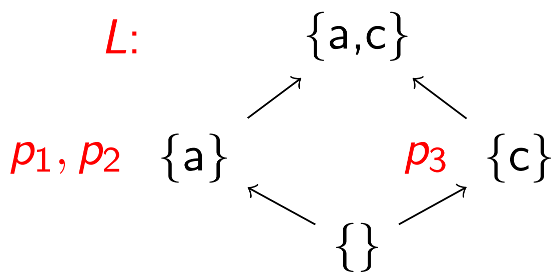
- Lattice Agreement
  [Hagit Attiya, Maurice Herlihy, and Ophir achman, 1995, Distributed Computing]
  Each process p_i has a input value x_i from a lattice X and must decide on some output y_i also in X.
  Downward-Validity: For all $i \in [1..n]$, $x_i \leq y_i$.
  Upward-Validity: For all $i \in [1..n]$, $y_i \leq \sqcup\{x_1, ..., x_n\}$.
  Comparability: For all $i \in [1..n]$ and $j \in [1..n]$, either $y_i \leq y_j$ or $y_j \leq y_i$, i.e, output values lie on a chain.



Useful Definitions

Height of value: The height of a value v in a lattice X is the length of longest path from any minimal value to v.
Height of lattice: The height of a lattice X is the height of its largest value.
Input sublattice L: Let L be the join-closed subset of X that includes all input values. $h(L) \leq n$.





Related Work

- Synchronous systems

Protocol	Time	Total #Messages
[Attiya et al,98,SIAM]	$O(\log n)$	$O(n^2)$
[Marios,2018]	$\min\{O(h(L)), O(\sqrt{f})\}$	$n^2 \cdot \min\{O(h(L)), O(\sqrt{f})\}$
LA_α	$O(\log h(L))$	$O(n^2 \log h(L))$
LA_β	$O(\log f)$	$O(n^2 \log f)$
LA_γ	$\min\{O(\log^2 h(L)), O(\log^2 f)\}$	$n^2 \cdot \min\{O(\log^2 h(L)), O(\log^2 f)\}$

- Asynchronous systems

Protocol	Time	Total #Messages
[Faleiro et al,2012,PODC]	$O(n)$	$O(n^3)$
LA_δ	$\min\{O(h(L)), O(f)\}$	$n^2 \cdot \min\{O(h(L)), O(f)\}$

n: the number of processes
f: the maximum number of crash failures
$h(L)$: the height of input sublattice L

The *Classifier* Procedure

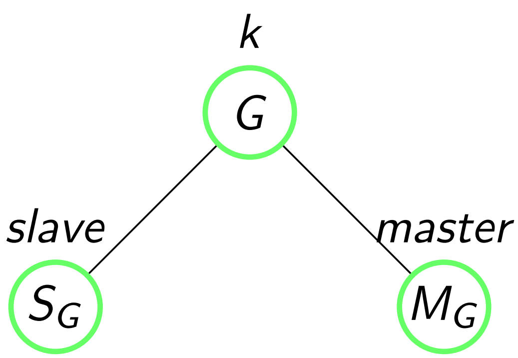
Motivation: divide processes into two groups and make sure one group dominates the other.

Classifier(v, k)**: return** (*value*, *class*, *decided*)
v: input value k: threshold value

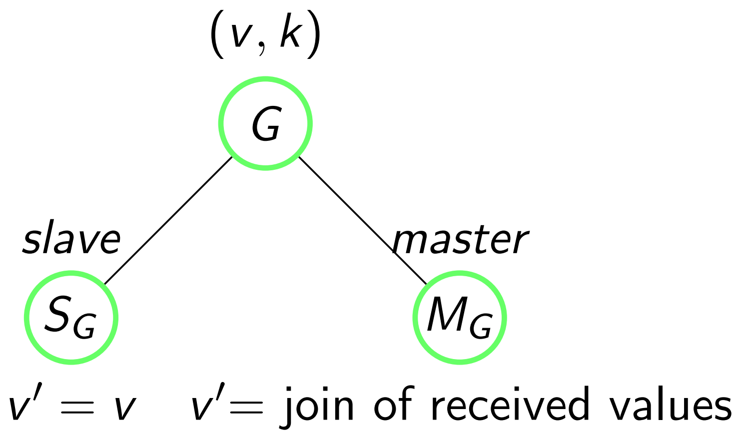
```
Exchange values within the group

/* Early Termination */
if v is comparable with all received values
    return (v, −, true)

/* Classification */
Let w denote the join of all received values
if h(w) > k
    return (w, master, false) //master
else
    return (v, slave, false) //slave
```



The *Classifier* Procedure



Property 1: The value of any slave process $\leq$ the value of any master process, i.e, $\forall p_i \in S_G$ *and* $p_j \in M_G, v_i \leq v_j$.

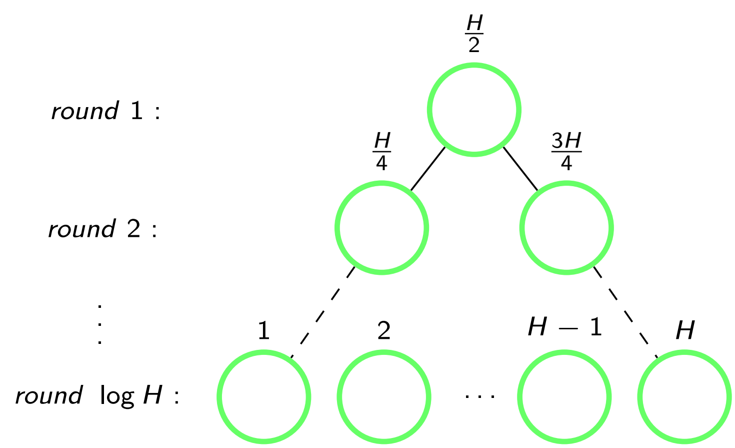
Property 2: The join of all values of slave processes $\leq$ the value of any master process, i.e, $\forall p_j \in M_G$, $v_j \geq \sqcup\{v_i : p_i \in S_G\}$

Algorithm LA_α: height is known

Assumption: the height of the L is known, denoted as H.

$LA_\alpha(H, x_i)$ for p_i:
H: given height x_i: input value

$v_i^1 := x_i$ // value at round 1
$l_i := \frac{H}{2}$ // label
$decided := false$
for $r := 1$ to $\log H + 1$
  $(v_i^{r+1}, class, decided)$
    $:= Classifier(v_i^r, l_i)$
  if $decided$
    return v_i^{r+1}
  else if $class = master$
    $l_i := l_i + \frac{H}{2^{r+1}}$
  else
    $l_i := l_i - \frac{H}{2^{r+1}}$
end for



Correctness: any two processes which decide in two different groups have comparable values and any two processes which decide in the same group have comparable values.

Algorithm LA_β: height is unknown

f is known by assumption

LA_β for p_i
$V_i := \{x_i\}$ // set of values, initially x_i $F_i := \emptyset$ // set of known failure processes $f :=$ the maximum number of failures **Phase A:** Exchange values and record failures Let V_i denote the set of values received Let F_i denote the set of failures /* LA with failure set as input */ **Phase B:** $F_i^{'} := LA_\alpha(f, F_i)$ Remove all values received from processes in $F_i^{'}$ from V_i Output the join of all remaining values in V_i

- Correctness
  Comparable views of failure set gives comparable values.
- Complexity
  Round: $\log f + 1$.
  Message: $n^2 * (\log f + 1)$.

Algorithm LA_γ: height is unkown but expects to be small

```
LAγ for pi
vi := xi // input value
decided := false

Phase A:
Exchange values and take join of all received values

/* Guessing Height */
Phase B:
guess := 2
while (!decided)
      vi := LAα(guess, vi)
      guess := 2 * guess
end while

yi := vi
```

- Complexity

  Round: $\min\{O(\log^2 h(L)), O(\log^2 f)\}$.

  Message: $n^2 \cdot \min\{O(\log^2 h(L)), O(\log^2 f)\}$

Algorithm LA_δ

```
LA_δ for p_i
    acceptVal := x_i // accept value
    learnedVal := ⊥ // learned value

on receiving prop(v_j, r) from p_j:
    if v_j ≥ acceptVal
        Send ACK("accept", −, r)
        acceptVal := v_j
    else
        Send ACK("reject", acceptVal, r)

for r := 1 to f + 1
    val := acceptVal
    Send prop(val, r) to all
    wait for n − f ACK(−, −, r) messages
    let V_r be values contained in reject ACKs
    let tally be number of accept ACKs
    if tally > n/2
        learnedVal := val
        break
    else
        acceptVal := acceptVal ⊔{v | v ∈ V_r}
end for
```

- Correctness
  Claim 1: a process only *accept* comparable values. Any two $n - f$ processes have at least one common process.
  Claim 2: if process p_i does not decide at a round, then the height of its value increases by at least one.
- Complexity
  Round: $min\{h(L), f\}$
  Message: $n^2 \cdot min\{h(L), f\}$

Generalize Lattice Agreement

- Generalized Lattice Agreement [Faleiro et al, 2012, PODC]
  Each process may receive a possibly infinite sequence of values as inputs from a finite lattice. Each process has to learn a sequence of output values with the following properties:
  Validity: Any learned value is a join of some set of inputs.
  Stability: The value learned by any process is non-decreasing.
  Comparability: Any two values learned by any two process are comparable.
  Liveness: Every value received by a correct process is eventually learned by every correct process.

Algorithm GLA_α

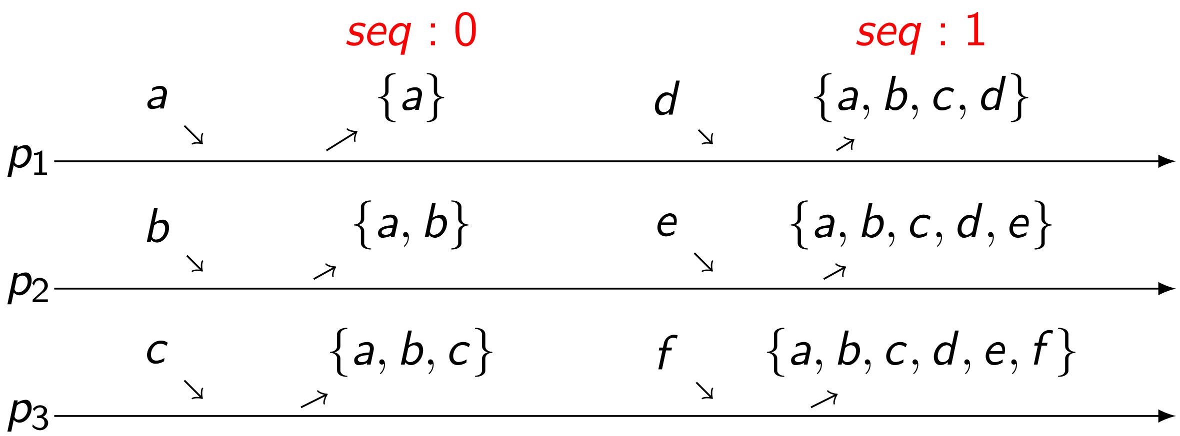
Adapt the lattice agreement protocol for generalized lattice agreement:

- Invoke a lattice agreement instance with a unique sequence number for each value.
- When receiving a value, buffer it until the current lattice agreement instance has finished.
- A process only *accept* a proposal when its current sequence number is higher.

Algorithm GLA_α

Comparability && Stability

- learned values for the same sequence number are comparable.
- learned value for a higher sequence number dominates learned value for a lower sequence number.



Future Work

- For asynchronous systems, is there a $O(\log f)$ algorithm? (In progress)
- Lower bounds for lattice agreement in both synchronous and asynchronous systems.