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.
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    |

\[
\begin{align*}
    & p_1: & add(b) & \rightarrow & read_1 & \rightarrow \\
    & p_2: & add(a) & \rightarrow & read_2
\end{align*}
\]
Road Map

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

Hagit Attiya, Maurice Herlihy, and Ophir achman, 1995, Distributed Computing

Each process $p_i$ has an 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 $i \in [1..n]$, $y_i \leq \sqcup \{x_1, \ldots, x_n\}$.

*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.

\[
\begin{array}{c}
\{a,b,c\} \\
\{a,b\} \quad \{a,c\} \quad \{b,c\} \\
\{a\} \quad \{b\} \quad \{c\} \\
\{\}\ \\
\end{array}
\]
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. $h(L) \leq n$. 

\[X: \{a,b,c\}\]

{a,b} \quad \uparrow \quad \{a,c\} \quad \{b,c\}

{a} \quad \{b\} \quad \{c\}

\{\}\quad \{\}\quad \{\}\quad \{\}\quad \{\}\quad \{\}

\[L: \{a,c\}\]

{a} \quad \{\} \quad \{\}

\{\} \quad \{\} \quad \{\}
## Related Work

### Synchronous systems

| Protocol                  | Time          | Total #Messages            |
|---------------------------|---------------|---------------------------|
| [Attiya et al, 1998, SIAM] | $O(\log n)$  | $O(n^2)$                  |
| [Marios, 2018]            | $\min\{O(h(L)), O(\sqrt{f})\}$ | $n^2 \cdot \min\{O(h(L)), O(\sqrt{f})\}$ |
| $LA_{\alpha}$            | $O(\log h(L))$ | $O(n^2 \log h(L))$       |
| $LA_{\beta}$             | $O(\log f)$  | $O(n^2 \log f)$          |
| $LA_{\gamma}$            | $\min\{O(\log^2 h(L)), O(\log^2 f)\}$ | $n^2 \cdot \min\{O(\log^2 h(L)), O(\log^2 f)\}$ |

### Asynchronous systems

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

$n$: 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
1: Exchange values within the group
   /* Early Termination */
2: if v is comparable with all received values
3:   return (v, -, true)
   /* Classification */
4: Let w denote the join of all received values
5: if h(w) > k
6:   return (w, master, false) //master
7: else
8:   return (v, slave, false) //slave
```
**The Classifier Procedure**

![Diagram of Classifier Procedure]

\[ (v, k) \]

\[ G \]

\[ S_G \]

\[ M_G \]

\[ \nu' = v \quad \nu' = \text{join of received values} \]

**Property 1:** The value of any slave process \( \leq \) the value of any master process, i.e, \( \forall p_i \in S_G \text{ and } p_j \in M_G, \nu_i \leq \nu_j \).

**Property 2:** The join of all values of slave processes \( \leq \) the value of any master process, i.e, \( \forall p_j \in M_G, \nu_j \geq \sqcup \{ \nu_i : p_i \in S_G \} \)
Algorithm $LA_{\alpha}$: 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
1: v_1 \equiv x_i \quad \text{// value at round 1}
2: l_i \equiv \frac{H}{2} \quad \text{// label}
3: decided \equiv false
4: for r := 1 to \log H + 1
5:   (v_{r+1}^i, \text{class}, \text{decided})
6:   \equiv \text{Classifier}(v_{r}^i, l_i)
7:   if decided
8:     return v_{r+1}^i
9:   else if class = master
10:     l_i \equiv l_i + \frac{H}{2^{r+1}}
11:   else
12:     l_i \equiv l_i - \frac{H}{2^{r+1}}
13: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 $LA_\beta$: height is unknown

$f$ is known by assumption

```
LA_\beta$ 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**: $n^2 \times (\log f + 1)$. 
Algorithm $LA_\gamma$: height is unknow but expects to be small

```
$LA_\gamma$ for $pi$
$v_i := x_i$ // input value
decided := false

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

/* Guessing Height */
Phase B:
guess := 2
while (!decided)
    $v_i := LA_\alpha(guess, v_i)$
    guess := 2 * guess
end while

$y_i := v_i$
```

- **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 $\text{LA}_\delta$

$\text{LA}_\delta$ for $p_i$
\begin{align*}
\text{acceptVal} & := x_i \quad \text{// accept value} \\
\text{learnedVal} & := \bot \quad \text{// learned value}
\end{align*}

\textbf{on receiving} $\text{prop}(v_j, r)$ \textbf{from} $p_j$:
\begin{align*}
\text{if } v_j & \geq \text{acceptVal} \\
& \quad \text{Send } \text{ACK}(\text{"accept"}, - , r) \\
& \quad \text{acceptVal} := v_j \\
\text{else} & \\
& \quad \text{Send } \text{ACK}(\text{"reject"}, \text{acceptVal}, r)
\end{align*}

\textbf{for} $r := 1 \text{ to } f + 1$
\begin{align*}
\text{val} & := \text{acceptVal} \\
& \quad \text{Send } \text{prop}(\text{val}, r) \text{ to all} \\
& \quad \text{wait for } n - f \text{ ACK}(-, -, r) \text{ messages} \\
& \quad \text{let } V_r \text{ be values contained in reject ACKs} \\
& \quad \text{let } \text{tally} \text{ be number of accept ACKs} \\
\text{if } \text{tally} & > \frac{n}{2} \\
& \quad \text{learnedVal} := \text{val} \\
& \quad \text{break} \\
\text{else} & \\
& \quad \text{acceptVal} := \text{acceptVal} \cup \{v \mid v \in V_r\}
\end{align*}

\textbf{Correctness}

\textbf{Claim 1}: a process only \textit{accept} comparable values. Any two $n - f$ processes have at least one common process.

\textbf{Claim 2}: if process $p_i$ does not decide at a round, then the height of its value increases by at least one.

\textbf{Complexity}

\textbf{Round}: $\min\{h(L), f\}$

\textbf{Message}: $n^2 \cdot \min\{h(L), f\}$
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 $GLA_\alpha$

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.
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.

seq : 0

\[
p_1: a \rightarrow \{a\} \quad p_2: b \rightarrow \{a, b\} \quad p_3: c \rightarrow \{a, b, c\}
\]

seq : 1

\[
p_1: a \rightarrow \{a\} \quad p_2: e \rightarrow \{a, b, e\} \quad p_3: f \rightarrow \{a, b, c, d, e, f\}
\]
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.