On bounded version vectors

Madhavan Mukund, Gautham Shenoy R, S P Suresh

Technical Report

Chennai Mathematical Institute

August 2012

Abstract

In this thesis we explore bounded representations for **version vectors** which are used to decide the most up to date replica in an eventually consistent system. We look at an existing bounded representation and reason about its correctness using a partial order framework. We use the insights from this analysis along with a solution for the **gossip problem** to come up with a more efficient bounded representation of version vectors.

Contents

1

Introduction

Brewer's CAP Theorem [GL02] states that distributed systems which are required to be highly available and partition tolerant cannot guarantee that the replicas are strongly consistent. In such cases, such systems make do with weaker notions of consistency. A popular weaker notion of consistency is *eventual consistency*, which allows for the states of the replicas to diverge for a finite (but not necessarily bounded) period of time with a guarantee that the states will eventually converge. The properties of such *eventually consistent* systems have been studied in [SS05].

The replicas in an eventually consistent systems perform updates locally and propagate the updates through mechanisms which include *epidemic propagation* and *pairwise synchronization*. Whenever a subset of replicas participate in the exchange of their knowledge about the local states of other replicas in the system, they require a mechanism to decide which amongst the partipants has the most updated information pertaining to every other replica in the system. *Version vectors* play an important role in this decision-making process.

A typical implementation of version vectors uses integer counters, which grow with time and thus are not bounded. However, we observe that given two replicas a and b with their states r_a and r_b, respectively, there can be only four possible relationships between them.

1. r_a is more up to date than r_b.
2. r_b is more up to date than r_a.
3. r_a and r_b cannot be compared since they have received concurrent updates.
4. r_a is the same as r_b.

Moreover, when these two replicas participate in *pairwise synchronization*, they jointly derive the new version vector that captures their state merger by locally comparing their respective version vectors.

The bounded number of relations they capture, and the fact that they can be locally computed without the need for the global view, provides the motivation to seek the existence of a finite representation for version vectors. One such finite representation for version vectors was presented in [AAB04].

For the first part of this thesis, we follow the presentation of [AAB04]. In that part, we also introduce the trace framework using which we shall reason about the properties of the bounded representation proposed in that paper. In the second part of the thesis, we shall use this gossip framework to provide an alternative bounded representation for version vectors.

2

Version vectors

2.1 Definitions

Consider an eventually consistent distributed system with N replicas $\{r_0, r_1, \dots, r_{N-1}\}$. Having initialized themselves by executing the initialization operation I, each of the replicas can perform one of the following operations:

- When a client requests an operation to be performed, replica r_i performs the update locally. This is denoted by U^i.
- Periodically, replicas participate in *pairwise synchronization*, where they exchange their states and arrive at a common state to reflect this exchange and update of knowledge. Pairwise synchronization between r_i and r_j is denoted by S^{ij}.

A finite sequence of operations performed by the distributed system is known as a *run*. (It is referred to as a *trace* in [AAB04], but we reserve that name for its more standard meaning, and use *run* here, which is also standard terminology.) We denote a run by $\alpha = I o_1 o_2 \dots o_n$, where o_i is either an update operation or a synchronization operation.

A *version vector* of replica r_i at the end of a run α is a vector $V_i(\alpha)$ of N integer counters. The jth entry of $V_i(\alpha)$, denoted by $V_i^j(\alpha)$, represents the most recent update of replica j that i is aware of, at the end of α. We say that a version vector $V_i(\alpha)$ *dominates* $V_j(\alpha)$ iff

$$\forall k \in [0 \dots N-1]: \quad V_i^k(\alpha) \geq V_j^k(\alpha)$$

Whenever $V_i(\alpha)$ dominates $V_j(\alpha)$ and $V_i(\alpha) \neq V_j(\alpha)$, sematically it means that at the end of a run α, replica r_i is more up to date than r_j, which implies that r_i has received all the updates that r_j has received.

We also define the pointwise join operation to be:

$$V_i^k(\alpha) \sqcup V_j^k(\alpha) = max(V_i^k(\alpha), V_j^k(\alpha))$$

The semantics of the version vectors for the various operations are presented in Table 2.1.

Example 1. *Consider a distributed system with three replicas denoted by* $\{0, 1, 2\}$. *The sequence of operations and the values of the version vectors of the three replicas is presented in Table 2.2.*

Operation	Semantics
Initialization	$V_i^k(I) = 0$
Update	$V_i^k(\alpha \cdot U^a) = \begin{cases} V_i^k(\alpha)+1 & \text{if } k=i=a \\ V_i^k(\alpha) & \text{otherwise} \end{cases}$
Synchronization	$V_i^k(\alpha \cdot S^{ab}) = \begin{cases} V_a^k(\alpha) \sqcup V_b^k(\alpha) & \text{if } i \in \{a,b\} \\ V_i^k(\alpha) & \text{otherwise} \end{cases}$

Table 2.1: Semantics of version vectors

Operation	Replica 0	Replica 1	Replica 2
I	[0,0,0]	[0,0,0]	[0,0,0]
U^0	[1,0,0]	[0,0,0]	[0,0,0]
U^2	[1,0,0]	[0,0,0]	[0,0,1]
S^{12}	[1,0,0]	[0,0,1]	[0,0,1]
S^{01}	[1,0,1]	[1,0,1]	[0,0,1]
S^{12}	[1,0,1]	[1,0,1]	[1,0,1]

Table 2.2: Evolutions of version vectors

2.2 Version vector matrix

We saw that version vectors of a replica is used to model the knowledge that the replica possesses about the most recent update about every other replica in the system. Similarly, we can define a second order object that models the knowledge that the replica possesses about the most recent version vector of every other replica in the system. Since this second order object also contains the version vector of the replica itself, it can also capture data-causality amongst replicas. We term such a second order object as a *version vector matrix*.

Formally, a *version vector matrix* for replica i at the end of a run α is defined to be an $N \times N$ matrix of integer counters, denoted by $M_i(\alpha)$ such that the j^{th} row of $M_i(\alpha)$ denoted by $M_i[j](\alpha)$ represents r_i's knowledge of the version vector of r_j. Thus $M_i[i](\alpha)$ represents the version vector for replica i. We denote the k^{th} entry in the j^{th} row of $M_i(\alpha)$ as $M_i[j]^k(\alpha)$.

We define a vector join operation for any two rows of any two version vector matrices $M_i[k](\alpha)$ and $M_j[l](\alpha)$, denoted by $(M_i[k] \sqcup M_j[l])(\alpha)$, as follows:

$$\forall p \in [0 \ldots N-1] : (M_i[k] \sqcup M_j[l])^p(\alpha) = M_i[k]^p(\alpha) \sqcup M_j[l]^p(\alpha)$$

The semantics for version vector matrix is presented in Table 2.3.

Proposition 2. *For any run α and any replica r_i,*

$$M_i[i](\alpha) = V_i(\alpha).$$

Operation	Semantics
Initialization	$M_i[j]^k(I) = 0$
Update	$M_i[j]^k(\alpha \cdot U^a) = \begin{cases} M_i[j]^k(\alpha)+1 & \text{If } i=j=k=a \\ M_i[j]^k(\alpha) & \text{otherwise} \end{cases}$
Synchronization	$M_i[j](\alpha \cdot S^{ab}) = \begin{cases} (M_a[a] \sqcup M_b[b])(\alpha) & \text{if } i,j \in \{a,b\} \\ (M_a[j] \sqcup M_b[j])(\alpha) & \text{if } i \in \{a,b\} \\ (M_i[j](\alpha) & \text{otherwise} \end{cases}$

Table 2.3: Semantics of version vector matrix

Proof. Induction on length of α. $\square$

Our goal is to arrive at a bounded representation for version vector matrices. Since version vector matrices encode version vectors (from Proposition 2), it follows that we have a bounded representation for version vectors themselves. To obtain a bounded representation for version vector matrices, we need to see them in a new light.

2.3 Version vector slices

Let $M_i(\alpha)$ be a version vector matrix for r_i at the end of a run α. The k^{th} column of M_i denotes what r_i knows about the updates originated from replica k that have been propagated to other replicas of the system. We term such a vertical slice of the version vector matrix a *version vector slice* (VVS). The k^{th} slice of M_i at the end of a run α is denoted by $C_i^k(\alpha)$. The j^{th} entry of the version vector slice C_i^k at the end of a run α is denoted by $C_i^k[j](\alpha)$.

Thus a version vector matrix can be thought of as a concatenation of version vector slices. In other words, $M_i(\alpha) = \langle C_i^0(\alpha), C_i^1(\alpha), \dots C_i^{N-1}(\alpha)\rangle$.

We define the operational semantics for version vector slices in table 2.4. These are derived from the semantics of version vector matrices.

In order to show a bounded representation for version vector matrices, it suffices to show that there exists a bounded representation for a version vector slice. In subsequent sections of the first part of this work, we shall obtain obtain a bounded representation for the version vector slice C_i^0. Hence we shall only concern ourselves with update events of replica r_0.

Operation	**Semantics**
Initialization	$(C_i^k[j])(I) = 0$
Update	$(C_i^k[j])(\alpha \cdot U^a) = \begin{cases} C_i^k[j](\alpha)+1 & \text{If } i=j=k=a \\ C_i^k[j](\alpha) & \text{otherwise} \end{cases}$
Synchronization	$(C_i^k[j])(\alpha \cdot S^{ab}) \begin{cases} C_a^k[a](\alpha) \sqcup C_b^k[b](\alpha) & \text{if } i,j \in \{a,b\} \\ C_a^k[j](\alpha) \sqcup C_b^k[j](\alpha) & \text{if } i \in \{a,b\} \\ C_i^k[j](\alpha) & \text{otherwise} \end{cases}$

Table 2.4: Semantics of version vector slices

3

Traces - An Introduction

In this section, introduce the framework of traces. We shall use this frame work to model the interactions between the replicas in a natural manner. The framework provides us a natural way to reason about the results given in [AAB04].

3.1 Traces of runs

Let $\alpha = I o_1 o_2 \dots o_n$ be a run. We define $Evs(\alpha)$ to be $\{e_\perp, e_1, \dots, e_n\}$, the set of events at which the operations in α occur. We define $Ops(\alpha)$ to be $\{I, o_1, \dots, o_n\}$, the set of operations in α. For replica r_k ($k \in \{0, 1, \dots, N-1\}$), $\alpha \downarrow k$ denotes to be the maximal subsequence of α, $I o_{j_1} o_{j_2} \dots o_{j_m}$ such that k is invoved in each of the operations o_{j_i}. This subsequence defines a linear order $\leq_k$ over the operations in $\alpha \downarrow k$.

Definition 3. *Given a run $\alpha = I o_1 \dots o_n$, its* **trace***, denoted $t(\alpha)$, is a triple $(\mathcal{E}, \leq, \lambda)$ such that:*

- $\mathcal{E} = \mathcal{E}(\alpha)$
- $\lambda : \mathcal{E} \to Ops(\alpha)$ *such that* $\lambda(e_\perp) = I$ *and* $\lambda(e_i) = o_i$ *for* $i \in \{1, \dots, n\}$
- $\leq$ *is the least partial order on* $\mathcal{E}$ *such that*

$$\exists k (\lambda(e_i) \leq_k \lambda(e_j)) \Rightarrow e_i \leq e_j.$$

A triple $t = (\mathcal{E}, \leq, \lambda)$ is a **trace** *if $t = t(\alpha)$ for some run α.*

Example 4. *Consider a distributed system with replicas $[0, 1, 2, 3]$.*
Let the run $\alpha = I \cdot U^0 \cdot U^2 \cdot S^{01} \cdot S^{23} \cdot S^{02} \cdot S^{13}$.

$$\alpha \downarrow 0 = I \cdot U^0 \cdot S^{01} \cdot S^{02}$$

$$\alpha \downarrow 1 = I \cdot S^{01} \cdot S^{13}$$

$$\alpha \downarrow 2 = I \cdot U^2 \cdot S^{23} \cdot S^{02}$$

$$\alpha \downarrow 3 = I \cdot S^{23} \cdot S^{13}$$

The trace of α, $t(\alpha)$, is given in figure 3.1.

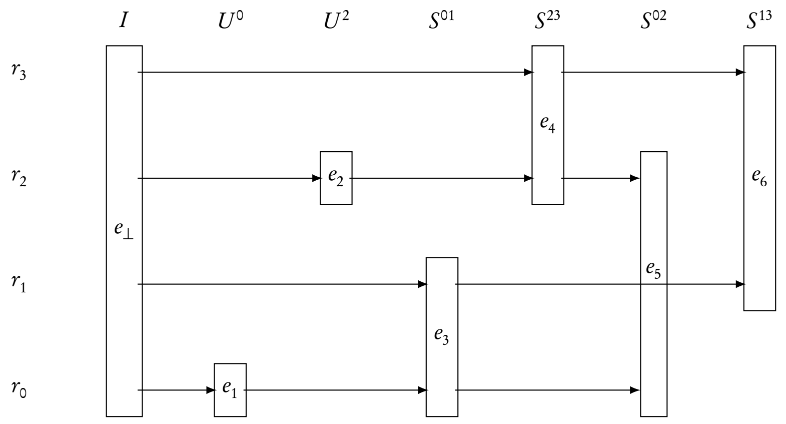

Figure 3.1: Trace for the run $\alpha = I \cdot U^0 \cdot U^2 \cdot S^{01} \cdot S^{23} \cdot S^{02} \cdot S^{13}$ in example.

Definition 5. *For a trace $t = (\mathscr{E}, \leq, \lambda)$ and $S \subseteq \mathscr{E}$, the subtrace of t induced by S is given by $t(S) = (S, \leq_S, \lambda_S)$ where:*

- $\leq_S = \leq \cap (S \times S)$
- $\lambda_S(e) = \lambda(e)$ for all $e \in S$

In this representation, for $i \neq j$, $e_i \leq e_j$ implies that the event e_j occurs strictly after e_i and thus the replicas participating in $\lambda(e_j)$ know about the occurrence of the event e_i. This transfer of this knowledge can be traced along the path from e_i to e_j in the trace. We formalize this notion by defining *ideals*.

3.2 Ideals

Definition 6. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace of the run α. A subset $\mathscr{I} \subseteq \mathscr{E}$ is said to be an* **ideal** *iff*

$$\forall e_i \in \mathscr{I} \text{ and } e_j \leq e_i, e_j \in \mathscr{I}.$$

Thus an ideal is a downward closed subset of $\mathscr{E}$ with respect to the partial order $\leq$.

Example 7. *In the run α mentioned in example 4, the subsets $\{e_\perp\}$, $\{e_\perp, e_1\}$, $\{e_\perp, e_2\}$, $\{e_\perp, e_1, e_3\}$, $\{e_\perp, e_2, e_4\}$, $\{e_\perp, e_1, e_2, e_3, e_4, e_5\}$, $\{e_\perp, e_1, e_2, e_3, e_4, e_6\}$ and $\{e_\perp, e_1, e_2, e_3, e_4, e_5, e_6\}$ are all ideals.*

We state a few properties of ideals:

Proposition 8. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace of some run α. Then the following are true:*

1. *$\mathscr{E}$ is an ideal.*
2. *If $e_i \in \mathscr{E}$, the set $\downarrow e_i = \{e_j | e_j \leq e_i\}$ is an ideal. (It is referred to as* **the ideal generated by** *e_i.)*
3. *Every ideal $\mathscr{I}$ is generated by its maximal elements. In other words $\mathscr{I} = \bigcup_{e_i \in \sup \mathscr{I}} \downarrow e_i$.*
4. *If $\mathscr{I}$ and $\mathscr{J}$ are ideals, so are $\mathscr{I} \cup \mathscr{J}$ and $\mathscr{I} \cap \mathscr{J}$.*

Our goal is to make use of the trace framework to model the flow of information amongst the replicas in the system. To that end, we define some terms which we shall use often in the proofs.

Definition 9. *Let $\mathscr{I}$ be an ideal and i be replica. An event e is the* **maximal event** *of i in $\mathscr{I}$, denoted by $max_i(\mathscr{I})$, iff $f \leq e$ for all i-events f. If $t = (\mathscr{E}, \leq, \lambda)$ is a trace, then we use the notation $max_i(t)$ to denote $max_i(\mathscr{E})$.*

Definition 10. *The* **view** *of a replica i in an ideal $\mathscr{I}$, denoted by $\partial_i(\mathscr{I})$, is $\downarrow max_i(\mathscr{I})$, the ideal generated by the maximal i-event in $\mathscr{I}$. For a trace $t = (\mathscr{E}, \leq, \lambda)$ and event $e \in \mathscr{E}$, we use the notations $\partial_i(t)$ and $\partial_i(e)$ to denote $\partial_i(\mathscr{E})$ and $\partial_i(\downarrow e)$, respectively.*

Definition 11. *Let $\mathscr{I}$ be an ideal and i and j be two replicas in the system. The* **latest information** *that i has about j in $\mathscr{I}$, denoted by $latest_{i \to j}(\mathscr{I})$, is defined to be $max_j(\partial_i(\mathscr{I}))$, the maximal j-event in the view of i. If $t = (\mathscr{E}, \leq, \lambda)$ is a trace, then we use the notation $latest_{i \to j}(t)$ to denote $latest_{i \to j}(\mathscr{E})$.*

Definition 12. *Let $\mathscr{I}$ be an ideal and i be some replica in the system. The* **latest update** *event of i in $\mathscr{I}$, denoted $update_i(\mathscr{I})$, is $max\{e \in \mathscr{I} : \lambda(e) = U^i\}$, the maximal i-event in $\mathscr{I}$ which is also an update event. For a trace $t = (\mathscr{E}, \leq, \lambda)$ and event $e \in \mathscr{E}$, we use the notations $update_i(t)$ and $update_i(e)$ to denote $update_i(\mathscr{E})$ and $update_i(\downarrow e)$, respectively.*

From the semantics of version vector matrices, we can observe that the value of the entry $M_k[k]^k$ increments during a k-update event.

Proposition 13. *1. If e_1 and e_2 are such that $e_1 \leq e_2$, then $update_i(e_1) \leq update_i(e_2)$.*

2. If $update_i(e_1) \leq e_2$, then $update_i(e_1) \leq update_i(e_2)$.

3. If e_1 and e_2 are j-events and $update_i(e_1) < update_i(e_2)$ then $e_1 < e_2$.

Proof. 1. Since $e_1 \leq e_2$, $update_i(e_1) = max\{e \in e_1 : \lambda(e) = U^i\} \leq max\{e \in e_2 : \lambda(e) = U^i\} = update_i(e_2)$.

2. This follows from the previous part and the observation that

$$update_i(update_i(e_1)) = update_i(e_1).$$

3. Since $update_i(e_1) < update_i(e_2)$, it follows that $update_i(e_2) \not\leq update_i(e_1)$, and hence (from item 1) that $e_2 \not\leq e_1$. But e_1 and e_2 are j events, and hence comparable. Thus it follows that $e_1 < e_2$.

□

Going back to the semantics of version vector matrices, at the end of a run α, the value $M_i[j]^k(\alpha)$ captures the knowledge that replica i has of the latest k-update known to replica j. We shall express this notion using the framework of traces following which we can resort to comparing corresponding trace events using the partial order instead of comparing the integer entries in the version vector matrices.

4

Version Vector Matrices and Trace-events

4.1 Relationship between Version vector matrix entries and trace events

In this section, we shall show why the trace-representation of a run captures the behaviour of communicating replicas more effectively than the runs themselves. To do this, we define the notion of linearization of a trace:

Definition 14. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace. We say that η is a* **linearization** *of t iff $\eta = e_{i_0} \cdot e_{i_1} \cdot \cdots \cdot e_{i_n}$ is some permutation of all the events in $\mathscr{E}$ such that if $e_{i_j} \leq e_{i_k}$ in t then $j \leq k$.*

Note: One can observe that η is a linearization of t iff $t = t(\lambda(\eta))$.

We want to define the notion of version vector matrices over traces using the semantics of version vector matrices over runs. To that end, we take a note of the following property about the version vector matrices over runs.

Proposition 15. *Let α and α' be runs such that for any replica i, $M_i(\alpha) = M_i(\alpha')$. Let o_1 and o_2 be operations such that the sets of participating replicas of o_1 and o_2 are disjoint. Then, for any replica i, $M_i(\alpha o_1 o_2) = M_i(\alpha' o_2 o_1)$.*

Proof. There are three cases to consider.

i does not participate in either o_1 or o_2: In this case $M_i(\alpha o_1) = M_i(\alpha o_2) = M_i(\alpha)$ and $M_i(\alpha' o_1) = M_i(\alpha' o_2) = M_i(\alpha')$. Hence, $M_i(\alpha o_1 o_2) = M_i(\alpha) = M_i(\alpha') = M_i(\alpha' o_2 o_1)$.

i participates in o_1 but not in o_2: In this case $M_i(\alpha' o_2) = M_i(\alpha') = M_i(\alpha)$. Hence $M_i(\alpha' o_2 o_1) = M_i(\alpha o_1)$. Similarly, $M_i(\alpha o_1 o_2) = M_i(\alpha o_1)$ since M_i doesn't get modified during o_2. Thus we see that $M_i(\alpha o_1 o_2) = M_i(\alpha' o_2 o_1)$.

i participates in o_2 but not in o_1: This case is similar to the above.

□

Definition 16. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace and let η be some linearization of t. For a replica i, the version vector matrix of i over t, denoted by $M_i(t)$, is defined to be $M_i(\lambda(\eta))$.*

Since t can have multiple linearizations, we need to show that our notion of version vector matrices over traces as presented here is well defined.

Lemma 17. *Let t be a trace and η_1 and η_2 be any linearizations of t. For any replica i, $M_i(\lambda(\eta_1)) = M_i(\lambda(\eta_2))$.*

Proof. We show this by an induction over the size of the trace t.

Suppose $t = (\{e_\perp\}, \leq, \lambda)$. Then $\eta_1 = \eta_2 = e_\perp$. Thus for any replica i, $M_i(\lambda(\eta_1)) = M_i(\lambda(\eta_2)) = M_i(I)$. Suppose the result is true for all traces of size smaller than n and let $t = (\mathscr{E}, \lambda, \leq)$ be a trace whose size is n. Let us denote the set of all maximal elements in t by Max_t. Let t' be the trace $t(\mathscr{E} \setminus \mathrm{Max}_t)$. Now any linearization η of t can be written as $\eta'\eta''$ where η' is some linearization of t' and η'' is some permutation of Max_t. Thus we can write $\eta_1 = \eta_1'\eta_1''$ and $\eta_2 = \eta_2'\eta_2''$ where η_1' and η_2' are linearizations of t' and η_1'' and η_2'' are permutations of Max_t. Thus $M_i(\lambda(\eta_1)) = M_i(\lambda(\eta_1'\eta_1''))$ and $M_i(\lambda(\eta_2)) = M_i(\lambda(\eta_2'\eta_2''))$. Since any pair of events $e_j, e_k \in \mathrm{Max}_t$ are incomparable, the sets of replicas participating in $\lambda(e_j)$ and $\lambda(e_k)$ are disjoint. Since by induction hypothesis, $M_i(\lambda(\eta_1')) = M_i(\lambda(\eta_2'))$, by Proposition 15, we get $M_i(\lambda(\eta_1'\eta_1'')) = M_i(\lambda(\eta_2'\eta_2''))$. Thus $M_i(\lambda(\eta_1)) = M_i(\lambda(\eta_2))$. □

Definition 18. *If i is any replica and e is some event in trace t and $t' = t(\downarrow e)$ the sub-trace associated with the ideal $\downarrow e$, then define $M_i(e) = M_i(t')$.*

Proposition 19. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace and i be a replica. Let e^i_{max} be the maximal i-event in t. Then, $M_i(t) = M_i(e^i_{max})$. .*

Proof. Let $t' = (\downarrow e^i_{max}, \leq_{e^i_{max}}, \lambda_{e^i_{max}})$ be the trace associated with the view of i in t. Observe that events in $\mathscr{E} \setminus \downarrow e^i_{max}$ do not lie below e^i_{max}. Thus there is a linearization of t of the form $\eta' \cdot \eta''$, where η' is a linearization of t' and η'' is a linearization of $t(\mathscr{E} \setminus \downarrow e^i_{max})$. Hence $M_i(t) = M_i(\lambda(\eta') \cdot \lambda(\eta''))$. Since η'' does not contain any i-operation, $M_i(\lambda(\eta') \cdot \lambda(\eta'')) = M_i(\lambda(\eta'))$. But this means that $M_i(t) = M_i(t')$. □

We now formalise our intuition which associates the entries of the version vector matrices with the events in the partial order.

Definition 20. *For a trace $t = (\mathscr{E}, \leq, \lambda)$, $lu^k_{i\to j}(t) \overset{\text{def}}{=} update_k(latest_{i\to j}(t))$.*

Theorem 21. *If t is a trace and i, j, k are replicas then $M_i[j]^k(t) = M_k[k]^k(lu^k_{i\to j}(t))$.*

Proof. We shall prove this result by induction on the size of the trace t. Suppose t contains only one event, I. Then, from the semantics of version vector matrices, for every i, j, k, $M_i[j]^k(t) = 0$. Also, since for every i, j, k, $lu^k_{i\to j}(t) = e_\perp$, and the trace corresponding to $\downarrow e_\perp$ is t itself, it follows that $M_k[k]^k(lu^k_{i\to j}(t)) = 0$.

Assume that for all the traces whose size is smaller than n, the result holds, and let $t = (\mathscr{E}, \leq, \lambda)$ be a trace of size n. Let e_{max} be some maximal event in t, and $o_{max} = \lambda(e_{max})$. Define $\mathscr{E}' = \mathscr{E} \setminus \{e_{max}\}$ and $t' = t(\mathscr{E}')$. By induction hypothesis, for any i, j, k, $M_i[j]^k(t') = M_k[k]^k(lu^k_{i\to j}(t'))$. There are the following cases to consider.

i does not participate in o_{max}: In this case $M_i[j]^k(t) = M_i[j]^k(t')$ and $lu^k_{i\to j}(t) = lu^k_{i\to j}(t')$. Thus $M_i[j]^k(t) = M_k[k]^k(lu^k_{i\to j}(t))$.

$o_{max} = U^i$ **and** $i \notin \{j,k\}$: We argue as in the previous case.

$o_{max} = U^i$ **and** $j = k = i$: From Proposition 19, we have $M_i[j]^k(t) = M_i[j]^k(e_{max})$. Since $i = j = k$ and since by definition, $lu^k_{i\to j}(t) = e_{max}$, $M_i[j]^k(t) = M_k[k]^k(lu^k_{i\to j}(t))$.

$o_{max} = S^{il}$ **and** $j \in \{i,l\}$: In this case, $M_i[j]^k(t) = max(M_i[i]^k(t'), M_l[l]^k(t'))$. By induction hypothesis, this is the same as $max(M_k[k]^k(lu^k_{i\to i}(t')), M_k[k]^k(lu^k_{l\to l}(t')))$. Since e_{max} is not an update event, $lu^k_{i\to j}(t) = max(lu^k_{i\to i}(t'), lu^k_{l\to l}(t'))$. It follows that $M_i[j]^k(t) = M_k[k]^k(lu^k_{i\to j}(t))$.

$o_{max} = S^{il}$ **and** $j \notin \{i,l\}$: In this case, $M_i[j]^k(t) = max(M_i[j]^k(t'), M_l[j]^k(t'))$. By induction hypothesis, this is the same as $max(M_k[k]^k(lu^k_{i\to j}(t')), M_k[k]^k(lu^k_{l\to j}(t')))$. Since e_{max} is not an update event, $lu^k_{i\to j}(t) = max(lu^k_{i\to j}(t'), lu^k_{l\to j}(t'))$. It follows that $M_i[j]^k(t) = M_k[k]^k(lu^k_{i\to j}(t))$.

□

Corollary 22. *If t is a trace and a,b,i,j,k are replicas then*

- $M_a[i]^k(t) < M_b[j]^k(t)$ *iff* $lu^k_{a\to i}(t) < lu^k_{b\to j}(t)$.
- $M_a[i]^k(t) = M_b[j]^k(t)$ *iff* $lu^k_{a\to i}(t) = lu^k_{b\to j}(t)$.

Proof. $M_a[i]^k(t) \le M_b[j]^k(t)$ iff $M_k[k]^k(lu^k_{a\to i}(t)) \le M_k[k]^k(lu^k_{b\to j}(t))$ (from theorem 21) iff $lu^k_{a\to i}(t) \le lu^k_{b\to j}(t)$ (since $lu^k_{a\to i}(t)$ and $lu^k_{b\to j}(t)$ are both k-update events). The statements in the corollary follow from this. □

4.2 Primary and secondary update events

We begin our quest for bounded representation by modelling the 0th slice of version vector matrix M_a for each replica a. From Theorem 21 we know that for any trace t, each entry $M_a[i]^0(t)$ of the version vector matrix has an associated 0-update event: $lu^0_{a\to i}(t)$. We shall call these events the **primary update events** for replica a in t. We shall denote this set of events by $PrimaryUpdate^0_a(t)$. The event $lu^0_{a\to a}(t)$, which is the latest 0-update event that a knows about shall be called the **principal event** for a.

We shall next state a few properties about the primary update events which we shall use repeatedly later.

Lemma 23. *Given replicas a,b and k, and a trace $t = (\mathcal{E}, \le, \lambda)$,*

1. $lu^0_{a\to b}(t) \le lu^0_{b\to b}(t)$
2. $lu^0_{a\to a}(t) \le lu^0_{0\to 0}(t)$
3. $lu^0_{a\to k}(t) \le lu^0_{a\to a}(t)$

Proof. 1. Since $latest_{a\to b}(t) \leq max_b(t) = latest_{b\to b}(t)$, by proposition 13 and by the definition of primary update events, we get $lu^0_{a\to b}(t) \leq lu^0_{b\to b}(t)$.

2. Since $lu^0_{a\to a}(t)$ is a 0-event, $lu^0_{a\to a}(t) \leq max_0(t)$. By proposition 13, $update_0(\downarrow lu^0_{a\to a}(t)) \leq update_0(max_0(t))$. But since $lu^0_{a\to a}(t)$ is a 0-update event, $update_0(lu^0_{a\to a}(t)) = lu^0_{a\to a}(t)$. Also, by definition, $update_0(max_0(t)) = lu^0_{0\to 0}(t)$. Hence $lu^0_{a\to a}(t) \leq lu^0_{0\to 0}(t)$ as required.

3. This follows from the fact that $latest_{a\to k}(t) \leq latest_{a\to a}(t)$ and Proposition 13 and the definitions.

□

Definition 24. *For a trace $t = (\mathcal{E}, \leq, \lambda)$, $latest_{a\to b\to c}(t) \overset{\text{def}}{=} latest_{b\to c}(\partial_a(t))$ and $lu^k_{a\to b\to c}(t) \overset{\text{def}}{=} update_k(latest_{a\to b\to c}(t))$.*

Observation 1. *1. $latest_{a\to b}(t) = latest_{a\to b\to b}(t) = latest_{a\to a\to b}(t)$.*

2. If $latest_{a\to b}(t) \leq latest_{c\to d}(t)$ then $latest_{a\to b}(t) \leq latest_{c\to d\to b}(t)$.

Note that for a given replica a, the size of $\{lu^0_{a\to i\to j}(t) \mid i, j < N\}$ is bounded by N^2. (Recall that N is the number of replicas.)

We now prove a few key results about how the various events of the form $lu^0_{a\to b}(t)$ and $lu^0_{a\to b\to c}(t)$ relate to each other. Preliminary to that we shall state and prove the following important lemma on paths that begin inside an ideal and end outside it. This lemma shall be used extensively in the rest of the paper.

Lemma 25 (Crossover point lemma). *Let $\mathcal{I} \subseteq \mathcal{E}$ be an ideal in a trace, and a be a replica. For events e_1 and e_2 such that $e_1 \in \partial_a(\mathcal{I})$, $e_2 \in \mathcal{I} \setminus \partial_a(\mathcal{I})$ and $e_1 < e_2$, there exists a replica c such that $e_1 \leq latest_{a\to c}(\mathcal{I}) \leq e_2$.*

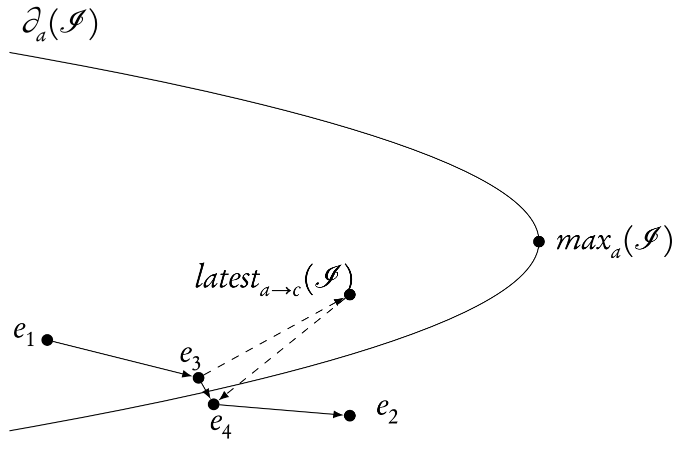

Figure 4.1: Crossover point lemma where e_3 and e_4 are both c-events and $e_1 \cdots \to e_3 \to e_4 \cdots \to e_2$ is some path from e_1 to e_2.

Proof. Let P be any path from e_1 to e_2 and let e_3 be the maximal $\partial_a(\mathcal{I})$-event on this path. Let e_4 be the minimal element along P such that $e_4 \notin \partial_a(\mathcal{I})$. Clearly $e_1 \leq e_3$ and $e_4 \leq e_2$. Also, there is an edge from e_3 and e_4, and this means that there is a replica c such that both e_3 and e_4 are c events.

Clearly $e_3 \leq latest_{a\to c}(\mathscr{I})$. Since $e_4 \notin \partial_a(\mathscr{I})$, $e_4 \not\leq latest_{a\to c}(\mathscr{I})$. But since e_4 is a c-event, $latest_{a\to c}(\mathscr{I}) < e_4$. Thus $e_1 \leq latest_{a\to c}(\mathscr{I}) \leq e_2$. □

Lemma 26. *For any trace $t = (\mathscr{E}, \leq, \lambda)$ and all replicas a, b, there exist replicas c, d such that $lu^0_{a\to b}(t) = lu^0_{0\to c\to d}(t)$.*

Proof. We need to consider the following cases:

$max_a(t) \in \partial_0(t)$: In this case $max_a(t) = max_a(\partial_0(t))$. Hence it immediately follows that $lu^0_{a\to b}(t) = lu^0_{0\to a\to b}(t)$.

$max_a(t) \notin \partial_0(t)$ **and** $latest_{a\to b}(t) \in \partial_0(t)$: By the crossover point lemma, there is a c such that $latest_{a\to b}(t) \leq latest_{0\to c}(t)$ and $latest_{0\to c}(t) \leq max_a(t)$. But from the first inequality it follows that $latest_{a\to b}(t) \leq latest_{0\to c\to b}(t)$, and from the second inequality it follows that $latest_{0\to c\to b}(t) \leq latest_{a\to b}(t)$. Hence $lu^0_{a\to b}(t) = lu^0_{0\to c\to b}(t)$

$max_a(t) \notin \partial_0(t)$ **and** $latest_{a\to b}(t) \notin \partial_0(t)$: By the crossover point lemma, there is a c such that $lu^0_{a\to b}(t) \leq latest_{0\to c}(t) = latest_{0\to c\to c}(t)$ and $latest_{0\to c}(t) = latest_{0\to c\to c}(t) \leq latest_{a\to b}(t)$. But from the first inequality it follows that $lu^0_{a\to b}(t) \leq lu^0_{0\to c\to c}(t)$, and from the second inequality it follows that $lu^0_{0\to c\to c}(t) \leq lu^0_{a\to b}(t)$. Hence $lu^0_{a\to b}(t) = lu^0_{0\to c\to c}(t)$

□

Lemma 27. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace and a, b, k be replicas such that $lu^0_{a\to a}(t) \leq lu^0_{b\to b}(t)$ and $lu^0_{b\to k}(t) \leq lu^0_{a\to k}(t)$. Then there is a replica l such that $lu^0_{a\to k}(t) = lu^0_{b\to l}(t)$.*

Proof. Observe that $lu^0_{a\to k}(t) \in \partial_b(t)$. Now suppose $latest_{a\to k}(t) \in \partial_b(t)$. Then $latest_{a\to k}(t) \leq latest_{b\to k}(t)$. Thus $lu^0_{a\to k}(t) \leq lu^0_{b\to k}(t)$. It follows that $lu^0_{a\to k}(t) = lu^0_{b\to k}(t)$.

If $latest_{a\to k}(t) \notin \partial_b(t)$, then by the crossover point lemma there is l such that $lu^0_{a\to k}(t) \leq latest_{b\to l}(t) \leq latest_{a\to k}(t)$. It immediately follows that $lu^0_{a\to k}(t) \leq lu^0_{b\to l}(t) \leq lu^0_{a\to k}(t)$, and hence $lu^0_{a\to k}(t) = lu^0_{b\to l}(t)$. □

Corollary 28. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace and a, b be replicas. Then $lu^0_{a\to a}(t) \leq lu^0_{b\to b}(t)$ iff $lu^0_{a\to a}(t) \in PrimaryUpdate^0_b(t)$.*

Proof. Suppose $lu^0_{a\to a}(t) \leq lu^0_{b\to b}(t)$. From Lemma 23, we have $lu^0_{b\to a}(t) \leq lu^0_{a\to a}(t)$. Setting $k = a$, from Lemma 27, we get $lu^0_{a\to a}(t) = lu^0_{b\to l}(t)$ for some replica l. Thus $lu^0_{a\to a}(t) \in PrimaryUpdate^0_b(t)$.

For the converse, suppose there exists an l such that $lu^0_{a\to a}(t) = lu^0_{b\to l}(t)$. From Lemma 23, we have $lu^0_{b\to l}(t) \leq lu^0_{b\to b}(t)$ thus giving us $lu^0_{a\to a}(t) \leq lu^0_{b\to b}(t)$ □

5

Bounding the version vectors: a la [AAB04]

5.1 Bounded Representation

We have thus far shown that, as a run proceeds, comparison between matrix elements can be reduced to comparison between the corresponding events in the trace of the run (Corollary 22). In fact, Corollary 22 tells us more: that at any point in the run, comparisons need to be performed only among elements of the form $lu^0_{a\to b}(t)$ in the corresponding trace. This is a set of size N^2. So we need to keep track of these elements as the run proceeds. But we need to compare elements of this form, so we need to keep track of the order between these elements. Furthermore, information about the events $lu^0_{a\to i}(t)$, as well as the ordering among them, is maintained by replica a locally, so we need to show that the information change, as the run proceeds, can be computed locally. If we manage to show this, we would have shown that the relevant order among the version matrix entries can be determined from a bounded number of events in each trace and their relative order. But even though the information needed at each stage of a run is bounded, this information keeps changing. So it would be nice to introduce a bounded set of labels that we use to represent the information at each stage of the run, and reuse the labels appropriately. To determine whether labels can be reused require us to also maintain the secondary update events $lu^0_{i\to j\to k}(t)$ for each trace. We need to show that these can also be computed locally as the run proceeds.

We start off with a technical lemma, which reduces order to containment, much like Corollary 28.

Lemma 29. *Let $t=(\mathscr{E},\leq,\lambda)$ and a,b,i be replicas. If $lu^0_{a\to i}(t)=lu^0_{b\to i}(t)$ and $latest_{a\to i}(t)\leq latest_{b\to i}(t)$ then $\{lu^0_{b\to i\to j}(t)\mid j<N\}\subseteq\{lu^0_{a\to i\to j}(t)\mid j<N\}$.*

Proof. Let $j<N$ be some replica. Since $latest_{a\to i}(t)\leq latest_{b\to i}(t)$, it follows that $latest_{a\to i\to j}(t)\leq latest_{b\to i\to j}(t)$, for any j. Thus $lu^0_{a\to i\to j}(t)\leq lu^0_{b\to i\to j}(t)$. If $latest_{b\to i\to j}(t)\leq latest_{a\to i}(t)$, then it easily follows that $lu^0_{b\to i\to j}(t)=lu^0_{a\to i\to j}(t)$.

Suppose $latest_{b\to i\to j}(t)\not\leq latest_{a\to i}(t)$. Then, since $lu^0_{b\to i\to j}(t)\leq lu^0_{b\to i}(t)=lu^0_{a\to i}(t)\leq latest_{a\to i}(t)$, we can appeal to the crosspoint lemma and obtain l such that $lu^0_{b\to i\to j}(t)\leq latest_{a\to i\to l}(t)$ and $latest_{a\to i\to l}(t)\leq latest_{b\to i\to j}(t)$. But from these two equations it immediately follows that $lu^0_{b\to i\to j}(t)\leq lu^0_{a\to i\to l}(t)$ and $lu^0_{a\to i\to l}(t)\leq lu^0_{b\to i\to j}(t)$. Thus $lu^0_{b\to i\to j}(t)=lu^0_{a\to i\to l}(t)$. □

Definition 30. *For any trace $t=(\mathscr{E},\leq,\lambda)$ and replica a, we define the "local" partial order $\leq^t_a$ as* *follows:*

$$e \leq^t_a e' \textit{ iff } [e' \in PrimaryUpdate^0_a(t) \textit{ and } (e \notin PrimaryUpdate^0_a(t) \textit{ or } e \leq e')].$$

For events e and e′ such that $\{e, e'\} \cap PrimaryUpdate^0_a(t) \neq \emptyset$,

$$max^t_a(e, e') = \begin{cases} e & \textit{if } e' \leq^t_a e \\ e' & \textit{if } e \leq^t_a e' \end{cases}$$

Theorem 31. *Let α be a run and $\alpha' = \alpha \cdot S^{ab}$, $t = t(\alpha) = (\mathscr{E}, \leq, \lambda)$ and $t' = t(\alpha')$. Let w (for "winner") be defined as follows:*

$$w = \begin{cases} b & \textit{if } \ lu^0_{a\to a}(t) \in PrimaryUpdate^0_b(t) \\ a & \textit{otherwise} \end{cases}$$

Also let S_i (for $i \notin \{a, b\}$) be defined as follows:

$$S_i = \begin{cases} \{lu^0_{a\to i\to j}(t) \mid j < N\} & \textit{if } \ lu^0_{a\to i}(t) >^t_w lu^0_{b\to i}(t) \\ & \quad \textit{or } [lu^0_{a\to i}(t) = lu^0_{b\to i}(t) \textit{ and} \\ & \quad\quad \{lu^0_{a\to i\to j}(t) \mid j < N\} \subseteq \{lu^0_{b\to i\to j}(t) \mid j < N\}] \\ \{lu^0_{b\to i\to j}(t) \mid j < N\} & \textit{otherwise} \end{cases}$$

Then:

1. *for all $i, j \in \{a, b\}$, $lu^0_{i\to j}(t') = lu^0_{w\to w}(t)$.*

2. *for all $i \notin \{a, b\}$, $lu^0_{i\to j}(t') = lu^0_{i\to j}(t)$.*

3. *for all $i \in \{a, b\}$, $j \notin \{a, b\}$, $lu^0_{i\to j}(t') = max^t_w(lu^0_{a\to j}(t), lu^0_{b\to j}(t))$.*

4. *for all $i \in \{a, b\}$ and for all j, k, $lu^0_{i\to j}(t') \leq^{t'}_i lu^0_{i\to k}(t')$ iff $lu^0_{i\to j}(t') \leq^t_w lu^0_{i\to j}(t')$.*

5. *for all $i \notin \{a, b\}$ and for all j, k, $lu^0_{i\to j}(t') \leq^{t'}_i lu^0_{i\to k}(t')$ iff $lu^0_{i\to j}(t') \leq^t_i lu^0_{i\to j}(t')$.*

6. *for all $c, i \in \{a, b\}$, for all j, $lu^0_{c\to i\to j}(t') = lu^0_{c\to j}(t')$.*

7. *for all $c \notin \{a, b\}$, for all i, j, $lu^0_{c\to i\to j}(t') = lu^0_{c\to i\to j}(t)$.*

8. *for all $c \in \{a, b\}$, $i \notin \{a, b\}$, $\{lu^0_{c\to i\to j}(t') \mid j < N\} = S_i$.*

Proof. We start with a key observation.

Suppose $lu^0_{a\to a}(t) \leq lu^0_{b\to b}(t)$. Then by Corollary 28, $lu^0_{a\to a}(t) \in PrimaryUpdate^0_b(t)$ and $w = b$. Otherwise $lu^0_{b\to b}(t) < lu^0_{a\to a}(t)$ and $w = a$. Therefore $max(lu^0_{a\to a}(t), lu^0_{b\to b}(t)) = lu^0_{w\to w}(t)$.

1. For $i, j \in \{a, b\}$, $lu^0_{i\to j}(t') = max(lu^0_{a\to a}(t), lu^0_{b\to b}(t)) = lu^0_{w\to w}(t)$.

2. For $i \notin \{a,b\}$ and any j, clearly $lu^0_{i\to j}(t') = lu^0_{i\to j}(t)$.

3. Suppose $i \in \{a,b\}$ and $j \notin \{a,b\}$. Then $lu^0_{i\to j}(t') = max(lu^0_{a\to j}(t), lu^0_{b\to j}(t))$. Assume that $w = b$. We consider two cases.

   $lu^0_{a\to j}(t) \le lu^0_{b\to j}(t)$: Then it is also the case that $lu^0_{a\to j}(t) \le^t_b lu^0_{b\to j}(t)$ (even when $lu^0_{a\to j}(t) \notin PrimaryUpdate^0_b(t)$).

   $lu^0_{b\to j}(t) \le lu^0_{a\to j}(t)$: Then $lu^0_{a\to j}(t) = lu^0_{b\to l}(t)$ for some l. Thus $lu^0_{b\to j}(t) \le^t_b lu^0_{a\to j}(t)$.

   Thus $lu^0_{i\to j}(t') = max(lu^0_{a\to j}(t), lu^0_{b\to j}(t)) = max^t_b(lu^0_{a\to j}(t), lu^0_{b\to j}(t))$. The argument is similar when $w = a$.

4. Observe that for any j, $lu^0_{i\to j}(t') = lu^0_{w\to l}(t)$ for some l. Thus $lu^0_{i\to j}(t') \le^{t'}_i lu^0_{i\to k}(t')$ iff $lu^0_{i\to j}(t') \le^t_w lu^0_{i\to k}(t')$.

5. This follows from item 2.

6. Suppose $c, i \in \{a,b\}$. Then $latest_{c\to i}(t') = latest_{c\to c}(t') = max_c(t')$ and hence $lu^0_{c\to i\to j}(t') = update_0(max_j(latest_{c\to i}(t'))) = update_0(max_j(max_c(t'))) = lu^0_{c\to j}(t')$.

7. Suppose $c \notin \{a,b\}$. Then $latest_{c\to i}(t') = latest_{c\to i}(t)$ and hence $lu^0_{c\to i\to j}(t') = lu^0_{c\to i\to j}(t)$.

8. We need to consider several cases.

   $lu^0_{a\to i}(t) >^t_w lu^0_{b\to i}(t)$: In this case, by Proposition 13, $latest_{a\to i}(t) > latest_{b\to i}(t)$. Thus for all $j < N$, $lu^0_{c\to i\to j}(t') = lu^0_{a\to i\to j}(t)$, and hence the statement of the theorem follows.

   $lu^0_{a\to i}(t) = lu^0_{b\to i}(t)$ **and** $\{lu^0_{a\to i\to j}(t) \mid j < N\} = \{lu^0_{b\to i\to j}(t) \mid j < N\}$: Let x be defined as follows:

   $$x = \begin{cases} a & \text{if} \quad latest_{a\to i}(t) \ge latest_{b\to i}(t) \\ b & \text{otherwise} \end{cases}$$

   Then it is clear that for all $j < N$, $lu^0_{c\to i\to j}(t') = max(lu^0_{a\to i\to j}(t), lu^0_{b\to i\to j}(t)) = lu^0_{x\to i\to j}(t)$. Thus

   $$\{lu^0_{c\to i\to j}(t') \mid j < N\} = \{lu^0_{x\to i\to j}(t) \mid j < N\} = \{lu^0_{a\to i\to j}(t) \mid j < N\} = S_i.$$

   $lu^0_{a\to i}(t) = lu^0_{b\to i}(t)$ **and** $\{lu^0_{a\to i\to j}(t) \mid j < N\} \subsetneq \{lu^0_{b\to i\to j}(t) \mid j < N\}$: It immediately follows that $\{lu^0_{b\to i\to j}(t) \mid j < N\} \nsubseteq \{lu^0_{a\to i\to j}(t) \mid j < N\}$. By Lemma 29, $latest_{a\to i}(t) \not\le latest_{b\to i}(t)$, and hence $latest_{a\to i}(t) > latest_{b\to i}(t)$. Thus

   $$lu^0_{c\to i\to j}(t') = max(lu^0_{a\to i\to j}(t), lu^0_{b\to i\to j}(t)) = lu^0_{a\to i\to j}(t)$$

   and hence the statement of the theorem follows.

$lu^0_{a\to i}(t) = lu^0_{b\to i}(t)$ **and** $\{lu^0_{a\to i\to j}(t) \mid j < N\} \nsubseteq \{lu^0_{b\to i\to j}(t) \mid j < N\}$: By Lemma 29, it follows that $latest_{b\to i}(t) \nleq latest_{a\to i}(t)$, and hence $latest_{a\to i}(t) < latest_{b\to i}(t)$. Thus

$$lu^0_{c\to i\to j}(t') = max(lu^0_{a\to i\to j}(t), lu^0_{b\to i\to j}(t)) = lu^0_{b\to i\to j}(t)$$

and hence the statement of the theorem follows.

$lu^0_{a\to i}(t) <^t_w lu^0_{b\to i}(t)$: In this case, by Proposition 13, $latest_{a\to i}(t) < latest_{b\to i}(t)$. Thus for all $j < N$, $lu^0_{c\to i\to j}(t') = lu^0_{b\to i\to j}(t)$, and hence the statement of the theorem follows.

□

Theorem 32. *Let α be a run and $\alpha' = \alpha \cdot U^0$, $t = t(\alpha) = (\mathscr{E}, \leq, \lambda)$ and $t' = t(\alpha') = (\mathscr{E} \cup \{e\}, \leq', \lambda')$. Then:*

1. *for $i \neq 0$ and any j, $lu^0_{i\to j}(t') = lu^0_{i\to j}(t)$.*
2. *for $j \neq 0$, $lu^0_{0\to j}(t') = lu^0_{0\to j}(t)$.*
3. *$lu^0_{0\to 0}(t) = e$.*
4. *for all $i \neq 0$, $\leq^{t'}_i = \leq^t_i$.*
5. *for all i, j, $lu^0_{0\to i}(t') \leq^{t'}_i lu^0_{0\to j}(t')$ iff either $j = 0$ or $lu^0_{0\to i}(t') \leq^t_i lu^0_{0\to j}(t')$.*
6. *for all j, $lu^0_{0\to 0\to j}(t') = lu^0_{0\to j}(t')$.*
7. *for all $i \neq 0$, for all j, $lu^0_{0\to i\to j}(t') = lu^0_{0\to i\to j}(t)$.*
8. *for all $c \neq 0$, for all i, j, $lu^0_{c\to i\to j}(t') = lu^0_{c\to i\to j}(t)$.*

Proof. All parts of the theorem follow from the definitions and the fact that e is the maximal 0-event and maximal update event in t. □

Theorem 33. *Let α be the run consisting of the single operation I, and let $t = t(\alpha) = (\{e_\perp\}, \emptyset, \lambda)$. Then:*

1. *for all i, j, $lu^0_{i\to j}(t) = e_\perp$.*
2. *for all i, j, k, $lu^0_{i\to j}(t) =^t_i lu^0_{i\to k}(t)$.*
3. *for all i, j, k, $lu^0_{i\to j\to k}(t) = e_\perp$.*

For the rest of this section, we fix a set $\mathscr{L} = \{l_0, \ldots, l_{N^2}\}$ of labels. Let $[N]$ denote the set $\{0, \ldots, N-1\}$. For any trace t, we let $P(t)$ denote $\bigcup_{b\in[N]} PrimaryUpdate^0_b(t)$. For any trace $t = (\mathscr{E}, \leq, \lambda)$, let $\mathscr{E}_0 = \{e \in \mathscr{E} \mid \lambda(e) = U^0\}$.

Definition 34. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace. We say that $(\delta, \sigma, \tau, \preceq_0, \ldots, \preceq_{N-1})$ is a* **valid $\mathscr{L}$-labelling** *for t iff:*

- *$\delta : \mathscr{E}_0 \to \mathscr{L}$ is a function such that for all $e, e' \in P(t) : \delta(e) = \delta(e') \Rightarrow e = e'$.*
- *$\sigma : [N]^2 \to \mathscr{L}$ is a function such that $\sigma(a,b) = \delta(lu^0_{a\to b}(t))$.*
- *$\tau : [N]^2 \to 2^{\mathscr{L}}$ is a function such that $\tau(a,b) = \{\delta(lu^0_{a\to b\to j}(t)) \mid j \in [N]\}$.*
- *for all $a \in [N]$, $\preceq_a$ is a partial order on $\mathscr{L}$ such that*

$$\textit{for all } e, e' \in \textit{PrimaryUpdate}^0_a(t) : \delta(e) \preceq_a \delta(e') \textit{ iff } e \leq^t_a e'.$$

Theorem 35. *Let α be a run and $t = t(\alpha) = (\mathscr{E}, \leq, \lambda)$. Then there is a valid $\mathscr{L}$-labelling for t which, further, is locally computable.*

Proof. We define a labelling valid for t by induction on the length of α.

$\alpha = I$: In this case take $\delta(e_\perp) = l_0$, $\sigma(a,b) = l_0$, and $\tau(a,b) = \{l_0\}$. We also take $l_0 \preceq_a l_0$ for all a. It is an immediate observation that $(\delta, \sigma, \tau, \preceq_0, \ldots, \preceq_{N-1})$ is a valid labelling for t.

$\alpha = \alpha' \cdot o$: Suppose $t' = t(\alpha') = (\mathscr{E}', \leq', \lambda')$, $\mathscr{E} \setminus \mathscr{E}' = \{e_{max}\}$, and let $(\delta', \sigma', \tau' \preceq'_0, \ldots, \preceq'_{N-1})$ be a valid $\mathscr{L}$-labelling for t'.

This means that for all $e, e' \in P(t')$, $\delta'(e) = \delta'(e')$ iff $e = e'$. So it follows that

$$lu^0_{a\to a}(t') \in \textit{PrimaryUpdate}^0_b(t') \quad \text{iff} \quad \sigma'(a,a) \in \tau'(b,b).$$

We now define the new labelling functions δ, σ, τ for t. There are two cases.

$o = U^0$: Let $l \in \mathscr{L}$ be the label with least index such that $l \notin \bigcup_{a\in[N]} \tau(0,a)$. (There is one such, since $\mathscr{L}$ is of size $N^2 + 1$ and there are at most N^2 events in $\bigcup_{a\in[N]} \tau(0,a)$.) By Lemma 26, for every $a, b < N$, there are $c, d < N$ such that $lu^0_{a\to b}(t') = lu^0_{0\to c\to d}(t')$. Thus, for all $a, b < N$, $\delta'(lu^0_{a\to b}(t')) \neq l$.
In this case, we define δ as follows:

$$\delta(e) = \begin{cases} l & \text{if} \quad e = e_{max} \\ \delta'(e) & \text{otherwise} \end{cases}$$

Observe from Theorem 32 that for $e \in P(t)$, either $e = e_{max}$ and $\delta(e) = l$ or $e \in P(t')$ and $\delta(e) = \delta'(e) \neq l$. Thus if $\delta(e) = \delta(e')$ then either $e = e' = e_{max}$ or $e, e' \in P(t')$ and $\delta'(e) = \delta'(e')$, whence by induction hypothesis $e = e'$.
We define σ as follows:

$$\sigma(a,b) = \begin{cases} l & \text{if} \quad a = b = 0 \\ \sigma'(a,b) & \text{otherwise} \end{cases}$$

We define τ as follows:

$$\tau(a,b) = \begin{cases} \{\sigma(0,c) \mid c \in [N]\} & \text{if} \quad a = b = 0 \\ \tau'(a,b) & \text{otherwise} \end{cases}$$

We define $\preceq_a$ as follows:

$$l' \preceq_a l'' \text{ iff } l'' = l \text{ or } (l'' \neq l \text{ and } l' \preceq'_a l'').$$

It is clear that from the assumptions on σ', τ', and $\preceq'_a$ and Theorem 32 that σ, τ, and $\preceq_a$ satisfy the conditions for a valid labelling of t, since the definitions mirror the properties proved in Theorem 32.

$o = S^{ab}$: We let $\delta = \delta'$. Observe from Theorem 32 that $P(t) \subseteq P(t')$. Therefore it follows that for $e, e' \in P(t)$, if $\delta(e) = \delta(e')$ then $e = e'$.

Let w be defined as follows:

$$w = \begin{cases} b & \text{if} \quad \sigma'(a,a) \in \tau'(b,b) \\ a & \text{otherwise} \end{cases}$$

We define σ as follows:

$$\sigma(i,j) = \begin{cases} \sigma'(w,w) & \text{if} \quad i,j \in \{a,b\} \\ \sigma'(i,j) & \text{if} \quad i \notin \{a,b\} \\ max_{\preceq'_w}(\sigma'(a,j), \sigma'(b,j)) & \text{otherwise} \end{cases}$$

We define $\preceq_i$ as follows:

$$l \preceq_i l' \text{ iff } \Big[(i \in \{a,b\} \text{ and } l \preceq'_w l') \text{ or } (i \notin \{a,b\} \text{ and } l \preceq'_i l')\Big]$$

It is clear that from the assumptions on σ' and $\preceq'_a$ and Theorem 31 that σ and $\preceq_a$ satisfy the conditions for a valid labelling of t. We just need to define τ appropriately.

For $c, i \in \{a,b\}$, we define $\tau(c,i) = \{\sigma(c,j) \mid j < N\}$.

For $c \notin \{a,b\}$, we define $\tau(c,i) = \tau'(c,i)$. For $c \in \{a,b\}$ and $i \notin \{a,b\}$, we define $\tau(c,i)$ as follows:

$$\tau(c,i) = \begin{cases} \tau'(a,i) & \text{if} \quad \sigma'(a,i) \prec_w \sigma'(b,i) \text{ or} \\ & \qquad [\sigma'(a,i) = \sigma'(b,i) \text{ and } \tau'(a,i) \subseteq \tau'(b,i)] \\ \tau'(b,i) & \text{otherwise} \end{cases}$$

We need to prove that $\tau(c,i) = \{\delta(lu^0_{c\to i\to j}(t)) \mid j \in [N]\}$. Since in most cases the definition mirrors the properties proved in Theorem 31, the only nontrivial case to be proved is when $\sigma'(a,i) = \sigma'(b,i)$. But this means that $\delta(lu^0_{a\to i}(t')) = \delta(lu^0_{b\to i}(t'))$, and by induction hypothesis on δ' (which is the same as δ), $lu^0_{a\to i}(t') = lu^0_{b\to i}(t')$. But in this case, by Lemma 29 that either $\{lu^0_{a\to i\to j}(t') \mid j < N\} \subseteq \{lu^0_{b\to i\to j}(t') \mid j < N\}$, or $\{lu^0_{b\to i\to j}(t') \mid j < N\} \subsetneq \{lu^0_{a\to i\to j}(t') \mid j < N\}$.

Now when $\tau'(a,i) \subseteq \tau'(b,i)$ there are two cases:

$\{lu^0_{a\to i\to j}(t') \mid j < N\} \subseteq \{lu^0_{b\to i\to j}(t') \mid j < N\}$: In this case $\{lu^0_{c\to i\to j}(t) \mid j < N\} = \{lu^0_{a\to i\to j}(t') \mid j < N\}$ and $\tau(c,i) = \tau'(a,i)$ and thus

$$\tau(c,i) = \{\delta(lu^0_{c\to i\to j}(t)) \mid j < N\}.$$

$\{lu^0_{b\to i\to j}(t') \mid j<N\} \subsetneq \{lu^0_{a\to i\to j}(t') \mid j<N\}$ **and** $\tau'(a,i)=\tau'(b,i)$: In this case we can see that $\{lu^0_{c\to i\to j}(t) \mid j<N\} = \{lu^0_{b\to i\to j}(t') \mid j<N\}$ and $\tau(c,i) = \tau'(b,i) = \tau'(a,i)$ and thus

$$\tau(c,i) = \{\delta(lu^0_{c\to i\to j}(t)) \mid j<N\}.$$

In the case that $\tau'(b,i) \subsetneq \tau'(a,i)$, it immediately follows that $\{lu^0_{b\to i\to j}(t') \mid j<N\} \subsetneq \{lu^0_{a\to i\to j}(t') \mid j<N\}$. In this case $\{lu^0_{c\to i\to j}(t) \mid j<N\} = \{lu^0_{b\to i\to j}(t') \mid j<N\}$ and $\tau(c,i)=\tau'(b,i)$ and thus

$$\tau(c,i) = \{\delta(lu^0_{c\to i\to j}(t)) \mid j<N\}.$$

Thus $(\delta,\sigma,\tau,\preceq_0,\ldots,\preceq_{N-1})$ is a valid $\mathscr{L}$-labelling of t.

□

5.2 Complexity of our solution

Since the labels are drawn from a set of size N^2+1, to represent each label requires $\theta(\log N)$ bits. Since we need to maintain N^2 entries of the form $\sigma(c,i)$ and N^2 sets (each of size N) of the form $\tau(c,i)$, and a linear ordering on N^2 elements, the amount of overall information that needs to be stored at any point is $O(N^3 \cdot \log N)$ bits. This is a representation for the version vector slices $C^0_a(t)$, for each a. We can represent the other slices also in a similar manner. So to represent all the version vector matrices would require $O(N^4 \cdot \log N)$ bits to be used.

5.3 A critique of the solution

Our goal was to obtain a bounded representation for version vectors of a distributed system comprising of N replicas. To achieve this, we modelled the problem using the traces of the run comprising of these operations. We associated with each version vector entry $V^k_a(t)$, the U^k event $lu^k_{a\to a}(t)$, which was the earliest event at which the value of V^k_k was the same as the current value of V^k_a.

In this model, we showed that during a synchronization operation performed by replicas a, b, pairwise comparison of entries $V^k_a(t)$ and $V^k_b(t)$ was equivalent to comparing the corresponding events $lu^k_{a\to a}(t)$ and $lu^k_{b\to b}(t)$. We solved this problem by noting the fact that $lu^k_{b\to b}(t)$ is the latest U^k event among the two iff the event $lu^k_{a\to a}(t)$ is present in the set of primary update events $PrimaryUpdate^k_b(t)$ of b with respect to the replica k.

Thus reducing the comparison problem to a set containment problem means that each replica a would end up maintaining the sets of primary update events $PrimaryUpdate^k_a(t)$ for each replica k. Thus it is convenient to work with version vector slices instead of version vectors themselves, since the entries in the version vector slice $C^k_a(t)$ correspond to the events in $PrimaryUpdate^k_a(t)$.

One drawback in this solution is that there is no way to check if $latest_{a\to k}(t)$ is strictly less than $latest_{b\to k}(t)$ directly. One has to resort to checking a complicated set inclusion. This would be solved if we had a way to label all events, rather than only the update events.

Another potential drawback is that the above solution works for only one slice, and we have to repeat it for the other slices. It would be better if there was a solution for the version vector matrices as a whole, with comparable space complexity. We devote the next section to such a solution based on the gossip problem.

6

Bounding the version vectors: Using gossip

6.1 A more efficient bounded representation

In this section we adapt a solution to the gossip problem [MS97] to provide a bounded representation for version vector matrices. In this solution, we maintain information about not just the primary update events, but also the primary events themselves. As we have seen already, this will require us to also maintain information about secondary update events and secondary events.

Suppose a and b are replicas with version vectors $V_a(t)$ and $V_b(t)$ at the end of a trace t. We have already argued (in Corollary 22) that comparing $V_a^k(t)$ and $V_b^k(t)$ is equivalent to comparing $lu^k_{a\to a}(t)$ with $lu^k_{b\to b}(t)$. Since for any replica i, $lu^k_{i\to i}(t) = lu^k_{i\to k}(t)$, comparing $V_a^k(t)$ and $V_b^k(t)$ is equivalent to comparing $lu^k_{a\to k}(t)$ with $lu^k_{b\to k}(t)$.

The following is an immediate consequence of Proposition 13.

Proposition 36. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace and $a, b, k < N$. If $lu^k_{a\to k}(t) \neq lu^k_{b\to k}(t)$ then*

$$lu^k_{a\to k}(t) < lu^k_{b\to k}(t) \quad \textit{iff} \quad latest_{a\to k}(t) < latest_{b\to k}(t).$$

Thus the problem of comparing distinct principal k-update events corresponds to comparing the latest k-events in the views of the appropriate a and b.

6.2 Gossip problem: Recap of the results

Definition 37. *Let $t = (\mathscr{E}, \leq, \lambda)$ be an ideal and a be a replica. We define the* **primary information** *of a in t, denoted by $Primary_a(t)$ to be the set of all the latest events in the view of a in t.*

Formally, $Primary_a(t) = \{latest_{a\to k}(t) \mid k < N\}$.

Our goal is to settle the question of comparing $latest_{a\to k}(t)$ and $latest_{b\to k}(t)$ using a finite amount of information that is also local. If $latest_{a\to k}(t) \leq latest_{b\to k}(t)$ then $latest_{a\to k}(t)$ is in the view $\partial_a(t) \cap \partial_b(t)$. Thus, $latest_{a\to k}(t)$ is in the view generated by the maximal elements of $\partial_a(t) \cap \partial_b(t)$. We show the following nice property about these maximal elements.

Lemma 38. *Let a and b are replicas and $t = (\mathscr{E}, \leq, \lambda)$ be a trace. If e is a maximal event in the ideal $\partial_a(t) \cap \partial_b(t)$ then $e \in Primary_a(t) \cap Primary_b(t)$.*

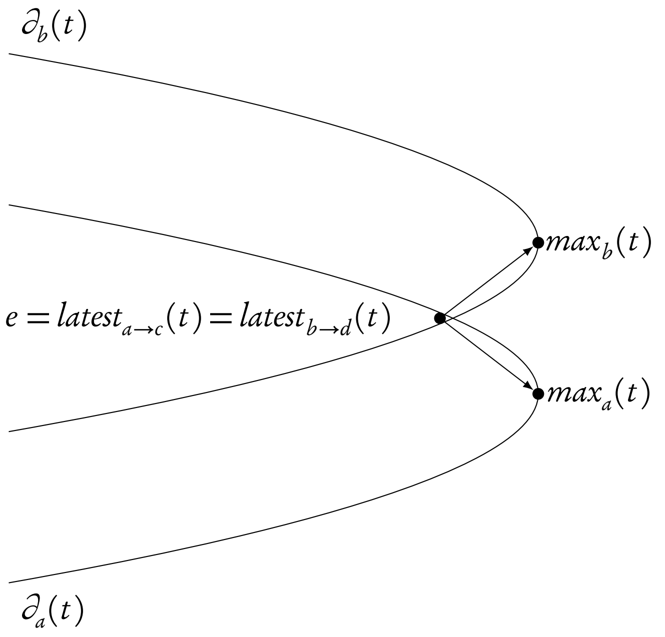

Figure 6.1: Figure for lemma 38. e is a maximal event in $\partial_a(t) \cap \partial_b(t)$.

Proof. If $max_a(t) \in \partial_b(t)$ then, $\partial_a(t) \cap \partial_b(t) = \partial_a(t)$ whose sole maximal element is $max_a(t)$ which also happens to be the same as $latest_{b\to a}(t)$ which is a member of $Primary_b(t)$ by definition. Thus $max_a(t) \in Primary_b(t)$. The case where $max_b(t) \in \partial_a(t)$ can be similarly handled.

Suppose it is neither the case that $max_a(t) \in \partial_b(t)$ nor the case that $max_b(t) \in \partial_a(t)$. Then $max_a(t) \in \partial_a(t) \setminus \partial_b(t)$. Similarly $max_b(t) \in \partial_b(t) \setminus \partial_a(t)$.

From the crossover point lemma that there is c such that $e \leq latest_{a\to c}(t) \leq max_b(t)$. Since e is a maximal event in $\partial_a(t)$, it follows that $e = latest_{a\to c}(t)$. Hence $e \in Primary_a(t)$. Similarly, $e = latest_{b\to d}(t)$ for some replica d and hence $e \in Primary_b(t)$. Thus any maximal $(\partial_a(t) \cap \partial_b(t))$-event e is in $Primary_a(t) \cap Primary_b(t)$. □

From this result we can reason out how to compare $latest_{a\to k}(t)$ and $latest_{b\to k}(t)$ as follows:

Corollary 39. *If $e = latest_{a\to k}(t)$ and $f = latest_{b\to k}(t)$ then*

$$e \leq f \quad \textit{iff} \quad \exists g \in Primary_a(t) \cap Primary_b(t) : e \leq g.$$

Proof. If $e \leq f$ then $e \in \partial_a(t) \cap \partial_b(t)$. On the other hand, $latest_{b\to k}(t)$ is later than any k-event in $\partial_b(t)$. Thus if $e \in \partial_a(t) \cap \partial_b(t)$, $e \leq f$.

Thus $e \leq f$ iff $e \in \partial_a(t) \cap \partial_b(t)$ iff e is dominated by some maximal element h of $\partial_a(t) \cap \partial_b(t)$, iff e is dominated by $g \in Primary_a(t) \cap Primary_b(t)$ (by Lemma 38). □

Thus solving the problem of comparing $latest_{a\to k}(t)$ and $latest_{b\to k}(t)$ is equivalent to performing the check in corollory 39 which requires us to maintain the partial order relation between the primary events of replicas. Since the comparison problem is reduced to finding a particular maximal element in the intersection of $Primary_a(t)$ and $Primary_b(t)$, we need to label these events uniquely. In our earlier solution we only assigned labels to the primary update events. However our current solution relies on the primary events which change with not just the update events, but also with the synchronization events. Hence we need to assign labels even to synchronization events.

However care must be taken to ensure that whenever two primary events $latest_{a\to i}(t)$ and $latest_{b\to j}(t)$ are assigned the same label, then they are indeed the same events. Otherwise the solution outlined above for comparing corresponding primary events would not work. Thus whenever we label a new S^{ab} event, we need to pick a label that is not currently in use for labelling any other primary event. We need to make this decision by looking at the local information of a and b. Since the information in the primary events is not sufficient, we need to maintain *secondary events*.

Definition 40. *Let a be any replica and $t = (\mathscr{E}, \leq, \lambda)$ be a trace. Then the* **secondary information** *for a in t, denoted $Secondary_a(t)$, is defined as*

$$Secondary_a(t) = \{latest_{a\to b\to c}(t) \mid b, c < N\}.$$

The following result relates the primary events corresponding to a replica a and the secondary events of replica a.

Lemma 41. *For any replicas b and c in a trace $t = (\mathscr{E}, \leq, \lambda)$, if $latest_{b\to c}(t)$ is an a-event then $latest_{b\to c}(t) \in Secondary_a(t)$*

Proof. The proof is very similar to the one for Lemma 26. We need to show that $latest_{b\to c}(t) = latest_{a\to d\to i}(t)$ for some $d, i < N$. If $max_b(t) \in \partial_a(t)$, then $latest_{b\to c}(t) = latest_{b\to c}(\partial_a(t)) = latest_{a\to b\to c}(t)$.

If $max_b(t) \notin \partial_a(t)$, since $latest_{b\to c}(t)$ is also an a-event, $latest_{b\to c}(t) \in \partial_a(t)$. By the crossover point lemma we can find d such that $latest_{b\to c}(t) \leq latest_{a\to d}(t) \leq max_b(t)$. Since $latest_{b\to c}(t)$ is the maximal c event in $\partial_b(t)$, we have $latest_{a\to d\to c}(t) = latest_{b\to c}(t)$. Thus $latest_{b\to c}(t) \in Secondary_a(t)$. □

Just as in the previous section, we can update secondary information "locally", as a run proceeds. Given two version vectors $V_a(t)$ and $V_b(t)$, comparing $V_a^k(t)$ and $V_b^k(t)$ is equivalent to comparing $lu^k_{a\to k}(t)$ with $lu^k_{b\to k}(t)$. When $lu^k_{a\to k}(t)$ and $lu^k_{b\to k}(t)$ are not equal, comparing them is equivalent to comparing the primary events $latest_{a\to k}(t)$ and $latest_{b\to k}(t)$. We will show that this comparison can be performed by maintaining an unambiguous labelling of the primary events and secondary events.

If our purpose was to decide if $V_a(t)$ dominates $V_b(t)$ or if they are incomparable, then the primary and secondary information would suffice. However, if we also want to verify whether the states of the two replicas are the same, we have to show that for each k, $lu^k_{a\to k}(t) = lu^k_{b\to k}(t)$. For this we maintain the primary update information. Note that this is different from the primary update information we used in the previous solution.

Definition 42. *For any replica a and an ideal $t = (\mathscr{E}, \leq, \lambda)$, the* **primary update information**, *denoted $PrimaryUpdate_a(t)$, is the indexed set $\{lu^k_{a\to k}(t) \mid k \in [N]\}$.*

However during an update event of replica k, we need to assign a label to this new event $lu^k_{k\to k}(t)$ which is different from the update event $lu^k_{a\to k}(t)$ for any other replica a. Since it is not guaranteed that the primary update event $lu^k_{a\to k}(t)$ is also a primary event for some replica, secondary information would not be sufficient to choose a unique label for the new update event $lu^k_{k\to k}(t')$. Hence we maintain **secondary update information**.

Definition 43. *For any replica a and a trace $t = (\mathscr{E}, \leq, \lambda)$, the* **secondary update information** *denoted as $SecondaryUpdate_a(t)$ is the indexed set $\{lu^k_{a\to b\to k}(t) \mid b, k < N\}$.*

For any replica a, we can show the relationship between the primary update information of a replica k corresponding to the replica a and the secondary update information for a.

Lemma 44. *For any trace $t = (\mathscr{E}, \leq, \lambda)$ and replicas a, b, $lu^a_{b\to a}(t) \in SecondaryUpdate_a(t)$.*

Proof. Once again, the proof is similar to the lemma 26. We need to show that for all a, b, there is c such that $lu^a_{b\to a}(t) = lu^a_{a\to c\to a}(t)$. If $max_b(t) \in \partial_a(t)$, then $lu^a_{b\to a}(t) = lu^a_{a\to b\to a}(t)$ and we are done.

Otherwise we use the crossover lemma to find a c such that $lu^a_{b\to a}(t) \leq latest_{a\to c}(t)$ and $latest_{a\to c}(t) \leq max_b(\mathscr{E})$. It then follows that $lu^a_{b\to a}(t) \leq lu^a_{a\to c\to a}(t)$ and $lu^a_{a\to c\to a}(t) \leq lu^a_{b\to a}(t)$, and we are done. □

Definition 45. *For any trace $t = (\mathscr{E}, \leq, \lambda)$ and replica a, we define the "local" partial order $\leq^t_a$ as follows:*

$$e \leq^t_a e' \textit{ iff } [e' \in Primary_a(t) \textit{ and } (e \notin Primary_a(t) \textit{ or } e \leq e')].$$

We now have all the definitions and tools with us to provide a bounded representation for version vector matrices.

Theorem 46. *Let α be a run and $\alpha' = \alpha \cdot U^a$, $t = t(\alpha) = (\mathscr{E}, \leq, \lambda)$ and $t' = t(\alpha') = (\mathscr{E} \cup \{e_{max}\}, \leq', \lambda')$. Then:*

1. *$latest_{a\to a}(t') = lu^a_{a\to a}(t') = e_{max}$.*
2. *for $c, d < N$, if $\neg(c = d = a)$, $latest_{c\to d}(t') = latest_{c\to d}(t)$ and $lu^d_{c\to d}(t') = lu^d_{c\to d}(t)$.*
3. *$latest_{a\to a\to a}(t') = lu^a_{a\to a\to a}(t') = e_{max}$.*
4. *for $c, d, i < N$, if $\neg(c = d = i = a)$, $latest_{c\to d\to i}(t') = latest_{c\to d\to i}(t)$ and $lu^i_{c\to d\to i}(t') = lu^i_{c\to d\to i}(t)$.*
5. *for $e, e' \in Primary_a(t')$, $e \leq^{t'}_a e'$ iff $e \leq^t_a e'$ or $e' = e_{max}$.*

Proof. Straightforward. All parts follow from the definitions. □

Theorem 47. *Let α be a run and $\alpha' = \alpha \cdot S^{ab}$, $t = t(\alpha) = (\mathscr{E}, \leq, \lambda)$ and $t' = t(\alpha') = (\mathscr{E} \cup \{e_{max}\}, \leq', \lambda')$. For each $i \notin \{a, b\}$, define w_i as follows:*

$$w_i = \begin{cases} a & \textit{if } \exists j, k : latest_{b\to i}(t) \leq^t_b latest_{b\to j}(\mathscr{E}) = latest_{a\to k}(t) \\ b & \textit{otherwise} \end{cases}$$

Then:

1. *for $c, i \in \{a, b\}$, $latest_{c\to i}(t') = e_{max}$, $lu^i_{c\to i}(t') = lu^i_{i\to i}(t)$.*
2. *for $c \notin \{a, b\}, i < N$, $latest_{c\to i}(t') = latest_{c\to i}(t)$ and $lu^i_{c\to i}(t') = lu^i_{c\to i}(t)$.*

3. *for* $c \in \{a,b\}, i \notin \{a,b\}$, $latest_{c\to i}(t') = latest_{w_i \to i}(t)$ *and* $lu^i_{c\to i}(t') = lu^i_{w_i\to i}(t)$.

4. *for* $c, i \in \{a,b\}, j < N$, $latest_{c\to i\to j}(t') = latest_{i\to j}(t')$ *and* $lu^j_{c\to i\to j}(t') = lu^j_{i\to j}(t')$.

5. *for* $c \notin \{a,b\}, i, j < N$, $latest_{c\to i\to j}(t') = latest_{c\to i\to j}(t)$ *and* $lu^j_{c\to i\to j}(t') = lu^j_{c\to i\to j}(t)$.

6. *for* $c \in \{a,b\}, i \notin \{a,b\}, j < N$, $latest_{c\to i\to j}(t') = latest_{w_i\to i\to j}(t)$ *and* $lu^j_{c\to i\to j}(t') = lu^j_{w_i\to i\to j}(t)$.

7. *for* $c \in \{a,b\}$ *and* $e, e' \in Primary_c(t')$, $e \preceq^{t'}_c e'$ *iff* $e \preceq^t_a e'$ *or* $e \preceq^t_b e'$ *or* $e' = e_{max}$.

Proof. To begin with, we observe that for any $i < N$ (and in particular for $i \notin \{a,b\}$), either $latest_{b\to i}(t) \leq latest_{a\to i}(t)$ or $latest_{b\to i}(t) < latest_{a\to i}(t)$. It now follows from Corollary 39 and the definition of w_i that $latest_{w_i\to i}(t) = max(latest_{a\to i}(t), latest_{b\to i}(t))$. Now we prove the various parts in the theorem.

1. Immediate from the definitions.

2. Immediate from the definitions.

3. $latest_{c\to i}(t') = max(latest_{a\to i}(t), latest_{b\to i}(t)) = latest_{w_i\to i}(t)$. Hence $lu^i_{c\to i}(t') = lu^i_{w_i\to i}(t)$.

4. Immediate from the definitions and the fact that e_{max} is an a-event and a b-event.

5. Immediate from the definitions.

6. Since $latest_{c\to i\to j}(t') = latest_{i\to j}(\downarrow latest_{c\to i}(t'))$ and $latest_{c\to i}(t') = latest_{w_i\to i}(t)$, it follows that $latest_{c\to i\to j}(t') = latest_{w_i\to i\to j}(t)$. It follows from this that $lu^j_{c\to i\to j}(t') = lu^j_{w_i\to i\to j}(t)$.

7. Clearly if $e' = e_{max}$ or $e \preceq^{t'}_a e'$ or $e \preceq^t_b e'$, it follows that $e \preceq' e'$ and hence $e \preceq^{t'}_c e'$. On the other hand, suppose $e \preceq^t_c e'$ and $e' \neq e_{max}$. This means that $e, e' \in Primary_a(t) \cup Primary_b(t)$ and $e \leq e'$. We have the following cases to consider:

   $e' \in Primary_b(t)$: Since $e \in Primary_c(t')$, it follows that for some $k < N$, $e = latest_{c\to k}(t') = max(latest_{a\to k}(t), latest_{b\to k}(t))$. Suppose $e' = latest_{b\to l}(t)$. Then $latest_{b\to k}(t) \leq e \leq latest_{b\to l}(t)$. But e is a k-event in $\partial_b(t)$ and so $e \leq latest_{b\to k}(t)$. Thus it follows that $e = latest_{b\to k}(t) \in Primary_b(t)$. Thus $e \preceq^t_b e'$.

   $e' \in Primary_a(t)$: By an argument symmetric to that in the previous case, we can show that $e \preceq^t_a e'$.

□

6.3 Labelling

Let $\{\mathscr{L}_{ij} \mid i \leq j < N\}$ be a collection of mutually disjoint sets of labels, each of size $4N + 1$. Let $\mathscr{L} = \bigcup_{i \leq j < N} \mathscr{L}_{ij} \cup \{l_\perp\}$. Also, for any trace $t = (\mathscr{E}, \leq, \lambda)$, let $P(t) = \{latest_{a \to b}(t) \mid a, b < N\}$, and $U(t) = \{lu^b_{a \to b}(t) \mid a, b < N\}$.

Definition 48. *Let $t = (\mathscr{E}, \leq, \lambda)$ be a trace. We say that $(\rho, \sigma_\ell, \sigma_u, \tau_\ell, \tau_u, \trianglelefteq_0, \ldots, \trianglelefteq_{N-1})$ is a* **valid $\mathscr{L}$-labelling** *for t iff:*

- *$\rho : \mathscr{E} \to \mathscr{L}$ is a function such that:*
  - *for all $e \in \mathscr{E}$, if $\lambda(e) = I$, $\rho(e) = l_\perp$.*
  - *for all $e \in \mathscr{E}$ and $a < N$, if $\lambda(e) = U^a$, $\rho(e) \in \mathscr{L}_{aa}$.*
  - *for all $e \in \mathscr{E}$ and $a \leq b < N$, if $\lambda(e) = S^{ab}$, $\rho(e) \in \mathscr{L}_{ab}$.*
  - *for all $e, e' \in P(t)$, if $\rho(e) = \rho(e')$, then $e = e'$.*
  - *for all $e, e' \in U(t)$, if $\rho(e) = \rho(e')$, then $e = e'$.*
- *$\sigma_\ell : [N]^2 \to \mathscr{L}$ is a function such that $\sigma_\ell(a, b) = \rho(latest_{a \to b}(t))$.*
- *$\sigma_u : [N]^2 \to \mathscr{L}$ is a function such that $\sigma_u(a, b) = \rho(lu^b_{a \to b}(t))$.*
- *$\tau_\ell : [N]^3 \to \mathscr{L}$ is a function such that $\tau(a, b, i) = \rho(latest_{a \to b \to i}(t))$.*
- *$\tau_u : [N]^3 \to \mathscr{L}$ is a function such that $\tau(a, b, i) = \rho(lu^i_{a \to b \to i}(t))$.*
- *for all $a \in [N]$, $\trianglelefteq_a$ is a partial order on $\mathscr{L}$ such that*

$$\textit{for all } e, e' \in Primary_a(t) : \rho(e) \trianglelefteq_a \rho(e') \textit{ iff } e \leq e'$$

Theorem 49. *Let α be a run and $t = t(\alpha) = (\mathscr{E}, \leq, \lambda)$. Then there is a valid $\mathscr{L}$-labelling for t which, further, is locally computable.*

Proof. We define a labelling valid for t by induction on the length of α.

$\alpha = I$: In this case take $\rho(e_\perp) = l_\perp$, $\sigma_\ell(a, b) = \sigma_u(a, b) = l_\perp$, and $\tau_\ell(a, b) = \tau_u(a, b) = \{l_\perp\}$. We also take $l_\perp \trianglelefteq_a l_0$ for all a. It is an immediate observation that $(\rho, \sigma_\ell, \sigma_u, \tau_\ell, \tau_u, \trianglelefteq_0, \ldots, \trianglelefteq_{N-1})$ is a valid labelling for t.

$\alpha = \alpha' \cdot o$: Suppose $t' = t(\alpha') = (\mathscr{E}', \leq', \lambda')$, $\mathscr{E} \setminus \mathscr{E}' = \{e_{max}\}$, and let $(\rho', \sigma'_\ell, \sigma'_u, \tau'_\ell, \tau'_u, \trianglelefteq'_0, \ldots, \trianglelefteq'_{N-1})$ be a valid $\mathscr{L}$-labelling for t'.

We now define the new labelling functions $\rho, \sigma_\ell, \sigma_u, \tau_\ell, \tau_u$ for t. There are two cases.

$o = U^a$: Let l_a be the label with least index in $\mathscr{L}_{aa} \setminus \bigcup_{j < N} \tau'_u(a, j)$. (There is one such, since $\mathscr{L}_{aa}$ is of size $> N$ and there are at most N events of the form $lu^a_{a \to j \to a}(t')$.)

We define ρ as follows:

$$\rho(e) = \begin{cases} l_a & \text{if} \quad e = e_{max} \\ \rho'(e) & \text{otherwise} \end{cases}$$

Observe from Theorem 46 that for $e \in P(t)$, either $e = e_{max}$ and $\rho(e) = l_a$ or $e \in P(t')$ and $\rho(e) = \rho'(e) \neq l_a$. Thus for $e, e' \in P(t)$, if $\rho(e) = \rho(e')$ then either $e = e' = e_{max}$ or $e, e' \in P(t')$ and $\rho'(e) = \rho'(e')$, whence by induction hypothesis $e = e'$.

Similarly from Theorem 46, for $e \in U(t)$, either $e = e_{max}$ and $\rho(e) = l_a$ or $e \in U(t')$ and $\rho(e) = \rho'(e) \neq l_a$. Thus for $e, e' \in U(t)$, if $\rho(e) = \rho(e')$ then either $e = e' = e_{max}$ or $e, e' \in U(t')$ and $\rho'(e) = \rho'(e')$, whence by induction hypothesis $e = e'$.

We define σ_ℓ as follows:

$$\sigma_\ell(c,d) = \begin{cases} l_a & \text{if} \quad c = d = a \\ \sigma'_\ell(c,d) & \text{otherwise} \end{cases}$$

We define σ_u as follows:

$$\sigma_u(c,d) = \begin{cases} l_a & \text{if} \quad c = d = a \\ \sigma'_u(c,d) & \text{otherwise} \end{cases}$$

We define τ_ℓ as follows:

$$\tau_\ell(c,d,i) = \begin{cases} l_a & \text{if} \quad c = d = i = a \\ \tau'_\ell(c,d,i) & \text{otherwise} \end{cases}$$

We define τ_u as follows:

$$\tau_u(c,d,i) = \begin{cases} l_a & \text{if} \quad c = d = i = a \\ \tau'_u(c,d,i) & \text{otherwise} \end{cases}$$

We define $\trianglelefteq_a$ as follows:

$$l' \trianglelefteq_a l'' \text{ iff } l'' = l_a \text{ or } (l'' \neq l_a \text{ and } l' \trianglelefteq'_a l'').$$

It is clear that from the assumptions on $\sigma'_\ell, \sigma'_u, \tau'_\ell, \tau'_u, \trianglelefteq'_a$ and Theorem 46 that $\sigma_\ell, \sigma_u, \tau_\ell, \tau_u$, and $\trianglelefteq_a$ satisfy the conditions for a valid labelling of t.

$o = S^{ab}$: Assume w.l.o.g. that $a < b$. Let l_{ab} be the label with least index in $\mathscr{L}_{ab} \setminus \bigcup_{j<N} \left[\tau'_\ell(a,j) \cup \tau'_\ell(b,j)\right]$. (There is one such, since $\mathscr{L}_{ab}$ is of size $> 4N$ and there are at most $4N$ events e with $\lambda'(e) = S^{ab}$ in $\mathit{Secondary}_a(t') \cup \mathit{Secondary}_b(t')$ – namely the events $\mathit{latest}_{c \to j \to d}(t')$, for $c, d \in \{a, b\}$ and $j < N$.)

We define ρ as follows:

$$\rho(e) = \begin{cases} l_{ab} & \text{if} \quad e = e_{max} \\ \rho'(e) & \text{otherwise} \end{cases}$$

Observe from Theorem 47 that for $e \in P(t)$, either $e = e_{max}$ and $\rho(e) = l_{ab}$ or $e \in P(t')$ and $\rho(e) = \rho'(e) \neq l_{ab}$. Thus for $e, e' \in P(t)$, if $\rho(e) = \rho(e')$ then either $e = e' = e_{max}$ or $e, e' \in P(t')$ and $\rho'(e) = \rho'(e')$, whence by induction hypothesis $e = e'$.

Similarly from Theorem 47, $U(t) \subseteq U(t')$ and $\rho(e) = \rho'(e)$ for $e \in U(t)$. Thus for $e, e' \in U(t)$, if $\rho(e) = \rho(e')$ then $e, e' \in U(t')$ and $\rho'(e) = \rho'(e')$, whence by induction hypothesis $e = e'$.
For $i \notin \{a, b\}$, define w_i as below:

$$w_i = \begin{cases} a & \text{if} \quad \exists j, k : \sigma_\ell(b, i) \trianglelefteq_b \sigma_\ell(b, j) = \sigma_\ell(a, k) \\ b & \text{otherwise} \end{cases}$$

We define σ_ℓ as follows:

$$\sigma_\ell(c, i) = \begin{cases} l_{ab} & \text{if} \quad c, i \in \{a, b\} \\ \sigma'_\ell(c, i) & \text{if} \quad c \notin \{a, b\} \\ \sigma'_\ell(w_i, i) & \text{otherwise} \end{cases}$$

We define σ_u as follows:

$$\sigma_u(c, i) = \begin{cases} \sigma'_u(i, i) & \text{if} \quad c, i \in \{a, b\} \\ \sigma'_u(c, i) & \text{if} \quad c \notin \{a, b\} \\ \sigma'_u(w_i, i) & \text{otherwise} \end{cases}$$

We define τ_ℓ as follows:

$$\tau_\ell(c, i, j) = \begin{cases} \sigma_\ell(i, j) & \text{if} \quad c, i \in \{a, b\} \\ \tau'_\ell(c, i, j) & \text{if} \quad c \notin \{a, b\} \\ \tau'_\ell(w_i, i, j) & \text{otherwise} \end{cases}$$

We define τ_u as follows:

$$\tau_u(c, i, j) = \begin{cases} \sigma_u(i, j) & \text{if} \quad c, i \in \{a, b\} \\ \tau'_u(c, i, j) & \text{if} \quad c \notin \{a, b\} \\ \tau'_u(w_i, i, j) & \text{otherwise} \end{cases}$$

For $c \in \{a, b\}$, define $\trianglelefteq_c$ as follows:

$$l' \trianglelefteq_c l'' \text{ iff } l'' = l_{ab} \text{ or } \left[l'' \neq l_{ab} \text{ and } (l' \trianglelefteq'_a l'' \text{ or } l' \trianglelefteq'_b l'') \right].$$

It is clear that from the assumptions on $\sigma'_\ell, \sigma'_u, \tau'_\ell, \tau'_u, \trianglelefteq'_a, \trianglelefteq'_b$ and Theorem 47 that $\sigma_\ell, \sigma_u, \tau_\ell, \tau_u, \trianglelefteq_a$, and $\trianglelefteq_b$ satisfy the conditions for a valid labelling of t.

□

6.4 Complexity of our Solution

The total number of labels in use is at most $4.N^3$. At every stage of a run we need to $2.N^2$ labels (corresponding to σ_ℓ and σ_u, N linear orders of N elements each, and $2.N^3$ labels (corresponding to τ_ℓ and τ_u). Thus the overall amount of information that needs to be stored is $O(N^3 \log N)$, which compares favourably to the $O(N^4 \log N)$ space that the earlier solution takes (to represent all the slices of all version vector matrices).

7
Conclusion and Summary

In this work, we have have shown that version vectors which are used in distributed systems to keep track of the states of the replicas do have a bounded representation. We introduced framework of traces using which we reworked the proofs from an earlier work which also provided a bounded representation for version vectors. We used the insights from this proof along with the results used in solving the gossip problem to arrive at a more efficient bounded representation for version vectors.

There are couple of interesting questions that one can pursue in this line of work. The bounded representation of a version vector presented in our work requires $O(N^3 \log N)$. Is this bound tight? Also the mode of communication used by the replicas in the model we looked at was pairwise communication. However, if we look at the model where the replicas use a different mode of communication, say message passing, then do the version vectors still have a bounded representation? What sort of restrictions should we place on the channels to achieve a bounded representation?

These questions have not been explored in our work. So in the future we hope to address some of them.

Bibliography

[AAB04] José Bacelar Almeida, Paulo Sérgio Almeida, and Carlos Baquero. Bounded version vectors. In *DISC*, pages 102–116, 2004.

[GL02] Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[MS97] Madhavan Mukund and Milind A. Sohoni. Keeping track of the latest gossip in a distributed system. *Distributed Computing*, 10(3):137–148, 1997.

[SS05] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.