The Bloom Clock to Characterize Causality in Distributed Systems

Ajay D. Kshemkalyani(✉) and Anshuman Misra

University of Illinois at Chicago, Chicago, IL 60607, USA
{ajay,amisra7}@uic.edu

Abstract. Determining the causality between events in distributed executions is a fundamental problem. Vector clocks solve this problem but do not scale well. The probabilistic Bloom filter data structure can be used as a Bloom clock to determine causality between events with lower space overhead than vector clock; however, the Bloom filter and hence the Bloom clock naturally suffer from false positives. We give a formal protocol of the Bloom clock based on Counting Bloom filters and study its properties. We formulate the probabilities of a positive outcome, a positive being false, and a false positive for Bloom clocks as a function of the corresponding vector clocks, as well as their estimates as a function of the Bloom clocks. We also indicate how to estimate the accuracy, precision, and false positive rate of an execution slice that is identified by the Bloom timestamps of two events.

Keywords: Causality · Vector clock · Bloom clock · Bloom filter · Partial order · Distributed system

1 Introduction

Determining causality between pairs of events in a distributed execution has many applications [8,15]. This problem can be solved using vector clocks [5,10]. However, vector clocks do not scale well. Several works attempted to reduce the size of vector clocks [7,11,16,18], but they had to make some compromises in accuracy or alter the system model, and in the worst-case, were as lengthy as vector clocks. A survey of such works is included in [6].

The Bloom filter, proposed in 1970, is a space-efficient probabilistic data structure that supports set membership queries [1]. The Bloom filter is widely used in computer science. Surveys of the variants of Bloom filters and their applications in networks and distributed systems are given in [2,17]. The accuracy of a Bloom filter depends on the size of the filter (m), the number of hash functions used in the filter (k), and the number of elements added to the set (n). Bloom filters suffer from false positives but no false negatives. Recently, the idea of using the Bloom filter as a Bloom clock to determine causality between events with lower space overhead than that of vector clocks was proposed, where, like Bloom filters, the Bloom clock will naturally inherit false positives [14].

However, the Bloom clock protocol was not given. We give a formal protocol of the Bloom clock based on Counting Bloom filters. We then formulate the expressions for the probabilities of a positive outcome, a positive being false, and a false positive as a function of the corresponding vector clocks, as well as their estimates as a function of the Bloom clocks. We also study properties of the Bloom clock. We give a way to estimate the accuracy, precision, and the false positive rate for a slice of the execution as identified by two given events' Bloom timestamps.

Section 2 gives the system model. Section 3 details the Bloom clock protocol. Section 4 studies properties of the Bloom clock and discusses ways to estimate the probability that a positive is false and the probability of a false positive. Section 5 presents an analysis of the probabilities as the distance between the events varies, and this is followed by a discussion. Section 6 presents a way to estimate the accuracy, precision, and false positive rate of an execution slice that is identified by the Bloom timestamps of two events. Section 7 concludes.

2 System Model

A distributed system is modeled as an undirected graph $(\mathcal{P}, \mathcal{L})$, where $\mathcal{P}$ is the set of processes and $\mathcal{L}$ is the set of links connecting them. Let $p = |\mathcal{P}|$. Between any two processes, there may be at most one logical channel over which the two processes communicate asynchronously. A logical channel from P_i to P_j is formed by paths over links in $\mathcal{L}$. We do not assume FIFO logical channels.

The execution of process P_i produces a sequence of events $E_i = \langle e_i^0, e_i^1, e_i^2, \cdots \rangle$, where e_i^j is the j^{th} event at process P_i. An event at a process can be an *internal* event, a *message send* event, or a *message receive* event. Let $E = \bigcup_{i \in \mathcal{P}} \{e \,|\, e \in E_i\}$ denote the set of events in a distributed execution. The causal precedence relation between events, defined by Lamport's "happened before" relation [9], and denoted as $\rightarrow$, induces an irreflexive partial order $(E, \rightarrow)$.

Mattern [10] and Fidge [5] designed the vector clock which assigns a vector V to each event such that: $e \rightarrow f \iff V_e < V_f$. The vector clock is a fundamental tool to characterize causality in distributed executions [8,15]. Each process needs to maintain a vector V of size p to represent the local vector clock. Charron-Bost has shown that to capture the partial order $(E, \rightarrow)$, the size of the vector clock is the dimension of the partial order [3], which is bounded by the size of the system, p. Unfortunately, this does not scale well to large systems.

Let $\downarrow e = \{f \,|\, f \in E \wedge f \rightarrow e\} \bigcup \{e\}$ denote the causal past of event e. The vector timestamp of $\downarrow e$, $V_{\downarrow e}$ is defined as: $\forall i \in [1, p], V_{\downarrow e}[i] = V_e[i]$. The set of events $\downarrow e \bigcap \downarrow f$ represents the common past of e and f. The vector timestamp of $\downarrow e \bigcap \downarrow f$, $V_{\downarrow e \bigcap \downarrow f}$ is defined as: $\forall i \in [1, p], V_{\downarrow e \bigcap \downarrow f}[i] = \min(V_e[i], V_f[i])$.

3 The Bloom Clock Protocol

The Bloom clock is based on the Counting Bloom filter. Each process P_i maintains a Bloom clock $B(i)$ which is a vector $B(i)[1, \ldots, m]$ of integers, where $m < p$. The Bloom clock is operated as shown in Fig. 1. To try to uniquely update

$B(i)$ on a tick for event e_i^x, k random hash functions are used to hash (i, x), each of which maps to one of the m indices in $B(i)$. Each of the k indices mapped to is incremented in $B(i)$; this probabilistically tries to make the resulting $B(i)$ unique. As $m < p$, this gives a space savings over the vector clock.

```
1. Initialize $B(i) = \overline{0}$.
2. (At an internal event $e_i^x$):
   apply $k$ hash functions to $(i, x)$ and increment the corresponding $k$
   positions mapped to in $B(i)$ (local tick).
3. (At a send event $e_i^x$):
   apply $k$ hash functions to $(i, x)$ and increment the corresponding $k$
   positions mapped to in $B(i)$ (local tick). Then $P_i$ sends the message
   piggybacked with $B(i)$.
4. (At a receive event $e_i^x$ for message piggybacked with $B'$):
   $P_i$ executes
   $\forall j \in [1, m], B(i)[j] = max(B(i)[j], B'[j])$ (merge);
   apply $k$ hash functions to $(i, x)$ and increment the corresponding $k$
   positions mapped to in $B(i)$ (local tick).
   Then deliver the message.
```

Fig. 1. Operation of Bloom clock $B(i)$ at process P_i.

The Bloom timestamp of an event e is denoted B_e. Let $\mathcal{V}$ and $\mathcal{B}$ denote the sets of vector timestamps and Bloom timestamps of events. The standard vector comparison operators $<$, $\leq$, and $=$ [5,10] apply to pairs in $\mathcal{V}$ and in $\mathcal{B}$. Thus, for example, $B_z \geq B_y$ is $\forall i \in [1, m], B_z[i] \geq B_y[i]$. The Bloom clock mapping from E to $\mathcal{B}$ is many-one. $(\mathcal{B}, \leq)$ is a partial order that is not isomorphic to $(E, \rightarrow)$. If the local tick (after merge) at a receive event is optionally omitted, this may introduce some added false positives (see the discussion in Sect. 4.4 for an analysis). In the development that follows next, we assume a local tick at a receive event.

Proposition 1. *Test for $y \rightarrow z$ using Bloom clocks: if $B_z \geq B_y$ then declare $y \rightarrow z$ else declare $y \not\rightarrow z$.*

4 Properties of the Bloom Clock

4.1 Accuracy of Causality Test

We have the following cases based on the actual relationship between y and z, and the relationship inferred from B_y and B_z.

1. $y \rightarrow z$ and $B_z \geq B_y$: From Proposition 1, this results in a true positive.

2. $y \rightarrow z$ and $B_z \ngeq B_y$: This false negative is not possible because from the rules of operation of the Bloom clock, B_z must be $\geq B_y$ when $y \rightarrow z$. Thus, given a negative outcome, i.e., $B_z \ngeq B_y$, the probability that the negative outcome is false, i.e., $y \rightarrow z$, is 0.
3. $y \not\rightarrow z$ and $B_z \ngeq B_y$: From Proposition 1, this results in a true negative. Given a negative outcome of the Bloom clock test, the probability that the negative outcome is true, i.e., $y \not\rightarrow z$, is 1.
4. $y \not\rightarrow z$ and $B_z \geq B_y$: From Proposition 1, this results in a false positive.

4.2 Probability of a False Positive and Probability that a Positive is False

We define probabilities along the following lines for the false positive case (Case (4)), and for the true positive and true negative cases.

1. The probability of a false positive, which we denote as pr_{fp}, is $pr(y \not\rightarrow z$ and $B_z \geq B_y)$. This probability is the equivalent of the *error rate*, which is defined as the percentage of causal relationships that are classified incorrectly.
2. We define the probability of a positive pr_p as $pr(B_z \geq B_y)$.
3. The probability $pr(y \not\rightarrow z)$ must be evaluated using only B_y and B_z as we do not have access to vector timestamps in practice. Thus, we approximate the probability that $\exists i \,|\, V_y[i] > V_z[i]$ as the probability that $\exists i \,|\, B_y[i] > B_z[i]$, which equals $1 - pr_p$. Then, $pr(y \not\rightarrow z) = 1 - pr_p$.
4. $pr_{fp} = pr(y \not\rightarrow z) \cdot pr(B_z \geq B_y) = (1 - pr_p) \cdot pr_p$.
5. Given a positive outcome of the Bloom Clock test, $pr_{pf} = 1 - pr_p$ is used to denote the probability that a positive is false.
6. Let pr_{tp} denote the probability of a true positive. $pr_{tp} = pr_p \cdot pr_p = pr_p^2$.
7. Let pr_{tn} denote the probability of a true negative. For a negative outcome having probability $1 - pr_p$, it is certain that $y \not\rightarrow z$ and hence in this case $pr_{tn} = 1$; and for a positive outcome having probability pr_p, in this case $pr_{tn} = 0$. So $pr_{tn} = 1 \cdot (1 - pr_p) = 1 - pr_p$.

The probabilities pr_{pf} and pr_{fp} are functions of pr_p. We now show how pr_p can be calculated for the false positive case (Case (4)) where $B_z \geq B_y$, otherwise if $B_z \ngeq B_y$ it is defined as 0. Observe that $pr(y \rightarrow z)$ and $pr(y \not\rightarrow z)$ are estimated as pr_p and $1 - pr_p$. However, as B_y and B_z are inputs, one could define the second term of $pr_{fp} = (1 - pr_p) \cdot pr_p$, which is $pr(B_z \geq B_y)$, as a step function $pr_{\delta(p)}$ which equals 1 if $B_z \geq B_y$ and 0 otherwise. Then pr_{fp} becomes $(1 - pr_p) \cdot pr_{\delta(p)}$ and pr_{pf} remains $1 - pr_p$ and evaluates to pr_{fp}. Also, pr_{tp} becomes $pr_p \cdot pr_{\delta(p)}$ and pr_{tn} becomes $1 - pr_{\delta(p)}$. It is a difference in perspective.

Events in $\downarrow y \bigcap \downarrow z$ contribute exactly equally to increments in B_y and B_z. Beyond those increments, we have the following. $\downarrow y \setminus \downarrow z \neq \emptyset$. Events in $\downarrow y \setminus \downarrow z$ contribute to increments in B_y. Disjoint events in $\downarrow z \setminus \downarrow y$ contribute to increments in B_z. This happens in such a way that for each increment to an index in B_y due to events in $\downarrow y \setminus \downarrow z$, there is an increment to the same index in B_z due to disjoint events in $\downarrow z \setminus \downarrow y$. The probability of this occurrence is pr_p.

We now formulate the precise expression for pr_p using vector timestamps V_y and V_z, if they were available. Then we estimate this pr_p using Bloom timestamps B_y and B_z.

Definition 1. $V_{\downarrow y \setminus \downarrow z} \equiv \forall i \in [1,p], V_{\downarrow y \setminus \downarrow z}[i] = V_y[i] - V_{\downarrow y \bigcap \downarrow z}[i]$.

Definition 2. For a vector X, $X^{sum} \equiv \sum_{i=1}^{|X|} X[i]$.

$V_{\downarrow y \setminus \downarrow z}$ gives the process-wise number of events in $\downarrow y \setminus \downarrow z$ whereas $V^{sum}_{\downarrow y \setminus \downarrow z}$ gives the total number of events in $\downarrow y \setminus \downarrow z$.

As analyzed above, for a false positive to occur, for each increment to $B_y[i]$ due to events in $\downarrow y \setminus \downarrow z$, there is an increment to $B_z[i]$ due to disjoint events in $\downarrow z \setminus \downarrow y$. The expected number of increments to $B_y[i]$, which we denote as c the *count threshold*, is $kV^{sum}_{\downarrow y \setminus \downarrow z}/m$. The probability pr_p of $B_z \geq B_y$ is now formulated. Let $b(l, n, 1/m)$ denote the probability mass function of a binomial distribution having success probability $1/m$, where l increments have occurred to a position in B_z after applying uniformly random hash mappings n times. From the above analysis, it follows that $n = kV^{sum}_{\downarrow z \setminus \downarrow y}$ times. Then,

$$b(l, kV^{sum}_{\downarrow z \setminus \downarrow y}, 1/m) = \binom{kV^{sum}_{\downarrow z \setminus \downarrow y}}{l} (\frac{1}{m})^l (1 - \frac{1}{m})^{kV^{sum}_{\downarrow z \setminus \downarrow y} - l} \tag{1}$$

The expected number of increments to $B_y[i]$ is $kV^{sum}_{\downarrow y \setminus \downarrow z}/m$. The probability that less than the count threshold $kV^{sum}_{\downarrow y \setminus \downarrow z}/m$ increments have occurred to $B_z[i]$ is given by:

$$\sum_{l=0}^{\lceil kV^{sum}_{\downarrow y \setminus \downarrow z}/m - 1 \rceil} b(l, kV^{sum}_{\downarrow z \setminus \downarrow y}, 1/m) \tag{2}$$

The probability that each of the m positions of B_z is incremented at least $kV^{sum}_{\downarrow y \setminus \downarrow z}/m$ times (after events in $\downarrow y \bigcap \downarrow z$), which gives pr_p, can be given by:

$$pr_p(k, m, V_y, V_z) = (1 - \sum_{l=0}^{\lceil kV^{sum}_{\downarrow y \setminus \downarrow z}/m - 1 \rceil} b(l, kV^{sum}_{\downarrow z \setminus \downarrow y}, 1/m))^m \tag{3}$$

Equation 3 assumed access to vector timestamps $V_{\downarrow y \setminus \downarrow z}$ and $V_{\downarrow z \setminus \downarrow y}$, which are derived from V_y and V_z. If only Bloom clocks are maintained, then we can approximate $pr_p(k, m, V_y, V_z)$ to $\widehat{pr_p}(k, m, B_y, B_z)$ as follows. Clearly, $B^{sum}_z \geq B^{sum}_y$ because $B_z \geq B_y$. We use B^{sum}_z as estimate of $kV^{sum}_{\downarrow z \setminus \downarrow y}$, while simultaneously varying l (for each i) from 0 to $B_y[i]$ instead of to B^{sum}_y/m (or to $kV^{sum}_{\downarrow y \setminus \downarrow z}/m$) across all i. Thus, in Eq. 3, $kV^{sum}_{\downarrow z \setminus \downarrow y}$ changes to B^{sum}_z, $\lceil kV^{sum}_{\downarrow y \setminus \downarrow z}/m - 1 \rceil$ in the summation bound changes to $B_y[i] - 1$ (i.e., the count threshold changes from $kV^{sum}_{\downarrow y \setminus \downarrow z}/m$ to $B_y[i]$), and rather than treating each position in B_y identically and raising to the exponent m, now a product is taken across all $i \in [1, m]$. This gives the following.

$$\widehat{pr_p}(k, m, B_y, B_z) = \prod_{i=1}^{m} (1 - \sum_{l=0}^{B_y[i]-1} b(l, B^{sum}_z, 1/m)) \tag{4}$$

Equation 4 treats the $V^{sum}_{\downarrow y}$ events in $V_{\downarrow y}$ and the $V^{sum}_{\downarrow z}$ events in $V_{\downarrow z}$ as disjoint and independent, whereas in reality, only the $V^{sum}_{\downarrow y \setminus \downarrow z}$ events in $V_{\downarrow y \setminus \downarrow z}$ and the $V^{sum}_{\downarrow z \setminus \downarrow y}$ events in $V_{\downarrow z \setminus \downarrow y}$ are disjoint and independent. Events in $\downarrow y \bigcap \downarrow z$ increment the Bloom clocks B_y and B_z identically in reality, whereas Eq. 4 assumes these events independently update the positions in B_y and B_z randomly through the k hash functions. This approximation is made as Bloom timestamps cannot identify the actual number of independent events.

4.3 Efficient Estimation of Probabilities

Equations 3 and 4 are time-consuming to evaluate for events y and z as the execution progresses. Specifically, Eq. 4 has to consider events in the entire causal past of y and z. A binomial distribution $b(l, n, 1/m)$ can be approximated by a Poisson distribution with mean n/m, for large n and small $1/m$. Also, the cumulative mass function of a Poisson distribution is a regularized incomplete gamma function. This provides an efficient way of evaluating Eqs. 3 and 4.

A more efficient-to-evaluate estimate $\widehat{pr_p}$ can be obtained by taking the Bloom clock equivalents of $V^{sum}_{\downarrow y \setminus \downarrow z}$ events in $V_{\downarrow y \setminus \downarrow z}$ and the $V^{sum}_{\downarrow z \setminus \downarrow y}$ events in $V_{\downarrow z \setminus \downarrow y}$, by trying to exclude the impact of events in $\downarrow y \bigcap \downarrow z$. For B_y and B_z, the common increments to each index are $\min(\min(B_y), \min(B_z))$, which we denote *reduce*. (Here, $\min(X)$ is the lowest element in vector X.) So we reduce each index entry of B_y and B_z by *reduce* to obtain $B\_reduce_{y|z}$ and $B\_reduce_{z|y}$ vectors, respectively.

Definition 3. For B_y and B_z,

1. $B\_reduce_{y|z} \equiv \forall i \in [1, m], B\_reduce_{y|z}[i] = B_y[i] - \min(\min(B_y), \min(B_z))$
2. $B\_reduce_{z|y} \equiv \forall i \in [1, m], B\_reduce_{z|y}[i] = B_z[i] - \min(\min(B_y), \min(B_z))$

We then use $B\_reduce_{y|z}$ and $B\_reduce_{z|y}$ instead of B_y and B_z in Eq. 4 to get the following.

$$\widehat{pr_p}(k, m, B_y, B_z) = \prod_{i=1}^{m}(1 - \sum_{l=0}^{B\_reduce_{y|z}[i]-1} b(l, B\_reduce^{sum}_{z|y}, 1/m)) \tag{5}$$

4.4 Ticking at a Receive Event

In the Bloom clock protocol given in Fig. 1, omitting the local tick at a receive event slows the growth of the Bloom clock but introduces more false positives which depend on the partial order induced by the communication pattern. Let s and r denote a send and receive event, respectively. Let the message sent at s^y_i be received at r^z_j and let $s^{z-1}_j \rightarrow s^y_i$. Then $B_{s^y_i} = B_{r^z_j}$. For events e^w_k such that $s^y_i \rightarrow e^w_k$, then $B_{r^z_j} \leq B_{e^w_k}$ even though $r^z_j \not\rightarrow e^w_k$ may be the case. The impact of such additional false positives on $\widehat{pr_p}$ is not considered in Eq. 4 or 5 and seems non-trivial to quantify. Note that the number of false positives increases further if multicasts are allowed in the system model and the local tick is omitted at a receive. This is because for one send event, there will be multiple receive events and all these receive events will have the same Bloom clock value if the tick is omitted at the receive events. If r_{j1}^{z1} and r_{j2}^{z2} are two such receive events, then for any event $e1$ such that $r_{j1}^{z1} \rightarrow e1$ and $r_{j2}^{z2} \not\rightarrow e1$, the false positive $r_{j2}^{z2} \rightarrow e1$ will be inferred.

In Fig. 1, with a local tick at a receive event, a more accurate test for a false positive gives the following instead of Proposition 1.

Proposition 2. *Test for $y \rightarrow z$ using Bloom clocks: if $B_z \geq B_y \bigwedge B_z^{sum} \geq B_y^{sum} + k$ then declare $y \rightarrow z$ else declare $y \not\rightarrow z$.*

To compute the new $\widehat{pr_p}$, the expression in Eq. 4 or 5 needs to be multiplied by the conditional probability that $B_z^{sum} \geq B_y^{sum} + k$, given that $B_z \geq B_y$. This probability depends on the selection of y and z, and on $|E|$. However, we expect it can be approximated to 1 and hence Eq. 4 and 5 which are based on Proposition 1 are still good estimates of $\widehat{pr_p}$.

5 Analysis and Discussion

For arbitrary events y and z, as $V_{\downarrow z \setminus \downarrow y}^{sum} - V_{\downarrow y \setminus \downarrow z}^{sum}$ increases, or equivalently in terms of the Bloom clock, as $B_z^{sum} - B_y^{sum}$ increases, we can predict the following trends from the definitions of pr_p, pr_{fp}, and pr_{pf}.

1. pr_p, the probability of a positive, is low if z is close to y and this probability increases as z goes further in the future of y. This is because, in Eq. 4, as B_z^{sum} increases with respect to B_y^{sum} or rather its m components, the summation (cumulative probability distribution function) decreases and hence $\widehat{pr_p}$ increases. Likewise for Eq. 5.
   This behavior is intuitive because intuition says that as z becomes more distant from y, the more is the likelihood that some causal relationship will get established from y to z either directly or transitively, by the underlying message communication pattern.
2. pr_{pf}, the probability that a positive is false, decreases as z goes further in the future of y. This is because pr_{pf} is defined as $1 - pr_p$.
   This behavior is also intuitive. Given a positive outcome, if z is close to y (B_z^{sum} is just a little greater than B_y^{sum}), it is unlikely that a causal relationship has been established either directly or transitively from y to z by the underlying message communication pattern, and thus pr_{pf} will tend to be higher; as z goes more distant from y, this likelihood increases, resulting in a lower pr_{pf}.
3. pr_{fp}, the probability of a false positive, which is the product of pr_p and pr_{pf}, is lower than the above two probabilities. It will likely reach a maximum of 0.25 and then decrease.

If $pr_{\delta(p)}$ were used instead of pr_p for $pr(B_z \geq B_y)$, then pr_{fp} would be higher for a positive outcome. Once $B_z \geq B_y$ becomes true, it steps up from 0 and then as z goes into the future of y, it decreases.

We remind ourselves that these probabilities depend on B_y, B_z, k, and m, and observe that they are oblivious of the communication pattern in the distributed execution.

There is a trade-off using Bloom clocks. m can be chosen, as desired, arbitrarily less than p, for space savings. To minimize the pr_p or $\widehat{pr_p}$, the expression for the optimal number of hash functions k as a function of m, n, and c (the m values of c if Eq. 4 or 5 is used) can be derived. Alternatively, for an acceptable pr_p or $\widehat{pr_p}$, the combination of values for m and k can be determined.

We observe that many applications in distributed computing require testing for causality between pairs of events that are temporally close to each other. In checkpointing, causality needs to be tracked only between two consistent checkpoints. In fair mutual exclusion in which requests need to be satisfied in order of their logical timestamps, contention occurs and request timestamps need to be compared only for temporally close requests. For detecting data races in multi-threaded environments, a causality check based on vector clocks can be used; however, in practice one needs to check for data races only between read/write events that occur in each other's temporal locality [13]. In general, many applications are structured as phases and track causality only within a bounded number of adjacent phases [4,12].

6 Estimating Accuracy, Precision, and False Positive Rate

Accuracy (*Acc*), Precision (*Prec*), Recall (*Rec*), and False Positive Rate (*fpr*) are metrics defined over all data points, i.e, pairs of events, in the execution. Let TP, FP, TN, and FN be the number of true positives, number of false positives, number of true negatives, and the number of false negatives, respectively. Observe that FN is 0 as there are no false negatives. We have:

$$\begin{gathered} Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, Precision = \frac{TP}{TP+FP}, \\ Recall = \frac{TP}{TP+FN}, fpr = \frac{FP}{FP+TN} \end{gathered} \tag{6}$$

Recall is always 1 with Bloom clocks. Given events y and z and their Bloom timestamps B_y and B_z, there is not enough data to compute these metrics. So we consider the slice of the execution from y to z and define the metrics over the set of events X in this slice. Specifically, we fix event y and we let z' be virtual events from y to z and estimate the TP, FP,TN, and FN of events $x \in X$ with respect to each other $x' \in X$. We define $v = (B_z^{sum} - B_y^{sum})/k + 1$ virtual events x, having timestamps such that $B_x^{sum} = B_y^{sum} + k(i-1)$, for $i \in [1, v]$.

- The contribution of each $B_{x'}$ (w.r.t. B_x) to TP is $pr(x \rightarrow x'$ and $B_{x'} \geq B_x)$, which is estimated as $pr(B_{x'} \geq B_x) \cdot pr(B_{x'} \geq B_x) = pr_p^2$ for that x' w.r.t. x. If $pr_{\delta(p)}$ were used for $pr(B_{x'} \geq B_x)$ in the second term, the contribution to TP would be $pr_p \cdot pr_{\delta(p)}$.
- The contribution of each $B_{x'}$ (w.r.t. B_x) to FP is estimated as $pr_{fp} = (1 - pr_p) \cdot pr_p$ for that x' w.r.t. x. (See the discussion and definitions at the start of Sect. 4.2.)
  If $pr_{\delta(p)}$ were used for $pr(B_{x'} \geq B_x)$ in the second term, the contribution to FP would be $(1 - pr_p) \cdot pr_{\delta(p)}$.
- The contribution of each $B_{x'}$ (w.r.t. B_x) to TN is $pr(x \not\rightarrow x'$ and $B_{x'} \not\geq B_x)$. If $B_{x'} \not\geq B_x$ then certainly $x \not\rightarrow x'$. (See Case 3 of Sect. 4.1). So the contribution is estimated as $1 \cdot pr(B_{x'} \not\geq B_x) = (1 - pr_p)$ for that x' w.r.t. x.
  If $pr_{\delta(p)}$ were used for $pr(B_{x'} \geq B_x)$ in the second term, the contribution to TN would be $1 \cdot (1 - pr_{\delta(p)})$.

Let $pr_p(x, x')$ denote pr_p for event x' with respect to event x, where $x, x' \in X$. We get the following estimates.

The equivalent of the error rate is given by $1 - Acc$.

$$\begin{aligned} 1 - \widehat{Acc} &= \frac{FP}{\sum_{x,x'} 1} \\ &= \frac{\sum_{x,x'} pr_{fp}(x, x')}{\sum_{x,x'} 1} \\ &= \frac{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x')}{\sum_{x,x'} 1} \end{aligned} \tag{7}$$

The equivalent of the error rate that a positive is false is given by $1 - Prec$.

$$\begin{aligned} 1 - \widehat{Prec} &= \frac{FP}{TP + FP} \\ &= \frac{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x')}{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x') + (pr_p(x, x'))^2} \\ &= \frac{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x')}{\sum_{x,x'} pr_p(x, x')} \end{aligned} \tag{8}$$

fpr is the proportion of actual negatives that are misclassified as false positives.

$$\begin{aligned} \widehat{fpr} &= \frac{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x')}{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x') + (1 - pr_p(x, x'))} \\ &= \frac{\sum_{x,x'} (1 - pr_p(x, x')) \cdot pr_p(x, x')}{\sum_{x,x'} 1 - (pr_p(x, x'))^2} \end{aligned} \tag{9}$$

Note that $pr_{\delta(p)}(x, x')$ cannot be used unless we have access to Bloom timestamps for events x, x' in the execution slice X. In a real execution, we would have access to these timestamps, and we have the following in terms of $pr_{\delta(p)}$.

$$1 - \widehat{Acc} = \frac{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} 1} \tag{10}$$

$$\begin{aligned} 1 - \widehat{Prec} &= \frac{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x') + pr_p(x,x') \cdot pr_{\delta(p)}(x,x')} \\ &= \frac{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} pr_{\delta(p)}(x,x')} \end{aligned} \tag{11}$$

$$\begin{aligned} \widehat{fpr} &= \frac{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x') + (1 - pr_{\delta(p)}(x,x'))} \\ &= \frac{\sum_{x,x'}(1 - pr_p(x,x')) \cdot pr_{\delta(p)}(x,x')}{\sum_{x,x'} 1 - pr_p(x,x') \cdot pr_{\delta(p)}(x,x')} \end{aligned} \tag{12}$$

7 Conclusions

Detecting the causality relationship between a pair of events in a distributed execution is a fundamental problem. To address this problem in a scalable way, this paper gave the formal Bloom clock protocol, derived expressions for the probability of false positives and the probability that a positive is false using Bloom clock, and studied the properties of the Bloom clock. We also gave a way to estimate the accuracy, precision, and the false positive rate for a slice of the execution as identified by two given events' Bloom timestamps.

The Bloom clock is seen to offer a trade-off between accuracy (minimization of false positives) and space overhead. The trade-off provides the Bloom clock with adaptability to different scenarios. It would be interesting to study such trade-offs in some practical applications of detecting causality between event pairs, for example, fair mutual exclusion, checkpointing, or dynamic race detection in multi-threaded environments. As future work, one could compute the values of the expressions of accuracy, precision, and false positive rates for a simulated execution to study their behavior. This will give an indication about the feasibility of the Bloom clock for real applications.

References

1. Bloom, B.: Space/time tradeoffs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
2. Broder, A.Z., Mitzenmacher, M.: Survey: network applications of bloom filters: a survey. Internet Math. **1**(4), 485–509 (2003)
3. Charron-Bost, B.: Concerning the size of logical clocks in distributed systems. Inf. Process. Lett. **39**(1), 11–16 (1991)
4. Couvreur, J., Francez, N., Gouda, M.G.: Asynchronous unison (extended abstract). In: Proceedings of the 12th International Conference on Distributed Computing Systems, Yokohama, Japan, 9–12 June 1992, pp. 486–493 (1992)

5. Fidge, C.J.: Logical time in distributed computing systems. IEEE Comput. **24**(8), 28–33 (1991)
6. Kshemkalyani, A., Shen, M., Voleti, B.: Prime clock: encoded vector clock to characterize causality in distributed systems. J. Parallel Distrib. Comput. **140**, 37–51 (2020)
7. Kshemkalyani, A.D., Khokhar, A.A., Shen, M.: Encoded vector clock: using primes to characterize causality in distributed systems. In: Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, 4–7 January 2018, pp. 12:1–12:8 (2018)
8. Kshemkalyani, A.D., Singhal, M.: Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press, Cambridge (2011)
9. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978)
10. Mattern, F.: Virtual time and global states of distributed systems. In: Proceedings of the Parallel and Distributed Algorithms Conference, pp. 215–226 (1988)
11. Meldal, S., Sankar, S., Vera, J.: Exploiting locality in maintaining potential causality. In: Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, PODC 1991, pp. 231–239 (1991)
12. Misra, J.: Phase synchronization. Inf. Process. Lett. **38**(2), 101–105 (1991)
13. Pozzetti, T.: Resettable encoded vector clock for causality analysis with an application to dynamic race detection. M.S. Thesis, University of Illinois at Chicago (2019)
14. Ramabaja, L.: The bloom clock. CoRR abs/1905.13064 (2019). http://arxiv.org/abs/1905.13064
15. Schwarz, R., Mattern, F.: Detecting causal relationships in distributed computations: in search of the holy grail. Distrib. Comput. **7**(3), 149–174 (1994)
16. Singhal, M., Kshemkalyani, A.D.: An efficient implementation of vector clocks. Inf. Process. Lett. **43**(1), 47–52 (1992)
17. Tarkoma, S., Rothenberg, C.E., Lagerspetz, E.: Theory and practice of bloom filters for distributed systems. IEEE Commun. Surv. Tutor. **14**(1), 131–155 (2012)
18. Torres-Rojas, F.J., Ahamad, M.: Plausible clocks: constant size logical clocks for distributed systems. Distrib. Comput. **12**(4), 179–195 (1999)