CS 553
Project: Bloom Clocks

Anshuman Misra and Ajay D. Kshemkalyani

Muhammad Abrar Tariq
mtariq5@uic.edu

TABLE OF CONTENTS

1- Introduction

Bloom clocks are probabilistic data structure help us in determine the causality between pairs of events in distributed an environment. The storage nature of bloom clocks is similar to that of the famous Vector clocks [], as they also store the information about an event in an array like structure, but the main advantage of Bloom Clocks is their reduction of space complexity. Other techniques such as Vector Clocks suffers of the O(n) space complexity which hinders them from scaling too well []. Bloom Clocks are based on Counting Bloom Filter, where each process maintains a Bloom Clock (a vector) of size 1 to m, where m is always supposed to be strictly less than the n (the total number of processes). Much Bloom filter, for every event a respective process hashes (maps) the tuple of its P_id and Event_Id to its own Bloom Clock. This probabilistically helps us make a distinctive clock for every event and simultaneously makes it space efficient in comparison to Vector Clocks.

2- Setup

My setup's theoretical model followed the work of Anshuman Misra [1]. My environmental setup is listed in the table below.

Setup Details	
CPU	Intel® Core™ i7-6500U - 2 Cores
RAM	12 GB DDR4
Storage	100 GB
OS	Linux (Ubuntu 18.04)
Framework	GO-Lang Ver 1.5
Hashes	Murmur3, FNV, CRC32, Maphash

Fig 2.1: Details of Environmental Setup

As per guideline, I did my experiments with Process Number N = [100, 200, 300], Bloom Size M = N * Bloom Factor (Ratio of Bloom Clock to Process Number) B = [0.1, 0.2, 0.3], Number of Hashes K = [2, 3, 4] and Probability of Internal Event P_I = [0, 0.5, 0.9]. I tried to schedule the experiments for N > 300 but due to limited CPU power the time grew exponentially. For this report I have bound myself to all set of combination of values mentioned previously. For each experiment, the process was allowed to run till the n^2+20n events and in every iteration, each process could at most send 1 send-message and queue up to n-1 receive-messages from other processes. The receiver of every send message was also chosen at random to prevent process starvation and maintain fair scheduling. Every process maintained a Bloom and Vector Clock which was updated passed on to a scheduler routine on FIFO queue.

I used hash functions mentioned above by evaluating their speed to collision ratio to by famous algorithms such as SHA-1 and MD5 etc. As per hashing the window (B*N - which is pretty small), I chose to 32-bit hashing to prevent my CPU from taking any extra load.

3- Experiments and Findings

In this project I conducted my experiment with varying number of process, clock sizes and hash functions to evaluate the metrics provided to us in the work of Anshuman Misra [1]. The evaluated metrics are Pr_p, Pr_{fp}, Accuracy, Precision and Fpr.

3.1- Pr_p

Pr_p is the probability of a positive outcome. It is estimated by applying Binomial distribution upon the bloom clocks of two events. The equation to calculate Pr_p is mentioned below.

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

Fig 3.1.0: Describes the equation to estimate Pr_p [1]

I evaluated the effects of N, M, H and P_I on Pr_p by changing each only one metric at a time. For my first experiment I kept varied N and kept all of the other factor's constant, for constant factor I normally took the median of non-changing factors to prevent myself from being overshadowed by choosing a minimum or maximum secondary factor.

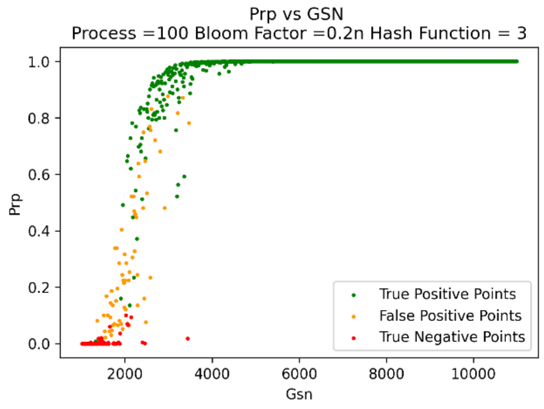

Fig 3.1.1: Prp Graph with P_I = 0.5

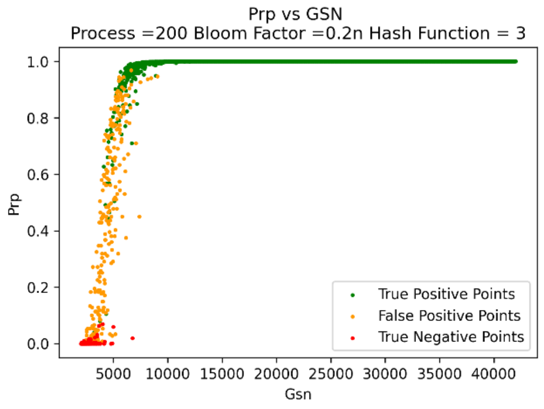

Fig 3.1.2: Prp Graph with P_I = 0.5

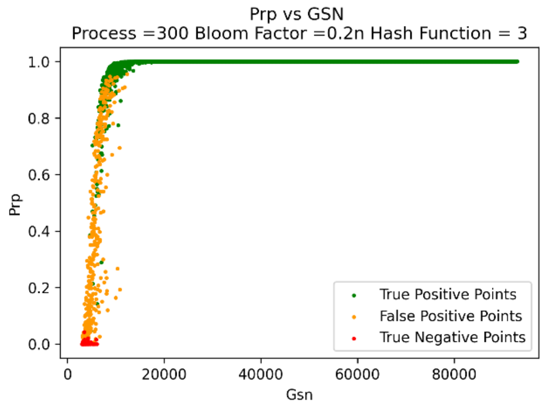

Fig 3.1.3: Prp Graph with P_I = 0.5

We can see from above graphs that the Pr_p, as we increase the total number of process (which also increases total size of our execution slice) the rate our Pr_p also increases. This is because with an increase process counts more events get executed in same time which gives our experiments more chance to isolate an event becoming positive (causal to others)

Second factor to which we check against our Pr_p is the Bloom Factor. As previously take the median of our other factors such as N = 200, K = 3 and P_I = 0.5.

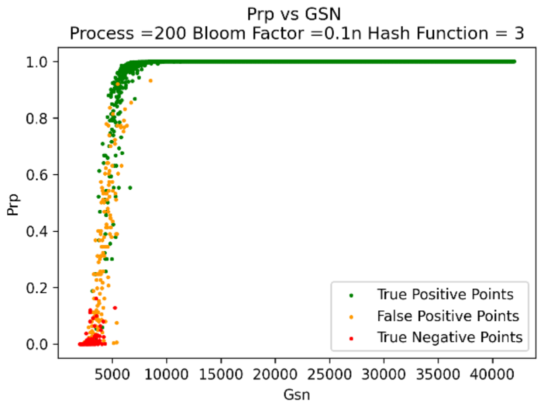

Fig 3.1.4: Prp Graph with P_I = 0.5

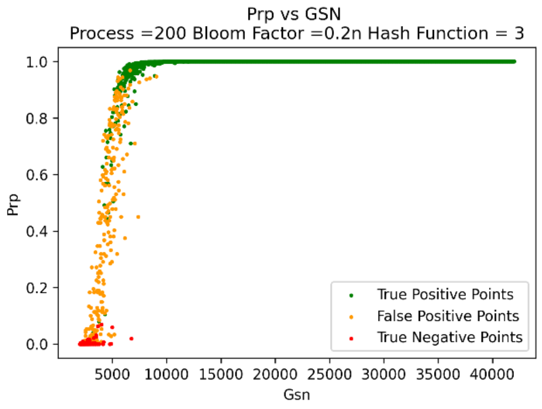

Fig 3.1.5: Prp Graph with P_I = 0.5

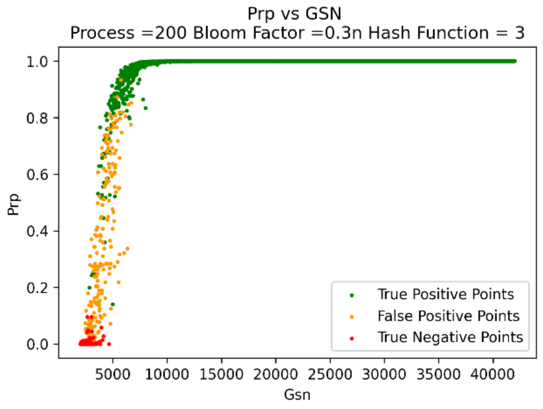

Fig 3.1.6: Prp Graph with P_I = 0.5

We can see from the experiments above that there is no noticeable impact of Bloom Factor to Pr_p. Intuitively, with more space to Hash Function our clocks should have a lower probability of false positives, but this might not be detectable by just looking at Pr_p.

Third we vary K and as per previously we take the median of our other factors such as N = 200, BF = 0.2 and P_I = 0.5.

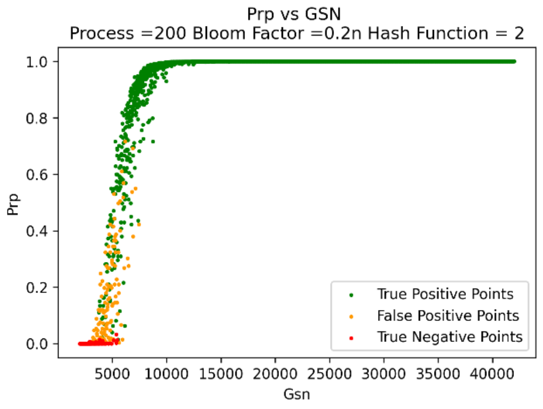

Fig 3.1.7: Prp Graph with P_I = 0.5

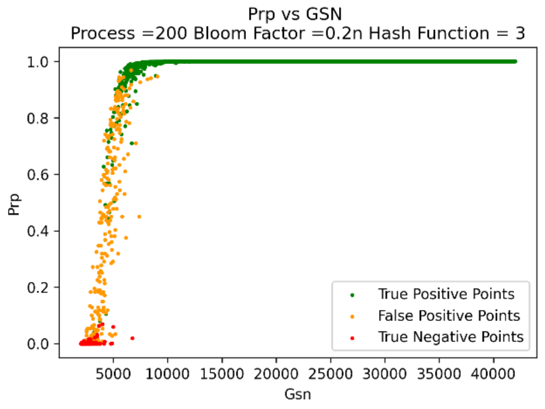

Fig 3.1.8: Prp Graph with P_I = 0.5

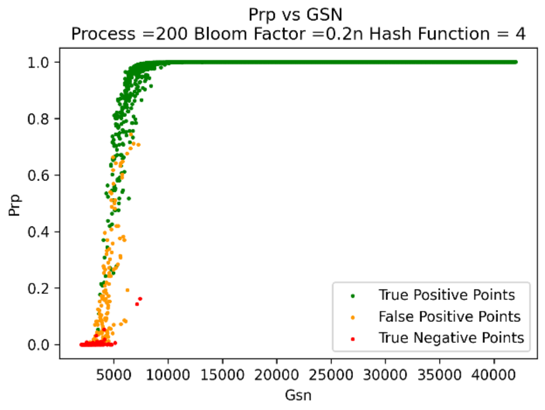

Fig 3.1.9: Prp Graph with P_I = 0.5

From the experiments if we look at the false positive in contrast to True Positives we should see decrease in number of False positive as we increase our K but fig 3.1.8 shows a sudden increase in False Positives. One could argue that the small size bloom clock might be the cause of it and that would have been true if we would have chosen the smallest Bloom Factor (0.1n) but as in this case we picked the median the reason that explains such variation is experimental anomaly which can get fixed by increasing the total number of runs.

Lastly, we vary the Probability of Internal Events (P_I). As previously we take the median of our other factors such as N = 200, BF = 0.2 and K = 3.

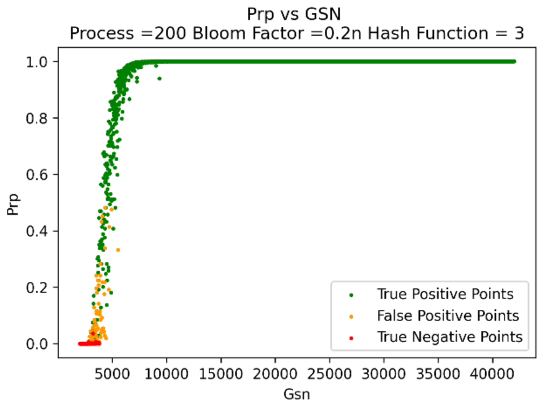

Fig 3.1.10: Prp Graph with P_I = 0

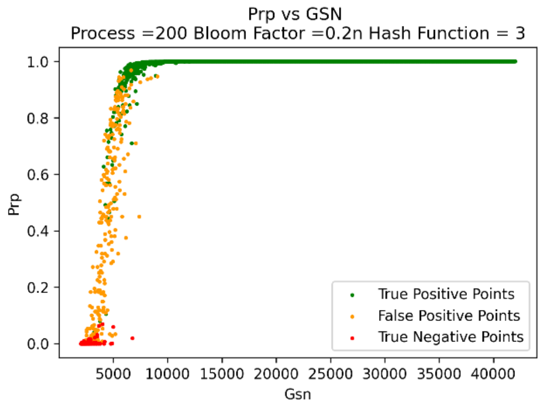

Fig 3.1.11: Prp Graph with P_I = 0.5

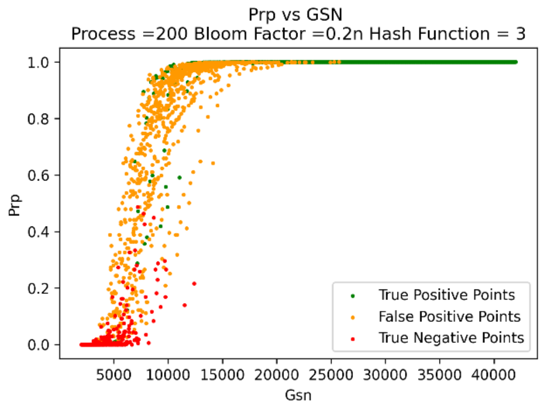

Fig 3.1.12: Prp Graph with P_I = 0.9

The graph from the above experiments exactly follow the logical conjecture. Every send event should have respective receive which in term make more events to causally related to each other. If we increase the P_I the internal events should increase and with more and more internal events there should be a relative decrease in the number causal relationships. These graphs tell us that bloom clocks will work better if there are less internal and more send/receive events.

3.2- Pr_{fp}

Pr_{fp} denotes the probability of an event being false positive. There are two ways to calculate this probability.

$$pr_{fp} = (1 - pr_p) \cdot pr_{\delta(p)} \quad (1)$$

$$pr_{\delta(p)} = \begin{cases} 1 & \text{if } B_z \geq B_y \\ 0 & \text{otherwise} \end{cases}$$

Fig 3.2.1: Method 1 for Pr_{fp}

$$pr_{fp} = (1 - pr_p) \cdot pr_p \quad (2)$$

Fig 3.2.1: Method 2 for Pr_{fp}

If Pr δ(p) were not precisely evaluated but used as a probability, we would get method 2, so one can use any of the above to evaluate Pr_{fp}. For the sake this report all figures used Method 1 to calculate Pr_{fp} unless stated otherwise.

As per our previous method we vary single factor and take median of the others.

For the first case we vary N and take the median of others BF = 0.2, K = 3, P_I = 0.5.

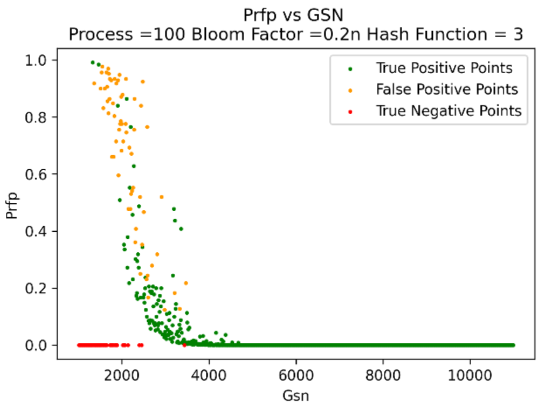

Fig 3.2.1: Prfp Graph with P_I = 0.5

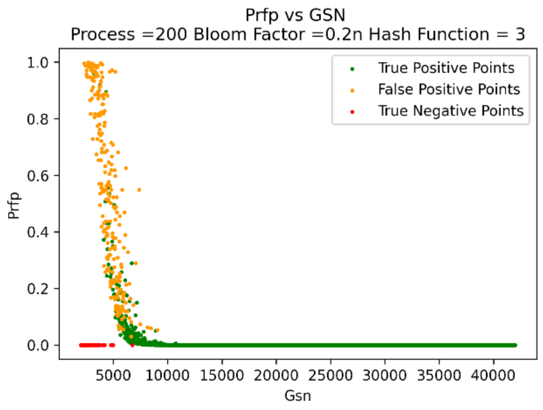

Fig 3.2.2: Prfp Graph with P_I = 0.5

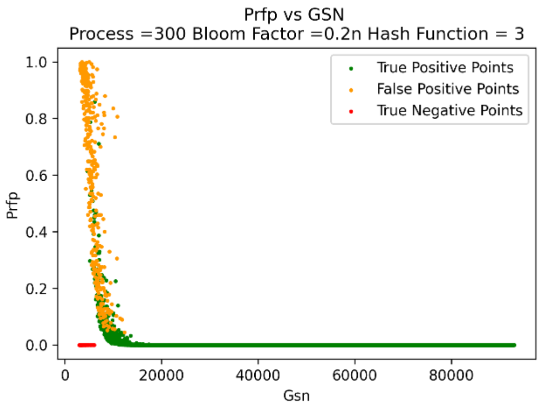

Fig 3.2.3: Prfp Graph with P_I = 0.5

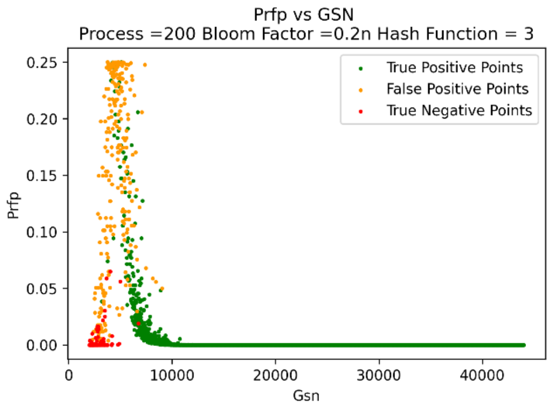

Fig 3.2.4 Prfp Graph with P_I = 0.5
Method 2

As Pr_{fp} is derived by the same Pr_p values from above so we similar trend of reaching Positive early on. The trend can be seen in Fig 3.2.1 – 3.2.3 to as we saw from Fig 3.1.1 - 3.1.3. The last figure, Fig: 3.2.4 is drawn by the other Method as described from the above. It is counter part to Fig 3.2.2. It is can been that the range Pr_{fp} in one is [0:1] while in the other is [0:0.25]. The reason for such difference in range is because Pr δ(p), as it has a step value of 0 to 1 where other value Pr_p has a rather continuous value in comparison. As discussed in another work of Anshuman Misra and Ajay D. Kshemkalyani [2], it is not apparent whether Equation 1 or 2 is better for modeling Pr_{fp}.

Secondly, we case we vary BF and take the median of others etc. N = 200, K = 3, P_I = 0.5. Much like the previous case we will be seeing the shadow effect of Pr_p from the above figures.

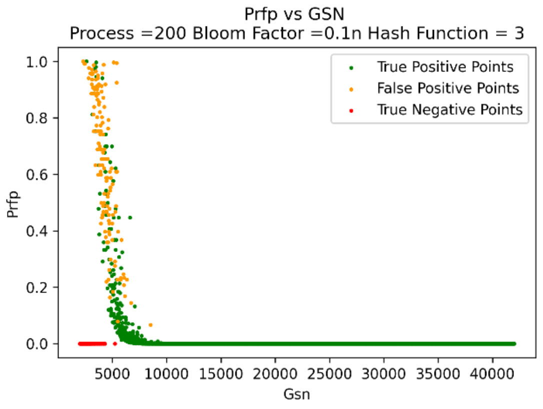

Fig 3.2.5: Prfp Graph with P_I = 0.5

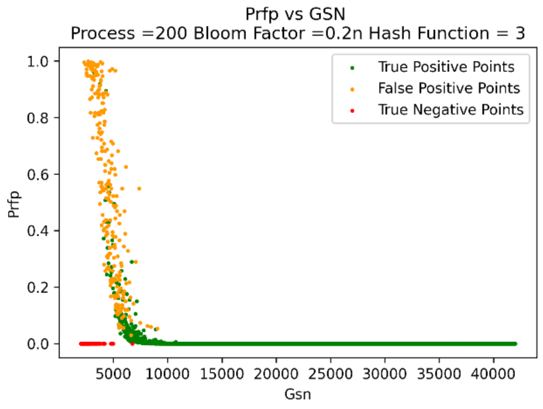

Fig 3.2.6: Prfp Graph with P_I = 0.5

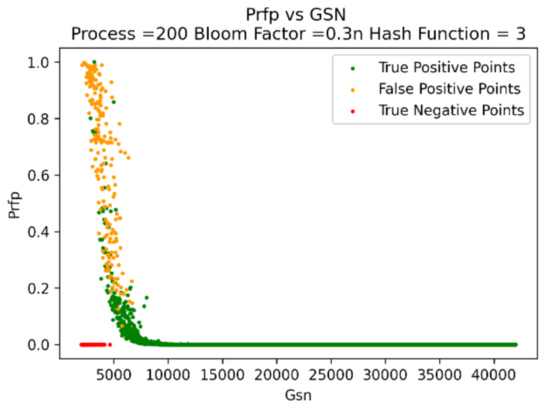

Fig 3.2.7: Prfp Graph with P_I = 0.5

As before, when we in the above section where we discuss the effect of BF on Pr_p. Same as before intuitively we should see an increase in False positives, but that change might be too less to be consistently visible to us across all graphs.

Thirdly we will change K and by being consistent with our previously chosen approach of taking Medians for the rest of values. N = 200, BF = 0.2, P_I = 0.5.

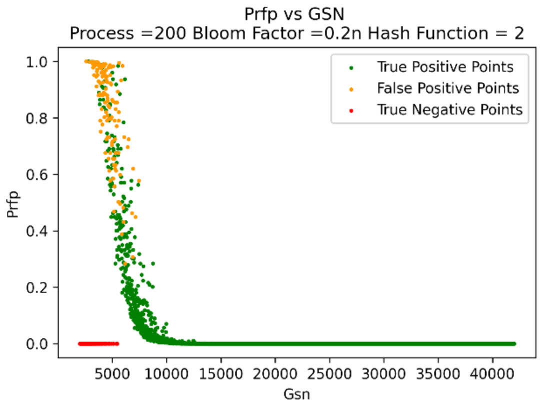

Fig 3.2.8: Prfp Graph with P_I = 0.5

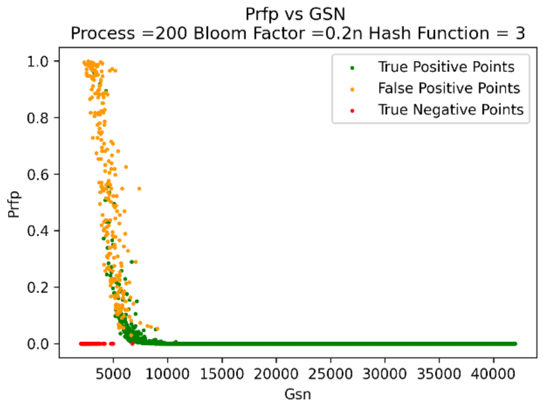

Fig 3.2.9: Prfp Graph with P_I = 0.5

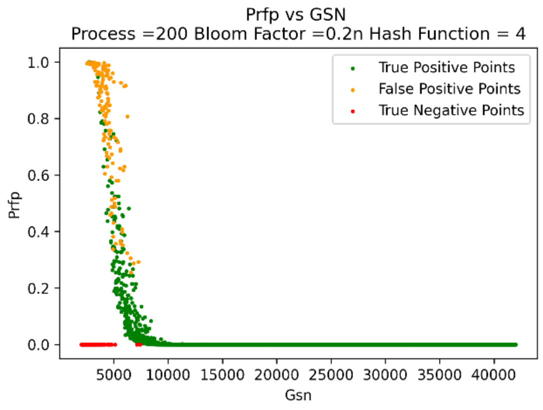

Fig 3.2.10: Prfp Graph with P_I = 0.5

We can see that like our Pr_p graphs of varying K, false positive rate decreases with an increase in K, but with the variation in Fig 3.2.9 and Fig 3.1.8 prevents us from making any strong conjecture about this. Much like before an increase number of Runs might help highlight this cause more clearly.

Last but not the least is the varying of P_I. Much like before we keep N = 200, BF= 0.2, K = 3.

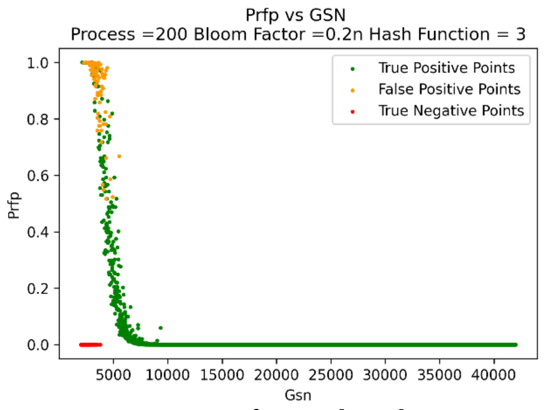

Fig 3.2.11: Prfp Graph with P_I = 0

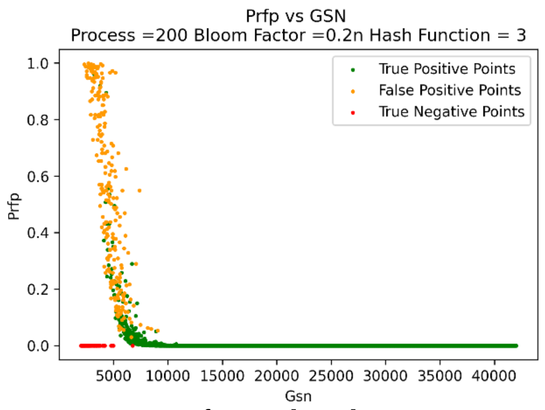

Fig 3.2.12: Prfp Graph with P_I = 0.5

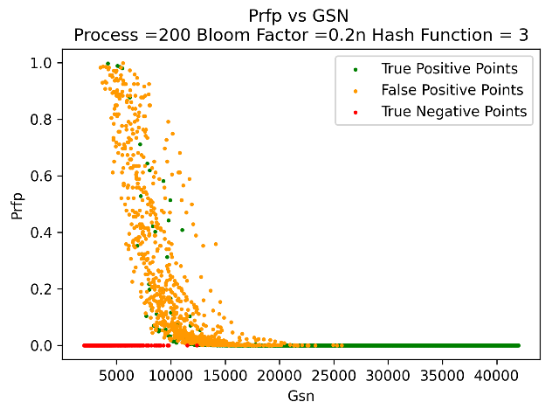

Fig 3.2.13: Prfp Graph with P_I = 0.5

The graphs follow the logical conjecture from Fig 3.1.10 - 3.2.12. Every send event should have respective receive which in term make more events to causally related to each other. If we increase the P_I the internal events should increase and with more internal events there should be relative less send/receive events which in-term should decrease in the number causal relationships. These graphs tell us that bloom clocks will work best with P_I = 0.

3.3- Metrics

Now we are to discuss the impact of metrics such Accuracy, Precision, Recall and Fpr. Since we know from the work that Recall is always 1 in Bloom clocks, we divert our full attention to computation of above metrics.

Following our previous trend, we check our metrics with varying N. BF = 0.2, K=3, P_I = 0.5

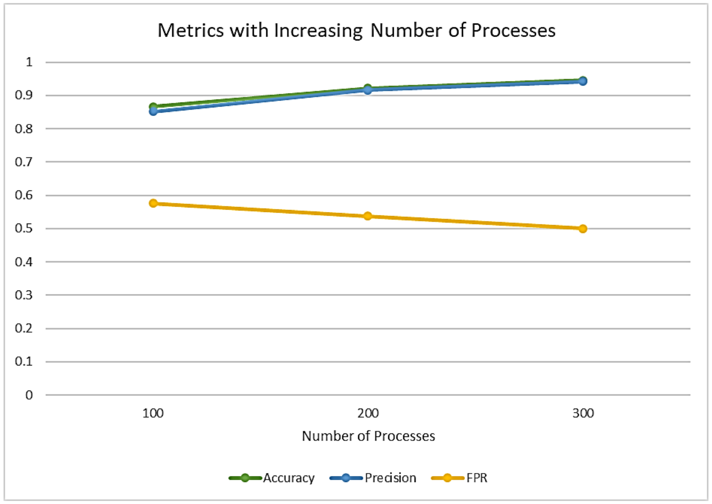

Fig 3.3.1: Metrics with increasing number of Processes

Processes	Accuracy	Precision	FPR
100	0.8656565657	0.8509302847	0.5762564991
200	0.9214160401	0.9157168969	0.5374989286
300	0.9450006180	0.9418052660	0.5004160763

We see in the table above that as we increase the number of process our Accuracy and Precision increases while simultaneously our FPR decreases. This shows that Bloom will work well with increasing number of processes. Thus, it can be a viable real-world utility.

Secondly, we vary BF, while keeping all the other values constant (median).

N = 200, K = 3, P_I = 0.5.

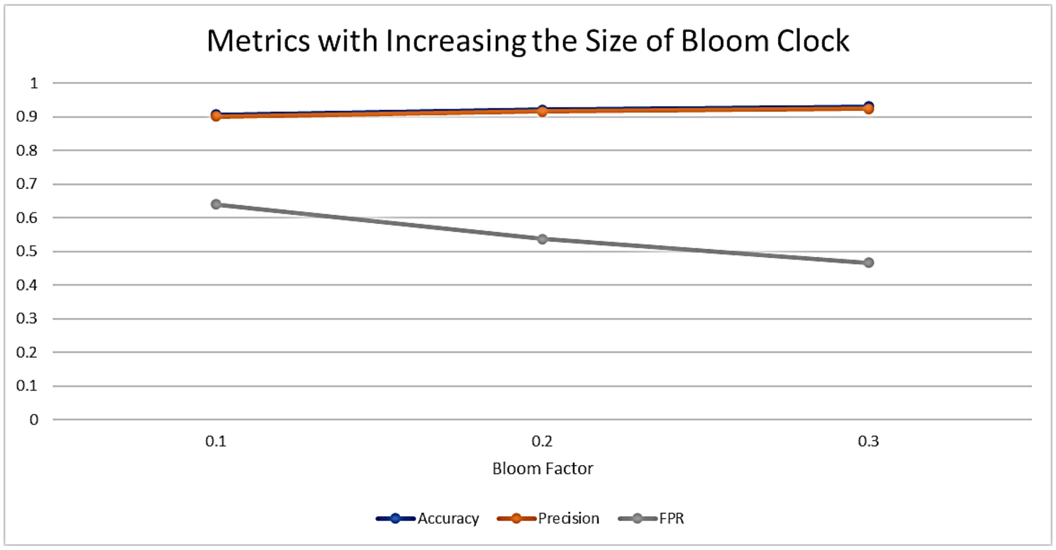

Fig 3.3.2: Metrics with increasing number of Processes

BF	Accuracy	Precision	FPR
0.1	0.906441	0.901246	0.640089
0.2	0.921416	0.915717	0.537499
0.3	0.930213	0.924174	0.467002

The improvement in the graph is completely logical because with a larger number of indices the probability of hash function outputs mapping to the same indices is lowered thus probability of false positives also reduces.

Thirdly, we vary K, while keeping all the other values constant (median).

N = 200, BF = 0.2, P_I = 0.5.

Hash Functions	Accuracy	Precision	FPR
2	0.925840	0.920111	0.508419
3	0.921416	0.915717	0.537499
4	0.924461	0.919033	0.529794

Table 3.3.3: Show our Metrics as we vary K

The difference in values is negligible to draw any valuable conclusion from the table.

Lastly, we vary the Probability of Internal Events (P_I). As previously we take the median of our other factors such as N = 200, BF = 0.2 and K = 3.

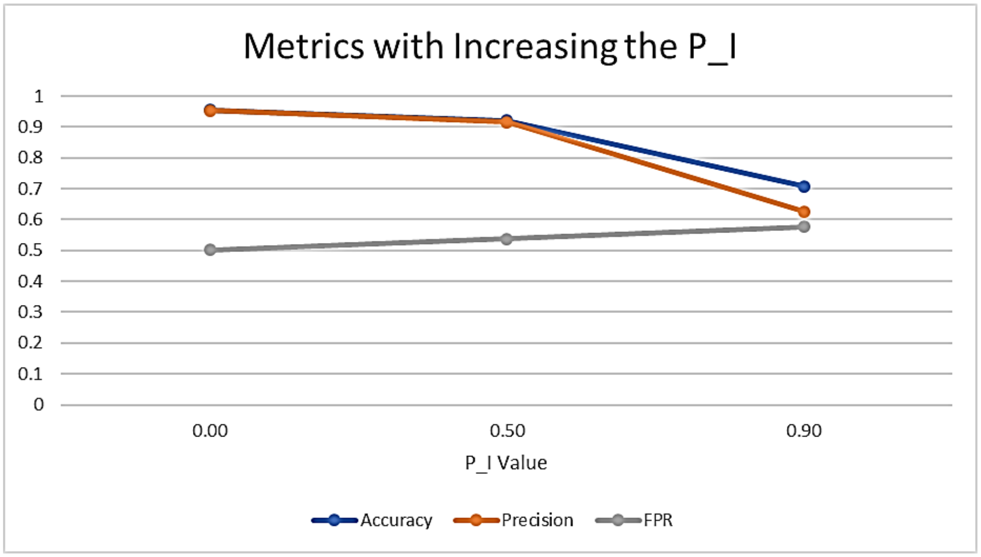

Fig 3.3.4: Metrics with increasing P_I

P_I Value	Accuracy	Precision	FPR
0.00	0.95501253	0.95290197	0.50097684
0.50	0.92141604	0.91571690	0.53749893
0.90	0.70711779	0.62657378	0.57589198

The graphs follow the logical conjecture from Fig 3.1.10 - 3.2.12. Every send event should have respective receive which in term make more events to causally related to each other. If we increase the P_I the internal events should increase and with more internal events there should be relative less send/receive events which in-term should decrease in the number causal relationships. These graphs tell us that bloom clocks will work best with P_I = 0.

3.4- Event Slice

I conducted the experiment with increasing my Event Slice Value from n^2+10n to n^2+20n but there were many minimal changes to (1-2 %) in our metrics so I did not append them in my report.

The files of each experiments were created and updated dynamically. All of the files pertaining to the project are present on a private Github repository, which can viewed upon a request to the email mentioned in the footer of title page.

4- References

1- https://www.cs.uic.edu/~ajayk/ext/BloomClockauthor.pdf
2- https://arxiv.org/pdf/2011.11744.pdf