Streaming Solutions for Fine-Grained Network Traffic Measurements and Analysis

Faisal Khan
fnkhan@ucdavis.edu
University of California, Davis

Nicholas Hosein
nhosein@ucdavis.edu
University of California, Davis

Chen-Nee Chuah
chuah@ucdavis.edu
University of California, Davis

Soheil Ghiasi
ghiasi@ucdavis.edu
University of California, Davis

ABSTRACT

Streaming network traffic measurements and analysis is critical for detecting and preventing any real-time anomalies in the network. The high speeds and complexity of today's network make the traditional slow open-loop measurement schemes infeasible. We propose an alternate closed-loop measurement paradigm and demonstrate its practical realization. To the heart of our solution are three streaming algorithms that provide a tight integration between the measurement platform and the measurements. The algorithms cater to varying degrees of computational budgets, detection latency, and accuracy. We empirically evaluate our streaming solutions on a highly parallel and programmable measurement platform. The algorithms demonstrate a marked 100% accuracy increase from a recently proposed MRT algorithm in detecting DoS attacks made up of synthetic hard-to-track elephant flows. Our proposed algorithms maintain the worst case complexities of the MRT, while empirically demonstrating a moderate increase in average resource utilization.

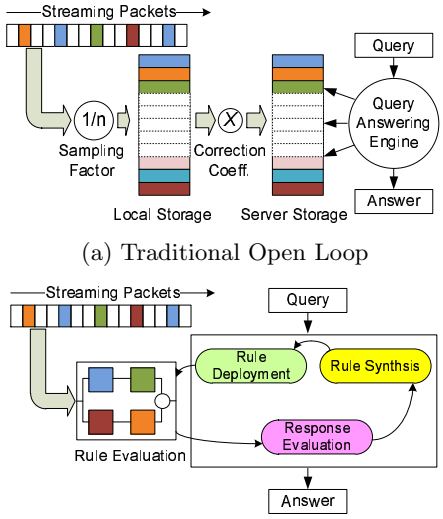
Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*; G.4 [**Mathematics of Computing**]: Mathematical software—*Algorithm design and analysis*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

1. INTRODUCTION

Accurate traffic measurement and monitoring is key in a wide range of network applications such as detection of anomalies and security attacks, and traffic engineering. A number of critical network management decisions such as blocking traffic to a victim destination, re-routing of traffic, or detection of anomalies, require extraction and analysis of real-time spatio-temporal patterns in network traffic. A high-quality network measurement tool is crucial for extracting such patterns of interest and making informed decisions to ensure proper network operation [1].

Today's high speed networks see huge amounts of streaming traffic, posing enormous computational and storage requirements for accurate traffic measurements. Traditionally, the measurements are performed by maintaining limited information of the streaming data. This is done by programming conservative sampling factors over to the routers, that maintain some very limited local storage. The collected sample is next periodically expired to high-end servers where it is post-processed in answering some higher level user-queries, such as traffic passing through a subnet or detecting a network anomaly [12]. A high level depiction of the traditional paradigm is shown in Figure-1(a). The traditional paradigm is *open-loop* based in which the measurements are typically blind, or *orthogonal*, to the user requirements.



(a) Traditional Open Loop

(b) Proposed Closed Loop

Figure 1: Network Measurement Paradigms

Being orthogonal, the open-loop schemes not only perform redundant measurements, but the scheme's reliance on sampling incurs significant measurement inaccuracies. There is a vast amount of research that is based on open-loop schemes and their shortcomings. However, the speed and scale, the size, and complexity of today's networks limit the feasibility of the paradigm in closing the loop between the measurements and the requirements in a streaming setup [7].

We address the problem using smart, goal-oriented closed-loop measurement solution, as shown in Figure-1(b). Central to our proposed scheme is a tight integration between the measurement requirements, available resources, and the actual measurements. This is achieved by breaking a higher

level user-query into multiple *rules* of finer granularities and their iterative refinement over time until the user-query gets answered. The contention is that the interesting traffic patterns could be detected or learned on the fly, via iterative rule based traffic measurements, online analysis of collected information, and closed-loop evolution of subsequent rules for further and finer traffic inspection. Each round in the iterative process guides the subsequent measurements towards the goal, thereby reducing redundant measurements and leading to answering the user-query over time.

The closed loop scheme thus temporally distributes the complexity in answering the user query by breaking it down into multiple rules, or a *rule-set*, to be answered over multiple iterations. Multi-Resolution Tiling (MRT) Algorithm [14] has been previously employed in coming up with the rule-sets in an offline closed loop settings, performing multiple passes over the same data. In an online setting, such a multiple-pass flexibility is not available, and may necessitate an aggressive formation of the rules, leading to large and redundant rule-sets. Thus the challenge in closed-loop streaming measurements is in the formation of a representative set of rules that can accurately answer the user query in reasonable time, while remaining within the computing budgets.

In this work, we present an online closed-loop measurement system that provides solution to the above challenges. Fundamental to our streaming solution are three novel algorithms that cater to different levels of computational and storage budgets, detection latencies, and user level knowledge of the anomaly. A key goal of our work is to reduce redundant measurements by closely associating the computational resources towards the required query, that is, directing the limited resources where they are needed the most. Another consideration in the design of our algorithms is to maintain scalabilities in computation and storage costs, while not compromising on the accuracy and latency of the final answer. We integrate our proposed algorithms using state of the art rule-processing platform, the BURAQ [10], that provides highly parallel and programmable computational resources on a commodity FPGA device.

We provide both empirical and analytical analysis of our algorithms. The worst case computational complexities of the proposed algorithms are similar to that of the MRT, whereas two of the three algorithms also demonstrate similar worst case storage complexities. We empirically evaluate our solution by injecting varying degrees of heavy flows [7], representing spatio-temporally distributed and intermittent DoS attacks. The results show a marked 100% increase in the detection accuracies from the MRT, with low to moderate resource utilization of the BURAQ platform. Finally, we provide mathematical upper bounds on expected false alarms while using our solution. The bounds yield interesting insights in fine tuning the system accuracy and latency, under any given network conditions and computational budgets.

2. BACKGROUND

Network traffic measurement fundamentally involves quantification of traffic that satisfies some criteria. The traffic is generally quantified in terms of *flows*, where a flow refers to a set of packets that have the same n-tuple value in their header fields. Typical definitions of the flow include 6-tuple: $\{prt, tos, sip, spt, dip, dpt\}$ where, prt is the protocol field, tos is type of service, sip and dip are the source and destination IP addresses and spt and dpt are the source and destination ports, respectively. We define a *flowset* to be an aggregation of flows. For instance, the CIDR prefix is a particular type of a flowset that aggregates over all the flows that have matching significant bits corresponding to the size of the prefix.

Traditional measurement schemes work by maintaining unique "*per-flow*" based statistics. The collected information is post-processed *offline* for answering higher-level user-queries [3] such as detecting an anomalous behavior. However, the per-flow schemes require storing information about potentially huge number of flows. The un-scalability of the per-flow scheme has traditionally been resolved using conservative packet sampling [6, 13], thereby infusing inaccuracies in answering the user-queries. Furthermore, as the collected sample is orthogonal to the user-query, the scheme involves potentially redundant data storage and subsequent data processing, making the traditional schemes practically infeasible for answering the queries in real-time.

Recently, there has been an interest in developing online closed-loop schemes to address the challenges in network measurement and analysis [11, 14]. The key observation of the schemes are top-down, goal-oriented measurements as desired by the user-query rather than the blind, bottom-up offline measurements as is done conventionally in sampling based approaches. The smart measurements are based on Multi-Tiling Resolution (MRT) Algorithm that iteratively tries to answer the user query through a progression of finite set of intermediate *rules*. A rule can be viewed as an intermediate question in pursuit of the user-query that if answered can help lead the search in a more intelligent manner. We will discuss the rules shortly in the context of MRT.

Modern networks are plagued with a variety of Denial of Service (Dos) attacks. One broad category of DoS attack tries to deplete available network bandwidth by spatio-temporal insertion of redundant data into the network. The inserted data could be in the form of a few over-sized or heavy flows, referred to as *Elephant* or *Heavy-Hitter (HH)* flows, or using a large number of small flows, the *Mice* flows.

There is a rich amount of research work that addresses the above types of attacks. Elephant flows have been the focus of researchers in [5, 7] that use sampling in the hope that it favors high intensity data. The mice attacks have been addressed by locating the heavy flowsets, or Hierarchical Heavy Hitters [4, 15], that aggregate over mice flows such that the aggregated flowsets meet the heavy-flow requirements.

2.1 Multi-Resolution Tiling Algorithm

Multi-Resolution Tiling (MRT) [14] is a recursive top-down heuristic that relies on a simple but powerful observation that if a flowset does not contain an anomaly, then no flow in that flowset can be anomalous. For instance, in the case of elephant flows, if a flowset does not consume θ-fraction of the entire network bandwidth, then no flows within that flowset may be an elephant flow. In terms of CIDR notation, the algorithm states that if a prefix is not elephant, then all its constituent prefixes of higher sizes (granularities) can be discarded from further consideration. The algorithm is tabulated in Algorithm-1.

An MRT iteration for two dimensional tuple space $\{source, destination\}$ involving a *Zoom Ratio* (ZR), or *Expansion Ratio*, of four is illustrated in Figure-2. The ZR dictates

Algorithm 1: MRT Algorithm

input : P_t: Packet enumerator at time t
input : Φ: The expansion/zoom ratio
input : R: Active rule-set
input : $R_i.Size$: Aggregate size for rule R_i
input : δ: Measurement Interval
input : θ: Threshold bandwidth consumption ratio
input : λ: Link rate (bits/second)
output: $E_f\{\}$: set of elephant-flows

$Size_{Th} \leftarrow \lambda * \theta * \delta$
/* Measurement Phase */
1 **while** $t \leq \delta$ **do**
2 **for** $R_i \in R$ **do**
3 **if** $R_i = P_t$ **then**
4 $R_i.Size \leftarrow R_i.Size + P_t.Size$

/* Decision Phase */
5 **for** $R_i \in R$ **do**
6 **if** $(R_i.Size > Size_{Th})$ **then**
7 **if** Granularity$(R_i) = MAX$ **then**
8 $E_f \leftarrow E_f + R_i$
9 **else** R.replace $\{R_i$, Expand $(R_i, \Phi)\}$
10 **else** R_i.drop

that the sample space is initially partitioned into four equal sub-regions. Statistics are next collected for the sub-region for a given measurement interval. This is achieved using rules that partition the subspace in four. Thus a rule in the context of the MRT can be viewed as a Boolean bit-mask on specific header bits that helps qualify the incoming packets. In the case of HH, the qualification helps in collection of statistics corresponding to the total bytes passing the sub-regions. The sub-regions that exceed the threshold θ, marked with a cross in the figure, are next selected for further *zooming-in*, or *expansion*, in the next-iteration. Thus each iteration in the given example results in resolution of two bits, one in each tuple space. MRT thereafter continues iterating between partitioning, statistics-collection and expansion phases until the anomalous flow is isolated.

The MRT's worst case expansion scenario corresponds to a spatio-temporal distribution of flowsets/flows such that every tracked MRT sub-region passes the threshold test. Assuming the tracked flowsets/flows remain consistent, the worst case leads to $log_\Phi(n)$ MRT iterations, where n being the number of bits of the search space resolved during the expansion, the expansion granularity. If MRT is viewed as tree structure with nodes defining the rules and levels describing the MRT iterations, then the algorithmic complexity of MRT's decision phase corresponds to the total number of nodes in the tree structure, given as $\Theta[\Phi^{log_\Phi(n)} - 1/(\Phi - 1)]$

3. MOTIVATION AND PROBLEM STATEMENT

The MRT algorithm helps in guiding the measurements in the vast n-tuple search space. However, the limited visibility under which the guided measurements have to base their decisions can lead to false negatives and positives in detecting an anomaly. For instance, a brief spike in activity may lead the MRT to incorrectly declare presence of a heavy flow, when it may only be a Flash crowd [9]. Similarly, a brief absence of an anomaly can lead the MRT to disregard a region from future consideration. Thus when the

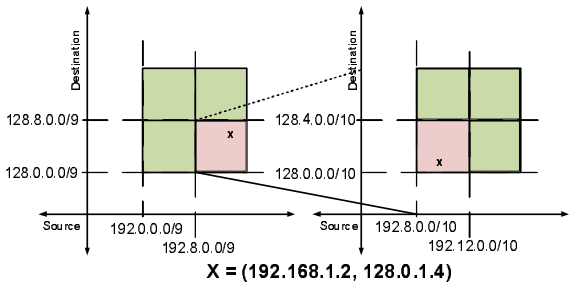


Figure 2: MRT with zoom ratio of four

anomalous behavior returns, the MRT will have to restart its tracking process from the highest granularities, resulting in false negatives and wastage of measurement resources as well as increased detection latencies.

The issue of false positives can easily be addressed by continuously tracking a declared anomalous flow. However, solving the problem of false negatives due to MRT resets is more involved. A naive solution to the problem could be to increase measurement intervals, δ, at the cost of increased tracking latency. However, as the network traffic is usually spiked and varied due to the traffic loads and network conditions, it is quite difficult to come up with an interval that can universally address the problem. Furthermore, the attacker can easily outsmart the solution by purposefully going under the measurement radars for a while.

Another solution to the problem of frequent MRT resets could be to continuously track all the sub-regions. However, the strategy could lead to tracking a huge number of regions, or rules, that may well overwhelm the measurement resources and render the detection infeasible in real-time. An intelligent measurement solution therefore needs to isolate the true rules from redundant rules for better utilization of limited computing resources. Unfortunately, it is quite difficult to predict if a rule is relevant to the search query unless it has been answered. Thus the challenge here is also to quantify the rules in determining an optimized rule-set that can yield accurate answers in optimal times while remaining within resource budgets.

In this work, we present streaming algorithms that address the above challenges in goal oriented rule-based online traffic measurements. A key challenge answered in designing the streaming measurement algorithms is to maintain measurement scalability, by maintaining an optimized rule-set. We demonstrate streaming solutions that can maintain the accuracies while significantly reducing the resource budget from naive methods. We demonstrate the algorithms in the context of heavy hitter flows. However, we will show that our solutions are quite generic and can easily be adapted for various other kinds of streaming anomalies.

4. STREAMING ALGORITHMS FOR SMART GUIDED-MEASUREMENTS

The key idea of the MRT is that one can, by observing a flowset, *infer* the characteristics of its subsets or objects (the flows). Therefore, one can selectively zoom into flowsets that might contain anomalies, such as heavy hitters, while ignoring others. As the algorithm explores the network landscape, it logs explored regions in a tree structure where nodes represent monitored regions in the IP-space. Parent nodes rep-

resent regions in IP-space which are supersets of the region covered by its children nodes, with the root node of this tree covering the entire IP-space. The number of children of a given parent node is determined by the expansion ratio.

The MRT algorithm requires persistence presence of an anomaly for its detection. This requirement is seldom met in practice where an anomaly may go under the radars, only to reappear shortly. In this section, we present streaming algorithms that overcome the challenges associated with isolating temporally distributed anomalies while preserving computational and storage scalabilities in the detection process.

4.1 Equilibrium Rollback

The Equilibrium Rollback (ER) Algorithm addresses the problem of MRT resets by adjusting the granularities of the expanded flowsets according to the temporal variations in the traffic. This is in contrast with the MRT that resets the granularities to the highest levels if the threshold requirements are not met. Instead, the Equilibrium Rollback algorithm takes into account the slow temporal variations in traffic patterns in collapsing, or zooming-out, of the expanded flowsets corresponding to the variations in the tracked regions. The algorithm thereby achieves an *equilibrium point* over the zoomed hierarchy that just passes the θ threshold requirements.

The pseudocode for ER is presented in Algorithm-2. The contention is that when the anomaly reappears, the graceful degradation over the zoomed granularity will lead to a quick re-expansion of the flowsets, instead of the slow expansion of the MRT. The cost of the algorithm is an increase in the storage requirements to keep the entire traversed hierarchy. It is to be noted that even though the entire hierarchy is stored, it is only the leaf nodes that are actively being evaluated, or constitute the active rule-set.

Algorithm 2: Equilibrium Rollback

```
   /* Decision Phase                                          */
1  for R_i ∈ R do
2      if (R_i.Size > Size_Th) then
3          if Granularity(R_i) = MAX then
4              E_f ← E_f + R_i
5          else R.replace {R_i, Expand (R_i,Φ)}
6      else if Granularity(R_i) > 1 then
7          R.replace {R_i, Collapse (R_i)}
8      else R_i.drop
```

Proposition 1. *The computational and storage complexity for Equilibrium Rollback is* $\Theta[\Phi^{log_\Phi(n)} - 1/(\Phi - 1)]$ *per rule.*

The computational complexity of ER (decision phase) is the same as that of MRT, assuming the consistency of the tracked flowsets/flows as in the case for MRT. However, whereas the storage complexity of MRT is $\Theta(2^n)$ per rule in maintaining just the leaf nodes, the ER algorithm's storage complexity corresponds to the storage requirements of the entire hierarchy for the worst case expansion.

4.2 Flow Momentum

The Equilibrium Rollback helps to gracefully degrade the zoomed granularity when an anomaly temporally goes under the measurement radars. However, in doing so, the algorithm ends up maintaining information about the entire traversed hierarchy in the hope that it may be needed if the algorithm needs to rollback. The Flow Momentum (FM) algorithm addresses the problem by instead giving the leaf nodes grace durations in the active rule-set, owing to the temporal variations in the anomaly. The grace durations are proportional to the intensity, or momentum, of the anomalous flows that guided the measurements towards the leaf-node in the first place. Thus in the case of the HH, a leaf node may be more active if the anomalous flow were of higher size.

The pseudocode for the FM Algorithm is presented in Algorithm-3. Besides reducing the storage overheads, another benefit of the FM over ER is an even faster re-expansion of the tracked flowsets when the anomaly re-appears. The price paid is a potentially bigger active rule-set size corresponding to the higher number of tracked leaf-nodes. However, the worst case storage and computational complexities for FM are the same as that of MRT.

Algorithm 3: Flow Momentum

```
   λ_i : Flow rate for flowset/flow i
   k : Algorithmic iteration
   /* Measurement Phase                                       */
1  while t ≤ δ do
2      for R_i ∈ R do
3          if R_i = P_t then
4              R_i.Size ← R_i.Size + P_t.Size
5              R_i.M ← R_i.M + P_t.Size
   /* Decision Phase                                          */
6  for R_i ∈ R do
7      λ_i ← R_i.M/k * δ
8      if (R_i.Size > Size_Th) then
9          if Granularity(R_i) = MAX then
10             E_f ← E_f + R_i
11         else R.replace {R_i, Expand (R_i,Φ)}
12         R_expanded.M ← R_parent.M
13     else if (λ_i/λ ≥ θ) then
14         R_i.Hold
15     else Ri.Drop
```

Proposition 2. *The computational and space complexity for Flow Momentum algorithm is* $\Theta[\Phi^{log_\Phi(n)} - 1/(\Phi - 1)]$ *and* $\theta(2^n)$ *per rule respectively.*

4.3 Directed Momentum

The streaming algorithms discussed above are quite generic in nature, that is, they do not take into account much opportunities or constraints presented by application or available computational platform. They are thus suited to the scenarios where the knowledge of the anomaly or the environment is limited. However, such an orthogonalization between the application/platform and the algorithm may lead to less than optimal use of the computing resources, leading to large active rule-sets. A very large rule-set can throttle the system by consuming the resources in processing redundant rules. An intelligent hacker could actually use

Algorithm 4: Directed Momentum

```
input  : R: Active rule-set
input  : Q{}: Set of Rule Processors
output: E_t{}: set of elephant-flows
Stretch ← |R| − |Q|
Pull ← |min(Stretch, 0)|
   /* Measurement Phase                                   */
1  while t ≤ δ do
2      for R_i ∈ R do
3          if R_i = P_t then
4              R_i.Size ← R_i.Size + P_t.Size
5              R_i.M ← R_i.M + P_t.Size
   /* Decision Phase                                      */
6  for R_i ∈ R do
7      if (R_i.Size > Size_Th) then
8          if Granularity(R_i) = MAX then
9              E_f ← E_f + R_i
10         else
11             R.replace {R_i, Expand (R_i,Φ)}
12             R_expanded.M ← R_parent.M
13             R_expanded.Static ← 0
14     else if DIR_MOM(R_i, Pull)/λ ≥ θ then
           R_i.Static ← R_i.Static + 1
           R_i.Hold
15     else R_i.Drop
   /* Calculates Directed Momentum                        */
    procedure DIR_MOM(R_i, Pull)
      λ_i ← (R_i.M/k ∗ δ)
      d_i^n ← exp^(Pull∗R_i.Static/Granularity(R_i))   /* Dir-Mom */
      return (λ_i/d_i^n)
    end procedure
```

this deficiency to outsmart the detection process in real-time by injecting a huge number of false flowsets, or leads, to be tracked. A smart algorithm therefore needs a mechanism to filter out redundant leads from the active rule-set.

As discussed earlier, it is quite difficult to predict how relevant a given rule is to the user-query unless it has been evaluated. However, the knowledge of an anomaly can help to intelligently quantify the rules using their past behavior. We make use of the anomaly information in designing a smart Directed Momentum (DM) algorithm that *directs* the search process by associating the limited resources where they are deemed more profitable. We develop the algorithm in the context of elephant flows. However, the ideas behind the algorithm are applicable for other types of anomalies.

A characteristic feature of elephant flows are their higher longevities. In the context of an iterative search process such as the Flow Momentum, the long lasting property of the elephant flows translates into higher expansion of the corresponding flowsets. We utilize this property in harnessing the *Momentum* to be *directed* towards the anomalous elephant flows by giving preference to the rules that have higher granularities. The Directed Momentum algorithm thus formed is tabulated in Algorithm-4.

Directed Momentum works by scaling the individual flowset longevities using a measure referred to as *Stretch*. The Stretch takes into account the availability of computational resources and could either be positive or negative. A positive Stretch describes a scenario where the active rule-set is smaller than the available rule processing resources. In contrast, a negative Stretch implies overshooting of the computational budget by the active rule-set. The DM algorithm uses the negative-stretch in coming up with a measure called *pull*, or algorithmic effort; such that the higher the pull, the higher the algorithmic effort in reducing the active rule-set. The algorithmic *pull* is combined with a rule's granularity to prefer more expanded and expanding flowsets over less-expanded or static flowsets, in a measure referred to as *Dir-Mom*, as shown in the pseudocode.

Proposition 3. *The computational and space complexity for Directed Momentum algorithm is* $\Theta[\Phi^{log_\Phi(n)} - 1/(\Phi - 1)]$ *and* $\theta(2^n)$ *per rule respectively.*

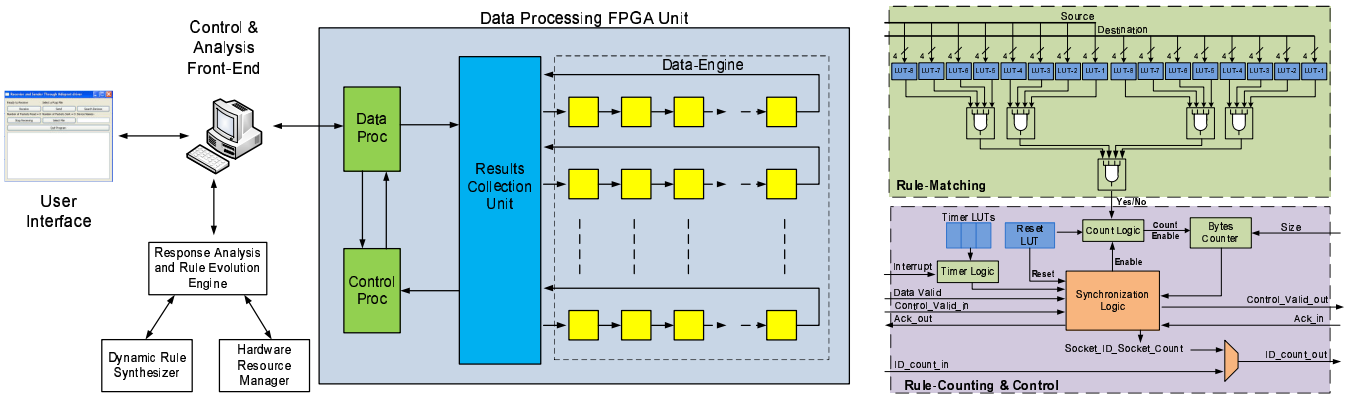
5. THE MEASUREMENT PLATFORM

The streaming algorithms discussed in previous sections are composed of two parts: (a) a data-plane to match incoming packets with the rule-set along with (b) a control-plane for algorithmic decisions to process rules that lead to measurements of finer granularities. These two planes have been previously mapped on software [14], hardware [8] and a software-hardware co-designed solutions [11]. In this work, we use the closed-loop BURAQ measurement and analysis framework [10], that combines the speed of a customized FPGA based rule-processing engine with the flexibility of a software based controller, as shown in Figure-3(a). By combining the speed with flexibility, the BURAQ framework envisions an online measurement framework that processes streaming network packets in real-time. We next present an overview of the system and study in its context the various constraints and challenges that are faced in an online measurement framework.

5.1 Rule Processing Data-Plane

BURAQ's data-plane is a custom architecture on an FPGA unit that combines a rule processing unit, the *Data-Engine*, with associated synchronization and data transfer logic. The Data-Engine itself is composed of a number of parallel rule processing units, dubbed as *sockets*, that are arranged similar to a systolic array as shown in Figure-3(a). The sockets are programmable rule matching units that are optimized to match the programmed rules and count the size of streaming network packets, independently and concurrently, in real-time. However, unlike the systolic arrays, the sockets pass on their results only to the next socket in a chain. These results from parallel chains in the Data-Engine are collected by the associated logic and communicated to higher control and analysis layers for further processing.

A high level depiction of a socket that can process two tuple $\{sip, dip\}$ rule is shown in Figure-3(b). The socket is composed of an array of Look-up-Tables (LUTs) that map the rules in the form of Boolean bit-vectors. As new rules get generated on-the-fly during algorithmic iterations, the LUTs are reprogrammed with updated Boolean bit-vectors corresponding to the new rules. One of the core features of the BURAQ framework is its novel use of fine grained Partial Dynamic Reconfiguration (PDR) of FPGA fabric in (re)programming the LUTs. The PDR programming paradigm does away with traditional just-in-time compilation of FPGA configuration data, a highly latency intensive operation. Instead, the socket (re)programming is based on minute logic changes involving specific LUTs whose entries are dynamically and directly updated in the FPGA's config-



(a) Platform Overview (b) Dynamically Reconfigurable Socket

Figure 3: BURAQ Measurement Platform [10]

uration memory. The dynamic nature of the PDR implies that only the operation of the LUT being (re)programmed is effected while the rest of the design operates as usual. We refer the interested reader to [10] for the details of the socket and its (re)programming paradigm.

5.2 Control & Analysis Front-End

The BURAQ's Control and Analysis Front-End is where user programs the high-level formulation of the measurement requirements. It incorporates a *Dynamic Rule Synthesizer* that translates the user requirements into socket deployable Boolean bit-vectors. The synthesizer also works in closed-loop with the measurements reported by the data-plane and a *Response-Evaluation Engine* in automating the exploration of the vast search space. It is the response-evaluation engine where the discussed streaming algorithms provide the streaming automation. The response evaluation engine analyzes the intermediate results from the back-end in assisting the dynamic synthesis of intermediate rules on-the-fly. The controller also maps the synthesized rules at the sockets using *Hardware Resource Manager* that performs a resource aware rule deployment at the data-plane.

5.3 Practical Constraints

Any rule-processing solution is constrained with a number of computational and communication constraints. A core constraint is the availability of computational resources for rule-processing. As rule-set grows, digging deeper in the vast n-tuple search space, it puts additional computing challenges on the limited resources. The standard practice is to *pipeline*, or *roll-over*, the additional rules, over the limited resources in multiple steps. Such a rule-pipelining increases the overall latency in yielding the final answer. For instance, in a platform employing N parallel rule-processing units, it takes $ceil(N/|R|)$ measurement cycles (each having measurement interval δ) for processing a rule-set of size $|R|$ in any given algorithmic iteration.

The active rule-sets may generally not be an integer multiple of N, and therefore there may exist available rule-processing resources in the last measurement cycles during certain algorithmic iterations. In practice, the controller can maximize the resource utilization by simultaneously mapping new rules for successive measurement phases while a portion of last rule-set is also active. However, it makes the design and analysis of an algorithm quite cumbersome. For simplicity, we therefore detach the rule-sets in discrete measurement cycles by assuming the controller only adapts new rules once a given rule-set is completely processed. We refer to such a controller implementation as *blocking*.

The synthesis of rules, deployment and collection of their results from the computing platform involve finite latencies, during which streaming packets may miss observation. We aggregate the above latencies together as reprogramming latency, denoted by ϵ. The BURAQ platform uses PDR to maximize the FPGA utilization in increasing the number of rule-processing sockets. A higher rule-processing opportunity reduces the pipelining effects, however the downside of PDR is a slight increase in reprogramming latencies as compared to a static solution [11]. Both rule pipelining and reprogramming latencies effect accuracy of reported results. We here discuss some accuracy measures for the Flow and Directed Momentum Algorithms with a more detailed analytical analysis discussed in the Appendix.

Theorem 1. (Measurement Accuracy) *If d_i^n represents DirMom in an algorithmic iteration j, then the Measurement Accuracy is given by $\delta/(\sum_{j=1}^{n} \left\lceil \frac{N}{|R^j|} \right\rceil (\delta+\epsilon).d_i^n)$, where $d_i^j = 1$ for Flow Momentum and $|R^j|$ represents the size of rule-set in algorithmic iteration j.*

Proof. For the measurement to be accurate, the observed measurements should match with ideal measurements. If $E[\alpha_i^j]$ and $E[\beta_i^j]$ represent expected values for observed and ideal measurements in an algorithmic iteration j for Flow or Directed Momentum algorithms, then for the system to be accurate

$$Measurement\ Accuracy = E[\alpha_i^j]/E[\beta_i^j] \leq 1$$

Equating the expressions for the expected values as described in Appendix leads to the desired result.

Corollary 1. *The Measurement Accuracy is less than 1 in a blocking measurement system.*

In a blocking implementation, the reprogramming (ϵ), and observation (δ) latencies do not overlap in time. This is to

say that the controller only adapts new rules at fixed time durations, rather than reading and reprogramming the sockets at different times to spread the load. In other words, the measurement and reprogramming phases are sequentially chained in blocking implementation, thereby leading to the above observation.

The system's accuracy in detecting streaming anomalies is not only a function of platform's measurement inaccuracies but also to the algorithmic inaccuracies. The algorithmic inaccuracies occur due to various abstractions that limit the complexity of the problem; for instance the finite measurement intervals that are iteratively processed. Such inaccuracies lead to possibilities of false positives and false negatives during the anomaly detection process. We discuss these possibilities in more detail and provide mathematical bounds to the presence of false alarms for heavy flow identification in the Appendix. The bounds provide tuning knobs for a user to fine-tune the system's accuracy and latency, given any platform and network conditions.

5.4 Cross-cutting Issues

Theorem-1 may appear to suggest that the Dir-Mom has a strictly inverse relationship with the system accuracy, leading to a conclusion that FM algorithm must always have superior accuracy than the DM algorithm. However, Dir-Mom also influences the rule-set size, that has its own effect on the system accuracy. As earlier stated, a rule-set that is larger than the available processing resources will have to be rolled over the platform's limited resources in successive measurement intervals. The rolling process leads to *measurement disentanglement* between the parent and the children rule-sets. Such a disentanglement reduces system accuracy and is represented by the factor $\left\lceil \frac{N}{|R|} \right\rceil$ in the above equation. Increasing Dir-Mom reduces the effect of the measurement disentanglement, through reduction in rule-set size, and as such can positively influence system accuracy. On the other hand, a very high Dir-Mom can also be counter-productive, as the factor $\left\lceil \frac{N}{|R|} \right\rceil$ is bounded by the minimum value of 1. A balanced Dir-Mom is therefore essential in maximally utilizing system resources. We will discuss the issue more when we discuss the results in the next section.

6. EMPIRICAL EVALUATION

The experiments are performed using a PC based workstation on Intel Core i7 Q740 Quad-core processor running at 1.73-GHz and having 4 GB memory. The BURAQ's rule-processing engine is mapped on a Xilinx Virtex-II Pro FPGA, XCV2VP30, running at 100-MHz, and employing $N = 169$ parallel sockets. The rule composition and results analysis is performed at the PC based controller, that is connected with the processing-engine over Ethernet. The complete system setup is shown in Figure-3(a).

We evaluate the proposed algorithms by injecting varying degrees of anomalies in CAIDA Backscatter data traces [2]. The anomalies piggyback the random trace data, and therefore, inherit real life data variations. We inserted 10 random heavy-hitter flows contributing from 0.5% to 1.4% of the total traffic in the trace. We used heavy-hitter threshold value θ to be 1%. As such, the inserted flows split evenly between true and false heavy-hitters around the threshold value, thereby producing edge test cases for evaluating the algorithms.

We also periodically inject new algorithmic seeds that have the effect of starting fresh algorithmic threads while the previous ones are active. This is done to capture new network conditions that may have been missed being captured by the previous threads. The rules produced by the threads are merged to avoid rule repetition.

We define a parameter *Score* to quantify the progress of MRT in identifying the inserted anomalies. Mathematically,

$$Score = \frac{\sum max|R_i|}{\sum |H_i|}$$

where $|R_i|$ represents the size, or expansion granularity, of the rule, R_i, in an MRT iteration that matches Heavy-Hitter, H_i. As there could be a number of rules of various sizes that match a heavy-hitter in any given algorithmic iteration, we only take into account the maximally matching rule or the rule with the highest number of matching bits with the heavy-hitter. The parameter thus represents the degree by which the algorithm has correctly (true-score) or incorrectly (false-score) identified the inserted true (false) elephant flows, with the maximum value 1 meaning all the inserted flows being completely identified.

6.1 Streaming Algorithms Analysis

The progression of True-Score (TS) with algorithmic iterations for the presented algorithms is shown in Figure-4. We also combine presented algorithms in various combinations as shown in the figure. It can be seen that the MRT performs quite poorly. As discussed in Section-3, this is due to the bursty nature of the streaming traffic where a rule falls off the rule-set if it could not consistently meet the MRT threshold condition, only to start all over again. The Rollback improvement avoids such resets by rolling an expanded rule back to its parent if it falls below the threshold, thereby showing TS improvement in the results. However, the bursty nature of the traffic means that the algorithm keeps fluctuating between the rollback and expansion phases, thus settling up with a sub-1 steady state score. The difficulties in isolation of the anomalous flows are addressed by the Momentum type algorithms that only saturate with the unit-score, that is, after identification of all the induced heavy-hitter flows. The results also show that the two Momentum type algorithms differ only slightly with respect to the algorithmic iterations in achieving the unity score.

The variation of rule-set size with algorithmic iterations for the proposed algorithms is given in Figure-5. As expected, the rule-set size for MRT is is seen to be suffering from frequent resets, corresponding to its difficulty in isolating the intermittent anomalies. Similarly, the ER algorithm is seen to be fluctuating around a steady state rule-set size, corresponding to its fluctuations in between the expansion and rollback phases. The inability of Rollback to consistently pursue a flow down to its unique identifiers is taken care of in Momentum type algorithms. The four variations of Momentum type algorithms do show nearly identical amounts of accuracies. However, they vary in the cost they require in terms of the rule-set size. The higher accuracies of FM and FM with ER can be seen to be tied with the steady increase in rule-set size. A bigger rule-set size has its consequences on the algorithm's convergence time that we will discuss shortly. The DM algorithm shows its efficacy in maintaining a steady rule-set size, while maintaining high accuracies. It therefore serves its purpose of a

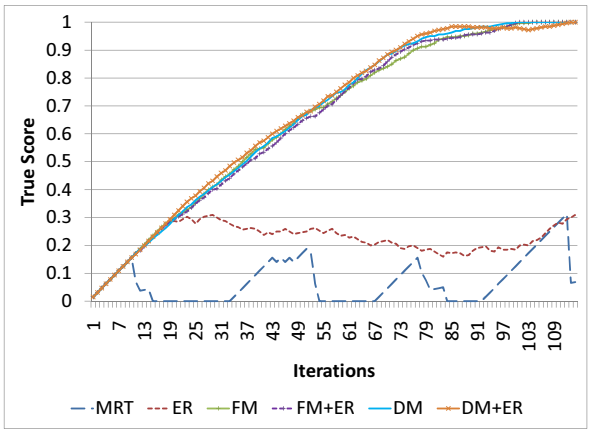


Figure 4: True Score progression with algorithmic iterations

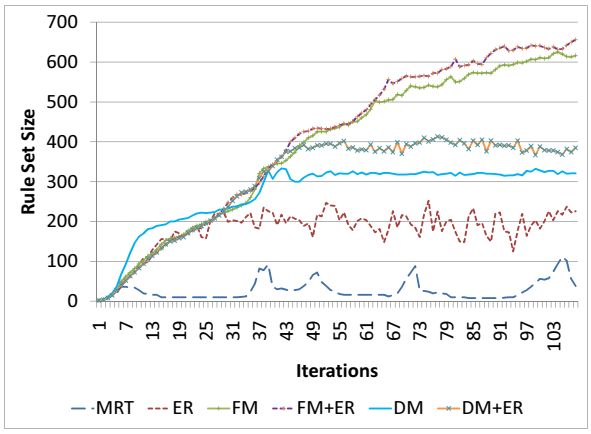


Figure 5: Rule Set Size with algorithmic iterations

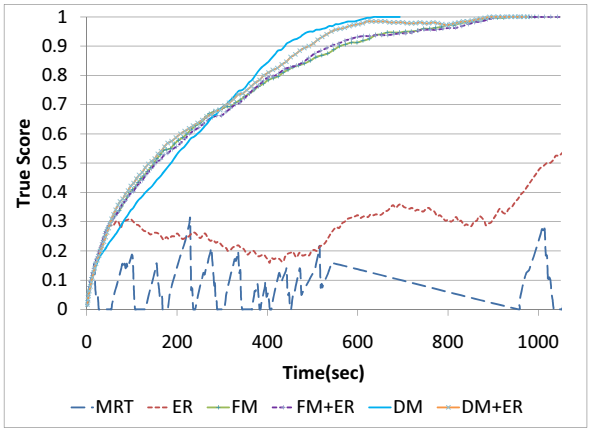


Figure 6: True Score progression with time

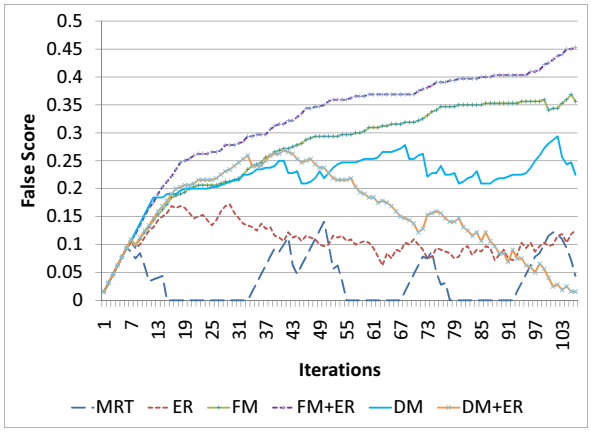


Figure 7: False Score progression with algorithmic iterations

more focused or *directed* search, by expiring the rules that are less relevant to the anomaly. A combination of DM with ER (DM+ER) also shows similar steady state rule-size convergence, albeit with a higher bias. This is because the rules that are discarded by DM are now being caught in the Roll-back catch-net. However, it is to be noted that this does not yield any improvement in TS, thus confirming the fact that the rules being discarded were indeed less relevant to the search process.

The latency in finding an anomaly using the streaming algorithms is a direct function of rule-set size. Figure-6 plots the progression of the presented algorithms with respect to their latencies. It is interesting to note that although the streaming algorithms perform quite uniformly with respect to algorithmic iterations, their differences get pronounced when algorithmic time is taken into account. It can be seen that DM outperforms all other streaming algorithms in anomaly detection latency, thanks to its ability in filtering out most relevant rule-set.

We also present variations in False-Score (FS) with algorithmic iterations in Figure-7. It is to be noted that the variations only show that the algorithms have filtered down to certain rules that match the induced false heavy-hitters with varying degrees. However, a FS value does not mean that some of the induced flows have been completely identified. Indeed, in our experiments, none of the false heavy-hitters ever got detected. This implies that although we have a False-Score, our algorithms never produced a false-positive. We note that the FS generally follows the variations in rule-set sizes. Such variations further highlight the fact that the FS is a by-product of rules that have randomly matching granularities with the induced false heavy-hitters, rather than an actual algorithmic progression towards the false flows. It is also to be noted that although we do not empirically have a false-positive, there is still a small but finite probability of false-positive. These probabilities are discussed in the Appendix.

The results yield an interesting observation that in a practical system, the efficiency of an anomalous flow detection may not necessarily be improved by increasing the rule-set size. In contrast, an intelligent search directed towards the most reliable leads can produce equivalent amount of accuracies while reducing the detection latency. The notion of 'intelligence' is of course application dependent and how efficiently it can be translated into a representative capture function. The results demonstrate significant gains that can be achieved.

6.2 Zoom Ratio

The Zoom (or Expansion) Ratio (ZR), has conventionally been associated with the speed of parsing through the search space. Conceptually, a higher ZR translates into quicker drilling down to higher granularities, and therefore should accelerate the isolation of an anomaly. We hereby investi-

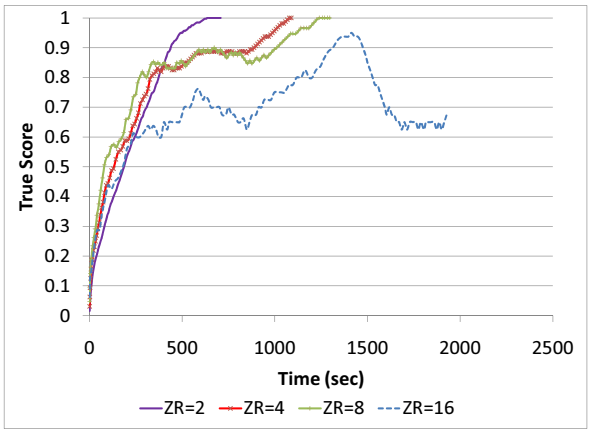


Figure 8: Variation of ZR with Directed Momentum

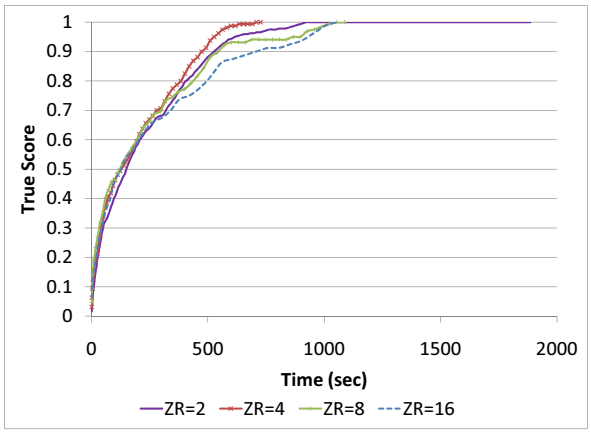


Figure 9: Variation of ZR with Flow Momentum

gate such a relation of the ZR using the realistic BURAQ measurement system.

Figure-8 shows the latencies in isolating the five anomalous heavy flows with increasing ZRs using the DM algorithm. The results, to our surprise, indicate that on a practical measurement system, the ZR actually demonstrate an inverse relationship with the detection latencies. A closer inspection reveals some interesting characteristics of the streaming algorithms. We note that the higher ZRs translate into an increased zoom granularity of the tracked rules (sub-regions). However, generally a network is mostly anomalous free and as such, the increased amounts of inspections with the higher ZRs translate into tracking sub-regions that do not contribute in isolation of the anomalies. In practical terms, this means allocation of higher number of resources to non-interesting rules/regions, and thereby lowering the overall detection latency. Thus although the higher ZRs reduce the number of iterations (not shown), in practice such a reduction in the algorithmic length may not translate into a better timing closure in isolating anomalies.

The results also reveal another interesting observation: that with higher ZRs, the DM exhibits difficulty in preserving a continuous true-score progression. This can be observed in the brief plateaus and dips in the TS. The reason being the nature of algorithm, where it aggressively tries to offset the rapid increase in the rule-set size due to the higher ZRs. Since the algorithm is designed to favor delving down the hierarchy, the algorithmic *pull* to counter the increase falls on the old nodes, during which some of good nodes also end up being discarded by the algorithm, resulting in the loss of the *momentum*, or TS. However, such a problem can be quickly resolved by adjustments in the algorithmic Stretch parameter, giving increased leniency with increasing ZRs.

One might argue that the characteristics of the DM algorithm could have played a stronger role in the inverse relationship exhibited by the ZRs with the detection latency. We further investigate this claim by experimenting using the FM algorithm that does not have any algorithmic *pulls* associated with the rule-set size. The latencies so obtained are plotted in Figure-9. The plots show that the inverse relationship generally holds true, with a singular exception while increasing ZR from 2 to 4. The reason for the special case lies in a combination of several factors, most notably in the relatively smaller overheads in the rule-set size while going to ZR 4. We therefore argue that in a practical system, one needs to keep the ZRs small by taking into account the available computational resources.

6.3 Latency Analysis

We also discuss the various latencies involved in the measurement framework. A pie-chart depicting the distribution of various latencies on the BURAQ framework is shown in Figure-10. Though the chart employs DM algorithm, the presented distribution is observed to be quite consistent across the proposed streaming solutions.

It can be seen that the major chunk of the latencies is the communication latency, which is an aggregation of the delays associated with transferring the sockets' statistics and (re)programming bit-vectors across the Ethernet. The actual latency involving collecting the rules' statistics, the statistics collection latency, comes up at the second spot. The third major chunk, the rule deployment latency, is made out of delays involved in (re)programming of the sockets for deployment of new rules. Finally, the actual delays involving the rules' evaluation and synthesis at the algorithmic layers is seen to be taking up only 1% in the total latency distribution.

As discussed earlier, the BURAQ framework employs fine grained PDR for (re)programming the LUTs. The novel PDR based mechanism does away with slow just-in-time compilation of FPGA programming data. Instead, by performing minute changes directly in the FPGA's configuration memory corresponding to the LUTs being (re)programmed, the mechanism takes only takes up $90\mu s$ for (re)programming a single LUT, or equivalently $1.8ms$ for the complete socket composed of 20 (re)programmable LUTs [10]. This latency is seen to be contributing 15% in the total simulation delays.

The above latencies can be sub-divided into algorithmic and platform-specific latencies. In particular, the rule synthesis and statistics collection are functions of the algorithm whereas the other two are by-products of the platform. The algorithmic latencies are interesting to us as they relate to the actual computation cost of the proposed streaming solutions.

For our simulations, we chose a statistics collection window size of $1s$ per iteration. The choice has been based on

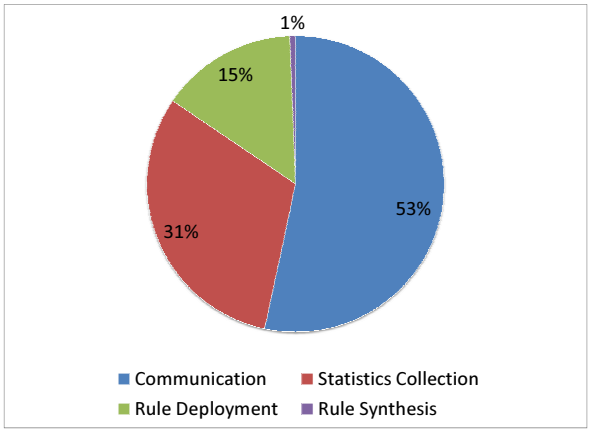


Figure 10: Latency Distribution

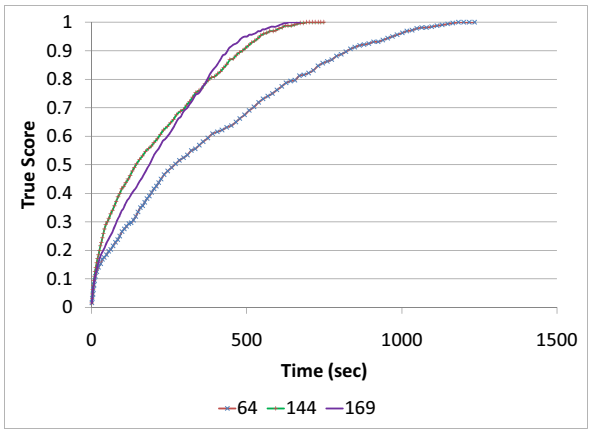


Figure 11: Algorithmic Convergence & Computational Resources

the amount of data we expected to gather during the collection window. A rule of thumb is that the collection window should be wide enough for capturing enough statistics. In other words, the collection latency is an inverse function of line-speed, and as such can be re-adjusted (and possibly reduced) depending on the framework's deployment.

The above discussion leads us in identifying the base cost of our streaming algorithms, the rule analysis and synthesis latency. The base cost relates the discussed algorithmic complexities in empirical figures and compares them with the overall latencies of the BURAQ's closed-loop measurement framework. It can be noticed that the base cost of the presented streaming solutions is quite minimal. In practical terms, this implies that there will be fewer packets that will miss inspection during rule evaluation and synthesis, thereby increasing the confidence in reported results.

6.4 Resource Constraints

We finally explore the effects of different amount of computing resources on the streaming solution incorporating the blocking controller as discussed in Section-5. The results are however orthogonal to the relative efficiencies of the streaming algorithms and are only presented to highlight the interplay of blocking implementation on the overall system performance.

We use DM algorithm on BURAQ system incorporating different amounts of rule-processing sockets. The results so obtained are shown in Figure-11. As expected, higher computational resources yield better timing closure to the anomalies, as they tend to reduce the need for rule-pipelining. However, the returns are seen to be un-proportional to the increase in computational budget. This is because of the blocking implementation of the controller that trades-off controller complexity with an underutilization of computational resources in certain algorithmic iterations as discussed earlier. The results thus demonstrate the effects of the trade-off over the performance of the streaming solutions.

7. CONCLUSION

The work addressed the challenges in closed-loop streaming measurements and analysis in today's high speed and complex networks. In particular, we addressed the issues in tracking down anomalous flows within a desired accuracy and latency, in a given resource budget. The work presented three streaming algorithms and integrated them with BURAQ measurement platform. We discussed their relative efficiencies with a recently proposed MRT algorithm. Our algorithms showed increased effectiveness, with two of the algorithms demonstrating a marked 100% accuracy increase from the MRT in isolating heavy-flows. We also presented mathematical bounds to the system accuracy while integrating the FM and DM algorithms in a practical measurement system.

The proposed algorithms offer parameterizable solutions in the wide spectrum of design choices involving available computing and storage resources, detection latencies, accuracy, and user's knowledge of the desired anomaly. The proposed ER and FM algorithms are suitable where detailed information about the anomalous behavior and computational platform is not available. The algorithms trade computational complexity with storage costs and detection latencies, with FM offering increased accuracy. The DM algorithm showcases superior accuracy and detection latency while employing minimal resources, making it an ideal candidate in scenarios where exceeding the available resources is prohibitive and/or increased characterization of an anomaly is possible.

8. REFERENCES

[1] Cisco NetFlow. ``http://www.cisco.com/warp/public/732/Tech/netflow''.

[2] CAIDA: Cooperative Association for Internet Data Analysis. http://www.caida.org/data/passive/backscatter_tocs_dataset.xml.

[3] N. Brownlee, C. Mills, and G. Ruth. Traffic Flow Measurement: Architecture. RFC 2722, 1999. http://www.ietf.org/rfc/rfc2722.txt.

[4] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Diamond in the rough: finding hierarchical heavy hitters in multi-dimensional data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 155–166, New York, NY, USA, 2004. ACM.

[5] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55:58–75, April 2005.

[6] N. G. Duffield. Sampling for passive internet measurement: A review. *Statistical Science*, 19(3), 2004.

[7] C. Estan and G. Varghese. New Directions in Traffic Measurement and Accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21(3):270–313, 2003.

[8] L. Jose, M. Yu, and J. Rexford. Online measurement of large traffic aggregates on commodity switches.

[9] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: characterization and implications for cdns and web sites. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 293–304, New York, NY, USA, 2002. ACM.

[10] F. Khan, N. Hosein, S. Vernon, and S. Ghiasi. BURAQ: A dynamically reconfigurable system for stateful measurement of network traffic. *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, 0:185–192, 2010.

[11] F. Khan, L. Yuan, C.-N. Chuah, and S. Ghiasi. A Programmable Architecture for Scalable and Real-time Network Traffic Measurements. In *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 109–118, 2008.

[12] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 165–176, New York, NY, USA, 2006. ACM.

[13] A. Ramachandran, S. Seetharaman, and N. Feamster. Fast monitoring for traffic subpopulations. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 257–270, 2008.

[14] L. Yuan, C.-N. Chuah, and P. Mohapatra. ProgME: towards programmable network measurement. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 97–108, 2007.

[15] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *IMC*, 2004.

APPENDIX

A. ANALYTICAL ANALYSIS

The closed-loop measurement system utilizes snapshots of the streaming data in coming up with an answer to the user-query. In a realistic system, these snapshots may differ with the actual traffic due to measurement inaccuracies, incurring inaccuracies in the final answer. Moreover, the iterative closed-loop measurement paradigm itself induces inaccuracies as any local decision taken in any given iterative step effects the subsequent search process and consequentially the system's answer to the user-query. We hereby model these inaccuracies and discuss the upper bounds in obtaining false alarms when given any network condition and system resource budget while using FM and DM algorithms. Let,

f_i represents flowset i.

$|f_i^t|$ and $|F_i^t|$ represent the measured and actual size for the flowset i in an iteration t.

δ: is sampling time during which incoming stream can be observed

ϵ: is configuration time during which incoming stream cannot be observed.

θ: represents threshold for elephant flows defined in terms of link usage.

N: represents the number of rule-processing resources.

$|R^n|$: represents the size of the rule-set in any given algorithmic iteration n.

λ: represents the link bandwidth (bits/second).

λ_i^n: represents the arrival rate for the flowset f_i in iteration n (bits/second). Thus,

$$\lambda = \lambda_i^1 \leq \lambda_i^2 \leq \lambda_i^n$$

α_i^n and $E[\alpha_i^n]$: the observed and expected momentum for flowset f_i in an algorithmic iteration n under limited observation window.

$$\begin{aligned} \alpha_i^n &= \sum_{j=1}^{n} |f_i^j|/(n.d_i^n) \\ E[\alpha_i^n] &= \frac{\sum_{j=1}^{n} \lambda_i^j \delta}{n.d_i^n} = \mu_{\alpha_i^n} \end{aligned}$$

where d_i^n represents the directed momentum (Dir-Mom) for f_i in iteration n and is given as

$$d_i^n = \begin{cases} \exp^{Pull * f_i.Static/f_i.Granularity} & \text{for Directed Momentum} \\ 1 & \text{for Flow Momentum} \end{cases}$$

β_i^n and $E[\beta_i^n]$: the actual and expected momentum for flowset f_i in an algorithmic iteration n under ideal measurement conditions.

$$\begin{aligned} \beta_i^n &= \sum_{j=1}^{n} \left\lceil \frac{N}{|R^j|} \right\rceil |F_i^j|/n \\ E[\beta_i^n] &= \frac{\sum_{j=1}^{n} \left\lceil \frac{N}{|R^j|} \right\rceil \lambda_i^j (\delta + \epsilon)}{n} = \mu_{\beta_i^n} \end{aligned}$$

Note that the multiplicative term $\left\lceil \frac{N}{|R^j|} \right\rceil$ accounts for rule pipelining on limited resources in any given algorithmic iteration j.

τ_n represents the threshold for the observed momentum to be classified as a heavy-flow in an iteration n

$$\tau_n = \theta * \lambda * \delta$$

We define two constants as follows

$$\begin{aligned} c_{\alpha_i^n} &= \frac{\tau_n}{\mu_{\alpha_i^n}} = \frac{n.d_i^n \theta \lambda}{\sum_{j=1}^{n} \lambda_i^j} \leq 1 \\ c_{\beta_i^n} &= \frac{\tau_n}{\mu_{\beta_i^n}} = \frac{n\delta\theta\lambda}{\sum_{j=1}^{n} \left\lceil \frac{N}{|R^j|} \right\rceil \lambda_i^j (\delta + \epsilon)} \leq 1 \end{aligned}$$

We assume the incoming stream as having Poisson distribution.

A.1 False Negative

A false negative is the probability of an incorrect absence of an alarm in an iteration, that is, it is the probability that the absence of an alarm in an algorithmic iteration t is incorrect given that there is no alarm in iteration t.

A false negative can occur in an iteration t when if f_i that can be classified as a heavy flow in iteration t is dropped at an earlier iteration j. This could happen

- when f_i falls below the threshold in iteration j due to measurement inaccuracies but it is heavy in iteration t. The probability of this happening is $P(\alpha_i^j < \tau)P(\beta_i^j \geq \tau)P(\beta_i^t \geq \tau)$.
- when f_i actually falls below the threshold in iteration j and it is heavy in iteration t. The probability of for this can be given as $P(\alpha_i^j < \tau)P(\beta_i^j < \tau)P(\beta_i^t \geq \tau)$.

The latter implies that there is a resurgence in the flowset activity such that it becomes heavy in iteration t.

If P_n^t represents probability of false negative in iteration t, then combining the above we have

$$\begin{aligned} P_n^t &= \frac{\sum_{j=1}^{t} P(\alpha_i^j < \tau)P(\beta_i^t \geq \tau)}{P(\alpha_i^t < \tau)} \\ &= \frac{\sum_{j=1}^{t} P(\alpha_i^j < c_{\alpha_i^j}\mu_{\alpha_i^j})[1 - P(\beta_i^t < c_{\beta_i^t}\mu_{\beta_i^t})]}{P(\alpha_i^t < c_{\alpha_i^t}\mu_{\alpha_i^t})} \end{aligned}$$

Applying Chernoff bound, we have

$$\begin{aligned} P_n^t < \sum_{j=1}^{t} [e^{-(\mu_{\alpha_i^j} - \tau)^2/2\mu_{\alpha_i^j} + (\mu_{\alpha_i^t} - \tau)^2/2\mu_{\alpha_i^t}}] * \\ [1 - e^{-(\mu_{\beta_i^t} - \tau)^2/2\mu_{\beta_i^t}}] \end{aligned}$$

Let

$$k_j = \mu_{\alpha_i^j}/\mu_{\alpha_i^t} \geq 1$$

Then

$$\begin{aligned} P_n^t < \sum_{j=1}^{t} [e^{-\frac{1}{2\mu_{\alpha_i^t}}\{(\mu_{\alpha_i^j} - \tau)^2 + k_j(\mu_{\alpha_i^t} - \tau)^2}] * \\ [1 - e^{-(\mu_{\beta_i^t} - \tau)^2/2\mu_{\beta_i^t}}] \end{aligned}$$

It is to be noted that $\mu_{\alpha_i^j} \geq \mu_{\alpha_i^t} \geq \tau$ for $j \geq t$, as well as $\mu_{\beta_i^t} \geq \tau$. The above inequality shows that probability for false negative increases with increase in parameters that uniquely influence threshold and configuration times (θ, λ and ϵ), while it has an inverse relationship with parameters that dictate the amount of data that can be observed in an algorithmic iteration (λ_i^n and δ).

A.2 False Positive

A false positive is the probability of an incorrect presence of an alarm, that is, it is the probability that an alarm in an algorithmic iteration t is incorrect, given that there is an alarm in iteration t.

For an alarm to happen in iteration t, f_i has to consistently meet the threshold requirements in all iterations $j \leq t$. The alarm will be an incorrect alarm if $\beta_i^t < \tau$ but $\alpha_i^t \geq \tau$.

If P_p^t represents the probability of false positive in iteration t, then using the above discussion we have

$$\begin{aligned} P_p^t &= \frac{\prod_{j=1}^{t} P(\alpha_i^j \geq \tau)P(\beta_i^t < \tau)}{P(\alpha_i^t \geq \tau)} \\ &= \frac{\prod_{j=1}^{t-1} P(\alpha_i^j \geq \tau)P(\alpha_i^t \geq \tau)P(\beta_i^t < \tau)}{P(\alpha_i^t \geq \tau)} \\ &= \prod_{j=1}^{t-1} P(\alpha_i^j \geq \tau)P(\beta_i^t < \tau) \\ &= \prod_{j=1}^{t-1} [1 - P(\alpha_i^j < c_{\alpha_i^j}\mu_{\alpha_i^j})]P(\beta_i^t < c_{\beta_i^t}\mu_{\beta_i^t}) \end{aligned}$$

Applying Chernoff Bound, we have

$$P_p^t < \prod_{j=1}^{t-1} [1 - e^{-(\mu_{\alpha_i^j} - \tau)^2/2\mu_{\alpha_i^t}}][e^{-(\mu_{\beta_i^t} - \tau)^2/2\mu_{\beta_i^t}}]$$

The above inequality shows that the probability for false positive can be positively influenced by increasing parameters that increase the size of the observed sample (λ_i^n, δ), while it gets negatively influenced by increasing parameters that uniquely effect threshold and configuration times (θ, λ and ϵ). The observation is similar to that for false negative, although the influence of the various parameters on false positive scales differently that can be bounded using the above expression.