Title
Automatically Inferring Patterns of Resource Consumption in Network

Authors
Estan, Cristian
Savage, Stefan
Varghese, George

Publication Date
2003-06-02

Peer reviewed

Automatically Inferring Patterns of Resource Consumption in Network Traffic

Cristian Estan, Stefan Savage, George Varghese
Computer Science and Engineering Department
University of California San Diego
cestan,savage,varghese@cs.ucsd.edu

ABSTRACT

The Internet service model emphasizes flexibility – any node can send any type of traffic at any time. While this design has allowed new applications and usage models to flourish, it also makes the job of network management significantly more challenging. This paper describes a new method of traffic characterization that automatically groups traffic into minimal clusters of conspicuous consumption. Rather than providing a static analysis specialized to capture flows, applications, or network-to-network traffic matrices, our approach dynamically produces hybrid traffic definitions that match the underlying usage. For example, rather than report five hundred small flows, or the amount of TCP traffic to port 80, or the "top ten hosts", our method might reveal that a certain percent of traffic was used by TCP connections between AOL clients and a particular group of Web servers. Similarly, our technique can be used to automatically classify new traffic patterns, such as network worms or peer-to-peer applications, without knowing the structure of such traffic a priori. We describe a series of algorithms for constructing these traffic clusters, minimizing their representation and the design of our prototype system, AutoFocus. In addition, we describe our experiences using AutoFocus to discover the dominant and unusual modes of usage on several different production networks.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*

Keywords

Traffic measurement, Network monitoring, Data mining

1. INTRODUCTION

The Internet is a moving target. Flash crowds, streaming media, CDNs, DoS attacks, network worms, peer-to-peer – these are but a few of the forces that shape traffic on today's networks. Each year, new applications and usage models emerge, and from these arise new communications patterns. This flexibility is a hallmark of the Internet architecture and can be credited with much of the Internet's success. At the same time, this quality also brings serious challenges for network management. Unlike the traditional voice networks, which are built around a single high-level abstraction for application data transfer ("calls"), managers of IP-based networks are forced to *infer* the type of traffic and how it relates to applications and users. Consequently, to understand and react to changes in network usage, a network manager must first analyze the bit patterns in individual packets, extract an appropriate traffic model and then reconfigure network elements to recognize that model appropriately.

To make this process feasible in practice, managers use a standard set of pre-defined patterns to identify well-known aspects of network traffic. For example, network managers frequently construct a model of application usage by classifying traffic according to the IP header fields: Protocol and SrcPort. Such an analysis might determine that 90 percent of traffic uses the TCP protocol, 75 percent of TCP traffic is for the HTTP service, 10 percent is for SMTP, 5 percent for FTP and so on. Similarly, to identify individual conversations between pairs of hosts, the five tuple (SrcIP, DstIP, Protocol, SrcPort, DstPort), is used to impart a "flow" abstraction on traffic. These kinds of analyses, exemplified by popular monitoring tools such as FlowScan, and Cisco's FlowAnalyzer, are a staple of modern network management [13]. However, they have two significant limitations when used in practice: *insufficient dimensionality* and *excessive detail*.

While network traffic may be characterized by many different criteria, it is easiest to aggregate traffic along one dimension at a time. Unfortunately, by aggregating traffic along any single dimension, the network manager inevitably hides any interesting, but orthogonal, structure. For example, by aggregating traffic according to an application-oriented view (i.e. Protocol and SrcPort), a network manager might conclude that peer-to-peer file sharing applications are in wide use, when in fact a small set of hosts are responsible for most of the file sharing traffic [14]. While the network manager can expose this structure by using finer-grained representations, such as flows, she then must manage the excessive detail contained in such a representation. Rather than identifying the file-sharing traffic concisely, the flow-oriented view decomposes it into thousands of individual network transfers.

Consequently, network managers spend considerable time manually "hunting for needles" in their data – trying to understand what are the real and significant sources of traffic in their network and which components of usage are changing over time. This problem is only exacerbated when there is a pressing need to understand and respond to sudden traffic spikes such as network worms or denial-of-service attacks.

The focus of our work is to help automate these tasks and

this paper makes four contributions in this direction. First, in Section 2 we motivate the need for a new approach to traffic monitoring that can automatically classify traffic into appropriate multi-dimensional clusters. We define a concrete and practical instance of such traffic clusters and describe a set of operations for reducing their size and increasing their utility. Based on this definition, Section 3 describes and evaluates algorithms for cluster construction and for implementing the key operations described in Section 2. Finally, in sections Section 4 and Section 5 we describe AutoFocus, our prototype system, and describe our experience using it on several production networks. At the end of our paper, we relate our efforts to previous work and then summarize our results.

2. MULTI-DIMENSIONAL TRAFFIC CLUSTERS

While the goal of every traffic analysis method is to empower the human operator with improved understanding, there is an inherent contradiction between the level of detail provided and the capacity of humans to absorb information: more detail can lead to a deeper understanding but makes the report harder to read – at one extreme is a bandwidth meter, at the other extreme are raw packet traces. The simplest solution is to report only the largest flows, so-called "top ten" reports, but this approach has a serious flaw: aggregates made up of many small flows can be important, but each individual flow may not be large. For example, a busy Web server might generate the bulk of the traffic, but since all of its flows are relatively small, the "top ten" report might only contain transfers from a nearby FTP server hosting a few large files.

Another solution is to aggregate the individual flows into a common category (e.g. by source or destination port, source or destination IP, prefix or Autonomous System number). However, if we chose the wrong dimension to aggregate over then we may miss the interesting characteristics of the traffic. For example, if we aggregate traffic by port number, we may miss the importance of traffic generated by a denial-of-service attack using random port numbers. In this case, aggregating traffic by destination address would likely be a more useful approach.

However, in some cases there is significant information that is hidden by aggregating on any *single* field, but is revealed by aggregating according to a combination of fields (multi-dimensional aggregates). For example, aggregating traffic by IP address might identify a set of popular servers and aggregating traffic by port might identify popular applications, but to identify which server generates which kind of traffic requires aggregating according to two fields simultaneously.

Our way out of this impasse is to focus on dynamically-defined traffic clusters instead of individual flows or other predefined aggregates. Our aim is to define the clusters so that any meaningful aggregate of individual flows is a traffic cluster. For example, a single cluster might represent all TCP client traffic originating from America Online's network destined for a cluster of replicated Web servers on Google's network. Another cluster might represent significant amounts of traffic originating from a host infected by the Sapphire worm destined to random addresses at UDP port 1434. While the goal is clearly attractive, automatically building such clusters is quite challenging. In practice, creating effective clusters requires balancing three key requirements:

- **Dimensionality**. The dimensionality of the problem is defined by how many distinct properties are considered in constructing a traffic cluster. If there is too little dimensionality then important traffic categories can be masked, while if there is too much, the computational overhead of computing cluster combinations can become infeasible.

- **Detail**. While multi-dimensional clusters allow us to capture the structure of the traffic being analyzed, this does nothing to reduce the magnitude of data that must be evaluated – one can easily create thousands of clusters from a traffic trace. To make such data useful, a clustering algorithm must carefully prune this set to remove "unimportant" clusters and tradeoff the loss in detail for corresponding gains in conciseness.

- **Utility**. In the end, network managers are not merely passive observers of traffic, but are active parties who attempt to control and react to changes in traffic load and usage. Therefore, while we could construct traffic clusters using arbitrary combinations of packet header bit patterns, it is far more useful to restrict our choices to header fields that are already well-classified by network hardware and can therefore be acted upon.

2.1 Defining Traffic Clusters

Based on these principles, we define our traffic clusters in terms of the five fields typically used to define a fine-grained flow: source IP address, destination IP address, protocol, source port and destination port. Unlike individual flows defined by unique values for each of these fields, clusters are defined by sets of values for each of these fields. These sets can contain a single value, all possible values (we use * to denote this case) or restricted subsets of possible values.

Evaluating all possible subsets of the values for each field would have made the problem of finding all large clusters unnecessarily difficult. Instead, we use instead the natural hierarchies that exist for each field. For IP addresses a cluster can be defined by prefixes of length from 8 to 32 (for individual IP addresses) or (*) for all IP addresses. For port numbers, we define clusters through a particular port number (e.g. port 80), through the set of high port numbers above 1023 (usually ephemeral ports allocated on-demand to clients), low port numbers below 1024 (usually well-known ports statically allocated to services), or the set of all possible values (*). Finally, the protocol field can take on exact values or (*).

For example, the cluster defined as (SrcIP=10.8.200.3, DstIP=*, Protocol=TCP, SrcPort=80, DstPort=*) represents Web traffic from the server with address 10.8.200.3. (SrcIP=*, DstIP=172.27.0.0/16, Protocol=TCP, SrcPort=low, DstPort=high) represents TCP traffic coming from low ports and going to high ports destined to a certain prefix. Finally, (SrcIP=*, DstIP=*, Protocol=ICMP, SrcPort=*, DstPort=*) represents all ICMP traffic. Notice that the first two clusters overlap with each other while the third cluster is unique.

There are many ways to further generalize the definition of traffic clusters. For example, we could accept sets of IP addresses defined by Autonomous System numbers, classify

traffic based on integer ranges of port numbers, or accept external input to group non-contiguous address ranges (e.g. Universities, Broadband Access, Data Centers, etc.) We could also employ heuristics such as those for identifying passive FTP and Napster traffic (that use random ports) in [13]. While these additions may provide greater value in some settings, they do not require any fundamental changes in our approach, merely a different set of aggregation criteria when constructing clusters. For the remainder of this paper we restrict our discussion to the "vanilla" cluster definitions we have described previously. There are three advantages to this cluster definition. First, our definition is sufficiently general to capture much of the usage structure in existing applications and networks. Second, our definition is consistent with current packet classifiers [7] and consequently a manager can apply controls, such as policy routing and hardware rate limiting, to the clusters we dynamically identify. Third, our definition allows a simple visually appealing rule-based display of clusters (Section 4). Finally, initial results on several distinct networks (Section 5) indicate that clusters defined in this way do identify interesting resource consumption patterns that managers care about.

Our definition of clusters satisfies two of the requirements for clusters outlined earlier: the need for multi-dimensional clusters (dimensionality) and field selection constrained by existing field hierarchies (utility). However, satisfying the remaining requirement, reducing detail, requires significant additional effort.

2.2 Operations on Traffic Clusters

A *traffic report* is a list of clusters presented to a manager. It is very easy to see that even restricting ourselves to IP prefixes and very simple port ranges, that there is an exponential number of raw clusters. There are approximately 2^{33} possible source IP prefixes alone! The first step in reducing this onslaught of data is to restrict the report to only include *high volume* clusters, where volume may be defined as the number of bytes or the number of packets contained in the cluster over a predefined measurement interval. While other criteria could be used to filter clusters, data volume is a categorization of inherent interest. A cluster containing 20 percent of all traffic is one that a network manager is likely to care about, while a cluster that only contains a few packets usually warrants less attention.

However, even with this restriction, it turns out that the number of such clusters identified in real traces is far too large to manage. Keep in mind that the precious resource we are optimizing for is not network bandwidth or CPU cycles, but a network manager's time; verbose and unstructured reports are not likely to be appreciated or useful. Consequently, to maximize the effectiveness of a traffic report, we believe that there are four essential operations that must be provided:

- **Operation 1, Compute:** Given a description of traffic as input (e.g., packet traces or NetFlow records), compute the identity of all clusters with a traffic volume above a certain threshold. This is the base operation.
- **Operation 2, Compress:** Having found the base set of clusters, one can compress the report considerably by removing a cluster C from the report if cluster C's traffic can be inferred (within some error tolerance) from that of cluster C' that is already in the report. For example, if all the traffic is generated by a single high-volume connection from source S to destination D is high volume, then one can infer that the traffic sent by S is also high volume. Thus one should retain the S to D cluster for the detail it shows, and omit the S cluster as it can be inferred from the S to D cluster. Intuitively, the rule we use is to remove a more general cluster if its traffic volume can be inferred (within some error tolerance) from more specific clusters included in the report.
- **Operation 3, Compare:** A good way to save the manager time is to concisely show how the traffic mix changes (deltas) from day to day, or week to week. This can be done by only displaying the high volume clusters that have changed from those listed in the last report. This is trickier than it seems, because a high volume Cluster on Day 1 may now become low on Day 2, or vice versa. Worse, more general clusters need not be larger than the sum of more specific non overlapping clusters. Thus combining Deltas (Operation 3) with Compression (Operation 2) is much harder than just implementing each operation in isolation.
- **Operation 4, Prioritize:** Even after compressing the report and computing Deltas, it is still desirable to prioritize the elements of the report in terms of their potential level of interest to a manager. We choose to equate the interest in a cluster to what we call its *unexpectedness*. While there are many ways to define this metric, we chose to use a relatively unsophisticated approach that is easy to compute. We define unexpectedness in terms of deviation from a uniform model in which the contents of different fields is mutually independent. For example, if prefix A sends 25% of the traffic and prefix B receives 40% of all traffic, then under the assumption of independence, we would expect the traffic from A to B to be 25%*40%=10% of the total traffic. If the actual traffic from A to B is 15% of the total traffic instead of 10%, the cluster is tagged with a score of 150%, indicating that it is unexpectedly large by a factor of 1.5. If the traffic from A to B is only 6%, then it is given a score of 60%, indicating that it is unexpectedly small. The closer a score is to 100%, the more boring it is, and the less important it is to highlight to the user. This construction of unexpectedness is, in effect, a very simple multi-dimensional gravity model.

3. ALGORITHMS

The last section motivated four fairly abstract operations on sets of clusters. In this section, we describe the specific algorithms we chose to *implement* these operations. These algorithms form the engine that underlies the core of our AutoFocus tool described in Section 4. Rather than directly present the algorithms for the multi-dimensional case, we first present the simpler algorithms for the unidimensional (i.e., single field) case. While AutoFocus implements the more general multidimensional case, the simpler unidimensional results are also included in the output. Addressing this simple case will also help build intuition. Furthermore, the multidimensional algorithms use the unidimensional algorithms to reduce their search space.

For many of the algorithms we also present theoretical upper bounds on the size of the report after performing the operation, and algorithm running time. Our practical results in Appendix A show much smaller reports than these upper bounds. Finally, our focus in this paper is maximizing information transfer to the manager, not algorithmic optimization; we believe that significantly faster algorithms that produce similar results may be possible.

Notation: In this section we use the terms dimension and field interchangeably since each field defines a dimension along which we can classify. We use k for the number of fields. In the actual system we implemented $k = 5$.

The sets (i.e., prefixes in IP address fields) for each field form a natural hierarchy in terms of set inclusion. This can be described by a tree where the parent is always the smallest superset of the child. The leaves of this tree are individual values the field can take. The root is always the set of all possible values, *. The sets denoted by two nodes are disjoint unless one of the nodes is an ancestor of the other case in which it is a superset of the other. We call the number of levels in this tree the depth of the tree (the maximum distance from the root to a leaf plus 1).

We use d_i for the depth of the hierarchy of the i-th of the k fields. The hierarchy for IP addresses we use in this paper has a depth of 26 (prefix lengths from 8 to 32 plus prefix length 0). We use a hierarchy of depth 3 for port numbers (individual port numbers, * and the two sets denoting high ports [1024,65535] and low ports [0,1023])[1] and for protocols we use the simplest possible hierarchy with a depth of d=2.

The raw data we build our algorithms on is a simplified version of NetFlow flow records: each flow record, which we sometimes refer to as a "flow" for conciseness, has a key that specifies exact values for all five fields and two counters, one counting the packets that matched the key during the measurement interval considered and one for the number of bytes in those packets. Transforming a trace with packet headers and timestamps into such flow records leads to no loss of information from the standpoint of traffic clusters.

We use n for the number of such records. Each traffic cluster is made up of one or more flow records and the corresponding byte and packet counters are the sum of the corresponding counters of the flow records it includes. Note that if a cluster contains exact values in all fields it is exactly equivalent to a single flow. For the rest of this section we ignore that the flow records contain two counters and work with a single counter. Our algorithms use a threshold H and focus on the traffic clusters that are above this threshold. We use s for the ratio between the total traffic T and the threshold $s = T/H$, so if H is 5% of the total traffic, then $s = 20$.

3.1 Unidimensional case

First we concentrate on the problem of computing high volume clusters on a single field, such as the source IP address. Note that even the unidimensional case is significantly more complex than traditional tools like FlowScan, in which managers define a static hierarchy by pre-specifying which subnets should be watched. By clustering automatically, we do not need to define subnets; the tool will automatically group addresses into "subnets" that contain a high volume of traffic. Let d denote the depth of the hierarchy in the field considered (e.g., 26 for source IP field). We also use $m \leq n$ for the number of distinct values for the field we consider among the n flow records in the input.

[1]For protocols that don't use ports, such as ICMP, packets/flows match * and nothing else.

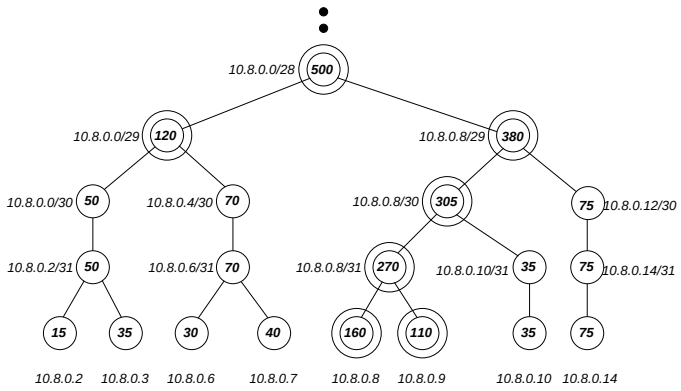


Figure 1: Each individual IP address sending traffic appears as a leaf. The traffic of an internal node is the sum of the traffic of its children. Nodes whose traffic is above H=100 (double circles) are the high volume traffic clusters. The Web server (10.8.0.12) is a large cluster in itself. While no individual DHCP address is large enough, their aggregate is, so 10.8.0.0/29 is listed as a large cluster.

3.1.1 Computing Unidimensional Clusters

Before we describe our algorithms for computing the high volume unidimensional clusters, it is useful to bound their number. Consider the IP source address. Each valid prefix corresponds to a valid one dimensional cluster in the source IP field. Thus, there are a huge ($\sim 2^{33}$) number of possible one-dimensional clusters. To reduce the report, we may filter the clusters to only include those whose traffic is over 5% of the total traffic.

However, implementing this filter is more complicated than it seems at first. A reasonable intuition might be that $s = 20$, because there can be at most 20 *disjoint* clusters, each sending 5% of the total. Unfortunately, our definition of clusters allows clusters to overlap. Thus, if $128.*.*$ is a high volume cluster, then so is $128.*.*.*$. Fortunately, a given source address cluster's traffic can at most be counted in 25 other clusters (the number of ancestors in its hierarchy tree – we do not consider prefixes with lengths from 1 to 7). Therefore, the maximum number of high volume clusters is not 20 but roughly $20 * 26 = 520$. This is formalized by:

LEMMA 1. *The number of clusters above the threshold is at most* $1 + (d-1)s$.

Proof The counter of each flow record can contribute to at most $d-1$ sets besides *. The sum of the counters of all flows is T. The sum of the sizes of all clusters other than * is at most $(d-1)T$. Therefore, there are at most $(d-1)T/H = (d-1)s$ clusters above H, and once we also add * we arrive at the final result. ■

For the unidimensional case, we now describe the algorithm to do **Operation 1**, computing the raw set of high volume clusters. When the number of sets in the hierarchy is relatively small, for example 257 for protocol and 65539

for port numbers, we can apply a brute force approach: keep a counter for each set and traverse all n flow records while updating all relevant counters; at the end, list the clusters whose counters have exceeded H.

If the number of possible values is much larger, as is the case for IP addresses, we use another algorithm illustrated by the example from Figure 1. As we go through the flow records, we first build the leaf nodes that correspond to the IP addresses that actually appear in the trace. For example, there are only 8 possible source addresses (leaves) in the trace that Figure 1 was built from. Thus, we make a pass over the trace updating the counters of all the leaf nodes. By the end of this pass, the leaf counters are correct; we also initialize the counters of all nodes between these leaves and the root to 0.

In a second pass over this tree, we now can determine which clusters are above threshold H: we only need to do the traversal post order (children before parents). Also, just before finishing with each node, the algorithm must add its traffic to the traffic of its parent. This way, when the algorithm gets to each node its counter will reflect its actual traffic.

The memory requirement for this algorithm is $O(1+m(d-1))$ and it can be reduced to $O(m+d)$ by generating the internal nodes only as we traverse the tree. The running time of the algorithm is $O(n+1+m(d-1))$. No algorithm can execute faster than $O(n+1+(d-1)s)$ because all algorithms need to at least read in the input and print out the result.

3.1.2 Compressing Unidimensional Traffic Reports

For the unidimensional case, we now describe the algorithm to do **Operation 2**, compressing the raw set of high volume clusters. The complete list of all clusters above the threshold is too large and most often it contains obvious information. Even if a /8 (prefix of length 8) contains exactly the same amount of traffic as a /24 all the prefix lengths in between are listed. This is obvious waste of space in the report.

More generally, perhaps an intermediate prefix length like /16 has a little more traffic than the /24 it includes but not a whole lot. Then we can argue that including the /16 adds little marginal value but takes up precious space in the report. Removing the /16 on the other hand, will mean that the manager's estimate of the /16 may be a little off. Thus we trade accuracy for reduced size.

In general, define the compression threshold C as the amount by which a cluster can be off. In our experiments we defined $C = H$. We did so to avoid unintended errors: if the manager wants all clusters above H, surely she realizes that the report can be off by H in terms of missing clusters of size smaller than H. By setting $C = H$, we are only adding another way to be off by H. Also, setting $C = H$ produces the following simple but appealing result.

LEMMA 2. *The number of clusters above the threshold in the* compressed report *is at most s.*

Proof Each cluster in the report has a traffic of at least H that was not reported by any of its descendants. The sum of these differences is at most T because each flow can be associated with at most one most-specific cluster in the report and these flows make up the difference between that cluster and the more-specific ones. Therefore, a report can contain at most $T/H = s$ clusters. ■

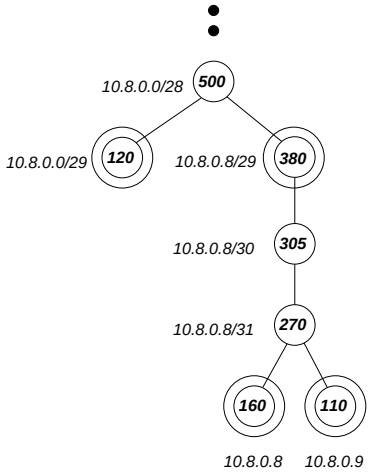


Figure 2: The clusters from the compressed report are represented with double circles. Node 10.8.0.8/31 is not in the compressed report because its traffic is exactly the sum of the traffic of its children. Node 10.8.0.8/30 is not in the compressed report because its traffic is within a small amount (35) of as what we can compute based on its two descendants in the report. Compression reduces the number of double circled nodes in this tree (the clusters included in the report) from 7 in Figure 1 to 4.

If we go back to our original example for computing all clusters that send over 5% of the total traffic, we find that the number of clusters in the compressed cluster (assuming $C = H = 5\%$) is at most 20 and not roughly $20 * 26$ predicted by Lemma 1. The compressed report corresponding to Figure 1 is shown in Figure 2. Note that the number of nodes retained in the report (nodes with double circles) has dropped from 7 to 4 which is actually less than the 5 Lemma 2 would have predicted for the 20% threshold.

Our algorithm for compressing the report (Figure 2) relies on a single traversal of the tree of the high volume traffic clusters. Each node in the tree gets two counters: one reflecting its traffic and one reflecting an estimate of its traffic based on the more specific clusters included in the report. We perform a post order traversal and decide for each node whether it goes into the report or not.

Thus, when we examine a given node in the tree, we already know which of its descendants made it into the report and which did not, but we actually need even less information than that. When we reach a node we compute its "estimate" as the sum of the estimates of its children. If the difference between this value and the actual traffic is below the threshold, the node is ignored, otherwise it is reported with its exact traffic and its "estimate" counter is set to its actual traffic.

3.1.3 Computing Unidimensional Cluster Deltas

While compressed reports provide a complete traffic characterization for a given input, sometimes we are more interested in how the structure of traffic has *changed* over a given interval. More specifically, the challenge is to produce a concise report that indicates the amount of the change for all the clusters whose increase or decrease in traffic is larger

than a given threshold.

There are two ways to define the problem: by looking at the *absolute* change in the traffic of clusters, or by looking at the *relative* change. If the length of the measurement intervals is equal and the total traffic doesn't change much, one can use absolute change: the number of bytes or packets by which the clusters increase or decrease. However, to compare the traffic mix over intervals of different lengths (e.g. how does the traffic mix between 10 and 11 AM differ from the traffic mix of the whole day), we can only meaningfully measure relative change and must normalize the sizes of traffic clusters so that the total traffic is the same in both intervals. Thus, even if the traffic of a given cluster changed very much, if it represents the same percentage of the total traffic, its relative change is zero. For the rest of this paper we assume that the we are computing the absolute change or that the traffic has already been normalized.

To detect the clusters that change by more than H, we can use the full traces from each interval, but a more efficient algorithm can be built simply using the uncompressed clusters reports computed earlier. Since each cluster in the uncompressed report is above a threshold of H, if a cluster was below H in both intervals it could not have changed by more than H overall. But operating on this input will leave some ambiguities: we cannot be sure whether a cluster that appears only in one of the intervals and is close to H was zero in the other one and thus changed by more than H, or was close to but below H and thus changed by very little. Of course if the threshold used by the input reports is much below H, the ambiguity is reduced and we can ignore it in practice. We can also use compressed reports as input since they allow us to reconstruct the uncompressed report (with some loss in accuracy). A simple preprocessing step can provide the exact input required for the delta algorithm as follows: using reports with threshold H for both intervals we compute the set of clusters that were above H in either of them and in one more pass over the trace we compute the exact traffic in both intervals for each of these clusters.

Given uncompressed reports of high volume clusters, we then apply a similar compression algorithm to compress the delta reports. We make decisions on whether to include a cluster or not into the delta report by comparing its actual change to the estimate based on more specific cluster already reported: if the estimate is lower or larger by at least H than the actual change, the cluster is included into the delta report. Note that this can (and does) lead to putting clusters that did not change by more than the threshold into the traffic report as shown by the following example. If the traffic from port 80 (Web) increased by more than the threshold and we put it into the report. No other individual low ports changed much, but the total traffic from low ports remained the same because many other low ports sent slightly less thus compensating for the increase in the Web traffic. Our delta report needs to indicate that the total traffic from low ports did not change because otherwise the manager would assume that it increased by approximately as much as the Web traffic.

LEMMA 3. *The number of clusters above the threshold in the* delta report *is at most* $s_1 + s_2$.

Proof Each cluster in the report covers a traffic of at least H from one of the intervals that was not reported by any of its descendants. The sum of the absolute value of these differences is at most $T_1 + T_2$ (T_1 is the total traffic of the first measurement interval and T_2 is the total traffic of the second one) because each flow can be associated with at most one most specific cluster in the report and the sum of the sizes of all flows is $T_1 + T_2$. Therefore, there are at most $(T_1 + T_2)/H = s_1 + s_2$ clusters in the delta report. ■

While this result suggests that delta reports could be double the size of compressed report, in practice traffic changes slowly, so the delta reports are much more compact than compressed reports using the same threshold.

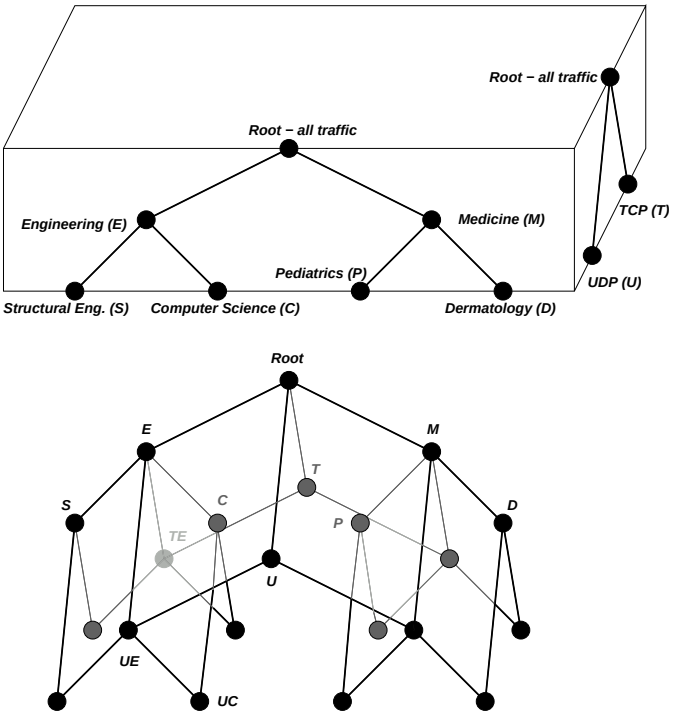


Figure 3: The multidimensional model combines unidimensional hierarchies (trees) into a graph. The hierarchy on the nearer side of the cube breaks up the traffic by prefixes; the hierarchy on the right side of the cube by protocol. For example, the node labeled C on the nearer side represents the Computer Science Department, the node labeled U on the right side represents the UDP traffic and the node labeled UC in the graph represents the UDP traffic of the Computer Science Department.

3.1.4 Computing "Unexpectedness"

Our definition of "unexpectedness" (**Operation 4**) relies on multiple dimensions to compute an expected size for a cluster. Therefore it is implicitly not applicable for the unidimensional case.

3.2 Multidimensional case

Compressed unidimensional reports can tell us that there is much UDP traffic, or that the traffic of the Computer Science Department is large, but they cannot tell us that the UDP traffic of the Computer Science Department is large. To capture such patterns, the multidimensional case combines multiple unidimensional hierarchies.

In the top of Figure 3, the closer face of the cube shows the prefix hierarchy that breaks up the traffic of a hypothetical university between the Engineering School and the Medical School, and breaks up the traffic of the Engineering School between the Structural Engineering Department and the Computer Science Department. On the right side of the cube we illustrate another hierarchy that breaks up the traffic by protocol into TCP and UDP.

When we combine these hierarchies in the bottom of Figure 3 we obtain a specific type of directed acyclic graph, a lattice[2]. Nodes in this graph have up to k parents instead of just one: one parent for each dimension along which they are not defined as *. For example node UC that represents the UDP traffic of the Computer Science Department has as parents the nodes UE (the UDP traffic of the Engineering School) and C (the total traffic of the Computer Science Department).

Unlike the unidimensional case, two clusters can overlap and still neither includes the other: one can be more specific along one dimension, while the second can be more specific along another one. For example clusters UE and C overlap (their intersection is UC) but none of them includes the other. The size of the graph is much larger than the sizes of the trees representing the hierarchies of individual fields: it is the product of their sizes.

We use the phrase unidimensional ancestor of cluster X along dimension i to denote the cluster defined the same way as X in its ith field, but having wildcards in all the other $k-1$ fields. This is an ancestor of X in the graph and it is also a unidimensional cluster along dimension i. In our example, C is the unidimensional ancestor of UC along the prefix dimension and U is its unidimensional ancestor along the protocol dimension. We use the phrase children of cluster X along dimension i to denote the clusters that have exactly the same sets for all other dimensions and for dimension i their sets are one step more specific (i.e. they are children of the set used by X in the hierarchy of field i). For example S and C are the children of E along the prefix hierarchy and UE and TE are its children along the protocol hierarchy.

3.2.1 Computing Multidimensional Clusters

For the multidimensional case, we now describe the algorithm to do **Operation 1** (computing all high volume clusters). Before we do so, we upper bound the number of high volume clusters in the multidimensional case.

LEMMA 4. *The number of clusters above the threshold is at most* $s\prod_{i=1}^{k} d_i$.

Proof The counter of each flow can contribute to at most d_i sets along dimension i (there can be less than d_i because some leaves of the tree might be closer to the root). There are at most $\prod_{i=1}^{k} d_i$ ways of combining these sets into clusters so each flow contributes to at most $\prod_{i=1}^{k} d_i$ clusters. The sum of the counters of all flows is T. The sum of the sizes of all clusters is at most $T\prod_{i=1}^{k} d_i$. Therefore, there are at most $T/H\prod_{i=1}^{k} d_i = s\prod_{i=1}^{k} d_i$ clusters larger than H. ■

[2]Actually we would need to add the special bottom cluster $\perp$ that matches no flows to turn this graph into a lattice. In its current form it is only a semi-lattice.

Our algorithm examines all clusters that may be above the threshold; for each such cluster, the algorithm examines all n flows, and adds up the ones that match. If the traffic is above the threshold, the cluster is reported, otherwise it is not. Explicitly going through the close to $n\prod_{i=1}^{k} d_i$ clusters generated by the n flows in the input is not an acceptable solution for the configurations we ran on. Therefore our algorithm restricts the search space (thereby reducing running time) based on a number of optimizations that prune the search space.

The first optimization is that all the unidimensional ancestors of a certain cluster include it, so the cluster can be above the threshold only if all its unidimensional ancestors are also above threshold. We first solve the k unidimensional problems. After this, we restrict the search to those clusters that have field values appearing in each of the unidimensional reports.

Next, observe that traversal of the search space is such that we always visit all the ancestors of a given node before visiting the node itself. Thus our second optimization considers only clusters with all parents above the threshold. This is very easy to check because in our graph nodes have pointers to their parents. A third optimization is to batch a number of clusters when we go through the list of flow records.

Even with all three optimizations, among all our algorithms, this is the one that produces the largest outputs and that takes the longest to run and we believe there are many ways to speed it up. For example, computing both packet and byte reports with a 5% threshold takes on average 16 minutes for a one day measurement interval, 2 minutes for a one hour measurement interval and 1 minute for a five minute measurement interval using a 1 GHz Intel processor. Using a threshold of 0.5% for a one day trace it takes on average around three hours to compute the uncompressed report. Even though there are pathological inputs that could force the size of the output close to its worst case bound (Lemma 4), the results we get on real data (see Appendix A) are often much smaller.

3.2.2 Compressing Multidimensional Traffic Reports

For the multidimensional case, we now describe the algorithm for **Operation 2**, compression. This is absolutely essential in this case because the uncompressed reports generated in the last subsection are very large, and would overwhelm the manager. On the other hand, the redundancy is also much higher than in the unidimensional case. Loosely speaking, this is because the inefficiencies along individual fields get multiplied together when looking at multiple fields. Thus, we have both a greater need as well as a greater opportunity. The following lemma bounds the maximum size of the compressed report.

LEMMA 5. *For any input data, there exists a compressed report of size at most* $(s\prod_{i=1}^{k} d_i)/(max\ d_i)$.

Proof Let m be the field with the deepest hierarchy ($max\ d_i = d_m$). Let T_j be the sizes of all clusters (indexed by j) that have * in field m. Applying the same method as in the proof of Lemma 4, we get $\sum T_j = T\prod_{i\neq m} d_i$. We can obtain any cluster by varying the mth field of the corresponding cluster j. We can compress all the clusters obtained from cluster j by varying field m using the unidimensional algorithm for field m, so by applying Lemma 2, we get that the number of

```
COMPRESS_REPORT
1  sort_more_specific_first(cluster_list)
2  foreach cluster in cluster_list
3      for field = 1 to 5
4          sum[i]=add_estimates(cluster.childlists[field])
5      endfor
6      cluster.estimate = max(sum[i])
7      if(cluster.traffic − cluster.estimate ≥ H)
8          add_to_compressed_report(cluster)
9          cluster.estimate = cluster.traffic
10     endif
11 endforeach
```

Figure 4: The algorithm for compressing traffic reports traverses all clusters starting with the more specific ones. The "estimate" counter of each cluster contains the total traffic of a set of non-overlapping more specific clusters that are in the compressed report. The clusters whose estimate is below their actual traffic by more than the threshold H , are included into the compressed report.

clusters in the result is bound by $s_j = T_j/H$. These reports for all j together cover all clusters, so for the total size of the report we get $\sum T_j/H = s\prod_{i\neq m} d_i = (s\prod_{i=1}^{k} d_i)/(max\ d_i)$. ■

We implemented a fast greedy algorithm for multidimensional compression (Figure 4). It traverses all clusters in an order that ensures that more specific clusters come before all of their ancestors (line 1). At each cluster we keep an "estimate" counter. When we get to a particular cluster we compute the sum of the estimates of its children along all dimensions (line 4) and set the estimate of the current cluster to the largest among these sums (line 6). If the difference between the estimate and the actual traffic of the cluster is below the threshold (line 7), it doesn't go into the compressed report. Otherwise we report the cluster (line 8) and set its "estimate" counter to its actual traffic (line 9).

The invariant that ensures the correctness of this algorithm is that after a cluster has been visited, its "estimate" counter contains the total traffic of a set of non-overlapping clusters that are in the compressed report. It is easy to see how this invariant is maintained: when computing the estimate for the cluster, for each dimension, the algorithm computes the sum of the estimates of the children of the node (cluster) along that particular dimension. Since the sets at the same level of the field hierarchy never overlap, these children will never overlap, so the invariant is maintained.

The problem statement allows the algorithm to consider all non-overlapping sets of more specific clusters reported when computing the estimate. Our algorithms does something simpler. There are two questions to consider: can this cause the results to be incorrect and can it cause the results to be suboptimal (i.e. larger than necessary). Our algorithm does not look at all sets of non-overlapping more specific clusters, just at the ones that can be partitioned along a dimension or another. Thus *it will sometimes include clusters into the report that could have been omitted, but it will never omit a cluster that does not meet the compression criterion* (i.e., is larger by more than H than the traffic of each of the sets of non-overlapping more specific clusters in the compressed report). In our experience this is a small price to pay for the big gains in performance we get by performing simpler local checks. Computing both byte and packet compressed reports took less than half a minute even for a low threshold of 0.5%.

Unfortunately there are inputs that make the size of the result produced by our algorithm get very close to the upper bound of Lemma 4, which is exponential in the number of fields. This did not seem to be a problem in practice when running AutoFocus, but we suspect there are tighter bounds and better algorithms that the one we use.

In Appendix A we present measurements of the size of the traffic reports produced by our algorithms when run on traces of actual network traffic. Compressed traffic reports are two to three orders of magnitude smaller than uncompressed reports and more than two orders of magnitude below Lemma 4. For a threshold of 5% of the total traffic the average report size is around 30 clusters. This is not influenced significantly by the length of the measurement interval or the diversity of the traffic (backbone versus edge), but the size of the report is proportional to the inverse of the threshold.

3.2.3 Computing Multidimensional Cluster Deltas

While in the unidimensional case the computation of delta reports was a simple extension of the compression algorithm, in the multidimensional case the interactions between compression (**Operation 2**) and deltas (**Operation 3**) are more complex.

Our algorithm takes as input uncompressed reports for the two intervals. We assume that these reports contain the exact traffic of all clusters that are above H in either of the intervals (see Section 3.1.3 for a discussion about the use of other types of input). Based on these two reports we build the graph representing the relationships between clusters above the threshold in any of the intervals. For all nodes we compute the change in traffic – the explicit list of the clusters that changed (increased or decreased) by more than H. This could be considered our uncompressed delta report. For compressing the delta report we use a greedy algorithm similar to our multidimensional algorithm: we traverse the clusters, more specific first, and for each node, based on the more specific clusters we already decided to add to the delta report, we decide whether the current cluster needs to be added or not. However, the procedure we use for deciding if a cluster is added or not is quite different.

When computing the compressed report in the previous section, we were looking for the set of non-overlapping clusters (each more specific than the current cluster and each in the compressed report) that had the largest traffic. The situation is different for the delta report. One thing that doesn't change is that if a set of more specific clusters includes another set we ignore the subset. Here by one set of clusters including another we mean that it matches all flows the other set matches, it can be the case that the second set has clusters that are not in the first set, but they are more specific than some clusters from the first set. We call a set of non-overlapping clusters *maximal* if there is no other set of non-overlapping clusters that includes it. Change can be negative or positive, so if we have two maximal sets we can not ignore the one with a smaller change (as we could in the case of simple compression).

Our rule for deciding whether to leave out a cluster from the delta report is to ensure *all* maximal sets of more specific clusters are within the threshold (either direction) of the actual change for the cluster. Because we need to look at all maximal sets of more specific clusters we cannot use the simplification used by the earlier report compression algorithm that made all decisions locally by restricting itself to sets that can be partitioned along one of the dimensions. Despite a lack of polynomial bounds for the running time of this algorithm, our implementation made decisions on clusters with up to 50 more specific cluster in the change report within seconds.

3.2.4 Computing "Unexpectedness"

Recall **Operation 4** which seeks to prioritize clusters via a measure of unexpectedness based on comparing the cluster percentage to the product of the percentages computed for each field in the cluster by itself. Computing the unexpectedness score of a given cluster is very easy using the graph describing the relations between the high volume clusters. We only need to locate in the graph the unidimensional ancestor along each of the dimensions (the cluster whose definition along that field is the same as that of our cluster and it has *s for all the other fields).

4. THE AUTOFOCUS TOOL

The AutoFocus prototype is an off-line traffic analysis system composed of three principal components:

- **Traffic Parser**. This component consumes raw network data, and builds a database of field values according to the cluster definitions described earlier. Our current system uses sampled packet header traces as input, but it could easily be modified to accept other forms of data such as sampled NetFlow records.

- **Cluster Miner**. The cluster miner is the core of the tool and applies our multidimensional algorithms (Section 3) to compute compressed traffic clusters, cluster deltas and unexpectedness scores over the database constructed by the parser. The output of this component is a traffic report written to a file.

- **Visual Display**. The visual display component is responsible for formatting the raw report and constructing graphical displays to aid understanding. To improve user recognition of individual elements, we postprocess the raw traffic report to attach salient names to individual addresses and ports. These names are generated from the WHOIS and DNS services, lists of well-known ports, as well as user-specified rules that contain information about the local network environment (e.g. that a particular host is a Web proxy cache or a file server). In addition to the base report, the display component also generates a series of time-domain graphs, using different colors to identify a set of key traffic categories. These categories can contain multiple clusters. Categories are ordered and each flow is counted against the first category it matches, traffic not falling any particular category is lumped into an "Other" category. Ideally, the categories are also constructed to be representative of "interesting" aggregates (e.g. outbound SSL traffic from any of our Web servers). Currently, the user specifies these categories – typically based on examining the clusters contained in the raw report. Heuristics for automatically selecting these traffic categories remains an open problem, complicated by the requirement that categories be meaningful. We expect that some amount of user involvement will always be beneficial.

Figure 5 depicts a byte volume report generated by AutoFocus using a 5% threshold on a trace recently collected from the SD-NAP exchange point. After identifying the size of the total trace and the threshold parameters, the report provides a series of five unidimensional traffic distributions – protocol, source address, destination address, source port and destination port. Indentation is used to reflect the address hierarchy present in CIDR-based addresses and each address prefix is named according to the best information available (e.g. DNS, WHOIS). For example, the report indicates that 66% of traffic in this trace originates from 192.128/10, which is not homogeneously allocated, however most of this traffic can be attributed to the more-specific prefix, 192.172.226.64/26, owned by CAIDA. Note that prefixes between these two records, from /10 to /26, are compressed away because traffic from them does not differ by more than 5% (17.7GB) from the more specific /26 prefix. Ultimately, an individual IP address 192.172.226.89/32 (magrathea.caida.org) is seen to be responsible for the majority of this activity.

Following the unidimensional reports, is a compressed traffic report, in which a wildcard (*), symbol is used to indicate the "don't care" fields. Starting from the least-specific clusters (such as arbitrary TCP traffic from servers using low ports to clients), the report includes all more-specific clusters that meet the threshold criteria. Note that the most-specific cluster shown exactly identifies the *particular transfer* that consumed the majority of total bandwidth (from magrathea.caida.org to hpss07.sdsc.edu). Moreover, its unexpectedness score of 596%, automatically identifies it as a good candidate for further investigation. Consequently, this cluster is also identified in the delta section at the end of the report.

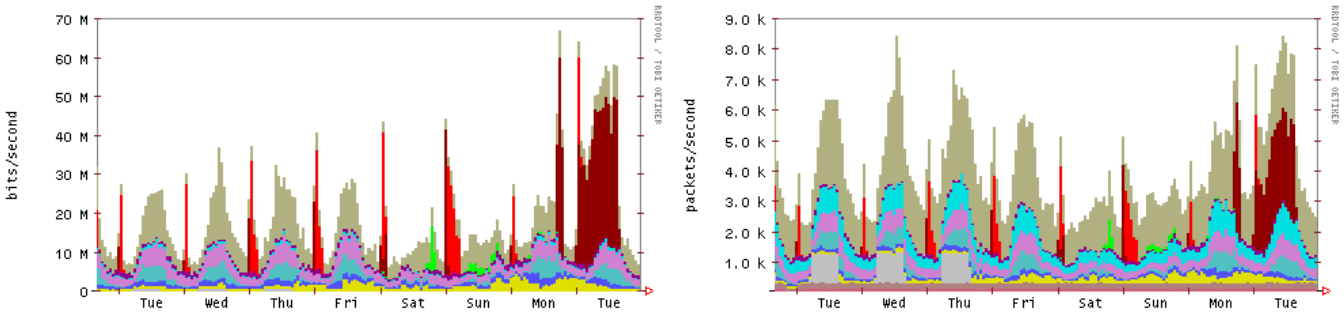
Finally, using a non-overlapping subset of the clusters (including the large backup transfer), the report includes time-series graphs of traffic colored by the appropriate clusters on two timescales: short (two days) and long (eight days). Only the eight day graphs are shown in Figure 5. In this particular example, two distinct patterns are immediately evident: regular diurnal activity across several applications (shown in light colors) and the backup process (shown in dark) which executes regularly in the middle of the night.

The AutoFocus prototype has another feature that proved useful on a number of occasions: drilling down into individual categories. For each of the categories, the tool provides separate time series plots and reports that analyze the internal composition of the traffic mix within that particular category.

5. EXPERIENCE WITH AUTOFOCUS

5.1 Comparing with unidimensional methods

In this section we contrast our multidimensional method with unidimensional analysis. Figure 6 presents a simplified version of the time domain plot generated by AutoFocus for




```
Total traffic is 354 GB. The threshold is 5%=17.7 GB.
Unidimensional reports
Protocol breakup
 6(TCP)    98.653%  350 GB
Source IP breakup
         137.131.0.0/16(Scripps)                                  7.778%  27.6 GB
   192.128.0.0/10                                                66.450%  235 GB
                  192.172.226.64/26[Caida]                       61.371%  217 GB
                        192.172.226.89(magrathea.caida.org)      55.424%  196 GB
Destination IP breakup
          132.249.0.0/17[SDSC]                                    5.506%  19.5 GB
          137.131.0.0/17[Scripps]                                 5.529%  19.6 GB
          137.131.128.0/17[Scripps]                               5.574%  19.8 GB
 192.0.0.0/8                                                     60.937%  216 GB
                        192.67.21.154(hpss07.sdsc.edu)           55.362%  196 GB
 198.0.0.0/8                                                      6.651%  23.6 GB
Source port breakup
 lowport    22.768%  80.8 GB
 80(http)   16.784%  59.6 GB
 highport   76.532%  271 GB
 4339       55.362%  196 GB
Destination port breakup
 lowport     6.121%  21.7 GB
 highport   93.178%  330 GB
 35904      55.362%  196 GB
Multidimensional report
 Source IP                            Destination IP                 Pr.  Src port   Dst port   Traffic   Label
 *                                    *                              TCP  lowport   highport   80.4 GB   108.2%
 *                                    *                              TCP  80(http)  highport   59.6 GB   108.8%
 *                                    *                              TCP  highport  lowport    21.1 GB   128.7%
 *                                    *                              TCP  highport  highport   248 GB     99.4%
 *                                    132.249.0.0/17[SDSC]           TCP  *         highport   19.0 GB   105.7%
 *                                    137.131.0.0/16(Scripps)        TCP  80(http)  highport   18.6 GB   306.0%
 *                                    137.131.0.0/16(Scripps)        TCP  highport  *          18.1 GB    60.9%
 *                                    137.131.0.0/17[Scripps]        TCP  *         *          19.5 GB   100.6%
 *                                    137.131.128.0/17[Scripps]      TCP  *         *          19.6 GB   100.6%
 *                                    192.0.0.0/8                    *    *         highport   214 GB    106.4%
 *                                    192.0.0.0/8                    TCP  *         *          214 GB    100.8%
 *                                    198.0.0.0/8                    TCP  *         highport   22.2 GB   102.2%
 137.131.0.0/16(Scripps)              *                              TCP  *         highport   20.5 GB    80.7%
 192.172.226.0/24(SDSC NAP)           *                              TCP  highport  highport   214 GB    139.9%
 192.172.226.0/25(Caida)              *                              TCP  *         highport   221 GB    110.5%
 192.172.226.64/26[Caida]             *                              TCP  *         *          217 GB    101.4%
 192.172.226.89(magrathea.caida.org)  192.67.21.154(hpss07.sdsc.edu)  TCP  4339      35904      196 GB    596.7%
Delta report ------------ traffic changed from 213 GB to 354 GB, threshold is 5%=17.7 GB ------------
 *                                    137.131.0.0/17[Scripps]        TCP  highport  *          -20.6 GB
 134.79.0.0/18[SLAC.stanford]         137.131.0.0/16(Scripps)        TCP  highport  lowport    -19.6 GB
 192.172.226.3(ra.caida.org)          192.67.21.168(hpss45.sdsc.edu) TCP  highport  highport   -30.5 GB
 192.172.226.89(magrathea.caida.org)  192.67.21.154(hpss07.sdsc.edu) TCP  4339      35904      196 GB
```

Figure 5: The report for the 17th of December 2002 (one of the 31 daily reports for this trace) contains compressed unidimensional reports on all 5 fields and the multidimensional compressed cluster report all using a threshold of 5% of the total traffic. In the unidimensional reports the percentages indicate the share of the total traffic the given set has. In the multidimensional cluster report they indicate the unexpectedness score (how the given cluster relates to the size we would expect based on the uniformity assumption). Note how much smaller the delta report is when compared to the full report.

Friday the 20th and Saturday the 21st of December 2002, together with unidimensional plots for all five fields (Figures 7 to 11). Besides one plot being more compact than five, our multidimensional view has the advantage of making it easier to see very specific facts about the network traffic.

For example the light colored "spot" between 7 AM and 2 PM on the first day is UDP traffic that goes to a specific port of a specific multicast address. While we could

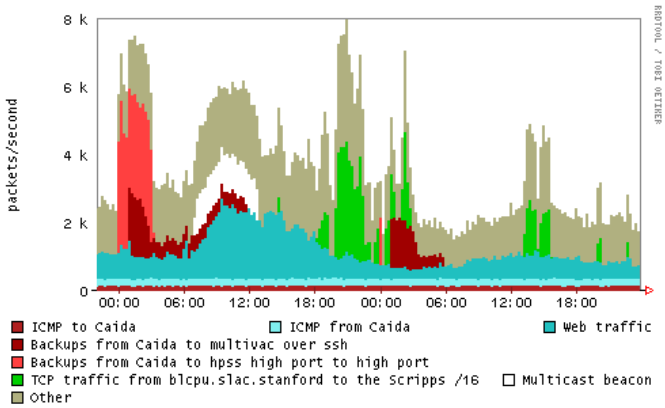


Figure 6: Multidimensional report

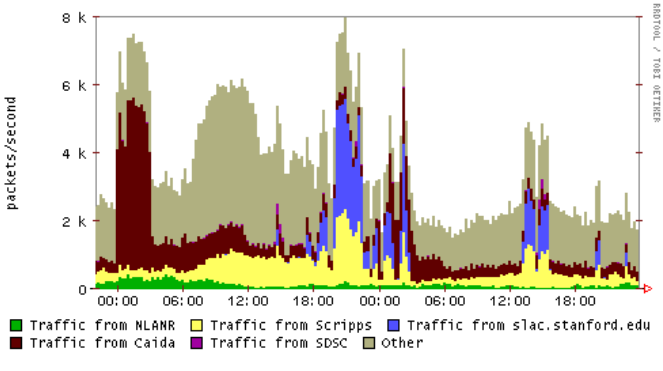


Figure 7: Unidimensional – source network

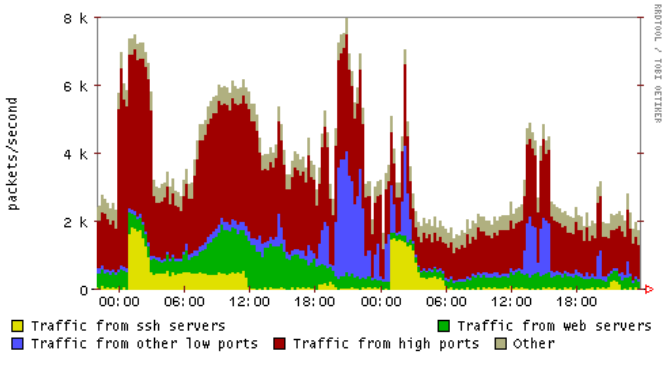


Figure 8: Unidimensional – source port

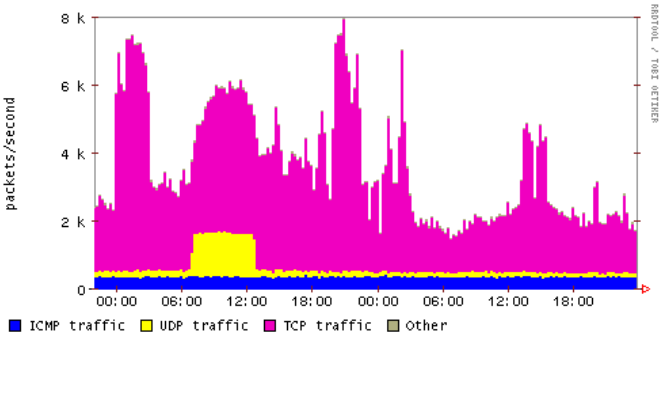


Figure 9: Unidimensional – protocol

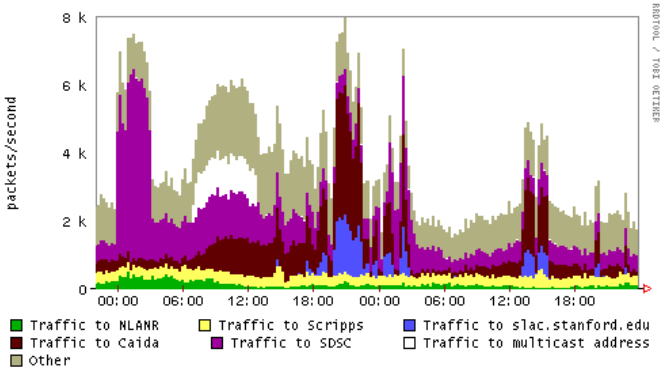


Figure 10: Unidimensional – destination network

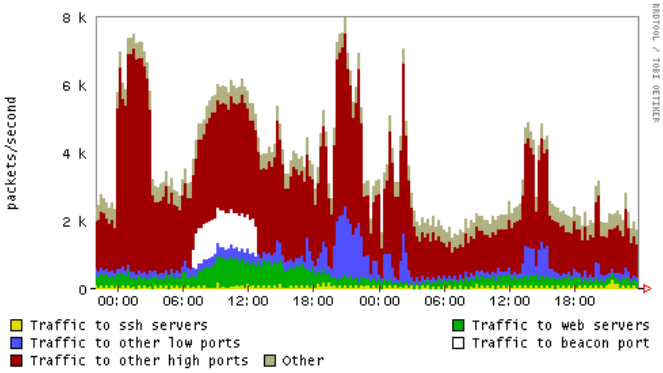


Figure 11: Unidimensional – destination port

Figure 12: Reports for Friday the 20th and Saturday the 21st of December 2002 using traffic clusters

manually correlate the corresponding "bright spots" from the protocol, destination prefix and destination port plots, the AutoFocus plot automatically identifies the key usage directly.

The very constant dark streak at the bottom of the protocol plot representing ICMP traffic might look puzzling. The AutoFocus report breaks it up into two distinct categories (at the bottom of the plot): ICMP traffic to and from a specific prefix (/28) belonging to CAIDA. The simple explanation for this large and consistent ICMP traffic is that servers in this prefix are involved in continuous network measurement activity (in particular, the Skitter network probe tool) using ICMP packets.

The massive red spike causing the traffic surge from 12:01 AM until 3 AM the first day and the longer dark brown traffic cluster from 1 AM until 11 AM that day and 1 AM and to 6 AM the second day are two different types of backups. It would be difficult to disentangle them using only unidimensional plots. Since they use different source ports (SSH versus high ports) they show up separately in the source port report, but since they come from the same source network and go to the same destination network they show up

together in those plots. Not only does the AutoFocus plot provide this information more readily but it also gives more specific information: the individual destination IP address used by all the backups in one of the categories.

5.2 Experience with analysis of traffic traces

This section presents our initial experiences using AutoFocus to analyze traffic taken from three large production networks. While the usefulness of the insights gleaned using AutoFocus is hard to quantify, we hope that presenting some of them will give the reader a more accurate idea about the power of our system.

5.2.1 Small network exchange point

Our first trace was collected from SD-NAP, a small network exchange point from San Diego, California, that connects many research and educational institutions and also connects some of them to the rest of the Internet. The trace is 31 days long and it starts on the 7th of December 2002. The collection point was connected to the Fast Ethernet LAN the NAP is built around[3], so packets in the trace do not have a direction (incoming/outgoing) associated with them. In addition, we were able to consult with those familiar with this network to calibrate our conclusions and receive useful feedback about out results.

When analyzing the raw reports we looked for traffic clusters that were large, but did not completely dominate the traffic (e.g. the cluster containing all TCP traffic is not very useful). In particular, we found that port 80 (TCP) Web traffic was so large that it was best subdivided into four categories: Web traffic from a particular server at the Scripps Institute, Web traffic destined for IP addresses within the Scripps /16 (Web client traffic), other outbound Web traffic and other inbound Web traffic. Of these, the first and second categories proved to be the most interesting. Traffic from the Web server showed a clear diurnal pattern with peaks before noon (sometimes a second peak after noon) and lows after midnight. It also showed a clear weekly pattern with lower traffic on the weekends and holidays than during the weeks. The second category (the Web clients) had similar trends, but the lows around midnight usually went down to zero and the difference between weekend and weekday peaks was much larger. In retrospect, this is to be expected since the client traffic requires the presence of employees at Scripps, while the server traffic can be driven by requests from home broadband machines or users outside of the Scripps environment.

Another interesting traffic cluster contained Web proxy traffic originating from port 3128 of a particular server at NLANR. The amount of traffic had a daily and weekly cycle, but the lowest traffic was observed at noon and the highest at midnight. Using AutoFocus' "drill-down" feature we were able to examine the subclusters of this traffic, which identified large clusters containing transfers to second-level caches from Taiwan, Indonesia, Spain and Hong Kong. Evidently the volume of the Web cache traffic was driven by the daily cycle of the clients in these other time zones.

We also identified two categories for non-backup traffic involving SSH servers (we discussed the backup traffic earlier in Section 5.1). When graphed in the time domain we observed spikes of uniform size (100MB) spaced evenly at 30 minute intervals. We conjectured that these were SCP transfers scheduled through crontab, a hypothesis confirmed by the network administrators of the IP addresses involved. We believe that this particular pattern would have been difficult to isolate using a purely protocol or address oriented view.

Finally, AutoFocus places the remaining non-categorized traffic into an amalgamated "Other" category. On the last day of the trace, we saw a huge sharp increase at 5:30 PM in the "Other" traffic that saturated the link followed by a sudden decrease at 11 PM. Again using the drill-down feature, we observed that this change was mostly attributable to traffic between two nearby universities: from UCSD to UCLA. However, traffic between these two did not appear in any other cluster over the previous 30 days. Upon investigation we determined that traffic bulge was due to a temporary network outage that forced traffic normally using the CalREN network to transit SD-NAP instead.

5.2.2 Large research institution

Our second trace was taken from the edge of a network that connects one large research institution (roughly 15,000 hosts) to the Internet. The second trace is 39 days long and it starts on the 12th of December. In this case, we had access to similar although less-detailed expertise concerning the operation of the network.

Many of our findings for the second trace were similar to those we have described earlier, but there were some differences. The Web traffic on the second trace is small enough to be put into just two categories: inbound and outbound Web traffic. Both have an unsurprising strong daily cycle, with traffic much lower on weekends and holidays.

We also observed a series of regularly scheduled backup transfers: one from a range of machines destined to TCP port 7500 that had regular daily spikes at 11PM and 5AM followed by periods of quiescence. Another example was a regular 40GB transfer that started at 8PM on each Wednesday (usually lasting until 10AM the following morning). This activity proved to be a full backup of a large RAID array.

We observed a single large cluster containing a series of regular TCP transfers from a single host to port 5002 on three other hosts distributed around the Internet. This usage was ultimately found to be a regularly scheduled network measurement experiment that was part of a distributed research activity.

The most interesting result for this second trace came by looking at the breakout of the Sapphire worm[10]. This worm exploits a vulnerability of the Microsoft SQL server running on UDP port 1434 and spreads extremely aggressively using single 404 byte packets to infect random destinations. We computed the traffic reports for three hour measurement intervals. The time at which the worm started was apparent in the time series plot (see Figure 13). It showed up as a huge increase in the "Other" traffic category. Drilling down into the report describing that category, the worm was conspicuous: 90% of the traffic was to UDP port 1434. A quick comparison between the packet and byte reports also gave us the average packet size. Furthermore, the report readily revealed the 6 internal IP addresses that generated 80% of the traffic: these were local hosts infected by the worm aggressively trying to infect the outside world. These

[3]Due to the network topology asymmetric routing does happen so we are not guaranteed to see both directions of all connections.

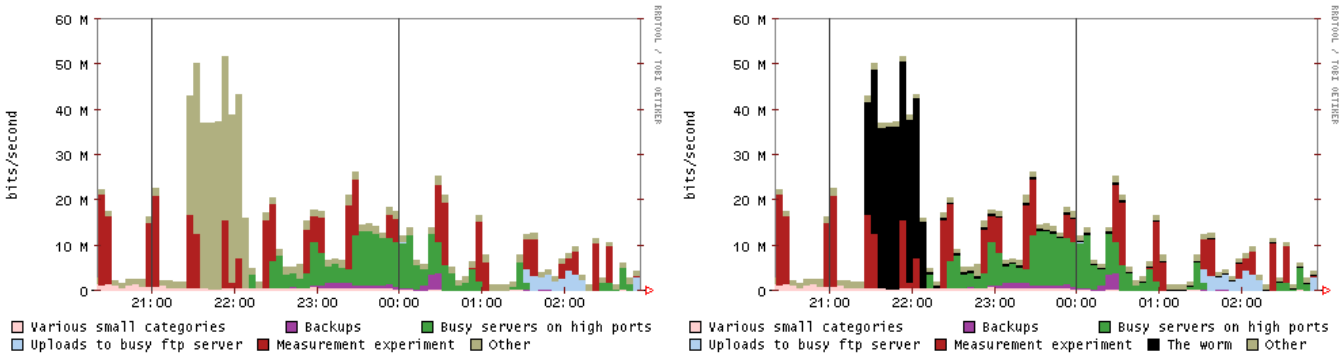


Figure 13: The Sapphire/SQL Slammer worm shows up in the time series plots as a big increase in the traffic of the "Other" category around 21:30. Once we highlight the worm by putting it into a separate category, it is evident that while its traffic is significantly reduced at 22:10 when the infected internal hosts were neutralized, worm traffic persists at a lower level because of outside hosts spreading it into our network.

compromised servers were promptly neutralized by the network administrators. In the next three hour interval, while the traffic in the "Other" category decreased to levels similar to those before the worm, still a substantial fraction of it (23%) was worm traffic. The report revealed that this was traffic originating on the outside: it consisted of incoming probes trying to infect internal hosts. We also performed an analysis of the trace with the worm traffic separated into its own category (the second plot from Figure 13). We were able to see fine details. For example some of the infected hosts did not spread their traffic uniformly over the whole address space, but focused on single /8s. This is consistent with the observation[10] that for some values of the random seed, the algorithm used by the worm to select target addresses choose them from a limited set. We believe that this particular example shows the strength of our multidimensional approach: AutoFocus is able to promptly bring to the network manager's attention and describe in great detail such unexpected and unpredictable event as a worm epidemic.

While none of the traces we worked with contained a massive denial of service attack, we believe that AutoFocus would bring that to the attention of the network operator the same way it showed the worm. The victim of the of the attack (whether it is an individual IP address or a prefix) will show up in the report with a very large number of packets (or bytes depending on the type of attack). Furthermore, the attack will reveal the protocol used by the attack and possibly the port number if it is kept constant. If the source address is faked at random from the whole IP address space, the report will not associate any particular source address with the attack traffic hitting the victim. However, if for some reason (e.g. egress filtering at the site the attack originates from), the source addresses are restricted to a certain prefix (or a small number of prefixes), the report will identify these, thus facilitating prompt and specific response.

We presented the output of AutoFocus to network managers of the first two networks we had traces from. Their reactions were very positive. It was easy for them to understand the output. They appreciated the intuitiveness of the time series plots, the large amount of information they convey and the ease with which the traffic reports and the drill-down feature provided them more detailed information when they needed it. The managers of both networks expressed interest in widely deploying our tool in their respective networks.

5.2.3 *Backbone*

A third trace we looked at was captured in August 2001 from an OC-48 backbone link and is 8 hours long. We looked at traffic reports for one hour measurement intervals. The reports reveal that around two thirds of the bytes on the link come from TCP port 80 and around one third came from high ports. The report also revealed that around one third of the traffic was from high ports to high ports. This is consistent with the behaviour of peer to peer traffic[4]. Through the unexpectedness labels the report revealed some further facts that seemed surprising at first. There were specific source and destination prefixes where the web traffic represented almost all of the traffic. Also there were some where the high port to high port traffic clearly dominated all other types of traffic. Our explanation for the prefixes that send almost only web traffic is the clustering of web servers in web hosting centers. The source and destination prefixes dominated by high port to high port traffic might be broadband providers. The prefixes that receive almost exclusively web traffic could be organizations with many web clients whose internal policy prohibits the use of peer to peer applications.

6. RELATED WORK

MRTG or Multi Router Traffic Grapher [12], is an extremely widely used tool written by Tobias Oetiker in 1995. It plots the time series of the usage of a link at various timescales based on SNMP interface counters and puts the results into web pages readable by any graphical browser.

FlowScan [13], dating from 2000 is a more recent tool by Dave Plonka. It uses NetFlow [11] data to give more detailed information about the traffic by breaking it down in a number of ways: by the IP protocol; by the well-known service or application; by IP prefixes associated with "local" networks; or by the AS pair between which the traffic was exchanged. There are many other applications that produce

[4]Although passive ftp transfers would also generate similar traffic mixes, we believe that peer to peer traffic is a more plausible explanation.

(among other things) similar breakdowns of the traffic such as CoralReef [1] or the IPMON project [2] based on packet traces instead of NetFlow data. In terms of our terminology, these reports visually display multiple one-dimensional (i.e., one field) traffic clusters. Also there are methods for reducing the size of the raw data describing the traffic mix in a way that does not preclude future analyses: through sampling[4] or sketches[6]. While our method also produces a very compact summary of the traffic mix, its primary purpose is not to be used by further analyses, but to convey a description of the traffic to the human operator.

In [5] Estan and Varghese present algorithms that automatically and efficiently identify large clusters, once the definition of clusters is fixed. The problems we are solving are related to classical clustering [9], but they differ in that we do not use an Euclidian space but the space defined by the field hierarchies. Another data mining problem, finding association rules requires finding the frequent item sets [3] and it is a well studied problem in the field of data mining. Finding all high volume clusters is a generalization of the problem of finding frequent item sets. The two important difference between these two problems are that traffic clusters are using the natural hierarchies while item set only use boolean attributes and that the frequent item set problem is usually applied to high dimensional spaces. Han and Fu address the the discovery of multiple-level association rules[8] using a single hierarchy similar to the per field hierarchies we use.

7. CONCLUSIONS

Managing IP-based networks is hard. It is particularly complicated by not understanding the nature of the applications and usage patterns driving traffic growth. In this paper, we have introduced a new method for analyzing IP-based traffic, *multidimensional traffic clustering*, that is designed to provide better insight into these factors. The novelty of our approach is that it automatically infers, based on the actual traffic, a traffic model that matches the dominant modes of usage. Unlike previous work, our algorithms can analyze traffic along multiple different "dimensions" (SrcAddress,DstAddress,Protocol,SrcPort,DstPort) at once, and yet is able to use compression to map results in this high dimensional space into a concise report. In essence, our approach exploits the locality created by particular modes of usage.

In addition to developing these algorithms, we have embodied them in the AutoFocus analysis system. We have developed a Web-based user-interface to allow managers to explore clusters across multiple time-scales and to "drill-down" to explore the contents of any clusters of interest. Our preliminary experiences with this tool have been extremely positive and we have been able to identify unusual traffic patterns that would have been considerably harder to identify using conventional tools. Moreover, we have received positive feedback from network mangers who have quickly been able to appreciate the benefits of our approach.

8. ACKNOWLEDGMENTS

We would like to thank Vern Paxson and Jennifer Rexford for the many discussions that lead to the clarification of the concept of traffic clusters. We would also like to thank David Moore and Vern Paxson for their help with the evaluation of the AutoFocus prototype. Support for this work is provided by NSF Grant ANI-0137102, the Sensilla project sponsored by NIST Grant 60NANB1D0118.

9. REFERENCES

[1] Coralreef - workload characterization. `http://www.caida.org/ analysis/ workload/`.

[2] Ipmon - packet trace analysis. `http://ipmon.sprintlabs.com/ packstat/ packetoverview.php`.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[4] Nick Duffield, Carsten Lund, and Mikkel Thorup. Charging from sampled network usage. In *SIGCOMM Internet Measurement Workshop*, November 2001.

[5] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM*, August 2002.

[6] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Quicksand: Quick summary and analysis of network data dimacs technical report, 2001.

[7] Pankaj Gupta and Nick Mckeown. Packet classification on multiple fields. In *Proceedings of the ACM SIGCOMM*, pages 147–160, August 1999.

[8] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland, September 1995*, pages 420–431, 1995.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. Springer, 2001. pages 453-479.

[10] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. The spread of the sapphire/slammer worm. Technical report, January 2003. `http://www.caida.org/ outreach/ papers/ 2003/ sapphire/ sapphire.html`.

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

[12] Tobias Oetiker and Dave Rand. Mrtg: Multi router traffic grapher. `http://people.ee.ethz.ch/ oetiker/ webtools/ mrtg/`, 1995.

[13] David Plonka. Flowscan: A network traffic flow reporting and visualization tool. In *LISA*, pages 305–317, December 2000.

[14] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. In *OSDI*, December 2002.

APPENDIX

A. THE SIZE OF TRAFFIC REPORTS

We use measurements to answer a number of questions about the size of traffic reports based on traffic clusters. We compare theoretical bounds against actual results. We investigate the effect of the threshold, the length of the measurement intervals and traffic diversity on the size of the

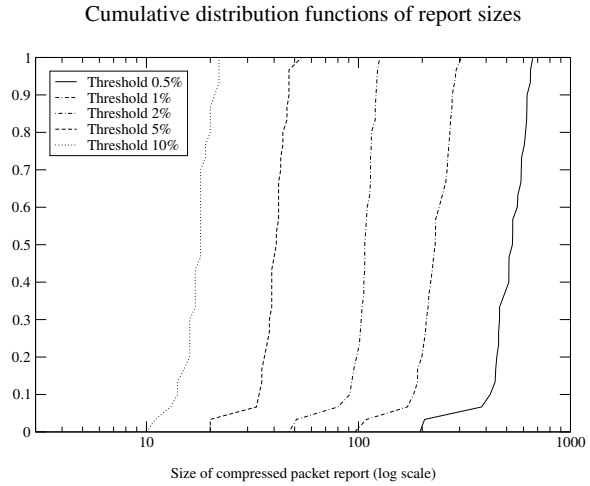


Figure 14: The size of compressed traffic reports is proportional to the inverse of the threshold. For the same configuration, most of the traffic reports have very close sizes, except a few that are much smaller due to high volume individual connections that dominate the traffic mix (and push the threshold up).

report. We also report running times for some of our algorithms. For these measurements we used our first trace is from the SD-NAP and a third one from a backbone link. Both of these traces use a packet sampling of one in 400.

A.1 Comparison of actual report sizes and theoretical bounds for different threshold values

We measured the number of high volume traffic clusters (the uncompressed report) and the size of the compressed report for each of the 31 days of our first trace, using both definitions: defining traffic as the number of bytes and also defining it as the number of packets. We used thresholds of 0.5%, 1%, 2% 5% and 10% of the total traffic. The results are in Table 1 together with the theoretical upper bounds on the sizes of the reports. The difference between the smallest and largest uncompressed report for the same configuration can be more than a factor of 100. A closer look at these extreme results explains these wide discrepancies. If there are a few very large TCP connections that dominate the traffic on a certain day, the uncompressed reports will be very large because there are very many more general clusters that include each of these fine-grained connections. Exactly the same type of traffic mix results in very small compressed reports because once the individual large connections are reported, there is not much other traffic to report. Because large connections dominate more the byte breakdown than the packet breakdown, the compressed reports are somewhat smaller when we look at bytes and the uncompressed ones are smaller when we look at packets. Figure 14 shows the cumulative distribution of compressed packet reports. For all configurations, the maximum number of high volume clusters is approximately within a factor of 10 of the bound of Lemma 4, but for all configurations there is more than a factor of 100 between the largest compressed reports and the bound of Lemma 5. This is encouraging because the compressed reports are what we actually use.

A.2 The effect of the length of the measurement interval

We measure the effect of the length of the measurement interval on the size of the report using a threshold of 5% of the total traffic. We use our entire first trace to obtain 31 one day long measurement intervals. We use two days of the trace with very different traffic mixes, Sunday the 8th of December and Monday the 9th, to obtain 48 one hour measurement intervals. We use as five minute measurement intervals from minute 22 to minute 27 of each of these one hour intervals.

The results from Table 2 suggest that the size of the compressed reports does not vary significantly with the length of the measurement intervals. Since shorter intervals have fewer flows, it is easier for the largest of them to dominate the traffic mix and this explains the slightly shorter compressed reports.

Note that except for the shortest intervals we considered (5 minutes) even the uncompressed reports are orders of magnitude smaller than the number of active flows which is a rough indication of the number of NetFlow records that would be generated during the measurement interval. Since we computed the number of flows active in the sampled trace this roughly matches the number of records sampled NetFlow would generate with the same sampling factor (1 in 400).

A.3 The effect of traffic diversity

We measure the effect of traffic diversity by comparing the size of the report using a threshold of 5% for the first trace and our third trace. Our third trace is very different: it is from an OC-48 backbone link. It is 8 hour long and it starts at 9:00 AM on the 14th of August 2002. This trace has both directions of the traffic. We divide the third trace into eight one hour measurement intervals. We use the same hours of the 16th of December (also a weekday) from our first trace.

The results from Table 2 suggest that the size of the compressed reports does not vary significantly with the size of the location the traffic, but the more diverse backbone traffic leads to slightly longer reports. Even though the running time on the backbone trace was double the running time on the first trace, we interpret our results as an indication that our method might scale to backbone links too.

H/T	Uncompressed size							Compressed size						
	Bound of Lemma 4	Byte report/10^3			Packet report/10^3			Bound of Lemma 5	Byte reports			Packet reports		
		min	avg	max	min	avg	max		min	avg	max	min	avg	max
0.5%	2,433,600	96.4	177	280	75	132	219	93,600	74	334	576	195	514	664
1%	1,216,800	54.2	96.3	166	15	50	120	46,800	47	155	281	96	228	306
2%	608,400	26.9	50.9	104	2.8	21.1	54.7	23,400	25	72	126	47	105	126
5%	243,360	7.16	18.1	46.1	0.75	8.85	42.4	9,360	9	29	50	20	40	54
10%	121,680	0.32	9.87	34.0	0.16	3.16	31.5	4,680	6	14	20	10	17	22

Table 1: Compressed reports are much shorter than uncompressed reports. Actual reports are orders of magnitude smaller than the theoretical bounds.

Number of intervals * length	Uncompressed report						Compressed report						Number of flows/10^3		
	Byte rep. size/10^3			Packet rep. size/10^3			Byte report size			Packet report size					
	min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max
31 * 1 day	7.16	18.1	46.1	0.75	8.85	42.4	9	29	50	20	40	54	164	267	420
48 * 1 hour	0.75	16.2	41.6	0.62	8.87	41.7	9	26	41	20	36	48	9.89	20.6	195
48 * 5 min	1.06	22.0	38.9	0.65	15.8	36.6	3	17	36	9	27	46	0.83	12.9	24.9

Table 2: The size of compressed reports is slightly smaller for shorter measurement intervals. The number of active flows is an indication of the number records NetFlow would generate.

Trace	Uncompressed report						Compressed report						Number of flows/10^3		
	Byte rep. size/10^3			Packet rep. size/10^3			Byte report size			Packet report size					
	min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max
trace1	0.75	4.69	13.2	0.77	1.64	7.39	28	36	41	32	37	41	22.4	64.9	357
trace3/dir0	0.69	0.75	0.82	0.56	3.66	19.5	34	38	43	34	40	45	320	340	376
trace3/dir1	1.00	1.19	1.30	0.69	0.75	0.81	33	40	45	42	45	50	471	514	588

Table 3: The greater diversity of backbone traffic (trace3) did not lead to significant increases in the size of the traffic reports.