Discovering Emergent Behavior From Network Packet Data: Lessons from the Angle Project

Robert L. Grossman* Michael Sabala Yunhong Gu
Anushka Anand Matt Handley Rajmonda Sulo
Lee Wilkinson
National Center for Data Mining
University of Illinois at Chicago

Abstract

We describe the design of a system called Angle that detects emergent and anomalous behavior in distributed IP packet data. Currently, Angle sensors are collecting IP packet data at four locations, removing identifying information, and building IP-based profiles in temporal windows. These profiles are then clustered to provide high-level summary information across time and across different locations. We associate certain changes in these cluster models with emergent behavior. Emergent clusters identified in this way are then used to score the collected data in near real time. The system has a visual analytics interface that allows different emergent clusters to be visualized, selected, and used for scoring of current or historical data. Each Angle sensor is paired with a node on a distributed computing platform running the Sector middleware. Using Sector, data can be easily transported for analysis or reanalysis.

1 Introduction

There are millions of computers connected to the Internet and billions of networks flows that access them. Unfortunately not all these flows are benign, and an increasing number of them are associated with some type of anomalous behavior, such as sending spam, probing for system vulnerabilities, attempting to install malware, and related behavior. Detecting suspicious flows across the Internet is a challenging problem in high performance analytics.

One of the reasons the problem is challenging is because the types of suspicious flows change all the time. One of the goals in this project was to detect

*Robert Grossman also works at Open Data Group behavior that is suspicious and different than the type of behavior that has been seen before. We call this type of suspicious behavior *emergent* and give a precise definition of it in Section 4.

There were three technical challenges in this project:

1. The first challenge was to develop an architecture for the system that supports discovery from very large, geographically distributed data sets.

2. The second challenge was how to define emergent behavior in a meaningful way and to develop algorithms to detect it.

3. The third challenge was to develop a visual analytics interface that could be used effectively by analysts.

There was also one practical challenge in the project:

4. The organizations we worked with were very reluctant to share data, due to the privacy required when working with IP traffic. A practical challenge was to develop an anonymization procedure and policies for handling data that are protective enough of privacy that institutions would share IP data but that left enough information so that sharing data was still useful.

In this paper, we introduce a system called Angle for making discoveries about emergent behavior from distributed IP data. Angle is based upon a framework so that information from geographically distributed locations can be combined together easily in order to detect emergent behavior that may not be readily apparent simply by analyzing data from one location.

The majority of prior work in this area is what is usually called signature based. Signatures of specific attacks are created and IP data is screened using these signatures. Snort is one of the most widely deployed systems for analyzing IP packet data using signatures [23]. There are also a variety of statistical based techniques. See, for example, [16] and the references cited there. Angle is a statistical based system.

This paper is organized as follows: Section 2 includes some background materials and describes related work. Section 3 describes the Angle architecture. Section 4 describes the data analysis methodology used by Angle. Section 5 describes the visual analytics interface. Section 6 describes several experimental studies and the status of the project. Section 7 includes some lessons learned and contains a summary and conclusion.

2 Background and Related Work

2.1 Emergent Behavior

There are many ways that emergent behavior can be defined. Our approach is to define emergence based upon two abstractions: windows and models (or summarizations). We give several examples below. Given these abstractions, assume that we have a sequence of windows w_1, w_2, w_3, $\ldots$ and a sequence of corresponding models M_1, M_2, M_3, $\ldots$. Loosely speaking, we say that there is emergent behavior at $\gamma > 1$ in case 1) the models M_i for $i \leq \gamma$ are similar; and 2) the models M_γ and $M_{\gamma+1}$ are dissimilar. More precisely, assume also that there is a distance function defined on the models:

$$d(M_i, M_j) \geq 0, \qquad d(M_i, M_j) = d(M_j, M_i), \qquad i \neq j, \qquad d(M_i, M_i) = 0.$$

More precisely, we say there is *emergent behavior* at $\gamma > 1$ in case there are constants $B_0 > 0$, $B_1 > 0$ and $D > 1$ such that

$$d(M_{i-1}, M_i) \leq B_0, \qquad i = \gamma - D, \gamma - D + 1, \ldots, \gamma - 1.$$

$$d(M_{\gamma-1}, M_\gamma) \geq B_1.$$

Of course, as a variant, we could require that the models M_i are uniformly similar within the windows $w_{\gamma-D}$, $\ldots$, $w_{\gamma-1}$.

Example. The set up for this example is that we have time stamped events and given a collection of events one can compute feature vectors. In the first example, we assume that the windows w_j are temporal and that each event can be assigned to one window. For each time window w_j, collect all the events that fall within the window and compute feature vectors. Let the model associated with window w_j be defined by a cluster model M_j containing k clusters, with centers $a_{j,1}$, $a_{j,2}$, $\ldots$ $a_{j,k}$, that is computed from the feature vectors associated with the window. Define a distance function

$$d(M_i, M_j) = \sum_{\alpha=1}^{k} \left(\min_{\beta} ||a_{i,\alpha} - a_{j,\beta}||^2 \right). \tag{1}$$

There are many variants of this example, depending upon how we define the clusters and how we measure the similarity (and dissimilarity) of two or more cluster models.

There is a large literature on outliers. A good overview of the subject is [3]. A standard definition is to define an outlier as an *observation (or set of observations) that appears to be inconsistent with the remainder of that set of data* [3] (page 7). Although one could argue that any method used to identify outliers could be used to identify emergent behavior, we are only interested in this paper in the much more narrow definition of emergent behavior defined above.

2.2 Algorithms for Detecting Emergent Behavior

The point γ at which emergent behavior occurs is an example of a change point [20]. Given any distance function $d(\cdot, \cdot)$, a variety of algorithms for the quickest detection of a change point could be applied to the time series $x_i = d(M_{i-1}, M_i)$. In general, these require some assumptions about the statistical properties of the time series x_i. See for example, [20].

Detecting emergent behavior is related to novelty detection. The paper [17] also uses models to detect new or novel behavior. The idea is to sound an alarm when a model no longer matches the observed data well, whereas the idea here is to look at the distance between models that describe the data in a sequence of windows.

The paper [25] also focuses on aggregate behavior in windows but, in contrast to the work described in this paper, is interested in the efficient detection of bursts in data streams.

Our preliminary studies involving Angle have primarily used cluster models. Using a large number of clusters to identify new types of behavior has been used previously in [1] and [9]. The approach described here is similar to these approaches in two ways: First, data points that are not closely associated with clusters are considered as candidates for emergent behavior. Second, data points belonging to clusters with small cardinality are also considered candidates for emergent behavior. The approach described above is different than [1] and [9] in that we analyze large numbers of different clusters (indeed on cluster model for each window) and analyze the behavior of these different cluster models to identify periods of stability followed by behavior that may be identified as emergent.

Another approach that is sometimes used is to use a one class learning method, such as one-class support vector machine (SVM) [21], [18], [5]. With this approach, *all* data in a training period is considered to be positive and a one-class SVM is used to find the boundary of the positively labeled data. This one class SVM is then used to score data, and anything not assigned by the SVM to the positive class is considered emergent.

2.3 Cloud-based Computing Platforms for Data Mining

The most common platform for data mining is a single workstation. There are also several data mining systems that have been developed for local clusters of workstations, distributed clusters of workstations and grids [11]. More recently, data mining systems have been developed that use web services and, more generally, a service oriented architecture.

By and large, data mining systems that have been developed to date for clusters, distributed clusters and grids have assumed that the processors are the scarce resource, and hence shared. When processors become available, the data is moved to the processors, the computation is started, and results are computed and returned [8]. To simplify, this is the supercomputing (and distributed supercomputing) model. With this approach, in practice for many computations, a good portion of the time is spent transporting the data.

An alternative approach has become more common during the last few years. In this approach, the data is persistently stored and computations take place over the data when required. In this model, the data waits for the task or query. To simplify, this is the data center (and distributed data center) model. This is an example of what is sometimes called a cloud-computing model. A storage or data cloud is used to manage the persistent data. A compute cloud is layered over the storage cloud and provides computing services. Examples of data or storage clouds include Amazon's S3 [2], the Google File System [10], and the open source Hadoop system [4].

To date, work on data clouds [10, 4, 2] has assumed relatively small bandwidth between the distributed clusters containing the data. In contrast, the Sector storage cloud used for the Angle application is designed for wide area, high performance 10 Gbps networks and employs specialized protocols such as UDT [14] to utilize the available bandwidth on these networks.

The most common way to compute these days over storage clouds is to use MapReduce [7]. In Angle, the Sphere compute cloud is designed to use a style of high performance distributed computing which is a generalization of MapReduce in which both the Map and Reduce functions are replaced by user-defined functions that can be executed simply over the Sector storage cloud using the Sphere libraries.

3 The Angle Architecture

In this section, we describe the architecture of the Angle System. Angle consists of three types of nodes:

1. Sensor Nodes. The first type of nodes are sensor nodes that are attached to the commodity Internet and collect IP data.

2. Cloud Nodes. The second type of nodes are connected via a wide area high performance network and run cloud-based storage and computing services. Each sensor node is associated to a cloud node and data is passed from the sensor node to this node for processing. The nodes in the cloud can be used both for processing data locally as well as for the distributed processing of data. It is important feature of this architecture that nodes in the cloud also provide persistent storage for data collected by the associated sensor node.

3. Grid Nodes. The third type of nodes are pools of nodes that run grid services. Grid nodes are used for specialized compute intensive tasks, such as the re-analysis of previously collected data. Data is moved to grid nodes as required.

Figure 1 contains a diagram of this architecture. If a computer has two network cards, one for the commodity internet and one for a high performance network connecting the Angle Cloud Nodes, then an Angle Sensor node and an Angle Cloud Node may both share the same physical computer.

Angle Cloud Nodes run a peer-to-peer storage system called Sector that is designed to manage large remote and distributed data over wide area high performance networks [13]. Sector is based on the Chord peer-to-peer routing API [24]. All participating Cloud Nodes belong to the storage system, which uses the Chord routing API to locate files by their names. A data channel is then set up between the Sector node that a storage client connects to and the Sector storage server node that holds the desired file. To transport data efficiently over high performance networks with high bandwidth delay products, Angle uses a network transport protocol called UDT [15], for the data channel.

Angle Sensors, running on sensor nodes, collect IP packet data and buffer it. In addition, Angle Sensors run code that extract features from IP packet data belonging to a rolling window and transport it to an Angle Cloud Node running Sector.

We close this section with three remarks.

First, a simple generalization of this architecture is to replace sensor nodes with data ingestion nodes whose purpose is to ingest data into the Sector Cloud. Once ingested, the data is managed persistently by the Sector Cloud and compute services are provided as required by the Sphere Cloud. We have used used this architecture to manage, analyze and mine a variety of different e-science data sets, including datasets from biology, astronomy and earth science.

Second, in contrast to a grid computing architecture where data is generally moved to a shared pool of compute nodes as required by computing tasks [8], the Angle cloud-based infrastructure provides long-term persistence for data and exploits locality whenever possible. Sphere compute services are designed so that as much as the computation as possible is done without moving the data. This is an important benefit of this type of architecture, as compared, for example, to grid services as commonly deployed.

Third, it is important to note that some of the Angle Cloud Nodes are connected by a 10 Gbps network, which is faster than the backplane of some computers. This type of wide area distributed computer was first popularized by the OptIPuter [22], and with the proper middleware, such as Sector and UDT, enables certain data intensive computing tasks to be completed almost as efficiently as if the data were co-located in one place.

4 The Angle Data Analysis Methodology

4.1 Overview

In this section, we describe the approach used by Angle to analyze distributed IP data.

Collecting and Processing Packets. Network data is captured by independently managed network monitoring servers running IP packet capture software that we have developed. Typically we use fast servers that monitor a port-mirror of an output port of a switch or router on the edge of a network.

Angle capture software was designed to preserve privacy while capturing sufficient packet information to allow behavioral data mining. Source and destination IP addresses are hashed using a randomly generated salt, which is changed automatically by the software every time it is restarted or when the previous salt is one week old. Payload checksum is computed and stored, and the payload itself is nulled. MAC address fields along with checksum are nulled.

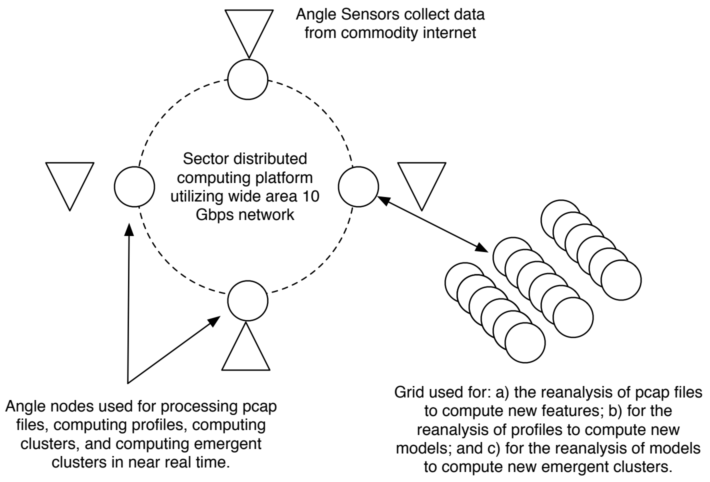

Angle Sensors collect data from commodity internet
Sector distributed computing platform utilizing wide area 10 Gbps network
Angle nodes used for processing pcap files, computing profiles, computing clusters, and computing emergent clusters in near real time.
Grid used for: a) the reanalysis of pcap files to compute new features; b) for the reanalysis of profiles to compute new models; and c) for the reanalysis of models to compute new emergent clusters.

Figure 1: This is an overview of the Angle architecture. Data is collected in near real time from sensors located on the commodity internet. Each Angle Sensor Node is paired to an Angle Cloud Node on a distributed computing platform running Sector and UDT middleware. Data analysis is done using the Angle Cloud Nodes.

Geo-location information is looked up based on IP addresses prior to their hashing and includes country, state, city and zip code (as available). The captured data is stored in a standard pcap format [19] to allow processing with standard tools and at no time are non-anonymized packets stored on disk. Furthermore, salts are non-recoverable. Uploads of pcap files are handled automatically by a data collection tool that manages a queue of files on the local disk of the sensor, spools the files, and uploads them to a near-by Angle Cloud Node. Care was required to build a tool that captures all the required data despite the variety of failures that happen from time to time at each site. Examples of failures machines going down, networks going down, disks getting full, etc. Currently a compressed pcap file is sent every ten minutes by each monitoring location.

Temporal Aggregation. We then choose a window length d and aggregate the data in pcap files into temporal windows of length d. Call these: w_1, w_2, w_3, ...

Computing Profiles. The next step is that the pcap data for each temporal window w_i is processed to produce profiles maintained by Angle, by which we mean summary data associated with a specific entity of interest as defined by one or more features associated with the entity of interest[1] In the experimental studies described below, we compute profiles associated with Source IP

[1]Currently, since we are still performing experiment studies to determine the optimal window size d, we compute feature vectors for each pcap file and then aggregate these feature vectors as required.

addresses, but other types of profiles can be easily generated using the Angle.

As an example of the features that we have used to compute profiles, some of our experimental studies have used the followed eight features: number of ports touched by the IP, number of destination IPs touched by the IP, number of packets sent, average packet size, average data size, maximum packets per destination, maximum packets per port and maximum inter packet interval. The features are normalized using data collected over a period of time and over several locations. The Angle system is designed so that features can be easily added or removed.

The pcap queuing process running on the Angle Sensor receives pcap files and queues them for processing by the Angle Cloud Nodes that compute profiles. The results are stored in files and are also tracked in a SQL database. The database stores information about each pcap file; site, time interval, number of packets, number of dropped packets, number of hosts; and stores URLs to the pcap file along with URL to the computed profile file.

Computing Models. Next each profile file is processed to determine clusters, which we view as a convenient way of summarizing the behavior at a particular sensor location over a particular time period. Currently, clusters are computed using the k-means algorithm, but other algorithms can also be easily used. More generally, from the viewpoint of Angle as a system, the k-means algorithm can be replaced by any algorithm that processes a dataset of profiles and produces a model that can be used for scoring, say as specified by PMML standard [12], [6].

More specifically, for the feature vectors associated with each window w_i, clusters are computed with centers $a_{i,1}$, $a_{i,2}$, $\ldots a_{i,k}$ and the temporal evolution of these clusters is used to identify certain clusters called emergent clusters.

Meta-analysis of Models. Next collections of models are analyzed to determine emergent behavior. Emergent clusters can be defined in various ways. As described above, our approach is to look for a period in which collections of clusters are relatively stable, and, then, after a period of such stability, to define as emergent any new cluster that arises. The details are in Section 4.2.

Real time scoring of profiles. The Angle system allows models to be browsed, visualized and selected. Once a model is selected, emergent clusters in the model are scored using the Angle scoring functions. This process is described below.

4.2 Computing Stable and Emergent Clusters

As mentioned above, a set of clusters for a selected combination of sensor locations and time window w_i are an example of a "model" M_i in the Angle application. As above, let $a_{i,\alpha}$ denote the centers (for $\alpha = 1, 2, \ldots, k$) of the cluster model M_i associated with window w_i.

For example, if the clusters are relatively stable for windows w_1, w_2, $\ldots$, $w_{\gamma-1}$, but there is a statistically significant change at w_γ, then one or more clusters from model $M_{\gamma+1}$ associated with window $w_{\gamma+1}$ can be identified.

These are the simply clusters that are responsible for the jump in the function $d(M_{\gamma-1}, M_\gamma)$ (see Equation 1). These clusters are called *emergent clusters*.

In the experimental studies reported here, we identify emergent behavior using the statistic (compare Equation 1):

$$\delta_i = \sum_{\alpha=1}^{k} \left(\min_\beta ||a_{i,\alpha} - a_{i-1,\beta}||^2 \right), \tag{2}$$

Figure 6 shows this statistic δ_i for windows of length d equals 10 minutes. Notice that it is quite choppy. On the other hand, Figure 7 shows the same statistic for windows of length d equals 1 day, where it is easy to identify three period of emergent behavior.

Every ten minutes, the cluster models for each site, as well as global cluster models for all the sites, are stored in a SQL database to allow easy access and search capability.

The clustering algorithm can be seeded with centroids from the previous run to identify and relate similar clusters over time as well as to achieve faster computation.

Both the features and cluster profiles are stored in a database as they are computed every ten minutes. This gives Angle the option to build or compute models from any time and sensor location that has been collected. Currently we have about a year's worth of data.

4.3 The Angle Score

In this section, we describe how the Angle score is computed.

1. Given TCP event data from a new pcap file, we retrieve the corresponding profile and update it using the event data.

2. We then choose a model. A model contains zero or more emergent or what we also call *red clusters*. We let μ_k denote the center of each red cluster and σ_k denote the variance of each red cluster in the model.

3. Each model has weights θ_k and λ_k associated to it. Here the weights θ_k sum to 1, while the weights λ_k control the influence of the particular cluster in the score.

4. Given the updated profile x, we assign a score ρ to it using the red clusters R using the formulas:

$$\rho(x) = \max_{k \in R} \rho_k(x)$$

$$\rho_k(x) = \theta_k \exp\left(\frac{-\lambda_k^2 ||x - \mu_K||^2}{2\sigma_k^2} \right)$$

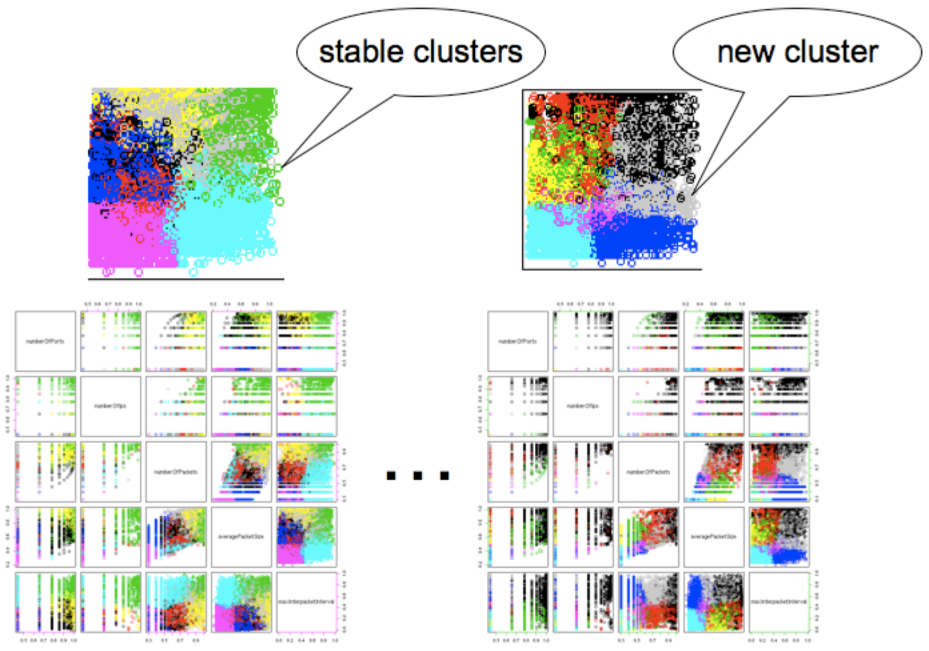
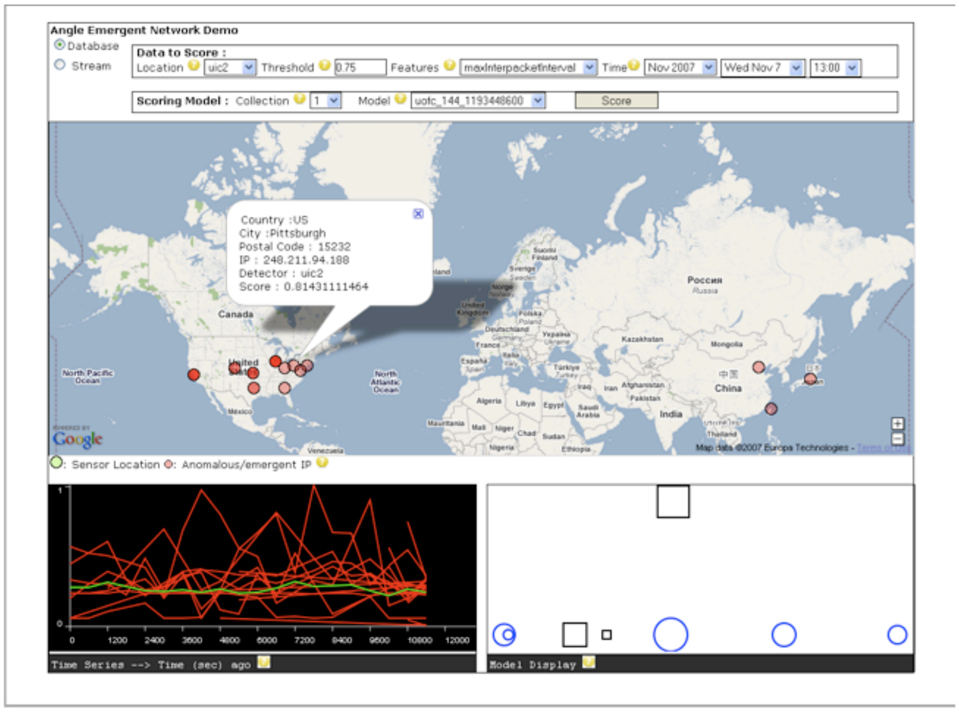


Figure 2: At the bottom of the figure are two scatter matrix plots (sploms). On the lower left is a splom from a period in which the clusters are stable. On the lower right is a splom as a new cluster emerges after the period of stability illustrated in the left splom. Above the sploms is a blow up in which the new cluster can be seen (the grey cluster).

5 Visual Analytics

The Angle visual analytics was developed to monitor anomalous flows or events across the system. Flows and events that are likely to be related to unusual behaviors are identified visually. Our visual analytics system represents statistical models in data viewers to allow easy recognition of both regularities and anomalies in data. The visual analytic components in Angle are the Map View, the Model View and the Inspector View. The Map View shows the spatial locations of IP hashes identified as anomalies.

To get a concise view of models and emergent behavior in the Model View, we utilize Multidimensional Scaling (MDS). MDS is a technique that allows the projection of points in a multidimensional space into a lower dimensional space (in our case, 2D space). The benefit of MDS is that it places similar clusters close together and dissimilar clusters far apart. The user can observe the spatial arrangement of clusters to infer relations between clusters within a model and across models. The axes of this plot are arbitrary (because the MDS model is invariant with respect to rotation and translation), so we omit other annotations



Angle Emergent Network Demo

[✓ Database] Stream

Data to Score : Location uic2 Threshold 0.75 Features maxInterpacketInterval Time Nov 2007 Wed Nov 7 13:00

Scoring Model : Collection 1 Model uofc_144_1193448600

Country :US
City :Pittsburgh
Postal Code : 15232
IP : 248.211.94.188
Detector : uic2
Score : 0.81431111464

Figure 3: This is a screen shot from Angle, that shows the visual analytics interface.

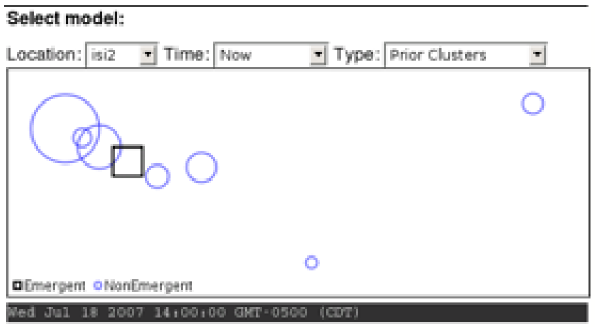
in the plot. The goal of this plot is simply to identify clusters of similar clusters and to recognize similarities among emergent clusters.

While the emergent behavior analysis and visualization helps identify anomalies, the time series Inspector View helps characterize the anomaly. In some cases certain features are more helpful for explaining a given type of behavior.

The baseline can be constructed by aggregating the information we capture from all our monitored locations or any single one of them. This helps the analysis of both global and local structure and will expose suspicious behavior that would not otherwise be obvious.

Angle takes advantage of new web technologies to provide a responsive and an intuitive interface for navigating large amounts of network packet data. The user browsing the web interface can select many options that regulate how data should be analyzed. These options are sent to the Angle server using AJAX (Asynchronous Javascript And XML). This technology lets a web page make a request for information in the background without disrupting the user's browsing experience. The server receiving these requests performs an analysis using the user's parameters. Once the analysis is complete, the results are returned to client's web browser and appropriate views are updated.

To use the interface, the first task for the user is to select a model to de-



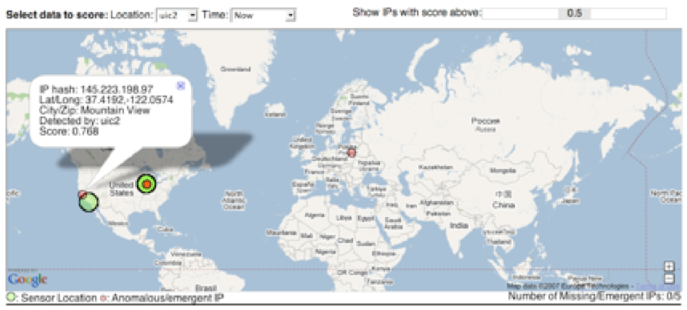
Select model:

Location: isi2 Time: Now Type: Prior Clusters

Wed Jul 18 2007 14:00:00 GMT-0500 (CDT)

Figure 4: This is a view of one of the Angle models. Squares represent emergent clusters, while circles represent regular clusters. The Angle system currently has access to over 100,000 separate models that can be selected and used to look for anomalies and emergent behavior in near real time or from historical data.

termine emergent behavior. This is accomplished using the lower-right Model View panel of the interface. Next, the user selects the type of comparison to occur (Prior clusters, Common clusters or Daily average) and the data capture location for Angle to use for signaling emergent behavior. Data sets captured at any interval may be specified to search for emergent clusters.



Select data to score: Location: uic2 Time: Now Show IPs with score above: 0.5

IP hash: 145.223.198.97
Lat/Long: 37.4192,-122.0574
City/Zip: Mountain View
Detected by: uic2
Score: 0.768

Greenland
United States
North Atlantic Ocean
China
India
Japan
Google
○: Sensor Location ●: Anomalous/emergent IP
Number of Missing/Emergent IPs: 0/5

Figure 5: In the map view above of emergent behavior, once a model is selected, emergent or anomalous IPs are identified in pink.

Once emergent behavior is located, a user can determine which IP address hashes (IPhash) are involved in the anomalous activity. The upper Map View panel, showing a map of the world, is utilized to place anomalous IPhashes onto the world map. The angle score slider at the upper-right corner of the panel controls the threshold of Angle scores to be visualized. The user can experiment with this slider, starting it near 0, and moving it up to filter visualization of low-scoring IPhashes. Clicking on red circular icons reveals information about the selected IPhash.

6 Experimental Studies

6.1 Angle Data Collection

Angle Sensors are currently installed at four locations: the University of Illinois at Chicago, the University of Chicago, Argonne National Laboratory and the ISI/University of Southern California.

Each day, Angle processes approximately 575 pcap files totaling approximately 7.6GB and 97 million packets. To date, we have collected approximately 140,000 pcap files which we compress to about 1 TB. For each pcap file, we aggregate all the packet data by source IP (or other specified entity), compute features, and then cluster the resulting points in feature space. This produces approximately 140,000 cluster models which we store in a database for subsequent analysis.

6.2 Experimental Studies - Detecting Emergent Behavior

We did a series of experiments to determine an appropriate window size for computing summary models. If the window is to short, there is not enough structure to the data to be useful. On the other hand, if the window is too long, interesting emergent behavior is lost.

We determined an appropriate window size empirically by varying the window size d from 10 minutes to 24 hours. For each fixed window size d, we divided the data into windows w_1, w_2, w_3, ... each of size d and computed clusters as described above for the data in each window w_i. Let $a_{i,\alpha}$ be the centers of the k clusters (as $\alpha = 1, \ldots, k$) computed for the data in window w_i. We then computed the following statistic (compare Equation 2):

$$\delta_i = \sum_{\alpha=1}^{k} \left(\min_{\beta} ||a_{i,\alpha} - a_{i-1,\beta}||^2 \right).$$

Figure 6 shows this graph for windows of length $d = 10$ minutes. Notice that the statistic δ_i is not well enough behaved to draw any conclusions. On the the other hand, Figure 7 shows the same statistic for windows of length $d = 1$ day. We manually investigated the packet behavior when the statistic δ_i crossed a threshold, as illustrated in Figure 7.

As a result of these manual investigations, we identified a variety of suspicious behavior, including port scans, email spam, and the probing of our network for vulnerabilities associated with a known Trojan attack.

6.3 Scalability of the Cloud Infrastructure

One of the ways this project is different than previous work in high performance, distributed data mining is the use of a cloud computing platform instead of web-service based or grid-based computing platform. We also performed several experimental studies to understand the scalability of the Sector/Sphere cloud. As

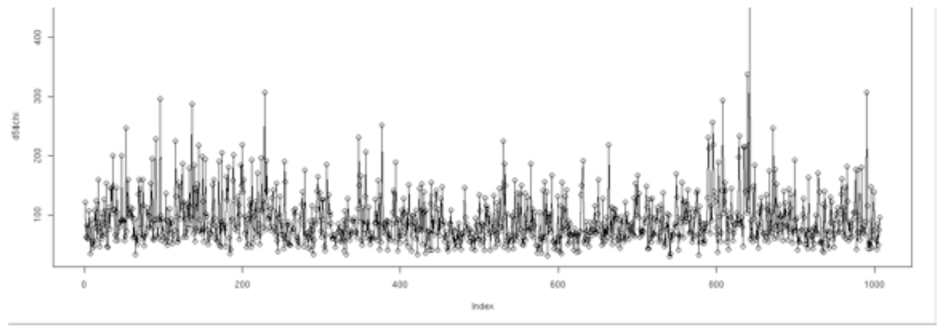


Figure 6: The graph above shows how the cluster centers move from one ten minute window to another as measured by the statistic δ_i.

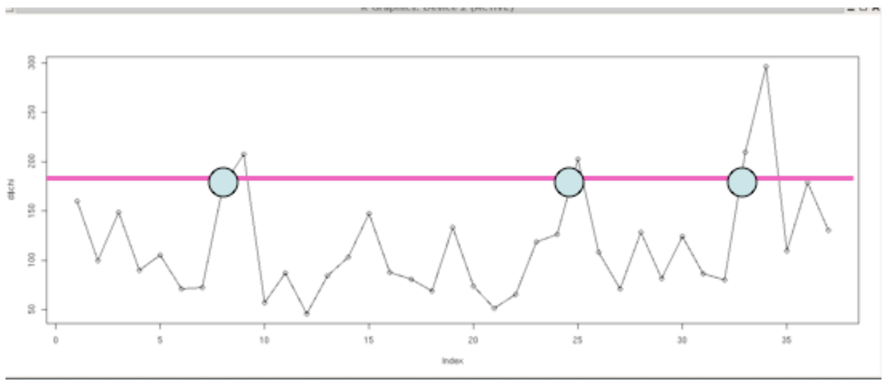


Figure 7: The graph above shows how the cluster centers move from one 1-day window to another as measured by the statistic δ_i. We manually investigated behavior associated with jumps of δ_i after periods of stability. In particular, we investigated the behavior associated with the three jumps of δ_i indicated.

illustrated in Table 1 we computed cluster models from the distributed pcap files ranging in size from 500 points to 100,000,000 points using Sector/Sphere. As the table indicates the time required for this computation scales in an acceptable manner.

We are continuing to explore the use of Sector/Sphere for computing cluster and related statistical models.

7 Conclusion

7.1 Lessons Learned

During the past year or so of the project as we developed a preliminary prototype of the system, we have learned several lessons.

Number records	Time	Average Error
500	1.9 s	0.021
1000	4.2 s	0.021
1,000,000	85 min	0.028
100,000,000	178 hours	0.036

Table 1: The time spent clustering using Sphere scales as the number of records increases is illustrated in the table above from 500 records to 100,000,000 records.

Lesson 1. The current grid-based distributed computing infrastructure is well-suited for sharing cycles, but less suited for making discoveries using distributed data. With the Angle architecture, a persistent distributed infrastructure is used to store and manage the distributed data over a wide area high performance network (Sector). With Sector, computations can be performed locally use Cloud Nodes or data may be moved using specialized protocols (UDT) when data is required to be co-located for computation.

Lesson 2. Much of the power of Angle derives from the ability to work with many different analytic models (there are currently over 100,000 different models available for identifying emergent behavior). It has been recognized for some time the necessity of developing algorithms that scale as the number of records increases and as the number of dimensions increases. Applications such as Angle also show the desirability of developing algorithms and infrastructure that scale as the number of analytic models increase.

Lesson 3. The Angle system is designed in a flexible way so that properly anonymized data can be shared, as well as models that contain no identifying information. Note that models built locally can be built on a richer set of features than models built on shared data. Even with this design, a great deal of effort was spent on developing policies and procedures so that anonymized data could be shared between organizations. This process substantially slowed the pace at which the project could proceed, which is an important lesson to keep in mind when designing projects that require sharing data.

7.2 Summary

In this paper, we have described the Angle Project that identifies emergent behavior in distributed IP data. Angle is noteworthy for several reasons.

First, Angle employs a distributed, cloud-based computing architecture called Sector that is designed to provide persistent storage for large distributed data sets. Layered over Sector, is a compute cloud called Sphere that moves the analytic computation to the data whenever possible. In contrast, many grid-based distributed computing systems are designed to move data to pools of compute servers when they become available.

Second, the Angle middleware supports computing with a large number of different analytic models, an approach, which for some applications, can lead to discoveries that are quite difficult when just one or a few analytic models are used, as is usually done. Angle also persists these models into a model repository, which simplifies their analysis, management, and deployment.

Third, Angle introduces statistical algorithm for identifying new types of behavior that have not been previously seen. This is an example of what is sometimes called emergent behavior. Angle also employs visual analytic techniques for identifying emergent behavior.

A fruitful area for research is to develop new algorithms for identifying emergent behavior. It is important to note that by emergent we do not simply mean anomalous behavior, but more specifically emergent behavior as defined in Section 2.1. The reasons for the importance of this area is very simple. First, there is very little work in the area. Second, the impact of many data mining problems is often gated by the number of analysts who can look at alerts generated by data mining systems in detail. Typically these analysts have to balance two types of investigations: investigations that look at instances related to *known* behavior of interest and those that look at instances of interesting behavior that is *unknown* in that it has not been seen previously. Algorithms to detect emergent behavior is targeted at the later type of behavior.

Another fruitful area of research is to develop high performance data mining systems that use cloud-based architectures. Again the reason is simple. Work to date in high performance data mining has focused primarily on data mining using high performance clusters and grids. With both of these systems, the *cycles usually wait for the data.* In contrast, with a cloud based system, a storage cloud can provide long term persistent storage for data, and a compute cloud can be layered over the storage cloud and designed to exploit data locality whenever possible. This leads in practice to good *end-to-end performance*, that includes not just the time required to compute a model once the data has reached the cluster, but the entire time required, including the time to transport the data, compute the model, and return the results.

Acknowledgments

This work was supported in part by the National Science Foundation through grants SCI-0430781, CNS-0420847, and ACI-0325013.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81—92, 2003.

[2] Amazon. Amazon simple storage service (amazon s3). www.amazon.com/s3.

[3] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley & Sons, New York, NY, third edition, 1994.

[4] D. Borthaku. The hadoop distributed file system: Architecture and design. retrieved from lucene.apache.org/hadoop, 2007.

[5] K. Crammer and G. Chechik. A needle in a haystack: local one-class optimization. In *ICML*, 2004.

[6] Data Mining Group. Predictive Model Markup Language (PMML), version 3.2. www.dmg.org, 2008.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[8] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, California, 2004.

[9] M. M. Gaber and P. S. Yu. Detection and classification of changes in evolving data streams. *International Journal of Information Technology and Decision Making*, 5(4):659—670, 2006.

[10] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *SOSP*, 2003.

[11] R. L. Grossman. Standards, services and platforms for data mining: A quick overview. In *Proceedings of the 2003 KDD Worskhop on Data Mining Standards, Services and Platforms (DM-SSP 03)*, 2003.

[12] R. L. Grossman, S. Bailey, A. Ramu, B. Malhi, P. Hallstrom, I. Pulleyn, and X. Qin. The management and mining of multiple predictive models using the predictive model markup language (pmml). *Information and Software Technology*, 41:589–595, 1999.

[13] R. L. Grossman, Y. Gu, D. Handley, M. Sabala, J. Mambretti, A. Szalay, A. Thakar, K. Kumazoe, O. Yuji, M. Lee, Y. Kwon, and W. Seok. Data mining middleware for wide area high performance networks. *Journal of Future Generation Computer Systems (FGCS)*, 22(8):940—948, 2006.

[14] R. L. Grossman, Y. Gu, X. Hong, A. Antony, J. Blom, F. Dijkstra, and C. de Laat. Teraflows over gigabit wans with udt. *Journal of Future Computer Systems*, 21(4), 2005.

[15] Y. Gu and R. L. Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, page to appear, 2007.

[16] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava, and V. Kumar. Evaluation of outlier detection schemes for detecting network intrusions. In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, 2003.

[17] J. Ma and S. Perkins. Online novelty detection on temporal sequences. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618, New York, NY, USA, 2003. ACM.

[18] L. Manevitz and M. Yousef. One class svms for document classification. *Journal of Machine Learning Research*, 2, 2001.

[19] Programming with pcap. www.tcpdump.org/pcap.htm, retrieved on September 1, 2007.

[20] H. V. Poor and O. Hadjiliadis. *Quickest Detection*. Cambridge University Press, 2008.

[21] B. Scholkopf, J. Platt, J. Shawe, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research, 1999.

[22] L. L. Smarr, A. A. Chien, T. DeFanti, J. Leigh, and P. M. Papadopoulos. The optiputer. *Commun. ACM*, 46(11):58–67, 2003.

[23] Snort. www.snort.com, retrieved on September 1, 2007.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM'01*, pages 149—160, San Diego, CA, 2001.

[25] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345, New York, NY, USA, 2003. ACM.