Detection of Denial-of-QoS Attacks Based On χ^2 Statistic And EWMA Control Charts

Vinay A. Mahadik[*], Xiaoyong Wu[†] and Douglas S. Reeves[‡]

January 31, 2002

Abstract

In this paper, we describe a method of detecting denial of Quality of Service attacks on DiffServ networks. Our approach focusses on real time and quick detection, scalability to large networks, and a negligible false alarm generation rate. Sensors sample QoS parameters like bit rate, packet dropping rate, and jitter of specific Virtual Leased Line (VLL) flows at predefined strategic points in their paths. We detect anomalies in sampled network flow statistics using the EWMA Control Chart test for the highly stationary measures and for the rest adapt SRI's χ^2 statistic based NIDES approach. Our implementation shows that the method has a 100% detection rate for attacks above its threshold level - those attacks that produce statistically significant QoS degradation. The detection time is low and less than about 15 minutes. The maximum inherent false alarm generation rate for both the tests and any of the monitored measures combined is of the order of 1 false alarm in 1000 valid status alerts of either *normal* or *under attack*. We believe that given the results of the tests on our implementation of the attacks and the detection system, the method is a strong candidate for QoS intrusion detection for a low-cost commercial deployment.

[*]Vinay A. Mahadik is pursuing Master of Science in Computer Networking at the NC State University, Raleigh. Email : vamahadi@unity.ncsu.edu

[†]Xiaoyong Wu is with the Advanced Networking Research Group, MCNC, Research Triangle Park. Email : xwu@anr.mcnc.org

[‡]Douglas S. Reeves is with the Department of Computer Science, NC State University, Raleigh. Email : reeves@unity.ncsu.edu

1 Introduction

As quality of service (QoS) capabilities are added to the Internet, our nation's business and research infrastructure will increasingly depend on their fault tolerance and survivability. Current frameworks and protocols, such as Resource ReSerVation Protocol (RSVP)[10] / Integrated Services (IntServ) [36] and Differentiated Services (DiffServ) [22, 4], that provide quality of service to networks are vulnerable to attacks of abuse and denial[4, 34, 10]. To date, no public reports have been made of any denial of QoS attack incidents. However, this absence of attacks on QoS is an indication of the lack of a large scale deployment of QoS networks on the Internet. Once, QoS deployments become commonplace, the potential for such attacks to maximize damages will increase and so would an adversary's malicious intent behind launching them. It is necessary both to make the mechanisms that provide QoS to networks intrusion tolerant and detect any attacks on them as efficiently as possible.

This work describes a real-time, scalable denial of QoS attack detection system with a low false alarm generation rate that can be deployed to protect DiffServ based QoS domains from potential attacks. We believe our detection system, named ArQoS, is the first and the only public research attempt at detecting intrusions on network QoS.

We describe the context of the problem in section 2 as a motivation for this work. Section 3 describes the simple, though distributed, framework that is used to monitor a DiffServ network. Sections 4 and 5 detail the mathematical background of the tests used for anomaly detection in individual flows. Sections 6, 7, and 8 justify the validity of the approach based on detection results for actual attacks on a test network. Section 9 briefly describes an important future direction we are exploring for going beyond just detection of attacks and trying to estimate the type, point, and intensity of attack using a Bayesian distributed event correlation approach.

2 Background

In the DiffServ architecture[4, 22], the entering traffic is classified and possibly conditioned at the boundary nodes, and assigned to different behavior aggregates. Each behavior aggregate is mapped into a single DiffServ CodePoint (DSCP), a field in the IP packet header, through a one-one or many-one mapping. At the interior nodes, packets are forwarded according to the per-hop behavior associated with their DSCPs. The architecture achieves scalability by aggregating the traffic classification state into the DSCP. Thus, the DiffServ nodes do not have to maintain per flow states as is the case with, say, IntServ. DiffServ are extended across a domain boundary by establishing a Service Level Agreement (SLA) between an upstream domain and a downstream domain, and a derived Traffic Conditioning Agreement (TCA) which may specify packet classification and re-marking rules and may also specify traffic profiles and actions to traffic streams which are in- or out-of-profile.

Since the DiffServ architecture is based on the Internet Protocols, in general, the DSCPs are not encrypted. The vulnerability then is that the architecture leaves scope for attackers who can modify or use these service class code points to effect either a denial or a theft of QoS which is an expensive and critical network resource. With these attacks and other non QoS-specific ones (that do not make use of DSCPs), there is the possibility of disrupting the entire QoS provisioning infrastructure of a company, or a nation.

Following are the attacks we have identified, all of which, we believe, can be detected or defended against by a DiffServ network in combination with our detection system. A malicious external host could flood a boundary router congesting it. A DiffServ core router itself can be compromised. It can then be made to remark, drop, delay QoS flows. It could also flood the network with extraneous traffic.

We focus only on attacks and anomalies that affect the QoS parameters typically specified in SLAs such as packet dropping rates, bit rates, end-to-end one-way or two-way delays, and jitter. For example, an Expedited Forwarding (EF)[13, 16] service class SLA typically specifies low latency/delay and low jitter. An EF flow sensor then monitors attacks on delay and jitter only.

Statistical Anomaly Detection is based on the hypothesis that a system being monitored for intrusions will behave anomalously when attacked and that these anomalies can be detected. The approach was first introduced in[7]. [9] compares it with other intrusion detection approaches. The statistical anomaly detection algorithm we use, henceforth referred to as STAT, is based on SRI's NIDES[17] algorithm. Statistical anomaly detection is, in general, well suited for monitoring subjects that have a sharply defined and restricted behavior with a substantial difference between intrusive and normal behavior. A QoS flow is an ideal candidate for such a subject and justifies the use of anomaly detection for monitoring it.

STAT uses the χ^2 goodness-of-fit test[14, 24] that makes no assumptions about the distribution of the base process. Further, by using an exponentially weighted estimation for linear equi-width bins' frequencies, STAT nullifies the statistical significance of the correlation of the individual QoS parameter measures or counts. In other words, the binning categorizes events that become fairly uncorrelated once estimated over a sufficiently large sample size and over a period of time. The choice of exponentially weighted estimation also reduces storage and processing requirements as against when using equal weight and fixed width moving windows.

The Exponentially Weighted Moving Average (EWMA) Control Chart[27] is a popular Statistical Process Control technique used in the manufacturing industry for product quality control. In this, a probability distribution curve for a statistic gives the rarity of occurrence of a particular instance of it. Outcomes that are rarer than a certain predefined threshold level are considered as anomalies. As will be explained here, the STAT algorithm is not suitable for a highly stationary measure, and it is for only such measures that we use the EWMA Control Charts to detect intrusions.

The statistical anomaly detection approach, as used in both the above techniques, has faced severe criticism from the security community due mostly to its inherent higher false alarm generation rate than the rule or signature-based detection approaches like

[15, 33, 12]. [20] explores the problems faced by the statistical anomaly detection approach in general. However, the anomaly detection approach is preferred for detection of new or uninvestigated attacks with undefined attack signatures - as is the case with the attacks on QoS networks.

In view of this, we also use a simple rule based detection technique, henceforth referred to as RULE, that works as the complement of the above two anomaly detection techniques, to quickly detect some discompliance of SLA or TCA or some known intrusions against the DiffServ network architecture. Currently, there are few known attacks against the QoS framework. The detection rules, that is, the attack signatures we use are derived from the DiffServ architecture documents [22, 4] and by heuristics [34].

3 Attack and Detection System Framework

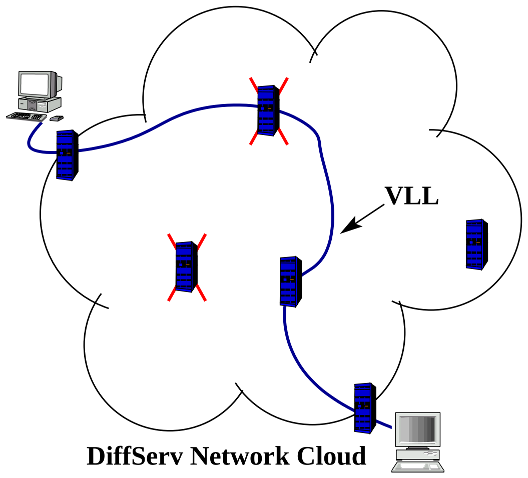

Figure 1: A typical DiffServ cloud between QoS customer networks

Figure 1 shows two QoS customer networks using a Virtual Leased Line (VLL) through a typical DiffServ network cloud between them. The VLLs can be achieved through traffic engineering on tag-switched networks like MultiProtocol Label Switching (MPLS) networks [28]. The rationale behind VLLs is that most QoS parameters are highly dependent on the packet routes and the nodes involved. A dynamic route reconfiguration affects these parameters unpredictably and hence can not guarantee QoS. VLLs also assist in predicting the compromised route, in using IP Security Protocol (IPSec) for data integrity and/or confidentiality, and also in *stealth probing* the network as explained later in this section.

We assume that the boundary routers are truly secure and have not been compromised. We do not trust any of the interior routers. We believe this is a reasonable requirement. Instead of expending security efforts on the entire domain, we need focus our personnel and other host-based intrusion detection systems on the boundary routers only which are typically much less in number than the core and boundary routers combined. The Figure 1 indicates interior or core routers as well as ingress/egress boundary routers. The (red) crossed ones have been compromised by the attacker and are being used to launch denial or theft of QoS attacks. The uncrossed ones are reliable DiffServ routers.

We use the following components for anomaly detection :

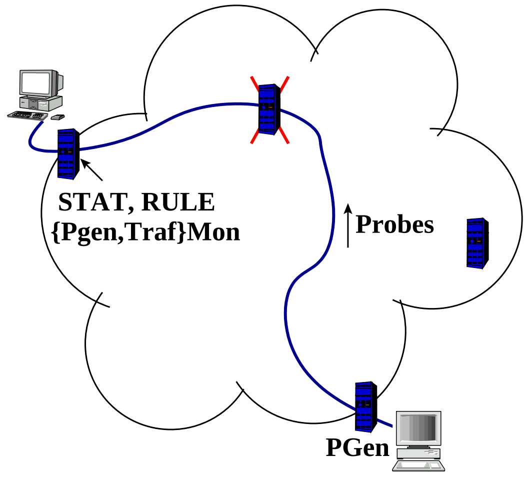

Figure 2: How the ArQoS components are placed about a VLL

- STAT : The **Stat**istical Anomaly Detection engine, accepts inputs from sensors on local or remote networks. STAT modules are placed on one of the boundary routers of a VLL being monitored. If that boundary router is shared by more than one VLL, the STAT process can be shared between them too.
- RULE : The **Rule**-based Detection engine, accepts inputs from local or remote sensors. It

is placed on either of the boundary routers of a VLL. The attack signatures we use include packet dropping rate being above a specified threshold rate for the Assured Forwarding (AF) probing traffic, EF packets being remarked to BE although the SLA/TCA specify dropping EF packets that are delayed significantly, etc. The rules are heuristically derived, and a function of the SLA for the VLL. We recommend the use of RULE mostly as a bounds and conformity checker for the SLA.

- DSMon : The **D**iff**S**erv aggregated-flows **mon**itor monitors BE, EF or AF aggregate flows (as against a specific VLL micro-flow that TrafMon monitors) on a core router. In order that DSMon finds that an upperbound exists on the burstiness of the aggregate flow it monitors, we suggest placing the monitor only on routers that are congested or where the incident links saturate the outgoing one(s). Due to the random nature of the flow it monitors, it is found to generate an unacceptably high false positive rate. Hence, we do not use the inputs from DSMon for any of our tests in this work. We expect to use it to get an estimate of the general *health* of the DiffServ network and for the Event-Correlation work to be taken up in the future.

- PGen : The stealth **P**robe Packet **Gen**eration module is placed on the boundary routers of the VLL to be monitored. PGen periodically makes a copy of a random packet from the VLL inward-flow, marks the new copy with a stealth cryptographic hash and reinjects both the packets into the VLL. The injection of probes is done in a soft real time manner at a low specified rate. The idea is to generate a highly stationary flow with a fixed and known packet rate and zero jitter before entering the DiffServ cloud. The QoS parameters that are monitored for this flow then are highly deterministic. Deviations from these stationary means are then a function of the DiffServ cloud. PGen also appends the hash with an encrypted sequence number for the probe packet that is necessary for jitter calculation at the egress router.

  For our tests, we have used unencrypted hashs and sequence numbers on the probes. In practice, we would use a secure hash to tag the packets based on a shared secret key between the boundary routers of the VLL and certain field(s) in the IP packets. The idea being that, then, the attacker would not be able to, with a significant level of certainty, distinguish between probe packets and the normal data packets. The use of IPSec with data encryption over the VLL clearly helps this requirement. The probes are terminated at the egress router. Interior routers simply forward packets and do not expend any processing effort on an encryption or decryption process.

  Then, all attacks on the QoS of the VLL are statistically spread over both the normal data and the probe packets. For jitter and delay parameters, we need to monitor only the low volume probe flow. In this respect, the secrecy of this mark should be considered as a single point of failure for ArQoS security for that VLL. The rate of probe generation can be considered negligible compared to the rate of the VLL flow. Typical rates are 1 or 2 probe packets in a second. Large rates would have flooded the network unnecessarily while lower rates would require unacceptably longer time windows over which to generate the short term profiles, that is, a sluggish detection rate.

- TrafMon & PgenMon : TrafMon monitors arbitrary flows non-invasively and independently of the probes. PgenMon monitors only the probes sent by Pgen at the other end of the VLL. Both are placed on one of the boundary routers of VLLs that they monitor. PgenMon is on the opposite end of the VLL as the corresponding PGen. We use TrafMon to monitor only the bit rates of (approximately) constant bit rate (CBR) flows. We use PgenMon to monitor the jitter and the packet rates of the probe flow inside a VLL.

4 STAT - A Mathematical Background

4.1 The χ^2 Test

For a random variable, let E_1, E_2, ..., E_k represent some k arbitrarily defined, mutually exclusive and exhaustive events associated with its outcome and let S be the sample space. Let p_1, p_2, ..., p_k be the long-term or *expected* probabilities associated with these events. That is, in N Bernoulli trials, we expect the events to occur Np_1, Np_2, ..., Np_k times as N becomes infinitely large. For sufficiently large N, let Y_1, Y_2, ..., Y_k be the actual number of outcomes of these events in a certain experiment. Pearson[24, 14] has proved that the random variable Q defined as

$$Q = \sum_{i=1}^{k} \frac{(Y_i - Np_i)^2}{Np_i} \tag{1}$$

has (approximately) a χ^2 distributed Probability Density Function with k-1 degrees of freedom if any k-1 of the events are independent of each other. The approximation is good if $Np_i \geq 5, \forall i$ so that no one event has so large a Q_i component that it dominates over the rest of the smaller ones. Rare events that do not satisfy this criterion individually may be combined so that their union may.

The p_i are the *long term* probability distribution of the events. Let $p'_i = \frac{Y_i}{N}$ be the *short term* (meaning the sample size, N, is relatively small) probability distribution of the events.

Our test hypothesis, H_0, is that the actual short term distribution is the same as the expected long term distribution of the events. Its complement, H_1, is that the short term distribution differs from the long term one for at least one event. That is,

$$H_0 : p'_i = p_i, \forall i = 1, 2, ..., k \quad \& \quad H_1 : \exists i, p'_i \neq p_i$$

Since, even intuitively, Q is a measure of the anomalous difference between the actual and the expected distributions, we expect low Q values to favor H_0 and high Q values to favor H_1.

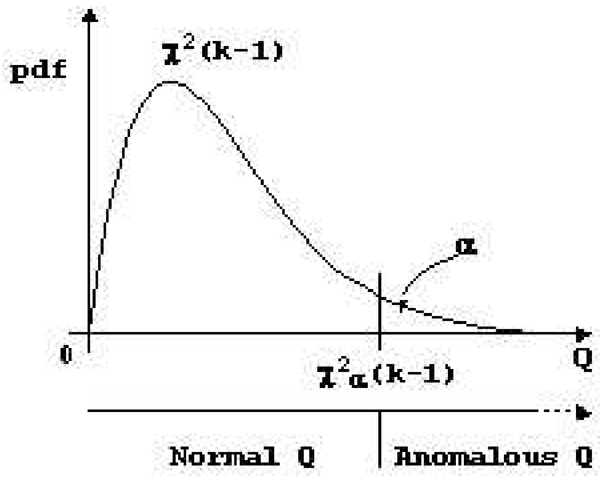

Figure 3: ArQoS χ^2 Statistical Inference Test

As Figure 3 indicates, we define α as the desired significance level of the hypothesis test. Also, let χ^2_α(k-1) be the corresponding value of Q, that is, such that, $Prob(Q \geq \chi^2_\alpha(k-1)) = \alpha$. For an instance of Q, say q_{k-1}, we reject the hypothesis H_0 and accept H_1 if $q_{k-1} \geq \chi^2_\alpha(k-1)$.

As will soon be evident, α also serves as the False Positive Rate (FPR) specification - fraction of the total (Normal or Anomaly) alarms generated that are False Positives. It is typically set at 0.01 for yellow alarms and 0.001 for red alarms, meaning a FPR of 1 red alarm in 1000 normal, yellow or red alarms. In other words, with an inherent chance α of error in our decision, we can flag an experiment with $q_{k-1} \geq \chi^2_\alpha(k-1)$ as an anomaly. α can not be arbitrarily low since it strikes a compromise between detection and false positive rates.

4.2 χ^2 Test applied to STAT

In the context of STAT, the random variable is the measured count of a QoS parameter of the network flow(s) being monitored. The parameters we use presently are

- Byte Count is the number of bytes of the network flow data that have flowed in a fixed given time.
- Packet Count is the number of packets of the network flow data that have flowed in a fixed given time.
- Jitter is defined as the average difference between the interarrival times of consecutive probe packet pairs arriving at the egress router of a VLL. The calculation makes use of the sequence numbers that the probe packets carry to identify the order in which the packets were actually injected into the system.

STAT can be extended to use other measures that matter to the SLA. STAT is trained with the maxima and the minima of every parameter. This range is linearly divided into several equal-width bins into which the count outcomes may fall. These are the events used by the χ^2 Test. The number of bins is arbitrarily set at 32. Hence, any 31 events out of the total 32 events are independent. Thus we expect a χ^2 Distribution for Q the maximum degrees of freedom of which are 31.

Sections 4.3 and 4.4 detail how the long term and the short term distributions of the counts are obtained.

STAT algorithm defines a variable S to *normalize* the values of Q from different measures and/or flows so that they may be compared against each other for alarm intensities. This is done such that S has a half-normal probability distribution satisfying

$$Prob(S\epsilon[s,\infty)) = \frac{Prob(Q > q)}{2}$$

That is,

$$S = \Phi^{-1}(1 - \frac{Prob(Q > q)}{2}) \tag{2}$$

where Φ is the cumulative probability distribution of the Normal $N(0, 1)$ variable. This ensures that S varies from 0 to ∞, but we limit the maximum value of S to 4 since that value is rare enough to be considered a certain anomaly.

4.3 Obtaining Long Term Profile p_i

To establish the long term distribution for a measure, STAT goes through the following sequence of phases

1. **Count Training :** where STAT learns the count maxima, minima, mean and the standard deviation over a specified, sufficiently large, number of training measures. The maxima and the minima help in the linear equi-width binning used by the χ^2 Test.

2. **Long Term Training :** At the end of a set number of long-term training measures, STAT initializes the p_i and the p'_i equal simply to the bin frequencies as

$$p_i = p'_i = \frac{CurrentCount_i}{CurrentTotal} \tag{3}$$

   where $CurrentCount_i$ is the current number of i^{th}-bin events and $CurrentTotal$ is the current total number of events from all the bins.

   These are the initial long term and short term distributions.

3. **Normal/Update Phase :** After every preset Long Term Update Period, STAT learns about any changes to the long term distribution itself. By using exponentially weighted moving averages, the older components in the average get exponentially decreasing significance with every update. This is important since the flow parameters, though typically wide-sense stationary in a QoS network, may shift in mean over the long update period and this shift needs to be accounted for to avoid false positives. This is done as

$$LastWTotal = WTotal \tag{4}$$

$$WTotal = b \times LastWTotal + CurrentTotal \tag{5}$$

   and

$$p_i = \frac{b \times p_i \times LastWTotal + CurrentCount_i}{WTotal} \tag{6}$$

   Here, b is defined as the weight decay rate at which, at the end of a preset $LTPeriod$ period (long term profiling period) of time, the present estimates of $CurrentCount_i$ and $CurrentTotal$ have just a 10% weight. Typically, $LTPeriod$ is about 4 hours. Clearly, equation (6) then gives the required long term frequencies p_i as the ratio of the EWMA of the current bin count to the EWMA of the current total count. Further, only the most recent $LTPeriod$ period has a (90%) significance to the long term profile.

4.4 Obtaining Short Term Profile p'_i

Unlike the long term frequencies p_i, the short term ones are updated with every count as

$$p'_i = r \times p'_i + 1 - r \tag{7}$$

and

$$p'_j = r \times p'_j, \forall j \neq i \tag{8}$$

where i is the bin in which the present count falls. r is the decay rate, defined in STAT as the rate at which the current short term bin frequency estimate has a weight of 1% at the end of a $STPeriod$ period (short term estimation period) of time. $STPeriod$ is about 15 minutes. The $1 - r$ in the equation (7) satisfies $\sum p'_i = 1$ and also serves as the weight for the present *count* of 1 for the bin to which the present measure belongs. Further, only the most recent $STPeriod$ period has a (99%) significance to the short term profile.

4.5 Generate Q Distribution

As each measure is received, STAT calculates the Q value associated with the count as per the equation (1). STAT maintains a *long term Q* distribution curve, against which each new *short term Q* is compared to calculate the anomaly score S. The long term Q distribution generation is very similar to the long term frequency p_i distribution generation. The procedure parallels the one in section 4.3, where equations (3) through (6) now become -

at the end of a specified long term training Q measures, we set

$$Q_i = \frac{CurrentQCount_i}{CurrentQTotal} \tag{9}$$

where we arbitrarily use a 32 bin equi-width partitioning.

Further, in the update/normal phase, after every long term update Q measures, we use

$$LastWQTotal = WQTotal \tag{10}$$

$$WQTotal = b \times LastWQTotal + CurrentQTotal \tag{11}$$

and

$$Q_i = \frac{b \times Q_i \times LastWQTotal + CurrentQCount_i}{WQTotal} \tag{12}$$

to get the Q distribution.

4.6 Calculate Anomaly Score and Alert Level

Equation (2) gives the degree of anomaly associated with a new or short term generated value of Q, say q. This S score also determines the level of alert based on the following categories -

- Red Alarm Level - S falls in a range that corresponds to a Q tail probability of $\alpha = 0.001$. This means that Q's and S's chances of having such high a value are 1 in a 1000.
- Yellow Alarm Level - S falls in a range that corresponds to a Q tail probability of $\alpha = 0.01$ and outside the Red Alarm Level range. Then, Q's and S's chances of falling in this tail area are roughly 1 in a 100.
- Normal *Alarm* Level - For all values of S below the Yellow Alarm Level, we generate a Normal Alarm that signifies an absence of attacks.

Certain parameters such as the drop rate associated with an AF probe flow in an uncongested network may be highly stationary. For such parameters, the long term frequency distribution p_i does not satisfy the $Np_i > 5$ rule for all expect possibly one or two bins. The χ^2 test does not gives good results for such parameters. Hence, as a rule, whenever the number of Q_i components calculated satisfying the above thumb-rule falls below 3, we use the EWMA Control Chart mode for finding the anomaly score. When it exceeds or equals 3, we switch back to the χ^2 mode.

If after an attack is detected, it is not eliminated within the *LTPeriod* period of time, both STAT and the EWMA chart *learn* this anomalous behavior as normal and eventually stop generating the attack alerts.

5 EWMA Charts - A Mathematical Background

5.1 Statistical Process/Quality Control

Shewhart[30, 31, 32] first suggested the use of Control Charts in the manufacturing industry to determine whether a manufacturing process or the quality of a manufactured product is in statistical control. The EWMA Control Charts, an extension of the Shewhart Control Charts, are due to Roberts[27].

In EWMA Control Charts, the general idea is to plot the statistic ϵ_k given as

$$\epsilon_k = \lambda \overline{x}_k + (1-\lambda)\epsilon_{k-1} \tag{13}$$

Here, a subgroup of samples consists of n independent readings of the parameter of interest (from n different manufactured products at any instant of time). Samples are taken at regular intervals in time. Then, $\overline{x}_k$ is the average of the k^{th} subgroup sample. λ is the weight given to this subgroup average. ϵ_k is the estimator for this subgroup average at k^{th} sample. The iterations begin with $\epsilon_0 = \overline{\overline{x}}$. $\overline{\overline{x}}$ is the (estimate of the) actual process mean, while $\hat{\sigma}$ is the (estimate of the) actual process standard deviation.

If $\lambda \geq 0.02$, as is typical, once $k \geq 5$, the process Control Limits can be defined as

$$\overline{\overline{x}} \pm 3\frac{\hat{\sigma}}{\sqrt{n}}\sqrt{\frac{\lambda}{2-\lambda}} \tag{14}$$

Let UCL denote the upper control limit and LCL denote the lower control limit, STE denote the short term estimate and LTE denote the long term estimate. For a base process that is Normally distributed, the STE estimator then falls outside the control limits less than 1% of times.

Ignoring the first few estimates then, when the iterations stabilize, we consider any estimate outside the control limits as an anomaly. The cause of the anomaly is then looked for and if possible eliminated. If the fault(s) is not corrected, the control chart eventually accepts it as an internal agent.

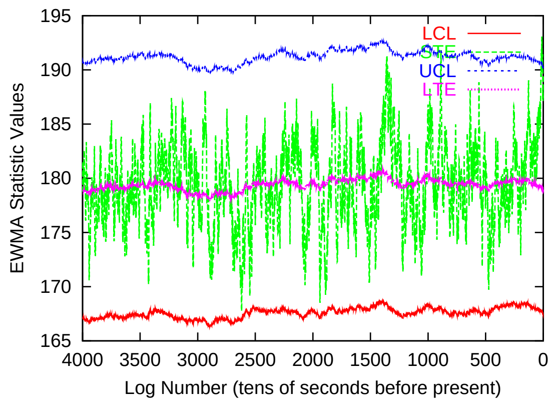

Figure 4: ArQoS EWMA SQC chart for a QoS measure recorded for over 10 hours

In case of samples that do not involve subgroups, that is, where $n = 1$, the equations (13) and (14) become

$$\epsilon_k = \lambda x_k + (1-\lambda)\epsilon_{k-1} \tag{15}$$

and

$$\overline{x} \pm 3\sigma\sqrt{\frac{\lambda}{2-\lambda}} \tag{16}$$

where we begin with $\epsilon_0 = \overline{x}$.

It is important to note that the $\overline{X}$ chart (based on subgrouping) is more sensitive in detecting mean shifts than the X chart[29] (based on individual measures). Hence, wherever possible the $\overline{X}$ chart should be preferred.

The value of the EWMA weight parameter λ is chosen as a tradeoff between the required sensitivity of the chart towards fault detection and the false positive rate statistically inherent in the method. A low λ favors a low false positive rate, whereas a large value favors high sensitivity. Typically, in industrial process control, it is set at 0.3.

5.2 Application to Network Flows

In ArQoS, we consider the QoS flows as quality controlled processes where the quality parameters of interest are the bit rates, packet drop rates, end-to-end one-way or two-way delays, and jitter of the flows.

In statistical process control, the EWMA control chart is used to detect small shifts in the mean of a process to indicate an out-of-control condition. In contrast, in ArQoS, we monitor flow processes that are only weakly stationary, and hence have to allow for a gradual shift in their means over sufficiently long periods of time. However, a swift shift will be detected as an anomaly and flagged.

In view of the above, an actual process mean $\overline{\overline{x}}$, which we represent as LTE, is found as an exponentially weighted moving average as

$$LTE_k = (1-r_{LT})LTE_{k-1} + r_{LT}\overline{x}_k \tag{17}$$

where the decay parameter r_{LT} makes LTE significant only for the past $LTPeriod$ period in time, just as with STAT. The flow processes are chosen such that the mean does not shift significantly over this period of time. Any small shift is reflected in a shift in LTE.

Similarly, the *long term count square*, $LTCntSq$ is found as

$$LTCntSq_k = (1-r_{LT})LTCntSq_{k-1} + r_{LT}\overline{x}_k^2 \tag{18}$$

and the *long term standard deviation*, $\hat{\sigma}$ or $LTStdDev$ is found as

$$LTStdDev = \sqrt{LTCntSq - LTE^2} \tag{19}$$

The EWMA statistic ϵ_k, which we represent by our short term estimate parameter STE_k, is given by

$$STE_k = rSTE_{k-1} + (1-r)\overline{x}_k \qquad (20)$$

where the decay rate r is the same as the short term decay parameter used in STAT. The STE estimate has $STPeriod$ as the moving window of significance.

Then, the equation (14) gives the EWMA control limits as

$$LTE \pm 3\frac{LTStdDev}{\sqrt{n}}\sqrt{\frac{1-r}{1+r}} \qquad (21)$$

where n equals 1 for byte and packet counts, but is greater than 1 for parameters like jitter and end-to-end delay measures that are averaged over several packets.

STE_k, LTE_k, $LTCntSq_k$ and the control limits UCL_k and LCL_k are calculated for every received measure and an alarm is raised whenever STE_k falls beyond either of the control limits.

Although the QoS parameters of interest to us are strictly not Normally distributed, nevertheless, our results indicate that the stationary and low-deviation parameters we study too fall beyond the control limits only under anomalous conditions, that is, when under attack.

6 Tests

6.1 Simulation Tests for Algorithm Validation

To verify the correctness of the STAT and the EWMA Control Chart algorithms, we use simulated normal network traffic and simulated attacks. We generate counts or measures of a particular QoS parameter, such as jitter say, such that it has an arbitrary but fixed distribution, such as Normal, Uniform or an arbitrary *bell-shaped* distribution, with known means and standard deviations. Once the STAT, RULE and EWMA Charts are trained and configured for the normal (unattacked) distributions, we gradually or incrementally vary or switch the mean or standard deviation of the distribution to effect simulated attacks.

The simulations help in studying the sensitivity of the detection system, measuring the false alarm generation rate, and to compare the performance of the three approaches in a fairly controlled environment. We also find the simulations immensely useful for empirical selection of the algorithm parameters.

6.2 Test Bed Setup and WAN Emulation

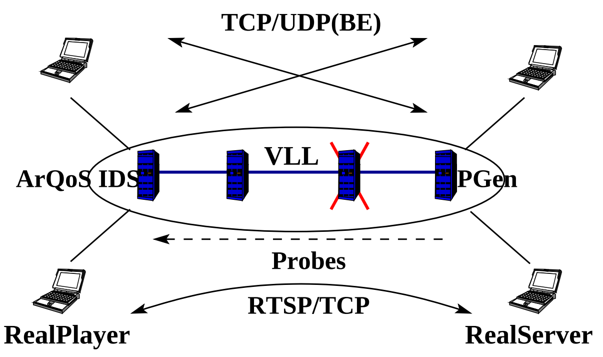

Figure 5: ArQoS network setup for tests

Figure 5 shows a condensed topology for our isolated DiffServ capable network test bed. All the routers we use are Linux kernels configured with DiffServ capabilities. For this, we use the traffic conditioning configuration utility tc that comes with iproute2[19] to configure the queueing disciplines, classes, filters and policers attached to the kernel network devices.

We setup a VLL between an ingress and an egress router. It is setup using only IP address based routing entries and DSCP based service class mappings. The VLL carries an audio-video stream that uses Real Time Streaming Protocol (RTSP) with TCP as the transport protocol, and either EF or AF (in separate tests) as the DiffServ class type for minimal delay or minimal drop rates in forwarding. The VLL is sourced and sinked by a RealVideo[1] client-server pair as shown.

We use MGEN/DREC[2] to generate a *Poisson* distributed UDP BE background traffic. thttp[8] generates a HTTP/TCP BE background traffic based on an empirical HTTP traffic model[21]. The idea is that it is easier to observe the relatively differentiated forwarding services in a moderately or heavily flooded network. Further, it is more interesting to attempt to distinguish effects on the QoS due to attacks from those due to random fluctuations in the background traffic. The combination of the Poisson and the HTTP models attempts to capture the burstiness, and the randomness typical to the aggregate Internet traffic. To emulate a Wide Area Network (WAN), we introduce end-to-end delays and jitter in each of the routers using NISTNet[6]. We use round-trip delays with means of the order of tens of milliseconds and standard deviations of the order of a few milliseconds to emulate a moderate sized DiffServ WAN with low congestion.

Probes are generated at the ingress side of the RTSP/TCP (data) flow and the PgenMon and TrafMon sensors, along with the detection engine components are placed on the egress router. We configure the RTSP/TCP stream server to stream a Constant Bit Rate traffic (CBR) within the congestion that the VLL normally experiences. PgenMon monitors the jitter and the packet rate of the probes, whereas TrafMon monitors the bit rate of the CBR (data, that is, one-way) flow.

For the attacking agents, we again use NISTNet to significantly increase the end-to-end delays (although we do not test for these presently) and the jitter in the QoS flows. We also use Linux kernel modules, we call ipt_drop and ipt_setTOS, as the packet dropping and packet remarking modules respectively. These are based on the IPTables'[11] packet capturing and mangling framework. The modules are written as IPTables' kernel extensions to be called whenever a predefined filter independently captures a packet.

Thus, based on a specified filter, a selected flow can be subjected to any combination of packet dropping, packet DSCP-field remarking, and introducing extra jitter & end-to-end delay variation attacks. The attack intensities are user configurable.

STAT and the EWMA chart module are trained with the VLL and the probe flows in the absence of any attacks, and RULE is configured for checking conformity with the SLA. Then the individual attacks are launched to check the detection capabilities of the modules. It is important to note that the attacks can not differentiate between the probe and the RTSP/TCP flow in the VLL.

Thus we monitor the VLL's QoS amidst a moderate background traffic and occasional attacks.

7 Test Results and Discussion

Figure 6 is a *screen capture* of a summary of alerts generated in real time for about just over an hour by the detection system. The anomaly detection results from our tests are provided in Table 1.

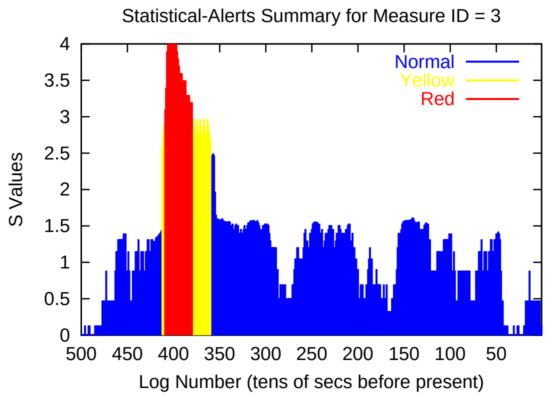

Figure 6: Screen capture of an Alerts Summary generated for over an hour. Shows an attack detection Red "alert-cluster". The rightmost end is the most recent alerts' end.

Attack Type	Detected?	By?	Time Taken
Dropping	Yes	PgenMon, EWMA, pps	<1 min
Dropping	Yes	TrafMon, STAT, bps	<15 min
Jitter+	Yes	PgenMon, STAT, jitter	<15 min
Remarking EF to BE	Yes	PgenMon, STAT, jitter	<1 min
Remarking BE to EF	Yes	PgenMon, STAT, jitter	<1 min

Table 1: ArQoS Anomaly Detection Test Results

We have been able to achieve a 100% detection rate for each of the attack types. For each type, there is a certain threshold level of intensity of the attack below which the detection takes more than 15 minutes - too slow detections that we have ignored. These intensity thresholds depend mainly on the statistical significance of the attack on the present distribution of the traffic parameter being monitored. For example, a sub-1% packet dropping rate attack does not produces a quickly detectable effect on a flow that normally experiences dropping rates upwards of 5%, whereas the same attack is quickly detected when the dropping rate has been close to 0%. As expected, the anomaly detection techniques fail to detect a slow and progressive attack in which the QoS of the flow is degraded over a long period of time.

RULE, a bounds-checker for the QoS measures, detects even these slow attacks once they eventually degrade the QoS beyond the threshold levels.

The detection rate of RULE is much higher than that of the anomaly detection methods for any attack that is detected by all the methods. This is due to the EWMA estimation delay associated with the short term profiling in the anomaly detection methods.

The selection of parameters used in the short term EWMA estimation process is made under the following constraints.

- The $STPeriod$ should be minimized for quicker detection. We suggest that it should not be greater than about 20 minutes.

- The number of short term estimate-components (, that are averaged in the short term EWMA process as given by the Equations (7) and (8),) which have more than 1% weight in the EWMA estimator at any given instance is the *effective short term EWMA sample size*, N_s. The FPR increases with decreasing N_s, since, even intuitively, a small *window of significance* implies a short term profile that is highly sensitive to a present profile fluctuation. Thus N_s should be maximized to reduce the FPR. We suggest that the N_s should always be set above about 15.

- The period between two consecutive logs from a given sensor is the sensor's *Inter-Logging Period ILPeriod*. Although not necessary, for administrative convenience, we recommend using the same value for $ILPeriod$ for all sensors and for all measures. Jitter involves an average over a subgroup of probe packets received between the times the receiving sensor makes two consecutive logs. Clearly then, this time between two consecutive logs should be large enough to allow the sensor to receive enough probe packets to average over. The probes are themselves generated at a low rate (recommended maximum of 2 packets per second). As a function of these constraints then, we recommend a $ILPeriod$ of at least 10 seconds, that guarantees at least 10 probe packets in that period at a 1 probe packet per second and a 0% packet dropping rate.

From Equations (7) and (8), it is straight forward to note that for a $STPeriod$ period to have a window of 99% significance translates to

$$r^{\frac{STPeriod}{ILPeriod}} = 0.01 \tag{22}$$

while N_s is obtained (approximately) as a geometric progression sum as

$$N_s \approx \frac{1 - r^{\frac{STPeriod}{ILPeriod}}}{1 - r} \approx \frac{1}{1 - r} \tag{23}$$

Then, eliminating r from the above two equalities gives

$$STPeriod = \frac{-2 \times ILPeriod}{log_{10}(1 - \frac{1}{N_s})} \tag{24}$$

Figure 7 is based on Equation (24). Considering the above mentioned constraints, we settled on a $STPeriod$ (short term estimation period) of 15 minutes, that corresponds to a N_s of about 20 at a logging or sampling frequency of the sensors of once in 10 seconds. We recommend a probe generation rate of 1 or 2 packets per second, to ensure a good subgroup sample size for jitter and a low network flooding (8 to 16 kbps in typically more than several hundreds of kbps of VLL normal/data traffic).

The long term period should be long enough to profile a (weakly) stationary distribution and should be short enough to capture a (slow) shift in the distribution mean typical to QoS flows. We suggest a $LTPeriod$ of about 4 hours.

Attacks that involve targeting only the packet bursts in a VLL while not significantly affecting the bit rates can not be detected in general. This is so because, the attacks appear to the detection system as normal policing by a router. Since the probes are spread uniformly in the VLL flow, the attacks on bursts statistically affect the actual data

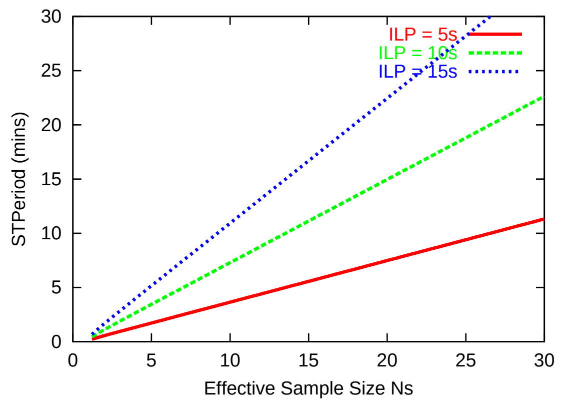

Figure 7: Curves illustrating the tradeoff between Detection Rate (from $STPeriod$) and False Positive Rate (from N_s)

flow more than the probe flow. This renders the PgenMon probe flow sensors insensitive to the attack, since they monitor the jitter and packet rates of only the probes. The only defense against such attacks, we believe, is that the VLL flow should be policed at boundary and some core routers to minimize its burstiness. Further, it is important to note that a dropping attack in particular that targets the bursts while maintaining the packet and bit rates is difficult to effect in a real world. This is because the end processes and their TCP transport protocol interpret the attack as a network congestion and adaptively vary the bit and packet rates of the flow between them.

We define the False Positive Rate (FPR) as

$$FPR = \frac{Total\ Y\ or\ R\ Alarms}{Total\ N,\ Y,\ or\ R\ Alarms} \tag{25}$$

measured over (say) a 24 hour period, given that the network has been isolated from any attacks or externally induced anomalies.

Both the STAT and the EWMA Chart methods have inherent FPRs that arise due to the tail probabilities of the corresponding χ^2 and the Normal distributions on which they are based. The χ^2 tail probability α and the 3σ Normal limits are chosen such that each leave a 1% chance for an error in detection. Indeed, the FPR that the modules produce together is about 0.01.

The FPR due only to STAT is even lower. The short term profile p_i' are obtained as EWMA estimates (Equation (7)). Thus they have a standard deviation reduced by a factor of $\frac{1}{\sqrt{n}}\sqrt{\frac{1-r}{1+r}}$ (Equation (14)). Hence the FPR is lower than if the p_i' were based on individual measure values.

Further, due again to the EWMA estimation process, STAT continues to generate anomaly alarms for a period of the same order as the short term estimation period, before returning to normal. Similarly, a definite shift in the short term QoS distribution results in a series of alarms from the EWMA Chart as against occasional individual false alarms that occur even in the absence of an anomalous mean shift. Hence, we recommend that only when a cluster or series of red alarms are detected (that thus last for at least a few minutes), do we declare the event as an anomaly detection. Else, the event is ignored as a false alarm. We find that this *elimination by inspection* procedure invariably reduces the overall FPR rate by upto an order of magnitude to 0.001. The *inspection* itself could be automated by an algorithm that looks for a certain rate or number of alarms to alert the Security Officer. Figure 6 illustrates an *alerts-cluster*.

A VLL which is terminated and re-initiated frequently (say more frequently than once in 24 hours) poses a problem for TrafMon which flags the events as anomalies. This is not a problem with PgenMon as it monitors only probes that are terminated only administratively. For such VLLs, then, we observe lower false positives with probe based detection.

Since the probe packets have sequence numbers, and since the jitter calculation takes these into consideration, the calculation is not skewed by attacks like packet dropping that indirectly vary the probe packet inter-arrival periods. Hence jitter and packet rates are fairly independently measured.

8 Conclusions

The ArQoS anomaly detection system detects QoS degradation quickly and in real time. All the identified attacks can always be detected. The DiffServ framework itself does not require any modifications. The possibility of generating a false alarm is very low and we believe the rate can be tolerated in most deployments. The sensors and the detection engine are all light weight threads that can monitor several VLLs at a time. Each VLL is profiled independently, plus each VLL has a separate anomaly score. Thus, the order of complexity or effort, as far as processing or attending to the alarms is concerned, is equal to the number of VLLs in the network cloud. Further, the profile updates do not require any human intervention. We thus believe that the detection system is highly scalable and can be used in moderately (typical) sized QoS networks. The system itself can be extended to other QoS parameters. For example, end to end delays can be monitored once boundary router components are time synchronized. RULE's attack signature database may be expanded as more attacks as investigated and defined.

9 Future Research Directions

With anomaly detection, typically, the number of false positives exceed the number of true positives. Hence, considerable effort has to be expended by security officers in investigating the causes of an attack alert before deciding whether they are either false alarms or actual attacks that need further looking into. Moreover, we find that a single point attack in the network may generate attack alerts at multiple other points in the QoS network. Plus, we may need to monitor multiple points of the network just to detect a single instance of an attack.

In view of this, we are exploring ways to allow us to generate audit reports that would aid a security officer to know the type of attack that occurred, the intensity of the attack and the exact node(s) or path(s) in the network that was attacked.

Presently, we are investigating the use of the Abductive Inference Model proposed in [23] for event management and correlation in a Bayes belief network. With this model, each alarm A_i would be associated with the probability of it being lost (l_i), the probability of it being generated spuriously (s_i), and a list of all possible attacks which would have caused it. The event correlation engine would then take alerts as input events from sensors distributed over the network and generate an audit report describing the most likely explanation for those alerts.

10 Acknowledgements

We would like to thank a number of co-researchers who have contributed to several components of this work. These include Daniel Stevenson at Advanced Networking Research, MCNC, Dr. Fengmin Gong at Intruvert Networks, Dr. Frank Jou at IBM, and Dr. Felix Wu of the University of California, Davis. This work is funded by a grant from the Defense Advanced Research Projects Agency (DARPA), administered by the Air Force Research Labs (AFRL), Rome under contract F30602-99-1-0540. We gratefully acknowledge this support.

References

[1] RealPlayer and RealServer from RealNetworks http://www.realnetworks.com/.

[2] B. Adamson. Multi Generator MGEN Toolset from Naval Research Laboratory, Available at http://manimac.itd.nrl.navy.mil/MGEN/.

[3] D. Anderson, T. Lunt, H. Javits, A. Tamaru, and A. Valdes. Detecting unusual program behavior using the statistical components of NIDES. Technical Report SRI-CSL-95-06, SRI International, Computer Science Laboratory, May 1995.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Technical Report RFC 2475, IETF DiffServ Network Working Group, December 1998.

[5] R. B. Blazek, H. Kim, B. Rozovskii, and A. Tartakovsky. New adaptive batch and sequential methods for rapid detection of network traffic changes with emphasis on detection of "denial-of-service" attacks. Technical report, University of Southern California, Center For Applied Mathematical Sciences, 2001.

[6] Mark Carson. NIST Net from National Institute of Standards and Technology, Available at http://snad.ncsl.nist.gov/itg/nistnet/.

[7] Dorothy E. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, Feb 1987.

[8] Don Smith et al. thttp from University of North Carolina, Chapel Hill.

[9] Herv Debar et al. Towards a taxonomy of intrusion detection systems. *Computer Networks*, 31:805–822, 1999.

[10] R. Braden et al. Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. Technical Report RFC 2205, IETF Network Working Group, September 1997.

[11] Rusty Russell et al. IPTables from the Netfilter/IPTables Project. Available at http://netfilter.samba.org/.

[12] T. D. Garvey and T. F. Lunt. Model based Intrusion Detection. In *Proceedings of the 14th National Computer Security Conference*, pages 372–385, 1991.

[13] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB group. Technical Report RFC 2597, IETF DiffServ Network Working Group, June 1999.

[14] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference.* Prentice Hall, 1997. 385-399.

[15] Koral Ilgun, Richard A. Kemmerer, and Phillip A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, 1995.

[16] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. Technical Report RFC 2598, IETF DiffServ Network Working Group, June 1999.

[17] H. Javits and A. Valdes. The NIDES statistical component: Description and justification. Technical report, SRI International, Computer Science Laboratory, March 1993.

[18] Y. F. Jou, F. Gong, C. Sargor, X. Wu, S. F. Wu, H. Y. Chang, and F. Wang. Design and implementation of a scalable intrusion detection system for the protection of network infrastructure. *DARPA Information Survivability Conference and Exposition(DISCEX) 2000*, 2:69–83, January 2000.

[19] Alexey Kuznetsov. iproute2 Available at http://diffserv.sourceforge.net/.

[20] Emilie Lundin and Erland Jonsson. Some practical and fundamental problems with anomaly detection. In *Proceedings of the Fourth Nordic Workshop on Secure IT systems*, Kista, Sweden, November 1999.

[21] Bruce A. Mah. An empirical model of HTTP network traffic. In *INFOCOM (2)*, pages 592–600, 1997.

[22] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of differentiated services field (ds field) in the ipv4 and ipv6 headers. Technical Report RFC 2474, IETF DiffServ Network Working Group, Dec 1998.

[23] David Alan Ohsie. *Modeled Abductive Inference for Event Management and Correlation.* PhD dissertation, Columbia University, 1998.

[24] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, 5(50):157, 1900.

[25] Diheng Qu. Statistical anomaly detection for link-state routing protocols. Master's thesis, North Carolina State University, Computer Science, 1998.

[26] Diheng Qu, B. M. Vetter, R. Narayan, S. F. Wu, F. Wang, Y. Frank Jou, F. Gong, and C. Sargor. Statistical anomaly detection for link-state routing protocols. *Proceedings of the 1998 International Conference on Network Protocols*, pages 62–70, October 1998.

[27] S. W. Roberts. Control chart tests based on geometric moving averages. *Techometrics*, 1(3):239–250, 1959.

[28] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Technical Report RFC 3031, IETF Network Working Group, January 2001.

[29] Thomas P. Ryan. *Statistical Methods for Quality Improvement.* Wiley Series, 2000. 133-134.

[30] W. Shewhart. The applications of statistics as an aid in maintaining quality of a manufactured product. 1925.

[31] W. Shewhart. Economic control of quality of manufactured product, 1931.

[32] W. Shewhart. Statistical method from the viewpoint of quality control, 1939.

[33] Shiuh-Pyng Shieh and Virgil D. Gligor. On a pattern-oriented model for intrusion detection. *Knowledge and Data Engineering*, 9(4):661–667, 1997.

[34] Aaron Striegel. Security issues in a differentiated services internet.

[35] Christina Warrender, Stephanie Forrest, and Barak A. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.

[36] J. Wroclawski. The Use of RSVP with IETF Integrated Services. Technical Report RFC 2210, IETF Network Working Group, September 1997.