ISSN:0975-9646
Sridhar kumar.Majoju et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1549-1554

Detecting multiple malevolent packet losses

Sridhar kumar.Majoju[#1], D.Sagar[*2], T.P.Shekar[#3]

[#]*Computer Science and Engineeringt,JNT University*
Sree chaitanya college of Engineering, ,karimnagar, Andhra Pradesh ,INDIA-505527.

[*]*sree chaitanya college of engineering*
karimnagar,andhar Pradesh,INDIA-505527.

***Abstract*- Identifying the victimized packets which are dropped in the router and revoking them for further utilization and also to protect from hackers. It is quite challenging to attribute a missing packet to a malevolent action because normal network congestion cannot produce the same effect. Static user-defined threshold and χ protocol approached this issue but it is fundamentally limiting, based on measured traffic rates and buffer sizes. This approach does not infer dynamically and here always a possibility of losing the packets. The number of congestive packet losses that will create an ambiguity. Once the ambiguity from congestion is removed, subsequent packet losses can be attributed to malevolent actions. Setting this threshold is at the best is an art, and will certainly create unnecessary false positives or mask highly focused attacks. A compromised router is malevolently manipulating its stream of Multiple packets. We have designed and implemented a compromised router detection protocol that will dynamically infer the multiple packet losses, based on measured traffic rates and buffer sizes. This paper is distinguished between a router dropping packets malevolently and a router dropping packets due to congestion. The present work consists Adaptive RED algorithm for finding the default packets in short time. Automatically setting ARED parameters and maintains a predictable average queue size and reduces RED's parameter sensitivity.**

***Keywords*— intrusion detection, distributed systems, reliable net works, malevolent routers, multiple packet losses, router, ARED**.

Introduction

Network packet loss is a symptom that when we use ping command to query the target, and because of all kinds of reasons, data packets lost in the channel. The command ping uses ICMP echo request and echoplex reply messages. ICMP echoplex request message is a inquiry from host or router to a specified target host, and machine received the message must send back ICMP echoplex reply message to source host. This inquiry message is used to test if the target could be reached and get to know its state. It should be noted that the command ping is an example directly using network level ICMP without passing through transportation level UDP or TCP.Main reasons of Network packet loss are: physical connection failure, device failure, virus attacks, router message error and so on. Next let's make an explanation with specific situation. Physical connection failure .Network administrator finds WAN connection on-and-off, when this happens, probably the connection line has some problem, or caused by users. To figure out if it is a connection failure, we can make the test below: If the WAN connection is realized by router, then login to router, and then test the router WAN connection by sending a mass of data packets. If it realize through three-level switch, then separately connect a computer at both sides of the wire, and set their IP addresses as the WAN connection address of the three-level switch, and use the command "ping the IP address of the other side computer -t" to test. If no packet loss happens during the test, then it indicates that the wire provided is fine, probably the reason is caused by users, so you need further test. If packet loss happens during the test above, then it indicates the failure is caused by the circuitry, you should contact with the circuitry provider as soon as possible to solve the problem. As there are many symptoms of packet loss caused by physical circuit, such as fiber connection problems, like jumper is not aimed at device interface, twisted-pair and RJ-45 tie-in problems. Besides, LOC gets affected by noise with machine or outburst noises, so data message error may occur, radio frequency signal interruption and signal attenuation may cause the loss of data packet. We can test the circuitry quality with network test machine.

I .INFERRING JAM-PACKED LOSS

In construction a traffic validation protocol, it is necessary to explicitly resolve the ambiguity around packet losses. Should the absence of a given packet be seen as malicious or benign? In practice, there are three approaches for addressing this issue:

- *Static Threshold.* Low rates of packet loss are assumed to be congestive, while rates above some predefined threshold are deemed malicious.
- *Traffic modeling.* Packet loss rates are predicted as a function of traffic parameters and losses beyond the prediction are deemed malicious.
- *Traffic measurement*. Individual packet losses are predicted as a function of measured traffic load and router buffer capacity. Deviations from these predictions are deemed malicious.

Most traffic validation protocols, including WATCHERS, Secure Trace route , and our own work described in ,analyze aggregate traffic over some period of time in order to amortize monitoring overhead over many packets. For example, one validation protocol described in maintains packet counters in each router to detect if traffic flow is not conserved from source to destination. When a packet arrives at router r and is forwarded to a destination that will traverse a path segment ending at router x, r increments an outbound counter associated with router x. Conversely, when a packet arrives at router r, via a path segment beginning with router x, it increments its inbound counter

Associated with router x. periodically, router x sends a copy of its outbound counters to the associated routers for validation. Then, a given router r can compare the number of packets that x claims to have sent to r with the number of packets it counts as being received from x, and it can detect the number of packet losses. Thus, over some time window, a router simply knows that out of m packets sent, n were successfully received. To address congestion ambiguity, all of these systems employ a predefined threshold: if more than this number is dropped in a time interval, then one assumes that some router is compromised. However, this heuristic is fundamentally flawed: how does one choose the threshold? In order to avoid false positives, the threshold must be large enough to include the maximum number of possible congestive legitimate packet losses over a measurement interval. Thus, any compromised router can drop that many packets without being detected. Unfortunately, given the nature of the dominant TCP, even small numbers of losses can have significant impacts. Subtle attackers can selectively target the traffic flows of a single victim and within these flows only drop those packets that cause the most harm. For example, losing a TCP packet used in connection establishment has a disproportionate impact on a host because the retransmission time-out must necessarily be very long (typically 3 seconds or more). Other seemingly minor attacks that cause TCP time-outs can have similar effects—a class of attacks well described in . All things considered, it is clear that the static threshold mechanism is inadequate since it allows an attacker to mount vigorous attacks without being detected. Instead of using a static threshold, if the probability of congestive losses can be modeled, then one could resolve ambiguities by comparing measured loss rates to the rates predicted by the model. One approach for doing this is to predict congestion analytically as a function of individual traffic flow parameters, since TCP explicitly responds to congestion. Indeed, the behavior of TCP has been excessively studied .A simplified1 stochastic model of TCP congestion control yields the following famous square root formula:

$$B = \left(\frac{1}{RTT}\right)\sqrt{(3/2bp)}$$

Where B is the throughput of the connection, RTT is the average round trip time, b is the number of packets that are acknowledged by one ACK, and p is the probability that a TCP packet is lost. The steady-state throughput of long-lived TCP flows can be described by this formula as a function of RTT and p. This formula is based on a constant loss probability, which is the simplest model, but others have extended this work to encompass a variety of loss processes. of these have been able to capture congestion behavior in all situations. Another approach is to model congestion for the aggregate capacity of a link. explore the question of "How much buffering do routers need?" A widely applied rule-of-thumb suggests that routers must be able to buffer a full delay bandwidth product. This controversial paper argues that due to congestion control effects, the rule-of-thumb is wrong, and the amount of required buffering is proportional to the square root of the total number of TCP flows. To achieve this, the authors produced an analytic model of buffer occupancy as a function of TCP behavior. We have evaluated their model thoroughly and have communicated with the authors, who agree that their model is only a rough approximation that ignores many details of TCP, including time-outs, residual synchronization, and many other effects. Thus, while the analysis is robust enough to model buffer size it is not precise enough to predict congestive loss accurately. we have to measuring the interaction of traffic load and buffer occupancy explicitly. Given an output buffered first-in first-out (FIFO) router, congestion can be predicted precisely as a function of the inputs,), the capacity of the output buffer, and the speed of the output link. A packet will be lost only if packet input rates from all sources exceed the output link Speed for long enough. If such measurements are taken with high precision it should even be possible to predict individual packet losses. We restrict our discussion to output buffered switches for simplicity although the same approach can be extended to input buffered switches or virtual output queues with additional adjustments (and overhead). Because of some uncertainty in the system, we cannot predict exactly which individual packets will be dropped. So, our approach is still based on thresholds. Instead of being a threshold on rate, it is a threshold on a statistical measure: the amount of confidence that the drop was due to a malicious attack rather than from some normal router function.

II. NET WORK MODEL

Our work proceeds from an informed, yet abstracted, model of how the network is constructed, the capabilities of the attacker, and the complexities of the traffic validation problem. In this section we describe and motivate the assumptions underlying our model.

Network Model: We consider a network to consist of individual homogeneous routers interconnected via directional point-to-point links. This model is an intentional simplification of real networks (e.g., it does not include broadcast channels or independently failing network interfaces) but is sufficiently general to encompass such details if necessary. Within a network, we presume that packets are forwarded in a hop-by-hop fashion each router following the directions of a local forwarding table. As well, we assume that these forwarding tables are updated via a distributed link-state routing protocol such as OSPF or IS-IS. This is critical, as we depend on the routing protocol to provide each node with a global view of the current network topology. Finally, we also assume the administrative ability to assign and distribute shared keys to sets of nearby routers. This overall model is consistent with the typical construction of large enterprise IP networks or the internal structure of single ISP backbone networks, but is not well-suited for networks that are composed of multiple administrative domains using BGP. At this level of abstraction, we can assume a synchronous network model of synchronized clocks and bounded message delays. Our goal is to extend the routing protocol to detect compromised routers. If the network behaves asynchronously for too long, then the routing tables will be updated, thereby changing the network topology. This assumption is common to all protocols we know of that have addressed the problem

of detecting compromised routers. We define a *path* to be a finite sequence *r*1, *r*2,...,*rn* of adjacent routers. Operationally, a path defines a sequence of routers a packet can follow. We call the first router of the path the *source* and the last router its *sink*; together, these are called *terminal routers*. A path might consist of only one router, in which case the source and sink are the same. An *x−path segment* is a sequence of *x* consecutive routers that is a subsequence of a path. A *path segment* is an *x*−path segment for some value of $x > 0$. For example, if a network consists of the single path _*a, b, c, d*_ then _*c, d*_ and _*b, c*_ are both 2-path segments, but _*a, c*_ is not because *a* and *c* are not adjacent. We do not rely on source routing, as has been done by some work in the past [4, 20, 36]. We do assume some knowledge of the path a packet will take, at least in the stable state. In link state protocols, this can be problematic, because they can take advantage of multiple paths with equal cost for load balancing purposes. a router can predict the path a packet will take in the stable state based on its own routing tables and the hash functions.

Threat Model: We assume that attackers can compromise one or more routers in a network and may even compromise sets of adjacent routers as well. In general, we parameterize the strength of the adversary in terms of the maximum number of adjacent routers along a given path that can be compromised. However, we assume that between any two uncompromised routers that there is sufficient path diversity that the malicious routers do not partition the network. In some sense, this assumption is pedantic since it is impossible to guarantee any network communication across such a partition. Another way to view this constraint is that path diversity between two points in the network is a

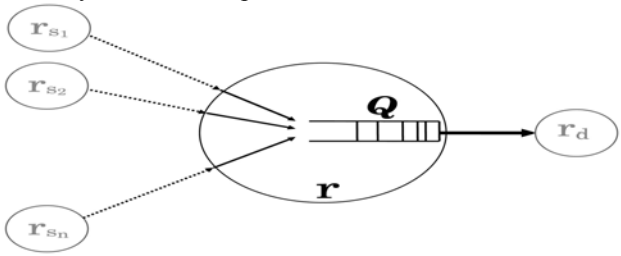

Fig.1: Validating the queue of an output interface.

Necessary, but insufficient, condition for tolerating compromised routers. What we are proposing is a set of protocols that offer the sufficiency condition in the presence of the necessary diversity. Similarly, many enterprise networks are designed with such diversity in order to mask the impact of link failures. Consequently, we believe that this assumption is reasonable in practice. It is worth noting however, that this diversity usually does not extend to individual local-area networks single workstations rarely have multiple paths to their network infrastructure. Consequently, the fate of individual hosts and of the router to which they are connected, are directly intertwined in practice. So, we assume that a terminal router is not faulty with respect to traffic originating or being consumed by that router. Since link-state protocols operate by periodically measuring and disseminating information, we assume a synchronous system. The failure of a router is defined in terms of an interval of time, which in practice corresponds with a period of time during which traffic measurements are made. Specifically, a router *r* is *traffic* faulty with respect to a path segment π during τ if π contains *r* and, during the period of time τ, *r* exhibits anomalous behavior with respect to forwarding data that traverses π. For example, router *r* can selectively alter, misroute, drop, reorder, or delay the data that flows through π, and it can fabricate new data to send along π such that the packets, if they were valid, would have been routed through π.

III. TRAFFIC VALIDATION

Protocol χ detects traffic faulty routers by validating the queue of each output interface for each router. Given the buffer size and the rate at which traffic enters and exits a queue, the behavior of the queue is deterministic. If the actual behavior deviates from the predicted behavior, then a failure has occurred. We present the failure detection protocol in terms of the solutions of the distinct sub problems: traffic validation, distributed detection, and response.

The first problem we address is traffic validation: what information is collected about traffic and how it is used to determine that a router has been compromised. Consider the queue Q in a router r associated with the output interface of link hr; rdi (see Fig. 1). The neighbor routers rs1; rs2; . . . ; rsn feed data into Q. We denote with T in f $O(r,Q_{dir,\prod,T})$the traffic information collected by router r that traversed path segment over time interval T. Q_{dir} is either Q_{in}, meaning traffic into Q, or Q_{out}, meaning traffic out of Q. At an abstract level, we present traffic, a validation mechanism associated with Q, as a predicate TV (Q,$q_{pred,}$t, S;D), where :

- $q_{pred}(t)$ is the predicted state of Q at time t. $q_{pred}(t)$ is initialized to 0 when the link <r; rd> is discovered and installed into the routing fabric. q_{pred} is updated as part of traffic validation. .

$C_{single} =$ Prob(*fp* is maliciously dropped)
= Prob(there is enough space in the queue to buffer *fp*)
= Prob($q_{act}(ts) + ps \le q_{limit}$)
= Prob($X + q_{pred}(ts) + ps \le q_{limit}$)
= Prob($X \le q_{limit} - q_{pred}(ts) - ps$)
= Prob($Y \le ((q_{limit} - q_{pred}(ts) - ps - \mu)/\sigma)$)
= prob($Y \le y1$)
= $(1+ erf(y1/\sqrt{2})/2)$

- S $=\{V_i \in \{1,2,\ldots,n\}:$Tin f $o(r_{si}, Q_{in,} <rsi,r,rd>,T)\}$, is a set of information about traffic coming into Q as collected by neighbor routers.
- D=T in f $O(rd,Q_{out},<r,rd>,T)$ is the traffic information about the outgoing traffic from Q collected at router rd.

If routers r_{s1}, r_{s2}. . . r_{sn} and rd are not protocol faulty, then TV (Q; $q_{pred(t),}$ S,D) evaluates to false if and only if r was traffic faulty and dropped packets maliciously during T. Tinfo(r,$Q_{dir,}\prod$,T) can be represented in different ways.We use a set that contains, for each packet traversing Q, a three-

tuple that includes: a fingerprint of the packet, the packet's size, and the time that the packet entered or exited Q (depending on whether Q_{dir} is Q_{in} or Q_{out}). For example, if at time t router rs transmits a packet of size ps bytes with a fingerprint fp, and the packet is to traverse ∏,then rs computes when the packet will enter Q based on the packet's transmission and propagation delay. Given a link delay d and link bandwidth associated with the link hrs; ri, the time stamp for the packet is t þ d þ ps=bw.TV can be implemented by simulating the behavior of Q. Let P be a priority queue, sorted by increasing time stamp. All the traffic information S and D are inserted into P along with the identity of the set (S or D) from which the information came. Then, P is enumerated. For each packet in P with a fingerprint fp, size ps, and a time stamp ts, q_{pred} is updated as follows. Assume t is the time stamp of the packet evaluated prior to the current one:

- If fp came from D, then the packet is leaving Q : $q_{pred}(ts) := q_{pred}(t)$- ps.
- If fp came from S and (fp € D), then the packet fp is entering and will exit: $q_{pred}(ts) := q_{pred}(t)$+ ps.
- If fp came from S and (fp €D), then the packet fp is entering into Q and the packet fp will not be transmitted in the future: $q_{pred}(ts)$ is unchanged, and the packet is dropped.
  – If $q_{limit} < q_{pred}(t)$+ ps, where q_{limit} is the buffer limit of Q, then the packet is dropped due to congestion.
  – Otherwise, the packet is dropped due to malicious attack.

III.1 One packet losses test

If a packet with fingerprint fp and size ps is dropped at time ts when the predicted queue length is q_{pred} (ts), then we raise an alarm with a confidence value c_{single}, which is the probability of the packet being dropped maliciously. c_{single} is computed. The mean μ and standard deviation σ of X can be determined by monitoring during a learning period. We do not expect μ and σ to change much over time, because they are in turn determined by values that themselves do not change much over time. Hence, the learning period need not be done very often. A malicious router is detected if the confidence value csingle is at least as large as a target significance level s^{level}_{single}.

Random variable $X=q_{act}(ts)-q_{pred}(ts)$With mean μ and standard deviation σ .Random variable Y = (X-μ)/ σy 1=q_{limit} –q_{pred} (ts)-ps-μ/ σ, erf is the error function.

III.2 Multiple packet losses test

Z- test[1] is useful when more than one packet is dropped during a round and the first test does not detect a malicious router.Let L be the set of n > 1 packets dropped during the last time interval. For the packets in L, let ps be the mean of the packet sizes, qpred be the mean of qpred(ts) (the predicted queue length), and qact be the mean of qact(ts) (the actual queue length) over the times the packets were dropped.We test the following hypothesis: "The packets are lost due to malicious attack": μ > q_{limit} _ q_{pred} _ ps. The Z-test score is

$$z1=\frac{(q_{limit} - q_{pred} - ps - \mu)}{\text{row } \sqrt{n}}$$

For the standard normal distribution Z, the probability of Prob(Z < z1) gives the confidence value $c_{combined}$ for the hypothesis. A malicious router is detected if $c_{combine}$d is at least as large as a target significance level $s^{level}_{combined}$.

One can question using a Z-test in this way because the set of dropped packets are not a simple random sample. But, this test is used when there are packets being dropped and the first test determined that they were consistent with congestion loss. Hence, the router is under load during the short period the measurement was taken and most of the points, both for dropped packets and for non dropped packets, should have a nearly full Q

III.3 Distributed Finding

Since the behavior of the queue is deterministic, the traffic validation mechanisms detect traffic faulty routers whenever the actual behavior of the queue deviates from the predicted behavior. However, a faulty router can also be protocol faulty: it can behave arbitrarily with respect to the protocol, by dropping or altering the control messages of χ.We masks the effect of protocol faulty routers using distributed detection.Given TV , we need to distribute the necessary traffic information among the routers and implement a distributed detection protocol. Every outbound interface queue Q in the network is monitored by the neighboring routers and validated by a router rd such that Q is associated with the link <r; d>.With respect to a given Q, the routers involved in detection are (as shown in Fig. 1)

- rs, which sends traffic into Q to be forwarded.
- r, which hosts Q.
- rd, which is the router to which Q's outgoing traffic is forwarded.

Each involved router has a different role, as described below.

3.3.1 Traffic Information Collection

Each router collects the following traffic information during a time interval T:

- rs: Collect Tinfo(rs,Q_{in};<rs,r, rd>,T).
- r: Collect Tinfo(rs,Q_{in};<rs,r, rd>,T).This information is used to check the transit traffic information sent by the rs routers.
- rd: Collect Tinfo(rd;Q_{out}; <r; rd>,T).

3.3.2 Information Dissemination and Detection

- rs: At the end of each time interval T, router rs,sends [T in f O(rs.,Qin; <rs.,r.rd>,T)]rs. That it has collected.$[M]_x$ is a message M digitally signed by x. Digital signatures are required for integrity and authenticity against message tampering.

IV. THE ADAPTIVE RANDOM EARLY DETECTION

The overall guidelines for Adaptive RED as implemented here are the same as those for the original Adaptive RED, that is, of adapting MAX_p to keep the average queue size between MIN_{th} and MAX_{th}. Our approach differs from original Adaptive RED in four ways:

- MAX_P is adapted not just to keep the average queue size between MIN_{th} and MAX_{th} but to keep the average queue size within a target range half way between MIN_{th} and MAX_{th}.

- MAX_P is adapted slowly, over time scales greater than a typical round-trip time, and in small steps.
- MAX_P is constrained to remain within the range [0.01, 0.5] (or equivalently, [1%, 50%]).
- Instead of multiplicatively increasing and decreasing MAX_P we use an additive-increase multiplicative decrease (AIMD) policy.

The guideline of adapting MAX_P slowly and infrequently allows the dynamics of RED of adapting the packet dropping probability in response to changes in the average queue size—to dominate on smaller time scales. The adaption of MAX_P is invoked only as needed over longer time scales.

The robustness of Adaptive RED comes from its slow and infrequent adjustments of max_p.The price of this slow modification is that after a sharp change in the level of congestion, it could take some time, possibly ten or twenty seconds, before *max p* adapts to its new value. To ensure that the performance of Adaptive RED will not be unduly degraded during this transition period, our third guideline restricts *max p* to stay within the range [0.01, 0.5]. This ensures that during the transition period the overall performance of RED should still be acceptable, even though the

Every *Interval* seconds:
If (*avg* >*target* and $max_p \leq 0.5$)
Increase *max p*:
$max_p \leftarrow max_p + \alpha$
elseif (*avg*<*target* and $max_p \geq 0.01$)
decrease max_p:
$max_p \leftarrow max_p * \beta$;
Variables:
avg : average queue size
fixed parameters:
interval :time 0.5 seconds
target : target for *avg;*
$[min_{th} + 0.4 * (max_{th} - min_{th}), min_{th} + 0.6 * (max_{th} - min_{th})]$.
α : increment; min(0.01, $max_p/4$

The algorithm for Adaptive RED is given in Figure :2

average queue size might not be in its target range, and the average delay or throughput might suffer slightly. We do not claim that our algorithm for Adaptive RED is optimal, or even close to optimal, but it seems to work well in a wide range of scenarios, and we believe that it could Safely be deployed now in RED implementations in the Internet. As a result of the slow adaptation of max_p , the design of Adaptive RED gives robust performance in a wide range of environments. As stated above, the cost of this slow adaptation is that of a transient period, after a sharp change in the level of congestion, when the average queue size is not within the target zone. Adaptive RED is thus consciously positioned in the conservative, robust end of the spectrum of AQM mechanisms, with the aim of avoiding the more finely-tuned but also more fragile dynamics at the more aggressive end of the spectrum.

Adaptive RED's algorithm in Figure :2 uses AIMD to adapt max_p while we experimented with other linear controls such as MIMD (Multiplicative Increase Multiplicative Decrease) as well as proportional error controls, as might be suggested by some control-theoretic analyses; our experiences have been that the AIMD approach is more robust.

IV.1 The range for max_p

The upper bound of 0.5 on max_p can be justified on two grounds. First, we are not trying to optimize RED for packet drop rates greater than 50%. In addition, because we use RED in gentle mode, this means that the packet drop rate varies from 1 to max_p as the average queue size varies from min_{th} to max_{th}, and the packet drop rate varies from *max p* to as the average queue size varies from max_{th} to twice max_{th}. With max_p set to 0.5, the packet drop rate varies from 0 to 1 as the average queue size varies from min_{th} to twice max_{th}. This should give somewhat robust performance even with packet drop rates greater than 50%. The upper bound of 0.5 on *max p* means that when the packet drop rate exceeds 25%, the average queue size could exceed the target range by up to a factor of four[4]

IV.2 The Parameters α and β

We note that it takes at least $0.49/\alpha$ intervals for max_p to increase from 0.01 to 0.50; this is 24.5 seconds for out fault parameters for α and *interval* similarly, it takes at least log 0.02/log β interval for max_p to decrease from 0.50 to 0.01; with our default parameters, this is 20.1 seconds. Given a sharp change from one level of congestion to another, 258 seconds is therefore a upper bound on the interval during which the average queue size could be outside its target range and the performance of the AQM might be some what degraded. Thus assuming p < max_p, when max_p increases by α, ***avg*** can be expected to decrease from $\min th + \frac{p}{\max p}(\max th - \min th)$ to $\min th + \frac{p}{\max p + \alpha}(\max th - \min th)$. This is a decrease of $\frac{\alpha}{(\max p + \alpha)} \frac{p}{\max p}(\max th - \min th)$.

As long as this is less than 0.2 (max_{th}-mi_{th}), the average queue size should not change from above the target range to below the target range in a single interval. This suggests choosing $\frac{\alpha}{(\max p + \alpha)} < 0.2$, or equivalently. $\alpha < 0.25\ max_p$. Our default setting of α obeys this constraint.

IV.3 setting RED parameters max_{th} and wq

As described above, Adaptive RED removes RED's dependence on the parameter max_p to reduce the need for other parameter-tuning for RED, we also specify procedures for automatically setting the RED parameters max_{th} and *wq*. The guidelines for setting *wq* given in the original RED are in

terms of the transient queue size accommodated by RED, and the time required by the estimator to respond to a step change in the actual queue size. if the queue size changes from one value to another, it takes -1/ln(1-*wq)* packet arrivals for the average queue to reach 63% of the way to the new value. Thus, we refer to -1/ln(1-*wq)* as the "time constant" of the estimator for the average queue size, even though this "time constant" is specified in packet arrivals and not in time itself. The default in the NS simulator is for *wq* to be set to 0.002; this corresponds to a time constant of 500 packet arrivals. However, for a 1 Gbps link with 500-byte packets,500 packet arrivals corresponds to a small fraction of a round-trip time (1/50-th of an assumed round-trip time of 100 ms). For RED in automatic mode, we set *wq* to give a time constant for the average queue size estimator of one second; this is equivalent to ten round-trip times, assuming a default round-trip time of 100 ms. Thus, we set

$$wq = 1 - \exp(-1/c)$$

Where C is the link capacity in packets / second, computed for packets of the specified default size.

V. SIMULATIONS

The simulations that Adaptive RED, by automatically setting *wq*. and continually adapting max_p achieves the goals of high throughput and low average queuing delays across a wide variety of conditions. In this section we more closely examine three aspects Adaptive RED's behavior: oscillations, effects of *wq* and response to routing dynamics.

V.1 Exploring Oscillations

The feedback nature of TCP's congestion control, oscillations in the queue length are very common. Some oscillations are "malignant", in that they degrade overall throughput and increase variance in queuing delay; other oscillations are "benign" oscillations and do not significantly effect either throughput or delay.

VI. CONCLUSION

However indentifying, validating and resending the dropped packets is typical but necessary task to minimize the loss of data. ARED allow us to identify and resend the data to their respective destinations by validating them with Chi-Square. Loop of this algorithm dynamically changes its parameter value to start the range of identification from beginning (0.01). Re-starting the loop takes time to change the parameter value and it depends on size of the dropped packets in queue. Due to change in average queue size the response time will be slow and infrequently changes. ARED is not optimal but it seems to work well in a wide range of scenarios and it also controls the proportional errors. ARED algorithm cannot identify the dropped packets if the packet(s) size is ranges from 0.01 to 0.5 (1% and 50% respectively). It also increases the range value in the next sprint of the algorithm but it cannot reach to 100% due to limit of link capacity. In future work, to explore the use of Adaptive RED in virtual queues with goal of providing minimal average queuing delays.

ACKNOWLEDGMENTS

We thank our thesis advisors, Associate professors D.Sagar, Khaja Ziauddin, T.P.Shekhar and Challa Thirupathi, for there Support and guidance. There energy and insight at all levels are a Constant inspiration.

REFERENCES

[1] X. Ao, Report on DIMACS Workshop on Large-Scale InternetAttacks,http://dimacs.rutgers.edu/Workshops/Attacks/internet-attack-9-03.pdf, Sept. 2003.

[2] R. Thomas, ISP Security BOF, NANOG 28, http://www.nanog.org/mtg-0306/pdf/thomas.pdf, June 003.

[3] [3] K.A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R.A. Olsson, "Detecting Disruptive Routers: A Distributed Network Monitoring Approach," Proc. IEEE Symp. Security and Privacy (S&P '98), pp. 115-124, May 1998.

[4] A.T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage, "Detecting and Isolating Malicious Routers," IEEE Trans. Dependable and Secure Computing, vol. 3, no. 3, pp. 230-244, July-Sept. 2006.

[5] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and Whisper: Security Mechanisms for BGP," Proc. First Symp. Networked Systems Design and Implementation (NSDI '04), Mar. 2004.

[6] J. Aweya, M. Ouellette, D. Y. Montuno and A. Chapman. Enhancing TCP Performance with a Loadadaptive RED Mechanism. International Journal of *Network Management*, V. 11, N. 1, 2001

[7] J. Aweya, M. Ouellette, D. Y. Montuno and A. Chapman.A Control Theoretic Approach to Active Queue Management. Computer Networks 36, 2001.

[8] J. Aweya, M. Ouellette, D. Y. Montuno and A. Chapman. An Optimization-oriented View of Random Early etection. *Computer Communications*, 2001,to appear.

[9] Brakmo, L., O'Malley, S., Peterson, L., "TCP Vegas: New Techniques for Congestion Detection and Avoidance," SIGCOMM'94

[10] Claffy, K., Braun, H-W., Polyzos, G., "A Parameterizable Methodology for Internet Traffic Flow Profiling," IEEE Journal on Selected Areas in Communications, March 1995.

[11] Eldridge, C., "Rate Controls in Standard Transport Protocols,"ACM Computer Communication Review, July 1992.

[12] Fall, K., Floyd S., "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," Computer Communication Review, July 1996.

[13] Eldridge, C., "Rate Controls in Standard Transport Protocols,"ACM Computer Communication Review, July 1992.

[14]] Adaptive RED: An Algorithm for Increasing the Robustness of RED'sActive Queue Management. Sally Floyd, Ramakrishna Gummadi, and Scott Shenker August 1, 2001.