

# The Need for Computer Network Connection Metrics

S TERRY BRUGGER  
University of California, Davis

October 29, 2007

## Abstract

Numerous metrics, or features, have been used to describe TCP connections. These features are frequently of interest to the network intrusion detection community, and occasionally the network engineering community. While researchers may understand these terms when used by others, there are no formal definitions of these terms such that two researchers looking at the same connection will necessarily calculate the same values for all the metrics. This paper will review a number of vague metrics that have appeared in published papers, why they are ambiguous, and how that ambiguity can cause problems when using them for analysis.

## 1 Introduction

Researchers frequently need to describe TCP connections, using metrics from the widely accepted “Duration” to the less commonly used, yet still understandable “URG packet rate”, through more obscure measures such as the “Wrong packet resend rate”. The advanced network intrusion detection community, which typically uses machine learning to attempt to identify malicious or anomalous connections, will use these metrics as the machine learning features (or attributes) that their methods train and recall on (Axelsson 2000). These metrics are also used by the network engineering community; however, in this arena, researchers tend to focus on the effects of a single metric (such as the “Interpacket latency”), on a device or network stack, at a time (Leland et al. 1994; Paxson and Floyd 1995; Racz et al. 2003). As such, these metrics tend to be better defined.

It is important that researchers agree on the definitions of these terms. This is somewhat easier in the network engineering community, where an assumption is typically made that TCP connections will behave to specification, and that misbehaviour is the result of correctable deficiencies in network stack implementations, or unforeseen interactions as networks scale in size. The lack of definitions is more problematic in the intrusion detection community, as attackers actively create connections that are outside protocol specifications in an attempt to break stack implementations, such as Christmas Tree Packets (Miller 2000). Since this misbe-

haviour may occur on such a small scale, relative to the total amount of traffic on the network, a difference in definitions may cause one researcher to measure a value of 0.01 for a metric, while another researcher measures a value of 0.02 – double the measurement of the first researcher.

This paper will begin by considering the effect of the viewpoint on the metrics we’re considering. Then we will look at the ambiguity in the term “TCP connection” itself. Next, we will review a selection of metrics that have been used in published works that are ambiguous, and how that ambiguity may cause problems when doing analysis. It’s worth noting that none of the works cited offer any guidance with respect to the questions we raise. We will limit our focus to first-order metrics of the connections. While the intrusion detection community frequently uses second-order (or calculated) metrics, such as “The number of new connections in the last  $n$  seconds” (Lee and Stolfo 1998; Lee et al. 1999a; Lee et al. 1999b; Lee and Stolfo 2000; Lee et al. 2000), we will not delve into those, as any ambiguity from those follows from the first-order metrics we do examine. Finally, we will encourage the reader to consider the formal definitions of TCP connection metrics that we have assembled, along with libpcap-based C code to capture them from pcap files.

## 2 Viewpoint

It turns out that one’s view of the network is rather important when considering TCP connection metrics. There are three general views one can have of network connections: from an endpoint, an active third-party observer, or a passive third-party observer. The reason this is important is that no single observation point has a comprehensive view of the connection. This section will look briefly at how each of these viewpoints differs from the others.

An endpoint is the source or destination host of a connection. The actions of an endpoint in response to any given TCP packet, given the current state of that connection, are well defined in RFC 793 (Postal 1981). An observer at this endpoint has the advantage of knowing what this state information is. While this is an important advantage, the major disadvantages come from coverage and complexity. A single endpoint can only see the connections to or from that one host (including broadcast traffic), hence, in order to get broad network coverage, a great number of hosts may need to be instrumented. Besides the additional effort this requires, it’s made harder by the second disadvantage: the complexity of instrumenting all the different operating systems that may appear on a given network. Certainly, the difficulty in instrumenting any given OS will vary depending on availability of the source, hooks provided for such instrumentation, and modularity of the code. However, even if we are given a kernel module (or equivalent) to do such instrumentation, it will likely not be acceptable to install it on many carefully managed production systems.

A third-party active observer is one that sits between the two endpoints of the connection, and has some control over that connection, for instance a firewall or intrusion prevention system. Such a viewpoint certainly has

better network coverage, and likely has useful state information as well, for example, most firewalls keep the state of current connections, and will only admit a packet if it is establishing a new connection, or part of an established connection. Most such devices also have the ability to log or otherwise output information about the connections they allow or deny. Unfortunately, the format of such logs varies widely from device to device, as does the level of detail provided about the connections. Some of the metrics we will consider below (such as the resend rate), aren't output by any firewalls or IPS devices that we are familiar with. Again, we could likely address the log format and metrics collected with internal access to the devices, but this is even less likely to happen, given the strict configuration management and device approval procedures required for such security sensitive equipment on many networks.

This leaves us with the third-party passive observer viewpoint. The advantage of this viewpoint is that it gives us the potential for coverage that the third-party active observer does, without the complications of tying into the internals of the device. It also allows for observation of networks without such devices (say, from an edge router), and allows us to use standard tools such as `tcpdump` (TCPDUMP Project 2006) to collect data. For these reasons, this is our preferred approach, and the viewpoint that we'll use for the rest of this paper. It is not without its disadvantages though:

1. We don't have the endpoint state, so even if we think a connection is established, if a host is rebooted, the connection may either suddenly be reset, or silently close.
2. We might miss packets because the observing machine can not keep up with the traffic on the wire.
3. Our observation may begin after some connections are already established, and some connections may continue after our observation ends.

Regardless of which of these techniques is used, none has a comprehensive viewpoint of all the connections on a given network. While either active or passive third-party might see a packet that gets dropped before making it to the end host, it is equally likely that such a packet drop occurs before even making it to the observation point. It is also worth noting that our primary interest is in internetwork connections (those that transit to or from the observed network). While there's no reason that a third-party observer couldn't watch the intranetwork connections through the trunking port of a switch, the engineering costs of capturing that much traffic are likely to be significant on contemporary high-speed networks.

### 3 Ambiguity in defining TCP connections

If we are interested in metrics on TCP connections, we must first have a good idea of what exactly a TCP connection is. For the purposes of this section, we will consider a TCP connection to just be the collection of packets that make up a single flow. While this seems easy by following

(Postal 1981) while observing a well behaved connection, many ambiguities result in the edge-cases.

The first ambiguity comes primarily from our viewpoint: what if we didn't start observing the link until after the connection started, or what if we stop observing the link before the connection ends? While our viewpoint may exaggerate this problem, it doesn't come solely from being a third-party observer. While a first-party observer may be able to utilize its host's TCP stack to determine if the host has initiated or accepted a new connection, what do we do with non-SYN packets that are not a part of an established connection? The host will just drop them, however we may want to collect metrics from them, as they may be part of some sort of nefarious activity, such as host fingerprinting. The question of when a connection ends is even more problematic, even for first-party passive observers: if many minutes have gone by since the last packet was observed in the connection, the observing host may consider the connection still open while the other host may have gone down (a half-open connection), which implicitly ends the connection, as the first host will discover if it attempts to send another packet.

Once we have an established connection, we will see packets that may or may not be part of the connection. For example, we may get the SYN flag set in an established connection – an obvious violation of protocol that should result in a RST. Is the illegal SYN packet part of the connection? What about packets with incorrect checksums? The destination will drop them and expect them to be resent. Do we count them as part of the connection? A couple related tricks frequently used to intrude onto a connection are sequence injection attacks and session hijacking. The first may be used as part of a social engineering attack by replacing some information in the connection with what the attacker wants the other party to see. The second attack is where an attacker tries to usurp a connection – for example a telnet connection after the real source has authenticated to the destination machine. For either of these attacks, if the attacker can not stop the flow of packets from the real source, the destination will end up seeing some repeated segments; if the attack fails, the destination will see a number of out of sequence packets. Either way, do we count these packets – which the destination host drops – as part of the connection?

What about when an error ensues? If an uncorrectable error occurs, the host is supposed to issue a RST to close the connection. Do we count that RST as part of the connection? What if instead of a RST, we get an ICMP error message, such as “Host not available”? On the subject of errors, what about connections that never make it to an ESTABLISHED state? Do we even count them as connections?

The purpose of this paper is to raise questions, but we should only worry about addressing these questions if they are relevant. Agreeing what packets are and are not part of a connection affects all other metrics about that connection. One of the most basic metrics is the packet count in the connection. This may be a small difference – for instance a difference of one if I count a RST packet as part of a connection and you don't. It can also be a big error though – for instance if the sender sends hundreds of more packets after the receiver sends a RST. These differences propagate

to all our other metrics, for example the count of the number of errors on a connection; if I consider the connection closed after the first error, I'll never get an error count of more than one, but if multiple packets get sent that each illicit a RST, you may count many more errors. Obviously, we won't be able to agree on anything meaningful about the network if I say I only saw one error and you say that you saw hundreds.

## 4 Ambiguous TCP connection metrics

There are numerous metrics one might use to characterize a TCP connection, for example the basic "Number of packets" measure discussed in the previous section. In this section we'll discuss a number of connection metrics which have been used by numerous researchers in the network intrusion detection community, how they might be interpreted differently by different researchers, and what effect this difference in interpretation might cause.

### 4.1 Number of data or control packets

Besides the total number of packets in a connection, sometimes we're interested in the number of data or control packets in a connection. This metric has been cited in Lee and Stolfo (1998), Lee (1999), and Singh and Kandula (2001).

Is a packet either a data packet or a control packet? Put another way, can we calculate either metric by subtracting the other from the total number of packets? Or might a packet be both? Since all the packets after the initial SYN should ACK a segment from the other host, in a manner all packets should be control packets. Certainly, the difference between these two interpretations will create a large difference in the calculated values we come up for the metrics. Why would anyone even use the second interpretation if it's always going to report 100% of the packets are control packets? Well, if the number of control packets is less than the total number of packets, it indicates that something odd occurred in the connection, such as someone attempting to exploit a flaw in the other host's TCP/IP stack. If an intrusion detection system is built with (or learns) this rule, then it's going to have a serious problem if use the other interpretation and tell it that only 50 of the 100 packets we counted in a given connection were control packets.

These metrics can have more subtle differences as well: is an ACK packet that doesn't include any data (that is, the length of the data portion of the packet is zero) a data packet, a control packet, or neither? Or, taking the opposite approach of above, maybe one system only counts the packets with the SYN, RST, or FIN flags as control packets, in which case we'll expect to see a control packet count of four (SYN, SYN-ACK, and two FINs) for a normal connection. That system is going to think something is seriously wrong if it's given some connection metrics that record 50% of the packets were control packets, like each of the four control packets had to be resent about a dozen times – maybe not an attack,

but certainly something about the network that should raise some red flags.

## 4.2 Resend rate

The resend rate is a common measure of the reliability of a network. Lee and Stolfo (1998), Lee (1999), Chittur (2001), and Singh and Kandula (2001) all used it as one of the metrics feeding their network intrusion detection systems as well. We contend that the resend count is a slightly more useful metric, as the rate can easily be found by dividing by the total number of packets in the connection<sup>1</sup>, and we don't have to worry about floating point error if we want to calculate more complex metrics, such as the resend rate over the past  $n$  connections. But how exactly do we count the number of resent packets in a connection?

For starters, we have a natural limitation from our viewpoint as a third-party observer. Ideally, we could observe a connection from both of the end hosts, in which case we could easily count how many times each of the hosts had to resend a given packet. Since we rarely, if ever, have this level of access, we must consider what we see monitoring the link between the hosts.

Do we only count packets as resent if we see the same packet multiple times? Or do we count the number of packets whose sequence numbers are less than the highest sequence number seen so far? That might be easy to do (and, in fact, is the approach used by the Ethereum network protocol analyzer (Ethereal, Inc. 2007)). Perhaps one should count the number of packets whose sequence numbers are less than the next logical sequence number we expect to see from the sender?

None of these solutions are foolproof given our position as a third party observer: keeping track of which packets have already been seen does not address packets that were lost between the sender and the observer. Counting packets with a sequence number less than the highest sequence number seen so far will count out-of-order packets as resends. Comparing instead against the next logical sequence number will only detect missing packets if we see a packet that follows it.

While no definition is perfect, what is more important is using a consistent definition. If one is attempting to compare the reliability of two networks, which in practice have the same average resend rate per connection, but one resend count is calculated by the first definition by an observer who doesn't see most of the duplicate packets, and the other is calculated by the third definition on data intensive connections (like file transfers) that allow it to easily see when packets were resent, will falsely show the second network to have a much higher resend rate (and hence, lower reliability) than the first network.

---

<sup>1</sup>At least we presume that these authors calculated the resend rate as the number of resends versus the total number of packets; one might also use the number of unique segments that were resent versus the total number of packets, this presents another ambiguity for consideration.

### 4.3 Number of establishment errors

One of the more popular, and most vaguely defined, connection metrics used by the network intrusion detection community is the number of establishment errors. This is used by Singh and Kandula (2001), Lee and Stolfo (2000), Lee et al. (2000), Lee and Stolfo (1998), Lee (1999), Chittur (2001), Lee et al. (1999b), Portnoy et al. (2001), and Dickerson and Dickerson (2000).

What exactly is an establishment error? Is it just a connection where we see a SYN packet, but that never makes it to the ESTABLISHED state? Can a connection that makes it to the ESTABLISHED state be considered to have had an establishment error? Can a connection have more than one establishment error? Can a connection have an establishment error after it reaches the ESTABLISHED state? Most of the work cited above was interested in the number of establishment errors over the past  $n$  connections or  $x$  seconds, so the use of this metric doesn't shed that much light on these questions. Some of these connections harken back to the question of exactly what is our definition for a TCP connection? If we don't consider something to be a connection unless it reaches the ESTABLISHED state, then that limits how we can interpret this metric.

With this large number of questions, it's easy to see how different definitions will cause confusion. If one researcher counts a single establishment error if they see a connection that does not make it to the ESTABLISHED state, and a different researcher counts the number of errors (bad checksums, incorrect TCP flags, exceeding window sizes, etc) on connections before they reach the ESTABLISHED state, comparing those two figures is comparing apples and oranges.

## 5 Moving forward

The goal of this paper has been to demonstrate how the term "TCP connection", as well as the various metrics that we might use for measuring that connection can be very ambiguous. That ambiguity can cause numerous problems if two researchers or systems are trying to talk in concrete terms about networks. The natural conclusion this leads to is that we need formal definitions for what these terms mean.

|                     |                   |                        |
|---------------------|-------------------|------------------------|
| Number of packets   | Duration          | Number control packets |
| Number data packets | Bytes transferred | Data bytes transferred |
| Urgent              | Resend            | Wrong resend           |
| Duplicate ACK       | Wrong ACK         | Wrong data packet size |
| Window exceeded     | Hole              | Connection errors      |
| Reset connection    | Other errors      | Disconnection errors   |
| Fragmented packets  | Bad fragment      | SYN-ONLY               |
| SYN-ACK             | Idle connection   | Half-open connection   |

Table 1: Connection metrics needing formal definitions.

Table 1 shows the metrics which we believe require formal definitions. We have attempted to create a set of definitions for these metrics, based on a reading of RFC793 from our third-party observer viewpoint. We've released this as a tech report available at <http://www.cs.ucdavis.edu/research/tech-reports/2007/CSE-2007-31.pdf>. While the proposed metrics are far too dry for a conference or journal paper, we hope this work will spur enough interest and discussion on the matter to further develop the proposed metrics into an IETF RFC. We have also implemented these definitions in libpcap (TCPDUMP Project 2006) based C code, available at <http://sourceforge.net/projects/pcaputils>. The code will read in a pcap (tcpdump) trace file and will output per packet and per connection metrics. We encourage readers to explore these resources and provide feedback on how these metrics can be improved.

## References

- Axelsson, S. (2000, March). Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ. of Technology, Göteborg, Sweden.
- Chittur, A. (2001). Model generation for an intrusion detection system using genetic algorithms. High School Honors Thesis, Ossining High School. In cooperation with Columbia Univ.
- Dickerson, J. E. and J. A. Dickerson (2000, July). Fuzzy network profiling for intrusion detection. In *Proc. of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society*, Atlanta, pp. 301–306. North American Fuzzy Information Processing Society (NAFIPS).
- Ethereal, Inc. (2007). Ethereal: A Network Protocol Analyzer. <http://www.ethereal.com/>.
- Lee, W. (1999). *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. Ph. D. thesis, Columbia Univ.
- Lee, W., R. A. Nimbalkar, K. K. Yee, S. B. Patil, P. H. Desai, T. T. Tran, and S. J. Stolfo (2000). A data mining and CIDF based approach for detecting novel and distributed intrusions. In H. Debar, L. Mé, and S. F. Wu (Eds.), *Proc. of Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000)*, Volume 1907 of *Lecture Notes in Computer Science*, Toulouse, France, pp. 49–?? Springer.
- Lee, W. and S. J. Stolfo (1998). Data mining approaches for intrusion detection. In *Proc. of the 7th USENIX Security Symp.*, San Antonio, TX. USENIX.
- Lee, W. and S. J. Stolfo (2000). A framework for constructing features and models for intrusion detection systems. *Information and System Security* 3(4), 227–261.
- Lee, W., S. J. Stolfo, and K. W. Mok (1999a, 9–12 May). A data mining framework for building intrusion detection models. In *Proc. of the*

- 1999 *IEEE Symp. on Security and Privacy*, Oakland, CA, pp. 120–132. IEEE Computer Society Press.
- Lee, W., S. J. Stolfo, and K. W. Mok (1999b, 15–18 August). Mining in a data-flow environment: Experience in network intrusion detection. In S. Chaudhuri and D. Madigan (Eds.), *Proc. of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, CA, pp. 114–124. ACM.
- Lee, W., S. J. Stolfo, and K. W. Mok (2000). Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review* 14(6), 533–567.
- Leland, W. E., M. S. Taqqu, W. Willinger, and D. V. Wilson (1994, February). On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Networking* 2, 1–15.
- Miller, T. (2000, 30 August). Intrusion detection level analysis of Nmap and Queso. Technical Report 1225, SecurityFocus.
- Paxson, V. and S. Floyd (1995, June). Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Trans. Networking* 3(3), 226–244.
- Portnoy, L., E. Eskin, and S. J. Stolfo (2001, 5–8 November). Intrusion detection with unlabeled data using clustering. In *Proc. of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, Philadelphia. ACM.
- Postal, J. (1981, September). RFC 793: Transmission control protocol.
- Racz, P. I., T. Matsuda, and M. Yamamoto (2003, 28–30 August). Contribution of the application, transport and network layers to the self-similarity of Internet traffic. In *Proc. of 2003 IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, Volume 2, pp. 760–763. IEEE.
- Singh, S. and S. Kandula (2001, May). Argus - a distributed network-intrusion detection system. Undergraduate Thesis, Indian Institute of Technology.
- TCPDUMP Project (2006). TCPDUMP public repository. <http://www.tcpdump.org/>.