Consistency of Network Traffic Repositories: An Overview

Elmer Lastdrager and Aiko Pras

University of Twente, The Netherlands
e.e.h.lastdrager@student.utwente.nl, a.pras@utwente.nl

Abstract. Traffic repositories with TCP/IP header information are very important for network analysis. Researchers often assume that such repositories reliably represent all traffic that has been flowing over the network; little thoughts are made regarding the consistency of these repositories. Still, for various reasons, the traffic capturing process may have missed packets. For certain kinds of analysis, for example loss measurements, such inconsistencies may lead to the wrong conclusions.

This paper proposes an algorithm to detect such inconsistencies, using the idea of "fake gaps". A prototype has been developed, and used to test two well-known repositories: the WIDE and Simpleweb repositories. The paper shows that both repositories contain several inconsistencies.

1 Introduction

A network traffic repository contains network traffic gathered from one or more location(s), often a router or backbone. Captured traffic is stored in data files in a repository; typically such files contain data captured over a longer period, for example minutes, hours or even days. Repositories can store different types of network data, for example TCP/IP header files, netflow records or SNMP packets. In this paper the focus is on the most common type: TCP/IP header data.

Using a repository can be very convenient for a researcher, as gathering data yourself can be very time-consuming, or even impossible. A potential issue in using a repository, is the consistency of the traffic inside the repository. When traffic inside a repository does not completely correspond with the actual traffic that was transmitted and received, this can influence measurements, analysis and therefore also conclusions that researchers draw. Hence, it is critical to have information about the consistency of the network traffic repository, so it can be taken into account when analysing the data.

Issues with consistency of the data in repositories have been reported by M. Timmer in [1]. While using a repository, it appeared that not all data was recorded properly. Timmer introduces the term "fake gap" to represent those parts of a TCP flow that are absent in the repository, although they were acknowledged at the TCP level. In [2] a relatively simple algorithm has been developed to find a sudden decrease in data in small intervals, which may indicate a problem with the repository. However it may very well be a temporary network

R. Sadre and A. Pras (Eds.): AIMS 2009, LNCS 5637, pp. 173–178, 2009.

problem and therefore does not necessarily affect the consistency of the repository. Although [2] has performed some initial research, the consistency analysis never exceeded a few data files. However, as researcher, knowledge about inconsistency in a repository is essential. At this moment, there are no statistics about possible inconsistency of repositories available. In this paper we will therefore analyse two well-known repositories: the WIDE and Simpleweb repositories. A tool has been developed that analyses TCP flows. We focus on TCP, because its state-full nature allows detection of fake gaps; for UDP this is, due to its stateless nature, not possible.

In this paper, the main question that will be answered is: *How can inconsistency be detected in a TCP traffic repository?*

To answer the main question, we first focus on detecting fake gaps by introducing an algorithm. The sub question we will answer is: *How can we detect fake gaps?*

Next, we built a prototype of the algorithm to test existing repositories in order to answer the second sub question: *How consistent are today's repositories?*

It should be noted that an earlier version of this paper has been presented at the tenth Twente Student Conference on Information Technology [3]. That conference is an internal conference of the University of Twente, of which the proceedings have not officially been published by a real publisher. This paper is short version of the earlier paper, and it is the first publication for an international audience.

The structure of this paper is as follows. In Section 2 we will propose an algorithm to detect inconsistency. In Section 3 we will discuss the results of testing two existing repositories using a prototype we built and finally in Section 4 we will answer our research questions, draw conclusions and discuss possible future work.

2 Detecting Inconsistency

In this Section, we will describe inconsistency called fake gaps and introduce an algorithm for detecting inconsistency. Although there may be different ways a repository can be inconsistent, we will only consider fake gaps.

2.1 Fake Gaps

In this paper's introduction, we said a fake gap to be representing those parts of a TCP flow that are absent in the repository, although they were acknowledged at the TCP level. To explain this more precisely we first have to consider a gap. To start with an example: when Alice wants to send some data to Bob, she sends ordered packets named A B C D E F. Bob may receive this as A B C E F **D**, but knows about the correct order and can therefore recreate the original message. This is called packet reordering. At the side of Bob, after having received C, there is a gap until D is received. In this example, the gap is filled when D is received.

We call a gap a fake gap, when one or more packets in a sequence of packets are not present, but are also not retransmitted; hence the original TCP flow is not affected. Consider Alice and Bob: Alice sends the sequence A B C D E F to Bob using TCP. Bob's network administrator records all traffic sent to Bob. According to the data recorded, Bob received A B C E F. When the connection between Alice and Bob is closed, we know Bob must have received D. We can then say that the recorded TCP flow between Alice and Bob contains a fake gap. All data was transmitted and received correctly, but the recorded data does not reflect this. Hence the recorded data is inconsistent. From now on, if we are referring to the flow of data packets within a TCP connection, we will abbreviate it to a flow.

In recorded traffic data, it is likely that there are flows that start and/or end outside the recorded time period. Therefore, detecting fake gaps in these flows is difficult. To avoid this, we only take flows into account that are sufficiently recorded. All packets of a single flow, from the first SYN-packet up to a FIN- or RST-packet should be recorded. If this is the case, we call the TCP flow a *usable flow*. Note that the final FIN-handshake can be partly outside the recorded time period.

2.2 Algorithm

In Section 2.1 we concluded fake gaps prove inconsistency. To detect this the algorithm we introduce, listed as Fig. 1, first extracts all usable flows from the complete set of packets. The second part of the algorithm loops over all usable flows. For every usable flow, it checks if there is an acknowledging packet that acknowledges a packet that has not yet been seen. If so, this indicates a fake gap and an identifier of the flow together with the packet that is used to detect the fake gap, is added to a list. So the final result of this algorithm is a list of flows combined with packets directly after the fake gaps.

It is important to see that this algorithm alone cannot detect the exact amount of fake gaps within the traffic dump. It can give an indication and show which usable flows are affected. If, for example, during a small interval no packets were recorded at the recording device, multiple flows can be affected by this. Our algorithm would detect a fake gap for each affected flow. We do not consider this a limitation. The results of our algorithm should provide a starting point for deep inspection of packets and flows.

3 Analysing Repositories

We made a prototype to analyse repositories using the algorithm from Section 2.2. We analysed two existing repositories using our prototype. The first repository we used was Simpleweb [4], maintained by the University of Twente. It was included in this paper because it is publicly available and easily accessible. We analysed a subset of data covering all 6 locations the Simpleweb repository provides, totalling to 224 data files. The second repository we included in this paper is the WIDE traffic repository [5], maintained by the MAWI working group. It contains data files with traffic of several trans-Pacific lines. We used one data file from samplepoint A to test our prototype and we analysed 27 data files from

```
INIT usableFlows to {}        # all usable flows found
INIT testFlows to {}          # all flows found so far
FOR each packet in data file
    IF packet is SYN
        CREATE flow from packet
        ADD flow to testFlows
    ELSE IF packet belongs to flow in testFlows
        SET flow to flowOf(packet)
        ADD packet to flow
        IF packet is FIN or RST
            REMOVE flow from testFlows
                ADD flow to usableFlows
END FOR
INIT fakeGaps to {}
FOR each flow in usableFlows
    FOR each packet in flow
        IF packet is ACK
            IF acked packets are not in this flow
                ADD tuple (flow, packet) to fakeGaps
    END FOR
END FOR
```

Fig. 1. Pseudo-code of the fake gap detection algorithm

samplepoint F, which were all over 1 GB in size. We chose samplepoint F since this one is daily updated while the other samplepoints are discontinued or were not accessible. The full table of results created by our prototype, and the prototype itself, can be found at [6]. Our prototype calculates fake gap statistics using various intervals. That is, fake gaps from different flows within this interval are grouped together and reported as a single fake gap.

Figure 2 shows the estimated number of missing packets per repository we tested and the estimated number of fake gaps within an interval of 0.05 seconds. The left part of the graph shows the extreme values. A first observation includes the presence of fake gaps in almost all tested data files from the WIDE repository. Also note that not all test data from analysing Simpleweb was plotted, instead we plotted the 28 highest values. It can be observed from the raw test data, that there is an absence of fake gaps in 50.8% of the tested data files of the Simpleweb repository. Data files from location 2 are almost solely responsible for this. A possible explanation could be the low amount of traffic at this location.

Figure 3 plots the percentage of affected usable flows that have at least one fake gap. For the WIDE repository, on average 0.49% of the usable flows is affected by at least one fake gap. When ignoring data files without fake gaps, the average percentage of affected usable flows is 0.51%. Of the tested data files of the Simpleweb repository, an average of 0.27% of the usable flows in all data files was affected by at least one fake gap. When ignoring consistent files, the average is 0.55%. The highest value is in the file loc2-20030718-1530, where 23.72% of the flows is affected by at least one fake gap.

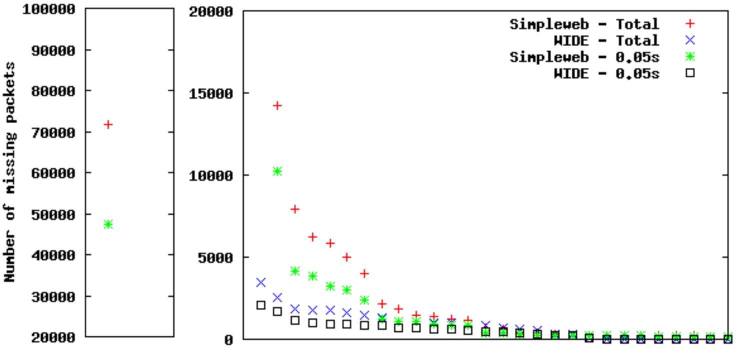

Fig. 2. Estimated number of missing packets and fake gaps

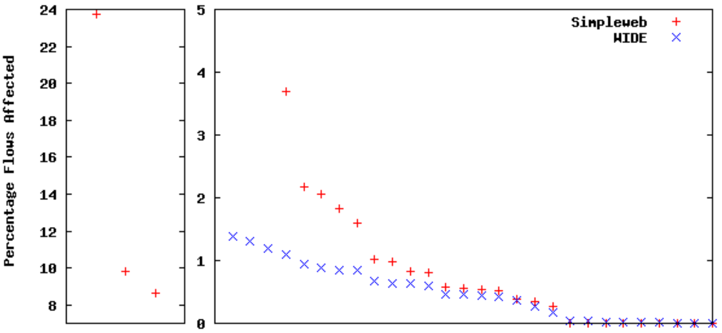

Fig. 3. Percentage of affected usable flows

4 Conclusions

This paper describes our research on the consistency of network traffic repositories. Before answering the main research question, we first look at two sub questions identified in Section 1.

The first sub question was *"How can we detect fake gaps?"*. We proposed an algorithm in Section 2, which extracts TCP flows. Then, it tries to identify fake gaps, packets that are not recorded by the recording device but were sent. The algorithm checks whether all data in the TCP flow is present by analysing TCP headers.

The last sub question, *"How consistent are today's repositories?"*, was answered in Section 3. We performed measurements on the Simpleweb [4] and WIDE [5] repositories. We showed both repositories contain inconsistencies. In the Simpleweb repository an average of 0.27% of the investigated TCP flows was affected by at least one fake gap. For the WIDE repository, this average was 0.49%. The research covered a substantial subset of data from both repositories.

We analysed 28 data files from the WIDE repository and 224 data files from the Simpleweb repository.

Going back to the main research question, *"How can inconsistency be detected in a TCP traffic repository?"*, we can now conclude detecting inconsistency is possible by using the proposed algorithm, which detects fake gaps. The knowledge that a repository is not always consistent is very important for research where it is critical to have all data recorded, like research on packet loss. For this kind of research, it is recommended to take possible inconsistency in the repository into account and, if no statistics are present, analyse the repository data before using it.

Future research could include extending the proposed algorithm to support TCP flows that are not completely present in the data file. This research can be used together with algorithms like the one described in [2], which checks for anomalies in traffic rate, to find the exact locations of fake gaps. This can, in turn, be used to draw conclusions about non-TCP traffic, thereby getting a better overview of the consistency of a network traffic repository.

References

1. Timmer, M.: How to identify the speed limiting factor of a TCP flow, `http://dacs.ewi.utwente.nl/assignments/completed/bachelor/reports/B-assignment_Timmer.pdf` (retrieved October 5, 2008)
2. Slomp, G.: Consistency of repositories. Presented at: 8th TSConIT, `http://referaat.cs.utwente.nl/new/paper.php?paperID=377` (retrieved October 5, 2008)
3. Lastdrager, E.E.H.: Consistency analysis of network traffic repositories. Presented at: 10th TSConIT, `http://referaat.cs.utwente.nl/new/paper.php?paperID=464` (retrieved February 20, 2009)
4. van de Meent, R., Pras, A.: Simpleweb/University of Twente – Traffic Measurement Data Repository, `http://traces.simpleweb.org` (retrieved October 5, 2008)
5. Cho, K., Mitsuya, K., Kato, A.: Traffic data repository at the WIDE project. In: Proc. USENIX Annual Technical Conference, p. 51 (2000)
6. Lastdrager, E.E.H.: Prototype and results, `http://www.vf.utwente.nl/~lastdragereeh/referaat`