Assessing Unknown Network Traffic

Remco van de Meent Aiko Pras

r.vandemeent@utwente.nl pras@ctit.utwente.nl

19th February 2004

University of Twente, Enschede
The Netherlands

Abstract

Recent measurements have shown that a growing fraction of all Internet traffic is *unknown*: it is unclear which applications are causing the traffic. Therefore we have developed and applied a novel methodology to find out what applications are running on the network. This methodology is based on the notion of "induced traffic": traffic cannot (wide-scale) be on unknown ports, thus, the hypothesis is that such traffic on unknown ports should be preceeded by traffic on known ports between the same peers. We have developed and implemented an algorithm to test this hypothesis. After applying the algorithm in two case studies we, unfortunately, have to conclude that although some improvement is made, there is still a significant fraction of traffic unidentifiable.

1. Introduction

In the early ninetees and before, the major part of traffic caried over the Internet was email. With the rise of the World Wide Web with all its multimedia features, came an incredible increase of Internet traffic. Most of it was HTTP traffic, the protocol most commonly used to transmit web-pages. Traffic is identified by looking at the transport protocol (i.e., UDP or TCP) port numbers. For example, port 80 is generally used to transmit web-pages, and port 25 is used to transport email.

Recent measurements of Internet traffic show, however, that a growing fraction of all network traffic is *unknown*, i.e., it seems not possible to tell what applications cause this traffic. The fraction of unknown traffic can be up to tens of percents of all traffic [6]. In order to effectively run networks, a good understanding of what is going on on the network is required. Hence, a situation where a significant proportion of all traffic is unidentifiable is undesirable.

1.1. Related Work

Off the record, some people suggest the unknown traffic might be caused by peer-to-peer (P2P) applications such as Napster or KaZaA, chat applications such as MSN or IRC, or multimedia streaming applications, but there is little evidence supporting this theory.

In the Internet2 NetFlow Statistics [6] initiative, routers on the Abilene network are monitored via NetFlow. The output data is processed to give an indication of, among other things, what applications are used on the network. These applications are identified by looking at the transport protocol port numbers. Up to tens of percents of all traffic remains unknown.

As part of their NG-MON project, Wong *et al.* [4] have looked at analysing network traffic to identify P2P traffic flows. Their recognition algorithm is based on the use of "frequently used ports" per application. This algorithm is able to identify considerable amounts of P2P traffic, although a non-negligible fraction of traffic remains unknown.

Worth mentioning is also the traditional way of recognizing applications: purely based on the transport protocol port numbers. Well-known listings are the port lists by IANA [2] and Graffiti [1].

1.2. Contribution

In this study we assess network traffic, in order to find out whether it is possible to reduce the fraction of unknown traffic, *without looking into the payload of the IP packets carrying the traffic*, because of obvious privacy concerns.

Doing so, we present a novel methodology of correlating traffic flows to each other. Our hypothesis is that certain type of traffic induces other traffic, for instance, an FTP (control) connection induces a data transfer, which is handled via a separate connection on other ports.

1.3. Approach

The approach is threefold: First we determine the amount of unknown traffic, by comparing traffic to the well-known transport protocol ports list from IANA [2]. Second we add entries to this list from various other sources. And third we *we assess unknown traffic by relating it to preceeding, known traffic*. This is motivated by our expectation that traffic is always caused by either (i) a human action; or (ii) a computer program; or (iii) earlier traffic. Reasons (i) and (ii) are obvious.

With regard to reason (iii), an example will help to understand: in an FTP session, before the actual file transfer, both peers negotiate with each other which TCP ports will be used to transmit the file. For FTP, this is a well-known and documented process. We assess the presence of a similar process for other applications of which it is not known whether they perform some negotiations to transfer

future data on other ports. Because of the enormeous amount of data, this “relating process” has been automated.

If we can relate unknown traffic to known traffic in this manner, we can say that the unknown traffic is no longer unidentified, but is induced by known traffic. Thus we are able to decrease the fraction of traffic that is not accounted for (i.e., unknown).

1.4. Organization

The remainder of this report is organized as follows: Section 2 details our methodology of relating unknown traffic to known traffic introduced above. Section 3 covers a prototype implementation of this methodology using Java and MySQL. We have applied the methodology in two case studies, to evaluate its effectiveness; the results are presented in Section 4. We provide some concluding remarks and topics for possible future research in Section 5. Some raw numbers illustrating the application recognition process are presented in the appendix.

2. Methodology

This section presents the methodology that has been developed and applied using traffic measurements performed on two different networks. The methodology itself is generically applicable; in this work we just describe a case study.

2.1. Data Collection

In this work we have used packet traces from measurements on two different networks (also see [7]):

- The *Campusnet*, the University of Twente’s residential network as described in [5], connecting about 2000 students to the Internet.
- The network of a *research institute* in the Netherlands, employing about 200 researchers and supporting staff

These networks have been chosen because of their different user population, although they should not be seen as respresentative for all networks.

The measurement process consists of two steps:

Table 1: Measurement PC Configuration

Component	Specification
CPU	Pentium-III 1 GHz
Mainboard	Asus CUR-DLS (64 bit 66 MHz PCI)
Hard disk	60 + 160 Gigabyte, UDMA/66
Operating system	Debian Linux, 2.4.19-rc1 kernel
Network interface	1 x Gbit/s Intel Pro/1000T
Main memory	512 MB reg. SDRAM

1. *Capturing and storing the headers of all packets* (i.e., the first 64 octets of each frame, preserving all information up to the transport (UDP or TCP) header, whilst ignoring the payload) flowing in and out of the network. The configuration of the PC used to capture packet headers is given in Table 1.
2. Processing this raw data with a custom made tool, *combining packets belonging to the same flow* (see next section for details).

Note that for privacy reasons, the *packet traces have been made anonymous* using the tcpdpriv [3] utility, i.c., scrambling all headers while preserving the same mapping between packets, which is needed to determine which packets belong to the same flow.

2.2. Flows

Traffic traces may be very voluminous, and contain more information than we need in order to assess the composition of the traffic in terms of applications. For example, we don't need information on all single packets that, taken together, form a file-transfer. Instead, we abstract from individual packets, and use the concept of *flows*. The definition of a flow is somewhat arbitrary, but in this work we use the common 5-tuple definition:

Flow:

(source IP address, destination IP address, transport protocol, source port, destination port)

Thus packets that share the same values for all of these five properties, belong to the same flow. Note that this definition closely resembles the notion of a TCP connection. One remark has to be made: if such packets are too long apart (in time), we consider them as belonging to separate flows. The maximum gap between packets to belong to the same flow, i.e., the *flow timeout* value, has been set to 20 seconds in this research. It is possible that packets that in fact belong to the same TCP connection, are considered to belong to different flows, when the connection is stalled (i.e., no single packet for this connection is transmitted) for more than 20 seconds, which happens only very rarely in practice.

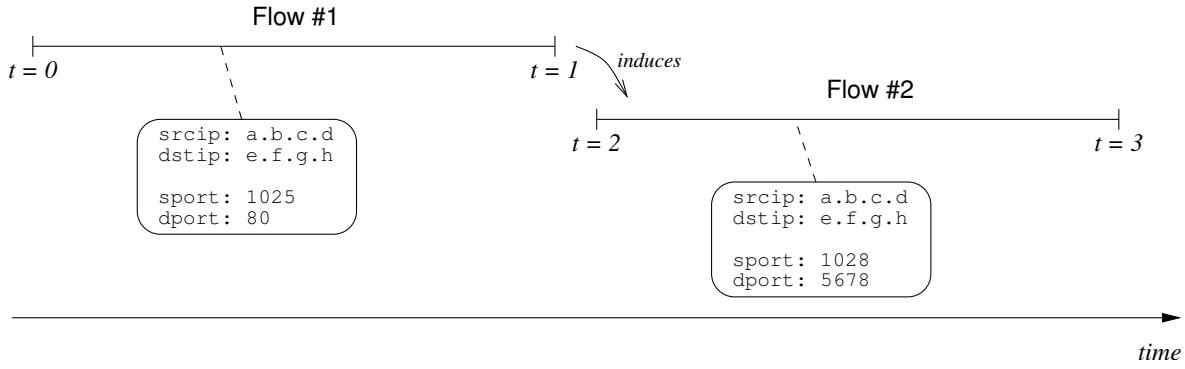

Figure 1: Flow #1 induces Flow #2

In the remainder of this report, when we refer to "traffic" we actually mean not raw traffic data, but traffic flows following the concept described above.

In order to identify traffic, we look at the *transport protocol destination port*. For example, packets in a TCP connection that are sent to port 80, regardless of their source ports, are all regarded as being web-traffic.

2.3. Traffic induces other traffic

In the Introduction section, we gave an example of traffic that induces other traffic: FTP. In Figure 1 a generic scenario is depicted: two peers[1] transmit data in flow #1, and after a while a new flow, on different ports but between the same peers, starts transmitting data.

Of course there is no way to guarantee that flow #2 is really induced by flow #1, at least not without looking into and analysing the payload of the traffic. Therefore we resort to statistics; if and only if the specific combination of (destination) port numbers in flows #1 and #2 occurs "often", we classify the second flow as being induced by the first, and as such no longer as unknown traffic.

One can think of other scenarios of different, possibly related flows between two peers:

- a flow that started and/or (too?) long before the unknown flow started
- a flow that started before, but did not end before the unknown flow started
- multiple (known) flows, before an unknown flow
- etc.

With all these different scenarios, the basic methodology remains the same.

[1] With a peer is meant the IP entity on a host, thus excluding the upperlying transport protocol.

3. Implementation

3.1. General

The methodology presented in the previous section has been implemented in a tool (see also [7]). The tool has been written in Java; the traffic (flows) on which it operates are stored in a MySQL database – because of the amount of data accesses while relating flows to each other, a fast system to access the data is required, a need better suited by a database than a flat file.

3.2. Parameters

There are two parameters that are important in the process of assessing network traffic: i) the port list that is used to identify (known) traffic; and ii) the amount of time that two flows between the same peers can be apart whilst still being possibly related.

The following well-known port lists have been used in order to see what destination port numbers corresponds to which applications:

Name	**Resource**	**Amount of ports**
Graffiti	http://www.graffiti.com/services	4427
IANA (<1024)	http://www.iana.org/assignments/port-numbers	505
IANA (all)	http://www.iana.org/assignments/port-numbers	2749
Own	Collected from the Internet via various resources	69

The maximum amount of time between two flows to be possibly related might typically be set in the range of 1–20 seconds.

3.3. Algorithm

In pseudo-code, the algorithm looks like this:

```
for all flows in DB {
  Flow A = GET flow from DB;
  if (portlist.isKnown(A)) {
    countAsIdentified(A);
  }
  else {
    Flow B[] = GET flows from DB between A.src_ip & A.dst_ip
                & satisfy time constraints;
    for (int i;i<B.lenght;i++) {
```

```
      if (portlist.isKnown(B[i])) {
        countAsIdentified(A);
        accountedTo(B,A);
      }
    }
  }
}
```

Explanation for all flows in the database, the destination port number is lookup in in a well known ports list. If this does not result in identification, other flows between the same peers are selected from the database, and if such a selected flow is identifiable, the original (unidentifiable) flow is related to the selected flow that is indentifiable.

As mentioned before, this algorithm has been implemented in the Java programming language and the datasets with flow information are stored in a MySQL database.

4. Results

We have applied the methodology of relating unknown traffic flows to known traffic flows in two case studies, using the various port lists and a "time window" of 10 and 30 seconds. The raw numbers are provided in Appendix A. Not that a distinction is made between i) relating an unknown flow to the last flow between the same peers, and ii) relating an unknown flow to all previous flows between the same peers within the set time window.

4.1. Discussion

We discuss the influence of the two parameters in the algorithm "port list" and "time window", and the distinction between looking back at only one, and all previous flows.

Note that specific results of the application of this methodology can be found at: `http://m2c-a.cs.utwente.nl/bsc-analysis/appB`. Detailed information for single measurements can be found at: `http://m2c-a.cs.utwente.nl/bsc-analysis/new`.

Changing the port list

It is expected that a larger fraction of the traffic is identified if the port lists become more extensive. This is confirmed by the numbers in the table.

However, using extensive port-lists introduces a new problem. It increases the chance of *false positives*. It is very doubtful if all the identified traffic is really generated by the application it is accounted

to. Careful analysis of the data indeed shows the occurence of false positives, e.g., almost at location 2 almost all traffic can be identified when all the port lists are used together; however, as it turns out, a large fraction of the traffic stems from one single port, and this traffic is *not* generated by the application the port lists accounts it to.

Changing the time window

Increasing this time results in more identified traffic. This is what was expected. The algorithm relates more unidentified flows to identified flows. This is because the chance of a flow between two computers increases when making the timewindow larger. Here the problem of false positives also exists. When making the time window too large, the algorithm might relate flows to each other that in real life had noting to do with each other.

Relate to last, or all previous flows

Results are highly influenced by the choice to which flows the algorithm relates the unidentified flows to. When the algorithm only relates to the last flow between the same peers, obviously, fewer flows are identified than relating to all previous flows.

Without detailed investigations, it is hard to say which approach is best. The chances of finding false positives are significant, but to detect a false positive is (almost) impossible without looking into the payload of individual packets.

5. Conclusions

Summerizing the conclusions from this study:

- Extensive port lists can be used to identify traffic, but the chances of false identifications are non-negligible. The more extensive the port list, the higher the fraction of identified traffic, and the higher the probability of false positives.
- Our novel approach of relating unknown traffic to previous, known, traffic gives some improvement in the fraction of identifiable traffic. The improvement, however, is rather marginal. The exact cause of this is yet unknown, but we intend to perform further study into this. One possibility is that “induced flows” are not between the same peers as the “causing flows” (e.g., a third party governing communications in a P2P network), for which our current algorithm will fail.
- The methodology presented in this report cannot be applied in real-time, due to the computational complexity. However, this is probably not a significant problem, as getting an idea of what applications are running on a network does not demand real-time analysis.

Acknowledgments

An important part of the work presented in this report has been carried out by Vincent Gaiser, a Telematics student at the University of Twente, under supervision of Remco van de Meent and Aiko Pras.

This research has been carried out as part of the *Measuring, Modelling and Cost Allocation (M2C)* project, which is sponsored by the Dutch Telematica Instituut. Partners in this project are the University of Twente (through its CTIT research institute) in Enschede, and the National Research Institute for Mathematics and Computer Science (CWI) in Amsterdam.

References

[1] Graffiti. Port Numbers, 2003. `http://www.graffiti.com/services`.

[2] IANA. Port Numbers, 2003. `http://www.iana.org/assignments/port-numbers`.

[3] Ipsilon Networks. tcpdpriv, 1997. `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`.

[4] Myung-Sup Kim, Hun-Jeong Kang, and James W. Hong. Towards Peer-to-Peer Traffic Analysis Using Flows. In M. Brunner and A. Keller, editors, *Proceedings of the 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM2003)*, number 2867 in Lecture Notes in Computer Science (LNCS), pages 55–67, October 2003.

[5] Remco Poortinga, Remco van de Meent, and Aiko Pras. Analysing campus traffic using the meter-MIB. In *Proceedings of the Passive and Active Measurement Workshop (PAM2002)*, pages 192–201, Fort Collins, Colorado, U.S.A., March 2002.

[6] Stanislav Shalunov. Internet2 NetFlow Statistics, 2003. `http://netflow.internet2.edu/`.

[7] Remco van de Meent. M2C Measurement Tools. Technical report, University of Twente, December 2003. M2C Project Deliverable 1.6.

A. Results of applying the algorithm

All numbers are percentages.

Portlists:	*IANA (< 1024)*		*Portlists:*	*IANA (< 1024)*	
Time window:	*10 seconds*		*Time window:*	*30 seconds*	
Location 1	identified flows	identified data	**Location 1**	identified flows	identified data
w/o algorithm	61.147	57.054	w/o algorithm	61.147	57.054
relate to last	61.725	64.456	relate to last	61.911	65.635
gain	0.578	7.402	gain	0.764	8.581
relate to all	63.231	68.205	relate to all	63.962	69.065
gain	2.084	11.151	gain	2.815	12.011
Location 2	identified flows	identified data	**Location 2**	identified flows	identified data
w/o algorithm	66.419	25.234	w/o algorithm	66.419	25.234
relate to last	66.610	27.123	relate to last	66.693	27.370
gain	0.191	1.888	gain	0.274	2.136
relate to all	67.209	29.025	relate to all	67.716	30.206
gain	0.790	3.791	gain	1.297	4.971

Portlists:	*IANA (all) and*	*Graffiti*	*Portlists:*	*IANA (all) and*	*Graffiti*
Time window:	*10 seconds*		*Time window:*	*30 seconds*	
Location 1	identified flows	identified data	**Location 1**	identified flows	identified data
w/o algorithm	78.671	67.082	w/o algorithm	78.671	67.082
relate to last	79.247	73.929	relate to last	79.416	74.852
gain	0.576	6.847	gain	0.745	7.770
relate to all	80.722	76.582	relate to all	81.338	77.168
gain	2.051	9.499	gain	2.667	10.085
Location 2	identified flows	identified data	**Location 2**	identified flows	identified data
w/o algorithm	82.548	90.331	w/o algorithm	82.548	90.331
relate to last	82.791	92.178	relate to last	82.958	92.381
gain	0.243	1.847	gain	0.410	2.050
relate to all	83.380	93.042	relate to all	83.859	93.753
gain	0.832	2.711	gain	1.311	3.422

Portlists:	*Own*		*Portlists:*	*Own*	
Time window:	*10 seconds*		*Time window:*	*30 seconds*	
Location 1	identified flows	identified data	**Location 1**	identified flows	identified data
w/o algorithm	72.098	39.684	w/o algorithm	72.098	39.684
relate to last	73.143	43.925	relate to last	73.382	45.537
gain	1.046	4.241	gain	1.284	5.853
relate to all	75.012	47.326	relate to all	76.184	48.282
gain	2.914	7.643	gain	4.087	8.598
Location 2	identified flows	identified data	**Location 2**	identified flows	identified data
w/o algorithm	60.788	19.914	w/o algorithm	60.788	19.914
relate to last	60.850	21.545	relate to last	60.877	21.642
gain	0.061	1.630	gain	0.088	1.728
relate to all	61.006	21.771	relate to all	61.228	21.997
gain	0.218	1.856	gain	0.439	2.083

Portlists:	*IANA (<1024) and*	*Own*	*Portlists:*	*IANA (<1024) and*	*Own*
Time window:	*10 seconds*		*Time window:*	*30 seconds*	
Location 1	identified flows	identified data	**Location 1**	identified flows	identified data
w/o algorithm	82.254	60.286	w/o algorithm	82.254	60.286
relate to last	83.250	67.723	relate to last	83.445	68.902
gain	0.996	7.437	gain	1.191	8.615
relate to all	84.992	71.508	relate to all	85.838	72.404
gain	2.738	11.221	gain	3.584	12.117
Location 2	identified flows	identified data	**Location 2**	identified flows	identified data
w/o algorithm	67.714	25.506	w/o algorithm	67.714	25.506
relate to last	67.900	27.394	relate to last	67.984	27.642
gain	0.186	1.888	gain	0.270	2.136
relate to all	68.490	29.296	relate to all	69.026	30.482
gain	0.776	3.790	gain	1.312	4.976

Portlists:	*All*		*Portlists:*	*All*	
Time window:	*10 seconds*		*Time window:*	*30 seconds*	
Location 1	identified flows	identified data	**Location 1**	identified flows	identified data
w/o algorithm	86.454	67.703	w/o algorithm	86.454	67.703
relate to last	87.377	74.556	relate to last	87.549	75.479
gain	0.924	6.853	gain	1.095	7.776
relate to all	88.960	77.243	relate to all	89.636	77.853
gain	2.506	9.540	gain	3.182	10.150
Location 2	identified flows	identified data	**Location 2**	identified flows	identified data
w/o algorithm	82.549	90.331	w/o algorithm	82.549	90.331
relate to last	82.792	92.178	relate to last	82.959	92.381
gain	0.243	1.847	gain	0.410	2.050
relate to all	83.381	93.042	relate to all	83.860	93.753
gain	0.832	2.711	gain	1.311	3.422