Traffic Analysis of E-Collaboration and E-Learning Applications used at ETH Zurich

Benedikt Köppel
bkoeppel@ee.ethz.ch

supervised by Dr. Wolfgang Mühlbauer
wolfgang.muehlbauer@tik.ee.ethz.ch

ABSTRACT

In the modern days at university, researchers use different web applications to collaborate in their teams. Students use similar applications to obtain lecture notes, additional reading materials and hand in their solutions to assignments.

In this paper usage patterns of those applications are studied. Section 2 describes how the underlying data was collected and the characteristics were obtained. In section 3, the patterns are discussed and results presented. Detailed graphs can be found in the appendix.

1. INTRODUCTION

Traffic analysis is done for various reasons such as traffic classification, anomaly detection and profiling. When classifying traffic, researchers and network operators want to find out about the services that are operated on their networks and what characteristics those services have. Anomaly detection, the second objective of traffic analysis, deals with intruders and attacks. One wants to detect unusual usage patterns of the network and conclude whether attacks are currently being performed and if hosts on the network are compromised. Thirdly, network profiling is done to measure the loads and usage of network links.

In this paper, we study three different e-learning and e-collaboration services of ETH Zurich. BSCW is a web application for team collaboration. Researchers of ETH Zurich and other European universities use BSCW to share their research documents and work on new papers. Moodle and ILIAS [1] are both web-based e-learning applications where lecturers and teachers can distribute problem sets, assignments and additional course material. Students use Moodle and ILIAS to hand in assignments. Each application is operated on a separate server at ETH Zurich and is studied in the following individually.

We have chosen those three services because they are well known at ETH and more and more courses are getting supported by our e-learning applications. In order to provide the needed performance, stability and security for more students, we will need to upgrade servers and infrastructure. To do that, it will be very helpful to understand how the services are used and how the network usage is characterized.

Additionally, the packet level traces give us the chance to detect anomalies and attacks of our servers. By measuring the network activities permanently, one could implement an intrusion detection system based on this data.

2. PROCEDURES AND TOOLS

As the basis for the traffic analysis, we had to obtain packet level traces of the services. There is a variety of tools to collect and analyze such traces. For this project, we have used `tcpdump` [2] to record the packet level traces. `tcpdump` was run on each application server individually and sniffed the headers of all Ethernet frames on the server's Ethernet interface. Only the first 68 bytes of the payload were captured to minimize the total amount of data. The tool was run during one week and collected a total of 52 GB of data. To analyze this amount of data, one needs to choose and design the tools carefully. To minimize memory and computational resources, we decided to split all traces into files of 100 MB size each. This was done by setting `tcpdump`'s option `-C` to 100 MB.

Initially, we had the idea to use command line utilities such as `sed`, `grep`, `sort` and `uniq` to analyze the packets. However it turned out that especially `sed` and `sort` do not perform well with large amounts of data. Therefore, we had to design the tools for the analysis more carefully. We developed a series of Perl scripts.

The process was divided into three separate steps. In the first step, each file was analyzed individually by the first Perl script. The script loads the dumped file by using Perl's `Net::TcpDumpLog` [3] module and then extracts the protocols, IP addresses and ports, packet sizes and the total number of packets per IP of the trace. Those characteristics are aggregated over a 10 minute window and then stored to text files using Perl's `Data::Dumper` [3] module.
This approach has the advantage that not all traces have to be analyzed at the same time and on the same host. It took around 3 to 5 minutes to process one file of 100 MB. To speed up the whole first step, we have distributed all traces to four workstations and let the script run during one night in parallel on those hosts.

In the second step, all the written text files from the first step were re-evaluated and aggregated to obtain continuous characteristics over the whole period of one week. This requires much less computational effort, because the files written by `Data::Dumper` [3] can be read in by using Perl's `eval()` method. Once we had computed the individual characteristics in the first step, it took us less than one hour in total to aggregate all characteristics of all services.

In the last step, the aggregated data was plotted using `gnuplot` [4]. This tool deals very well with big data sets. It has the advantage that the data does not have to be sorted, because `gnuplot` can draw points on a 2D graph in an arbitrary order. The drawn plots can be found in the appendix.

For this project, no special actions to guarantee privacy and anonymity of the data in the traces have been taken. Instead, only the appended plots are distributed publicly but not the traces itself.

During the capturing of the traces, parts of the dumped files were already moved to another host via SSH. In addition to that, disk accesses to NFS disks and the transfer of log files to the central logging server also generated some traffic overhead. All this traffic, which is not of interest for this project, has been filtered out in the first Perl script.

During the night of the 27th October, no traces could be collected due to server maintenance and upgrades. Thus, there is no data available within this period.

3. RESULTS

The measured data had a traffic mix of 99.52% TCP and 0.28% UDP. For the following plots, we considered only incoming TCP connections. All applications are plotted separately. The data is aggregated in windows of 10 minutes. The detailed plots of the characteristics can be found in the appendix.

The first thing to notice is the daily pattern. In all graphs, one can clearly see that there is a higher usage during the day and a lower usage in the night. In average, there are around 750 to 1000 flows per 10 minutes during the day, while we have only a few ones in the night hours. If the number of flows would significantly increase in a very short time period, this could be a hint to a virus outbreak on the host. This has been done before in the DDoSVax project at ETH Zurich [5].

An interesting characteristic is the number of source IPs. This gives a good impression of how many users are concurrently accessing the web applications. For BSCW and ILIAS, there are between 10 and 20 users during the day. Moodle has a maximum of up to 50 concurrent users during the day hours. Even in the night there are some source IPs accessing all three services. This does not necessarily mean that those requests come from humans, they could also stem from automated robots, for example crawlers of search engines and our internal monitoring agent.
In the future, we expect a much higher number of users on our system, because there will be e-assessments and online exams for whole classrooms soon.

By considering the plot of the throughput, there is an average of around 1 Mbps of incoming traffic within 10 minutes. Still, temporal maxima could be much higher than 1 Mbps. Plots such as this one could be used for network profiling purposes.

When comparing throughput and the number of packets, there are a few obvious peeks visible. There can be different reasons for those peeks. It could be legitimate traffic, for example when students are handing in assignments shortly before a due date. On the other hand, these peeks could also reveal attackers.
When people are monitoring throughput and the number of packets during a virus outbreak, one can clearly see a rise in those characteristics. They can thus be used for profiling purposes, but also for anomaly detection.

For the last plot, we split the numbers of packets into three groups: packets originating in Switzerland, those from the European Union, and the rest. To resolve the IP addresses to a geographical location we used the hostip.info service [6]. This service provides a simple HTTP API, which can be accessed easily with Perl.
Most of the packets come from Switzerland; packets from EU play the second role. As all analyzed web applications are mainly targeted at students in Zurich in the case of Moodle and ILIAS, respectively European research groups in the case of BSCW, these patterns are not surprising.

Besides the plots shown here, we could look at many other interesting characteristics. By looking at entropies of source IPs or ports, one could detect anomalies. Low entropy means that a lot of traffic originates from one IP or is targeted to one destination port. This would identify different viruses and worms [7].
On the other hand, interarrival times and flow lengths would give more information about the browsing habits of the users. With such statistics, one could see how long a user remains active on the application and how many requests one makes during a certain time interval.
These characteristics were not studied for this project but could be other interesting fields for further analysis. Because the initial packet level traces contain all needed information, one could easily modify the Perl scripts and obtain such statistics in short time.

4. CONCLUSION

While working with 52 GB of data, we experienced different problems and difficulties. The handling of such large files is cumbersome and requires a lot of computer power for the analysis. Collecting packet level traces of a core router would result in even bigger dump files. In traffic analysis, it is always a crucial point to be able to deal with large data sets efficiently.

Still, by using carefully designed tools, we were able to extract interesting characteristics within a short time. Those characteristics will help us plan and design the next generation of e-learning and e-collaboration services, which will be able to deal with much larger numbers of users.

5. REFERENCES

[1] BSCW, Moodle and ILIAS at ETH Zurich
`http://www.let.ethz.ch`

[2] `tcpdump`, a tool to obtain packet level traces
`http://www.tcpdump.org`

[3] The Perl scripts are based on Perl's `NetPacket::Ethernet`, `NetPacket::IP`, `Net::TcpDumpLog` and `Net::Dumper` modules.
`http://www.cpan.org`

[4] `gnuplot`, a tool to plot big datasets efficiently
`http://www.gnuplot.info`

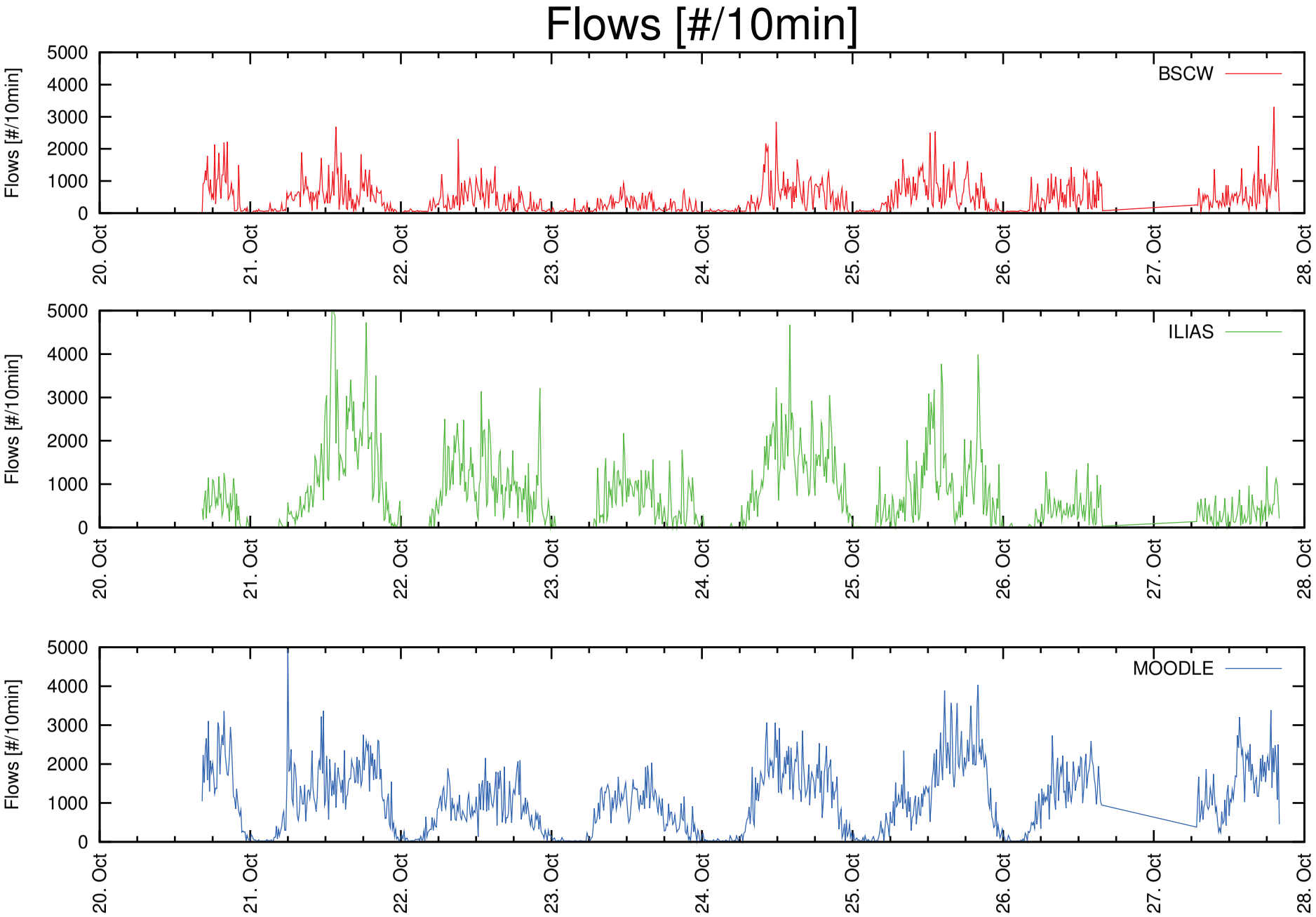
[5] DDoSVax Project, ETH Zurich
`http://www.tik.ee.ethz.ch/~ddosvax/`

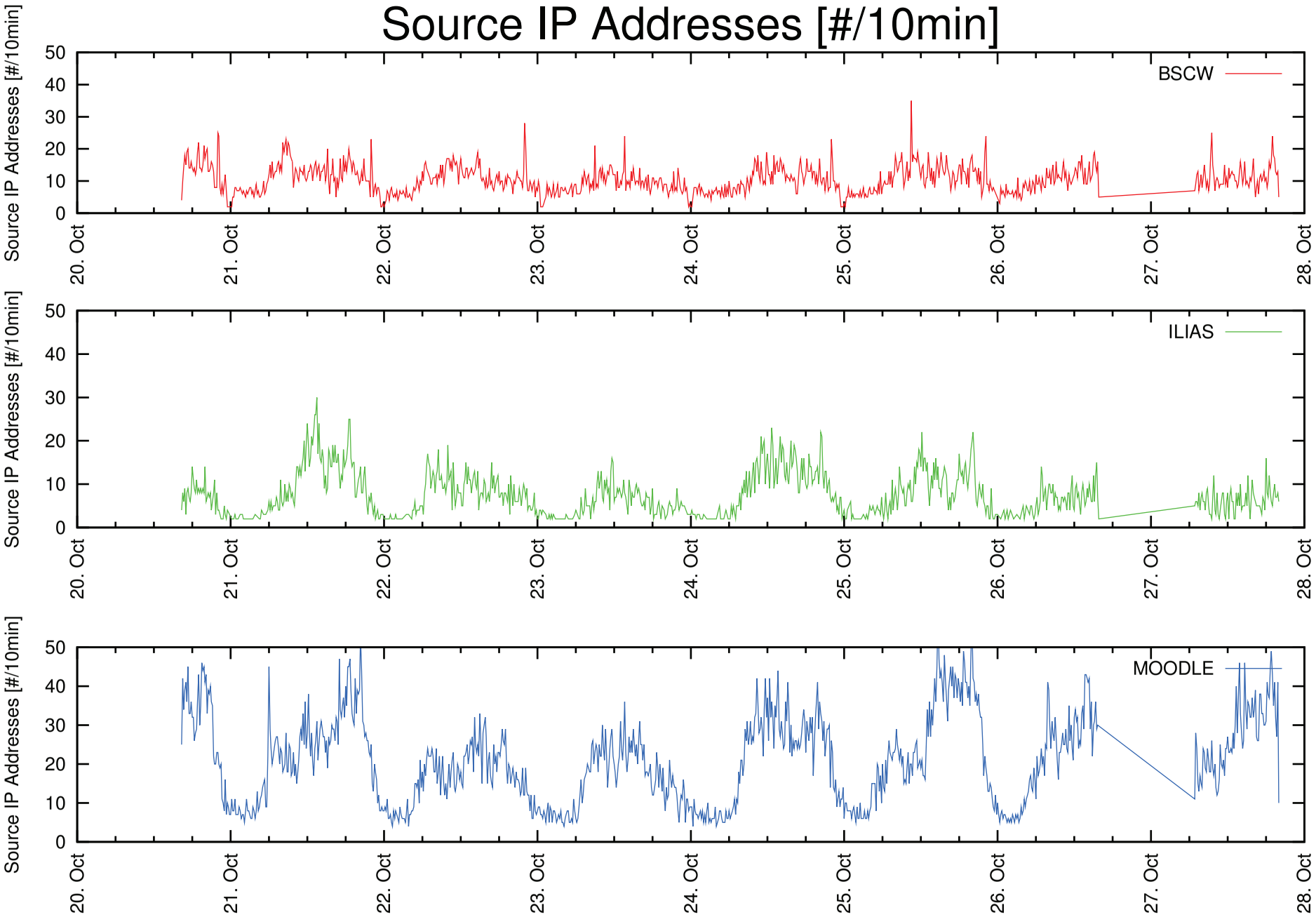
[6] hostip.info, a GeoIP service
`http://www.hostip.info`

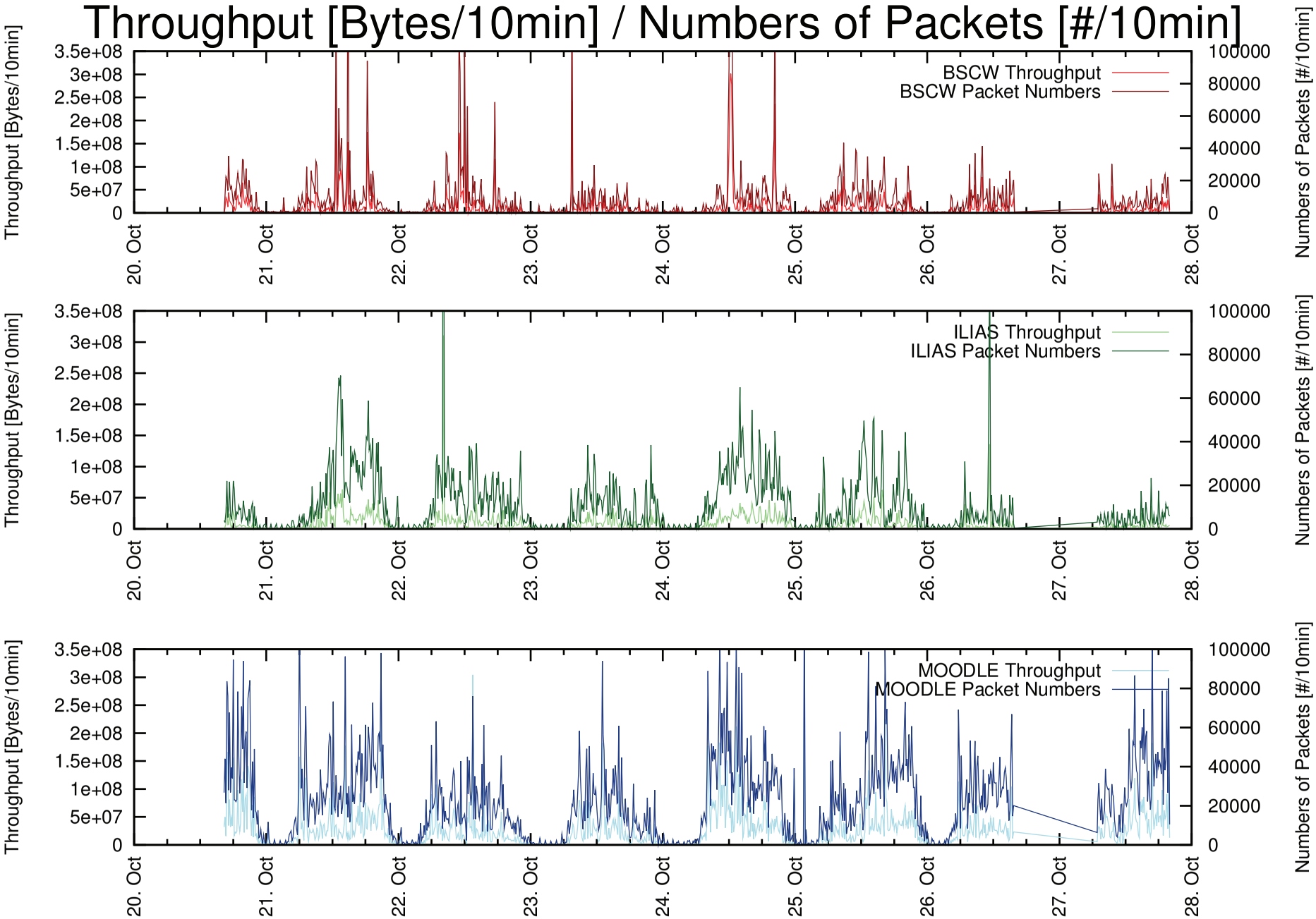
[7] Lakhina, A., Crovella, M., and Diot, C. Mining anomalies using traffic feature distributions ACM SIGCOMM, 2005.

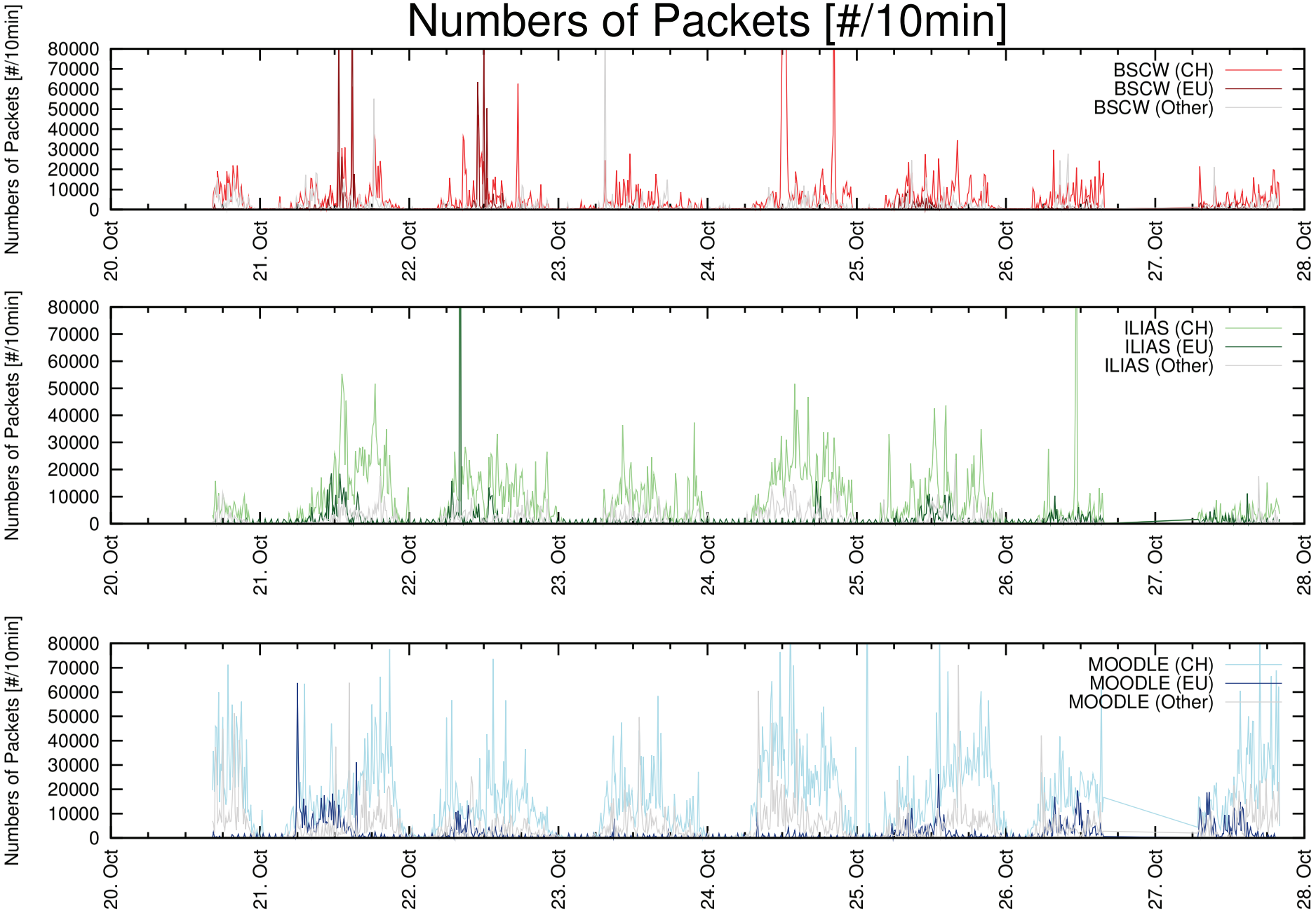
APPENDIX

The plots shown in the appendix were discussed in section 3.


Flows [#/10min]
Flows [#/10min]
5000
4000
3000
2000
1000
0
BSCW
ILIAS
MOODLE
20. Oct
21. Oct
22. Oct
23. Oct
24. Oct
25. Oct
26. Oct
27. Oct
28. Oct


Source IP Addresses [#/10min]
Source IP Addresses [#/10min]
BSCW
ILIAS
MOODLE
0
10
20
30
40
50
20. Oct
21. Oct
22. Oct
23. Oct
24. Oct
25. Oct
26. Oct
27. Oct
28. Oct


Throughput [Bytes/10min] / Numbers of Packets [#/10min]
Throughput [Bytes/10min]
Numbers of Packets [#/10min]
3.5e+08
3e+08
2.5e+08
2e+08
1.5e+08
1e+08
5e+07
0
100000
80000
60000
40000
20000
0
BSCW Throughput
BSCW Packet Numbers
ILIAS Throughput
ILIAS Packet Numbers
MOODLE Throughput
MOODLE Packet Numbers
20. Oct
21. Oct
22. Oct
23. Oct
24. Oct
25. Oct
26. Oct
27. Oct
28. Oct


Numbers of Packets [#/10min]
Numbers of Packets [#/10min]
80000
70000
60000
50000
40000
30000
20000
10000
0
BSCW (CH)
BSCW (EU)
BSCW (Other)
ILIAS (CH)
ILIAS (EU)
ILIAS (Other)
MOODLE (CH)
MOODLE (EU)
MOODLE (Other)
20. Oct
21. Oct
22. Oct
23. Oct
24. Oct
25. Oct
26. Oct
27. Oct
28. Oct