Atheris: A First Step Towards a Unified Peer-to-Peer Traffic Measurement Framework

Philipp M. Eittenberger, Udo R. Krieger
Faculty of Information Systems and Applied Computer Science
Otto-Friedrich University,
96045 Bamberg, Germany
Email: {philipp.eittenberger, udo.krieger}@uni-bamberg.de

***Abstract*—The extensive growth of peer-to-peer video applications (P2PTV) indicates the evolutionary path towards the next generation of networking. To enhance the quality of experience for P2PTV users, new P2P video traffic models are required. These models are often developed by measurement-driven performance modeling based upon statistical analysis. However, this type of analysis can be a challenging task, due to the complexity and especially, with large sample sizes, the sheer quantity of the data. For this reason, preliminary data examination by graphical means and appropriate visualization techniques can be of great value. In this paper, the versatile network traffic analyzer Atheris will be introduced. It is an essential part of a comprehensive traffic measurement and analysis concept to cope with the rapid deployment of P2PTV applications and their overlay structures. Atheris has been designed specifically for measuring P2P applications and for visualizing their inherent properties. As a part of its functionality, Atheris performs traffic measurements at the packet layer, data extraction, flow analysis and enables the visual inspection of statistical characteristics. Furthermore, an evaluation of overlay structures by bandwidth estimation techniques and an overview of the statistical P2P traffic features are provided.**

I. Introduction

In recent years, we have seen the advent of a strong multimedia pervasion in the Internet. Numerous new applications have changed the way we are using the Internet. Along with the first file-sharing networks, a new data dissemination paradigm evolved. This so called peer-to-peer (P2P) mechanism is based on overlay networks in the application layer, which implement the control plane and the content dissemination. After the thriving success of P2P file-sharing, the Skype [1] P2P VoIP network was another step towards the next generation of networking. The last success of this evolutionary process is the dissemination of video data with support of this new paradigm. Since 2004 a hole plethora of P2PTV applications has sprung up (e.g. PPLive [2], PPStream [3], SopCast [4] etc.).

Accurate traffic models are the important first step towards an improved quality of experience for P2PTV users. Therefore, we need new statistical models of P2P video traffic to design networks that deliver satisfying video quality. In order to develop these traffic models and to be able to reveal optimizations, a necessary requirement is given by the measurement and analysis of these P2PTV applications. Measurement-driven traffic modeling always involves trace-based analysis of captured packet streams, to detect, identify and quantify the intrinsic characteristics. Despite various measurement studies (e.g. [5], [6], [7] among others), publicly available measurement tools, which are tailored to analyze the features of this new type of applications, are rarely provided yet. To overcome this deficiency, we have developed the traffic analyzer Atheris based on a new teletraffic analysis concept proposed by Markovich et al. [8]. This paper is organized as follows. Next, in Section II, we evaluate publicly available traffic measurement software and discuss, how their lack of graphical capabilities can impede traffic modelling studies. After, in Section III, we introduce Atheris as a part of a suitable solution to address these drawbacks and demonstrate some of its capabilities by an exemplary measurement study. In Section IV, we discuss related work ansubsequently, we give an outlook to future work in Section V. Finally, we conclude in Section VI.

II. Traffic Measurement Software

To investigate the characteristics of this new breed of applications and use subsequently the obtained results for optimizations, new measurement tools and methodologies are required. We will elaborate this statement in the following section by evaluating freely available measurement tools, which are used to analyze the traffic of P2P applications.

A. Libpcap / WinPcap

Libpcap [9] and its Windows counterpart WinPcap [10] constitute the foundation of most of the measurement tools, i.e. most packet-capture and filtering engines of conventional packet sniffers and protocol analyzers are built on top of them. These libraries are used to intercept the packets and copy them from the kernel to the user space. Afterwards, sniffers and analyzers can dissect the packets and save the packet traces to storage media, usually in the pcap (**p**acket **cap**ture) file format.

B. Wireshark

Wireshark [11] (formerly known as Ethereal) is certainly the most prominent representative of all packet sniffers. It is by far the most powerful publicly available protocol dissector, i.e. its ability to interpret a lot of different protocols is unmatched by any other freely available tool. Many measurement studies can

be found, which use either Wireshark or its predecessor Ethereal (e.g. [5], [12] or [6]). One of the drawbacks of Wireshark is the big memory footprint, especially when working with large traces (> 500 MB). If the trace file is getting too large for the memory of the computer, Wireshark will even crash without a warning. One possibility to circumvent this problem, is to split the large capture file into several smaller files. And one item is even more painful, since Wireshark is analyzing many protocols, it is fairly slow when it comes to analyzing large trace files. Since most of the communication protocols of P2PTV applications are proprietary and therefore unpublished, these capabilities are unnecessary for this type of analysis. Another pitfall of Wireshark, which we will elaborate later, is given by the possibility to loose some packets occasionally.

C. TCPDUMP / WinDump

TCPDUMP [13] is still quite useful. There are a lot of scientific studies (cf. [14] or [15]), which use tcpdump, often in combination with command line tools like grep, awk or sed (see for instance [7]) or scripting languages like Perl or Python. The advantage of this combination is determined by the rapid development and the quick results obtained by customized scripts. Such scripts, which are aligned to the individual needs of the researcher, can be rapidly implemented and are able to give a fast glance on the characteristics, which are to be investigated. Apart from the fact that this approach is fast in the development, tcpdump is also able to capture fast links, i.e. a lot of traffic. The analysis of the trace can then be performed off-line later on. Apparently, this approach has also its drawbacks. Such scripts are good for a first glance, but often the analysis needs to be conducted iteratively. Therefore, this approach can be tedious to use and time consuming. If you want to dig deeper in the data, you need to adjust and run the scripts every time anew. Apart from this issue, there is often no verification of correctness for the scripts. Hence, the results gained by such an approach can be easily prone to error. From a software engineering standpoint, scripts are often not reusable due to unique systems requirements or exotic software dependencies and as a result are difficult to maintain. The Windows counterpart WinDump [16] has also a special handicap: WinDump looses packets without reporting to the user [17]. We could observe this phenomenon as well with Wireshark in combination with WinPcap, i.e. on a Windows machine. We were very surprised to find this problem also on Linux. We encountered this flaw, when we simultaneously captured ICMP packets with tcpdump and Wireshark on a Linux machine. Due to that failure, some studies, like [18], which used this methodology, could have obtained partly biased results.

D. Tstat

The last of the presented tools is called Tstat [19]. It is able to provide statistical data from packet traces and has also some abilities to present the obtained results graphically. Albeit, it has a different focus compared to the approach we follow. Tstat is usually placed between the Internet and an edge node, connecting an internal network to an outside network. Therefore, it is presenting condensed information of the whole internal network rather than from an individual perspective.

E. Problem Formulation

The quest of a statistical analysis is a multi-layered endeavor. For the purpose of performance modeling, graphical methods are intuitive and appealing ways for a preliminary examination of the data, especially when the sample size is massive. As an example, the initial process of distribution selection is usually a combination of using visual inspection and summary statistics. Since all of the presented tools lack sophisticated visualization capabilities, and it seems that there is always a trade-off between ease of usability and functionality (respectively performance). Obviously, tcpdump, as a command line tool does not provide any graphical displays. Wireshark includes some visual displays, like the Flow Graph or the TCP Stream Graphs, which are almost useless for a P2P analysis. There is also the I/O Graph that is quite capable, but it is so awkward to handle that its use is rather cumbersome. Thereby, after the packet capture has been finished, the usual procedure for performance modeling is to export the properties of interest (such as packet length, inter-arrival time etc.) to a text file. Subsequently, the data in the file will be imported for further statistical analysis to a standard software of choice, e.g. R or S-PLUS. Of course, it would be beneficial to get a first glance upon common criteria without this media discontinuity. Since such a statistical analysis is an explorative process that is normally conducted iteratively, the repetition of exporting and re-importing data sets can be an annoying and time-consuming task. To summarize, for the purpose of performance modeling, visual inspection can be an integral complement to summary statistics and there is no solution available that provides powerful visualization techniques.

III. ATHERIS

To address the issues raised, we have developed Atheris. The target of this development was not only to enable straight through processing, but also to lay the foundations for an unified measurement framework. We presented a short demonstration of Atheris at the poster session of the P2P'2010 conference [20]. Additionally, we have already proposed a comprehensive P2P traffic measurement, modeling and teletraffic analysis concept [8]: This framework integrates four orthogonal dimensions to cope with the analysis of P2P structures and the characterization of P2P traffic: (1) single site traffic measurements at the packet layer combining passive and active monitoring techniques, (2) data extraction, analysis and inspection of P2P overlays based on a hierarchical multi-layer modeling concept, (3) a characterization of the overlay structure by techniques and metrics of complex networks, and (4) a statistical characterization of P2P traffic features using the analysis and estimation of the bivariate distributions $\mathbb{P}\{X_i \leq x, Y_i \leq y\}$ of packet inter-arrival times X_i and packet lengths Y_i extracted from corresponding flow data $\{(X_1, Y_1), ..., (X_n, Y_n)\}$ of collected i.i.d samples. Atheris,

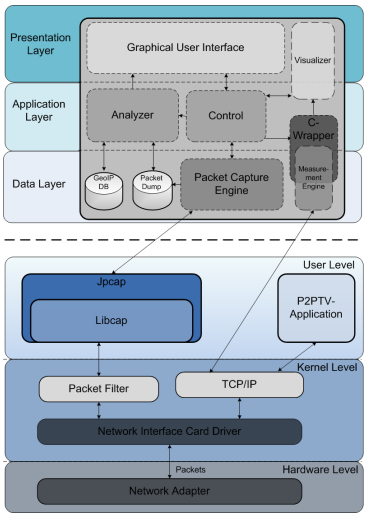

Figure 1. **Software Architecture of Atheris**

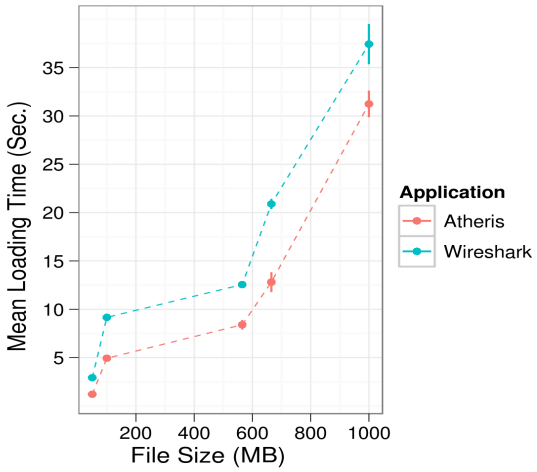

Figure 2. **Loading time: Wireshark vs. Atheris**

as a traffic analyzer, serves as a basis of such a framework and, furthermore, facilitates simultaneously monitoring and analyzing P2P applications in a true on-line manner.

A. *Software Architecture*

Atheris is developed in Java 1.6. It has been chosen due to its platform independence and the availability of suitable libraries. It follows a true object oriented design and consists of several modular components, which can be easily extended. In Figure 1 the architecture of the Linux version of Atheris is depicted. A similar Windows version is also available, which, in contrast, relies on the WinPcap library. The version 0.1 of Atheris uses Jpcap [21] as a Java wrapper for the libpcap/WinPcap library. Since the development of Jpcap has been discontinued and there are serious limitations and bugs, Atheris v0.2 will be migrated to use jNetPcap [22] as a libpcap wrapper. The heart of Atheris is given by its capture engine. We have implemented a multi-threaded approach, i.e. one thread captures the packets and one thread is responsible for the analysis. Version 0.2 we have extended the multi-threaded capture engine, which will allow us to capture simultaneously on multiple devices or to capture multiple sessions seperately on the same device. For very fast links, there is also the possibility, just to analyze the packets without storing them. Since Atheris relies on the libpcap/WinPcap APIs, the usual tcpdump expressions to filter the capture can be used, e.g. to minimize the capture length of each packet. If a packet is stored, Atheris allows, like Wireshark, the inspection of the raw data for every single packet. After capturing a packet, another dedicated thread is used for the packet analysis. It collects some basic connection informations, like the total number of packets and bytes seen up to a given time, the used protocols, time stamps, port numbers etc. In addition, the evaluation of statistical session information is carried out by this thread, i.e. these data are also gathered for every single session. These session informations are accessible bidirectional and for both flow directions separately. In this context, we define a session as the bidirectional conversation among two IP addresses.

B. *Performance Comparison with Wireshark*

At first, we had concerns about the performance of the presented approach, due to the object oriented design and Java as the programming language. In comparison, most parts of Wireshark are written in plain ANSI C. Thus, Wireshark suffers not from the overhead of object orientation and C as a relatively low-level programming language should improve its performance.

To provide a comparison, we have performed some measurements, which evaluate the necessary time span for loading and analysing large pcap files for both applications, Wireshark v1.2.7 and Atheris v0.1. The traces were recorded with Wireshark and the measurements were conducted on a machine with a Intel Core i7 processor 2.8 Ghz and 4 GB RAM. A Linux distribution with kernel version 2.6 served as operating system. On Wireshark, the auto-scroll and the colorization of the packet list was switched off, to provide equal conditions with Atheris, since especially the coloring of the rows needs a lot of computational overhead. Additionally, since it is not possible in Wireshark to switch off particular protocol dissectors, only UDP traces, which induce the least computational costs for the protocol dissectors, have been used for the comparison. Figure 2 shows the 95% confidence intervals on the mean loading times. These numbers are promising and assure a better scalability of Atheris. When comparing the loading times of TCP traces this graphic looks extremely differently. We encountered that Wireshark may need up to half an hour to load a 1 GB trace file, while Atheris needed for the same file in its longest attempt 21 seconds. Of course, the longer loading times of Wireshark are resulting from the many protocol dissectors Wireshark

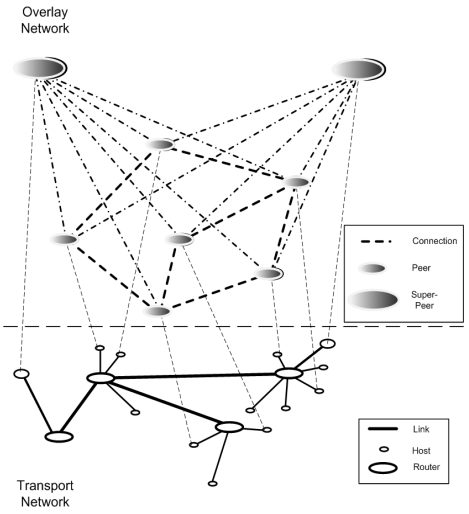

Figure 3. **Two-layered P2P overlay network model**

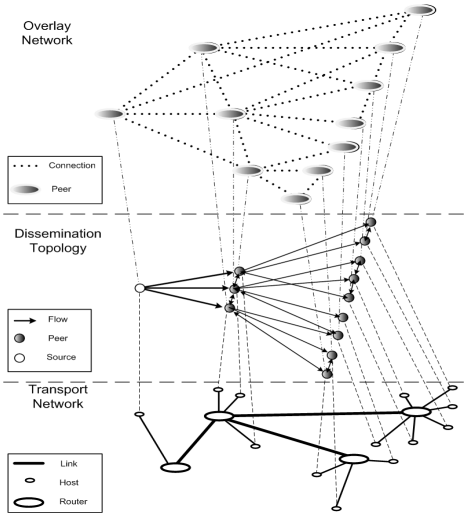

Figure 4. **Multi-layer model of a mesh-pull P2PTV application**

executes. However, when performing a P2P traffic analysis, most of these protocols are not of any interest. Since Atheris is only evaluating the most common protocol headers, this is resulting in much faster loading times. Compared to Jpcap, the jNetPcap library is more sophisticated and makes even use of its own memory model written in native C. Therefore, we expect even further performance improvements, in terms of faster loading times and less memory consumption, after the migration to jNetPcap. In addition, first prototypical tests confirm this conclusion.

C. Exploring the Multi-Layered Topology of P2P Overlays by Active Measurements

The overlay structure of a peer population can be modeled by an undirected neighborhood graph $G_N = (V_N, E_N)$ among an object specific, time dependent finite population $P(Oi)(t) = p_1(t), ..., p_n(t)(t) \subset U$. Recent studies have shown that particular P2P applications elect, according to desired criteria (like bandwidth, delay, uptime etc.), so called super-peers, to support the operation of the P2P network (cf. e.g. [6] for Skype). One characteristic of super-peers is their high degree of connectivity, since they establish connections with many peers for the purpose of maintaining the operational infrastructure of the overlay. Figure 3 illustrates such a situation, some super-peers in the overlay of the depicted P2P network are connected with a higher degree compared to the normal peers. To model P2PTV networks, this two-layered model needs to be extended by a third layer to take the dissemination of the data into consideration. This new layer can be modeled as a directed dissemination flow graph $G_V = (V_V, E_V, f, c), V_V \subseteq V_N$, among pairs (p_i, p_j) of peers exchanging video data as well as the underlying TCP/UDP-IP transport network connecting the peers p_i by flow paths p along capacity constrained links (e_k, c_k) (see Figure 4). When packets are lost during transmission or when the inter-arrival times of frames are either large or variable, the video quality degrades. Hence, P2PTV networks should build their dissemination topology with well-provisioned peers at the core to increase the quality of the video playback. Atheris offers the possibility to probe actively the connections $\{p_i, p_j\} \in E_N$ and, thus, to evaluate the underlying dissemination route p to a home peer p_i. These estimates can be used to evaluate the bandwidth awareness and assess the responsiveness of the particular P2P application to the dynamically changing network conditions. The obtained insights can reveal performance bottlenecks and are useful for optimizing the dissemination networks of P2P applications. With the limitation that we have just the control of the sender and not the receiver of the measurement probes, we evaluated the unidirectional tools for estimating the available bandwidth in a static laboratory test bed. Thereafter, a first prototype of a measurement server was implemented to conduct the active measurements. The measurement server is written in C and performs the measurements with the help of the tool Pathneck [23], which delivered a reasonable compromise between intrusiveness of the probe traffic and accuracy of the estimates in our evaluation. To estimate the peers' bandwidth, Atheris sends their IP addresses along with the needed parameters for Pathneck to the server. After performing the measurements, the server returns the results to Atheris (see Figure 5). Since the available bandwidth fluctuates dynamically, the measurements can be conducted periodically. The results of several such measurement runs can subsequently be compared with one another. Additionally, the round-trip time and hop value along the path to every peer can be obtained, too. (Due to space limitations, we omit these graphics here). Basically, there are two approaches to traffic measurement: *active* and *passive*. Active measurement tools, like Pathneck, Ark etc., try to interact with the network to induce a measurable effect. In contrast, in the passive approach, the network traffic is just observed, bearing in mind

Figure 5. **Bandwidth estimates visualized by Atheris**

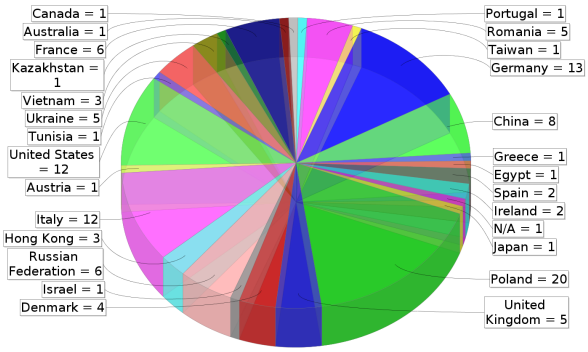

Figure 6. **Geographical distribution of the peer population**

not to interfere the observation. Atheris uses a combination of both, i.e. active and passive measurement techniques, to explore the investigated P2P applications.

D. Investigating P2PTV Traffic - Atheris in Action

As already mentioned, graphical displays are an excellent way to grasp and assess the information in the data. Right now, Atheris enables the investigation of packet flows by adequate inspection techniques using common statistical visualization methods such as histograms, bivariate scatter diagrams or pie charts. In the following section, we will present some of the visualization capabilities of Atheris, that means that all of the following graphics, except Figure 7, were generated by Atheris. To demonstrate some of these capabilities, we performed an exemplary measurement study, to explore the traffic features of Sopcast, a thoroughly investigated representative of a mesh-pull P2PTV application (cf. [12], [7] among others).

	Download	Upload	Download	Upload	Total
Packets	44.55 %	55.45 %	466,980	581,254	1,048,234
Data Volume	14.28 %	85.72 %	92.72 MB	556.68 MB	649.4 MB

Table I
TRAFFIC CHARACTERISTICS

The configuration of the measurement test bed is depicted in Figure 7. The measurement was performed on a laptop with a Core 2 Duo 2.4 GHz as CPU and 3.24 GB of RAM, Windows XP served as the operating system. The computer was connected to the university network with a 100Mbit switch, which is connected through a router to the Internet. In total, 13 minutes of a live transmission of the 30th European swimming championships in Budapest, Hungary, were captured. The trace file consists of roughly 650 MB with roughly one million captured packets, with a fraction of more than 99% UDP packets. More detailed information of the trace are provided in table I. It is quite obvious that Sopcast uses the broad upload capacity of the Internet connection with upload traffic accounting 85.72% of the total exchanged data volume.

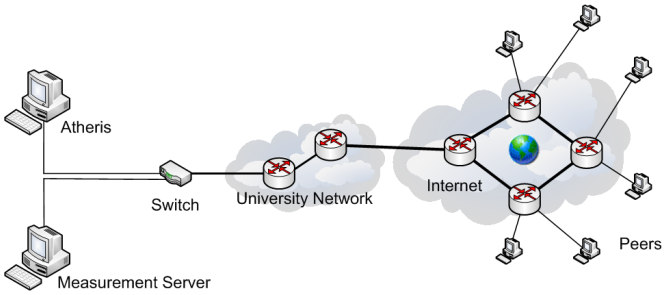

Figure 7. **Measurement test bed**

At the beginning, we will explore the geographical distribution of the peer population. The geographical IP address location check is performed with the support of the GeoIP API by MaxMind [24]. The result is depicted in Figure 6. In total, 118 peers were identified. The largest share of the peer population, 83 peers, is stemming from all over Europe, with the biggest group in Poland with 20 peers, followed by Germany with 13 and Italy with 12 peers. There is also a rather large group in the USA with 12 peers. The rest of the peer population is allocated all over the world, with the exception that no peers come from South America. Since this were European championships, this result is not entirely surprising.

As we have seen, Sopcast clients are typically connected with up to hundreds of different peers during just tens of minutes. Therefore, it is necessary to investigate the preference relationship among the peer flows, to understand the inherent hierarchical structure of the mesh-pull topology. One can use as metric, for instance, the number of transferred packets or the data volume of the sessions. Atheris is able to visualize both metrics, either bidirectional or separated by direction. Exemplary, the distribution of exchanged data volume in all sessions, i.e. the amount of traffic that was exchanged in each of the sessions, will be illustrated. For this purpose, Atheris sorted all the sessions according to their exchanged data volume and then plotted them on a log-log scale (see Figure 8). This reveals the hierarchical structure of the dissemination network, only a small fraction of the peers contributes most of

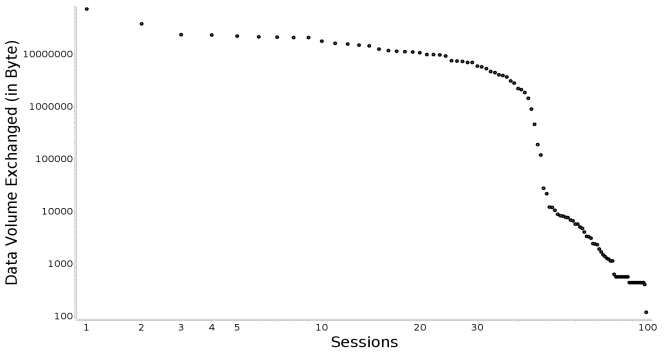

Figure 8. **Exchanged data volume of every session on a log-log scale**

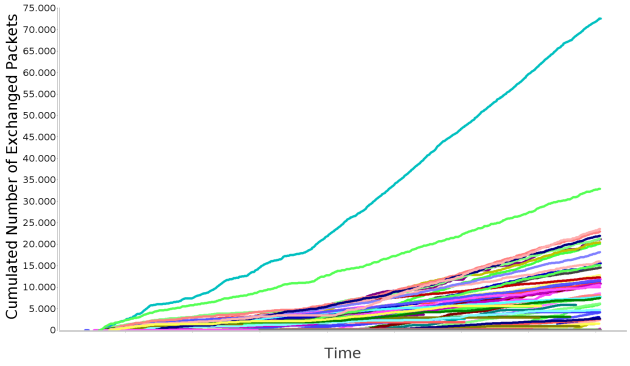

Figure 9. **Cumulated outbound traffic of every session**

the data. When we let Atheris separately analyze the inbound and outbound traffic, the same outcome as before can be identified, only a small fraction of the peers sends, respectively receives, most of the data. In both cases, the end of the long tail consists of unsuccessful connection attempts, i.e. peers did not reply to these attempts. By these graphical means, we can easily come to the assumption that parts of these distribution functions obey some kind of power law.

To identify the top-peers, i.e. the peers which contributed most of the data, Atheris is also able to show the cumulated traffic volume, again either for the in- or outbound direction or bidirectional. Since it is not possible to conclude from the limited local view, if these peers are super-peers, in the sense that they are chosen from the P2P network to support its infrastructure, we call them top-peers subsequently, to draw a clear distinction. The three top-peers come from Taiwan, Ukraine and Denmark. The analysis of the cumulated session inbound traffic illustrates two interesting facts, in particular, all three sessions started quite early in the trace, and the three top-peers contributed by far the biggest share of the total data volume. One should assume that mesh-pull P2PTV networks embed a peer requesting a video stream into a dense mesh of feeding peers to guarantee a reliable and timely supply of chunk sequences to the streaming engine of that host. Unexpectedly, and despite the fact that the measurement was conducted in Central Europe, a single peer from Taiwan contributed roughly 45% of the total inbound traffic. This is even more surprising, since the next 11 of the top-peers are all located in Europe. This may possibly have been due to the fact that the Taiwanese peer could be part of a dedicated infrastructure of the Sopcast network to support its data dissemination. It can be a legitimate approach to ensure the timely delivery of the video data to the peer's media player. The cumulated outbound traffic is depicted in Figure 9. Surprisingly, one single peer receives by far the biggest amount of data. From the choking mechanism of BitTorrent [25] it is known that capping the number of simultaneous uploads can improve the TCP performance. However, Sopcast uses for the data dissemination only UDP and it transmits simultaneously video data to a multitude of peers, but, astonishingly, one peer receives the biggest share of the data. To the best of our knowledge, this transmission pattern of Sopcast has not been revealed before. The intensity of the in- and outbound traffic, i.e. the amount of transferred packets or bytes per second, shows this pattern as well. Figure 10 illustrates the intensity of the packet exchange of the inbound traffic in all sessions. The outbound intensity of the data exchange is illustrated by Figure 11. By these graphics, it is easy to spot that for both traffic directions one peer is accountable for the largest proportion of the data volume, respectively the number of packets (cf. also with Figure 9). Another interesting observation, which both graphics 10 and 11 reveal, are several spikes, possibly due to scene changes and two global minima which occur simultaneously. Most of the data was transmitted to peers in Europe, 16 out of the top 20 peers are resident in Europe. The peer which received the biggest share of the data is located in Ukraine. From this perspective, it is also possible to conclude that the Sopcast network is somehow location-aware. It will be an interesting task to measure the available bandwidth along the paths to such top-peers with Atheris, but we will postpone this task for future investigations. The work of Tang et al. [7] has revealed some of the interaction patterns of Sopcast. The study has also shown that the packets of the signaling traffic have approximately a packet length between 40 and 150 bytes and the video frames are transported in IP packets with 1348 bytes. If the captured frame length of the packets is displayed on a scatter-plot along the time axis (see Figures 12 and 13 for outbound respectively inbound traffic), several levels can be identified. The frames with 1362 bytes (=1348 bytes + 14 bytes MAC header) build a straight line for both directions, and several levels in the region between 54 and 164 bytes can be detected. Another interesting fact can be observed, when the measurement host starts to receive a large number of packets within the frame range between 170 and 350 bytes, it starts to send packets with a frame length ranging from 550 to 1360 bytes. This happens at the beginning of the trace, again after a while and then, this interaction pattern continuous from the second half of the trace until the end. An attempt to explain that behavior could be the creation and maintenance of the overlay. When joining the P2P network,

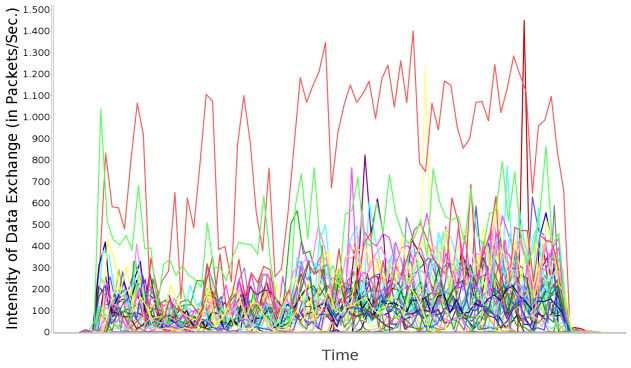

Figure 10. **Intensity of packet exchange in all sessions (inbound traffic)**

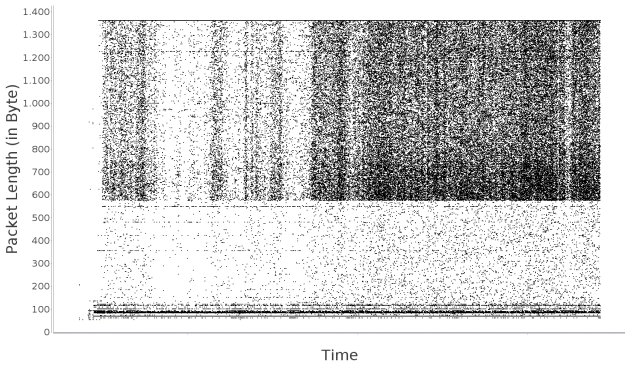

Figure 12. **Packet length scatter plot (outbound traffic)**

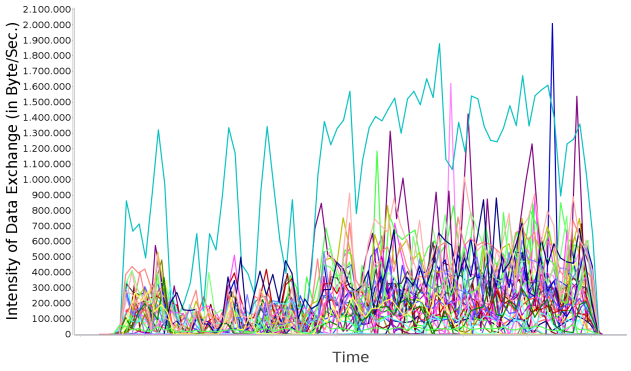

Figure 11. **Intensity of data exchange in all sessions (outbound traffic)**

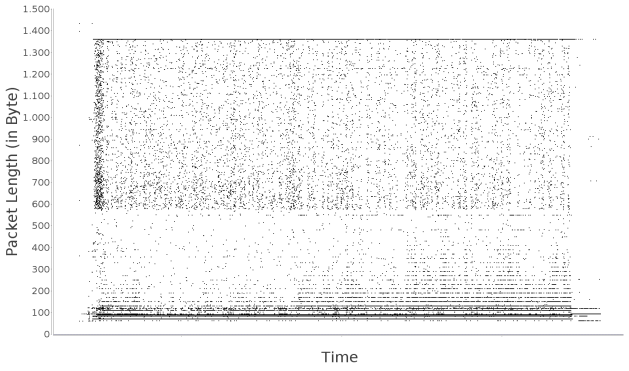

Figure 13. **Packet length scatter plot (inbound traffic)**

the host needs information of other peers, to start the video data transmission. At this time, it receives a larger number of packets within the frame range between 550 and 1360 bytes, because it has no prior knowledge about the overlay network. When the Sopcast network gets aware of the high upload capacity of the host, it could have been chosen as a super-peer to support Sopcast's infrastructure, for example, to distribute the peer lists. For getting an overview on the amount of exchanged packets there is simply "too much ink" on both plots. Hence, it is a suited approach to use histograms for displaying the distribution of the packet length. Figures 14 and 15 illustrate this matter. Now, we can easily grasp that most of the sent packets are video frames, and most of the received packets are in the range of the signaling messages. Since the host received much more video packets than it consumed, it was used as a relay node for the video dissemination.

Several studies have performed distributed experiments to explore hidden properties of P2PTV applications (e.g. [7] or [26]). Most of these applications use unpublished communication protocols and are proprietary systems, thereby, one don't has access to e.g. server log files. To circumvent such limitations, these experiments are often conducted on the PlanetLab [27] test bed in order to reverse engineer the interaction patterns of these applications. Apart from reverse engineering and experiments on the PlanetLab platform, which both require a great deal of time and energy, the only possibility to investigate such closed systems is given by single site measurements. We have shown that it is possible to reveal some of the characteristics of a P2PTV dissemination network by statistical visualizations, in combination with summary statistics, only through single site observations. Of course, these observations are somehow still limited and thus, the natural next step is the deployment of Atheris in a true P2P manner. That implies that several distributed instances of Atheris will run simultaneously and exchange their local information to construct a holistic view of the P2P overlay networks.

IV. Related Work

Apart from the capture software we have already presented in section II, there are some other efforts, which shall be discussed in the following: The IETF Application-Layer Traffic Optimization (Alto) [28] project is not a packet sniffer, but rather a distributed measurement infrastructure. Alto tries to provide a complete infrastructure with dedicated measurement servers incorporated into the Internet to optimize the data dissemination of P2P networks. Plab [29] is a traffic analyzer, which enables session analysis according to different criteria. For each session, several specified statistics can be gathered, but all the data is written into plain text files, without the possibility of visualizing properties of interest. Caida provides several tools to measure and analyze Internet traffic [30]. For

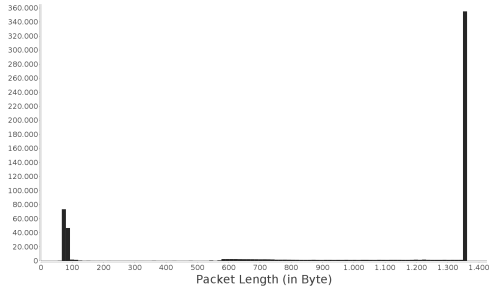

Figure 14. **Packet length distribution (outbound traffic)**

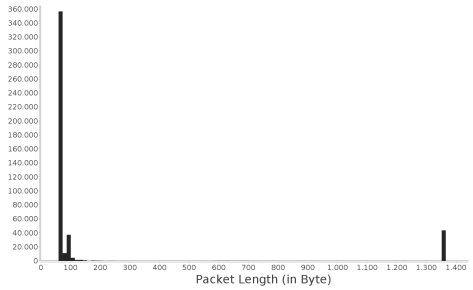

Figure 15. **Packet length distribution (inbound traffic)**

instance, one noteworthy project of Caida is called CoralReef [31], which includes a lot of basic scripts to analyze trace files, e.g. to separate specific flows or to generate reports etc. All of these tools consist more or less of a bundle of scripts and lack serious graphical capabilities. Tstat and CoralReef provide some limited visualization techniques, but not as an integral part of the particular tool. One effort in the direction of the visualization of P2P traffic is given by Blink [32], but this tool is not available in the public domain.

V. Future Work

After completing the migration to the jNetPcap library, an open source version of Atheris will be made publicly available. One of the new features in version 0.2 of Atheris will be the possibility of storing the captured packets into a database to enable packet filtering with SQL statements. Afterwards, the next goal is a P2P version of Atheris, which will enable us to explore P2P application overlays even more thoroughly. Thereby, we will also have a better starting situation to measure, for instance, the available bandwidth between the peers with more sophisticated network measurement methods, like MGRP [33].

VI. Conclusion

In this paper, we introduced the traffic analyzer Atheris. First of all, we evaluated the available traffic measurement solutions and explained why their lack of graphical displays can impede traffic modeling. Then, we introduced a multi-layered approach to model P2PTV applications and presented the possibilities of Atheris to reveal hidden characteristics from single site measurements. To demonstrate some of these capabilities, we conducted a small measurement study with the P2PTV application Sopcast and presented the obtained results, visualized by Atheris.

Acknowledgment

The authors acknowledge the partial financial support by the projects COST IC0703 and BMBF MDA 08/015.

References

[1] Skype. [Online]. Available: http://www.skype.com
[2] Pplive. [Online]. Available: http://www.pplive.com
[3] Ppstream. [Online]. Available: http://www.ppstream.com
[4] Sopcast. [Online]. Available: http://www.sopcast.com
[5] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," in *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.
[6] S. Baset and H. Schulzrinne, "An analysis of the Skype peer-to-peer internet telephony protocol," in *INFOCOM'06*. IEEE, 2006, pp. 1–11.
[7] S. Tang, Y. Lu, J. M. Hernández, F. Kuipers, and P. Mieghem, "Topology dynamics in a P2PTV network," in *NETWORKING'09*. Springer, 2009, pp. 326–337.
[8] N. M. Markovich, A. Biernacki, P. M. Eittenberger, and U. R. Krieger, "Integrated measurement and analysis of peer-to-peer traffic," in *WWIC'10*. Springer, 2010, pp. 302–314.
[9] Libpcap. [Online]. Available: http://www.tcpdump.org
[10] Winpcap. [Online]. Available: http://www.winpcap.org
[11] Wireshark. [Online]. Available: http://www.wireshark.org
[12] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *In Proc. of ICST Workshop on Recent Advances in Peer-to-Peer Streaming*, 2006.
[13] Tcpdump. [Online]. Available: http://www.tcpdump.org
[14] K. Tutschku, "A measurement-based traffic profile of the edonkey filesharing service," in *PAM'04*, 2004, pp. 12–21.
[15] D. Ilie and A. Popescu, "Statistical models for Gnutella signaling traffic," *Journal of Computer Networks*, vol. 51, pp. 4816–4835, 2007.
[16] Windump. [Online]. Available: http://www.winpcap.org/windump/install/default.htm
[17] P. Arlos and M. Fiedler, "A method to estimate the timestamp accuracy of measurement hardware and software tools," in *PAM'07*. Springer, 2007, pp. 197–206.
[18] T. Silverston and O. Fourmaux, "P2P IPTV measurement: a case study of TVants," in *CoNEXT'06*. ACM, 2006, pp. 1–2.
[19] Tstat. [Online]. Available: http://tstat.tlc.polito.it/index.shtml
[20] P. M. Eittenberger, U. Krieger, A. Biernacki, and N. Markovich, "Integrated measurement and analysis of peer-to-peer streaming traffic by the java tool Atheris," in *P2P'10*. IEEE, 2010, pp. 155–157.
[21] Jpcap. [Online]. Available: http://netresearch.ics.uci.edu/kfujii/jpcap/doc
[22] jnetpcap. [Online]. Available: http://jnetpcap.com/node
[23] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements, and implications," in *SIGCOMM'04*. ACM, 2004, pp. 41–54.
[24] Maxmind geoip country database. [Online]. Available: http://www.maxmind.com/app/geoip_country
[25] Bittorrent. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html
[26] A. Sentinelli, G. Marfia, G. Pau, and L. Celetto, "IPTV-P2P clients at home," in *IWSSIP'09*, 2009, pp. 1–4.
[27] Planetlab. [Online]. Available: http://www.planet-lab.org
[28] J. Seedorf and E. Burger, "Application-layer traffic optimization (ALTO) problem statement," RFC 5693, October 2009.
[29] Plab. [Online]. Available: http://www.grid.unina.it/software/Plab
[30] Caida. [Online]. Available: http://www.caida.org/tools
[31] Coralreef. [Online]. Available: http://www.caida.org/tools/measurement/coralreef
[32] R. Ando, Y. Kadobayashi, and Y. Shinoda, "Blink: Large-scale P2P network monitoring and visualization system using vm introspection," in *NCM*, 2010, pp. 351–358.
[33] P. Papageorge, J. McCann, and M. Hicks, "Passive aggressive measurement with MGRP," in *SIGCOMM'09*. ACM, 2009, pp. 279–290.