SopCast P2P Live Streaming: Live Session Traces and Analysis

Alex Borges Vieira[1], Ana Paula Couto da Silva[1], Francisco Henrique[1],
Glauber Goncalves[2], Pedro de Carvalho Gomes[3]
[1]Computer Science Department, Universidade Federal de Juiz de Fora (DCC-UFJF)
Juiz de Fora, Brazil
[2]Computer Science Department, Universidade Federal de Minas Gerais (DCC-UFMG)
Belo Horizonte, Brazil
[3]School of Computer Science and Communication, KTH Royal Institute of Technology (CSC-KTH)
Stockholm, Sweden
{alex.borges, anapaula.silva, francisco.henrique}@ufjf.edu.br;
ggoncalves@dcc.ufmg.br; pedrodcg@csc.kth.se

ABSTRACT

P2P-TV applications have attracted a lot of attention from the research community in the last years. Such systems generate a large amount of data which impacts the network performance. As a natural consequence, characterizing these systems has become a very important task to develop better multimedia systems. However, crawling data from P2P live streaming systems is particularly challenging by the fact that most of these applications have private protocols. In this work, we present a set of logs from a very popular P2P live streaming application, the SopCast. We describe our crawling methodology, and present a brief SopCast characterization. We believe that our logs and the characterization can be used as a starting point to the development of new live streaming systems.

Categories and Subject Descriptors

D.5.1 [**Multimedia Information System**]: Video

General Terms

Internet metrics, Performance, Measurement

Keywords

P2P live streaming; SopCast

1. INTRODUCTION

P2P live streaming applications are very important nowadays. Recently, these applications have attracted interest from both the academia and from the industry. Such systems generate a large amount of data, which impacts the network performance. As a natural consequence, the characterization of these systems has become a very important task in order to develop better network and multimedia systems.

However, the characterization and modeling of P2P live streaming systems is a challenging task [1] once most of these applications have private protocols. Moreover, the client behavior and the application system information are vital to the business model. Due its importance, these information are store in tracking servers which are not available for public use, not even for research purposes.

In this scenario, it is difficult to produce a precise reconstruction of the characteristics from popular P2P live streaming networks. Despite the number of works characterizing P2P live streaming application [2–6], we are not aware of any public available dataset. Moreover, most of these works characterize a static view of the P2P systems, ignoring important dynamic aspects.

In this work, we present datasets from the most popular P2P live streaming application, the SopCast [7]. The datasets have been used in important works from our team. For instance, we have characterized the SopCast client behavior [2, 8], producing models that researchers can use to test their own P2P live streaming systems. We discuss deeply our crawling methodology and present a brief SopCast characterization. We believe that our datasets and the characterization are specially useful for the development of new streaming media services. The most recent set of datasets and its description are available at: `http://netlab.ice.ufjf.br/traces`.

2. THE SOPCAST APPLICATION

SopCast is the most popular P2P live streaming application according to Google Trends. Figure 1 shows that during 2012 the SopCast has attracted about 50% of the searches for P2P-TV applications. It means that SopCast has attract more queries than all other similar popular application together.

Similar to other P2P applications, SopCast implements a mesh based overlay topology. On it, peers actively request pieces of the streaming to its partners (pull based data driven). Each SopCast channel has its own P2P overlay.

MMSys '13, February 26-March 1, 2013, Oslo, Norway.

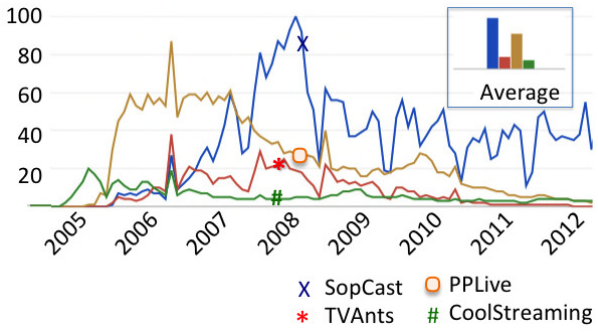

Figure 1: SopCast Popularity on Google Trends.

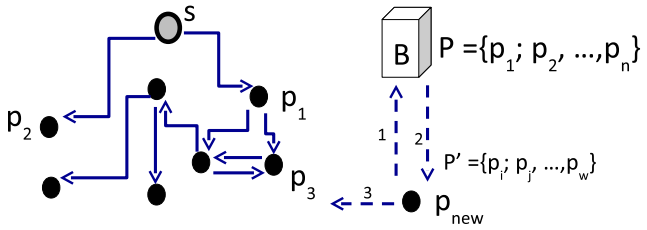

Figure 2: Mesh-pull based P2P Live Streaming System.

Figure 2 shows the main components of the SopCast application: a live streaming server (S), a bootstrap server (B) and a set of clients ($P = \{p_1, p_2, \ldots, p_n\}$).

The live streaming server is a special client that encodes the media and splits it into small data chunks. The partners of the server request these chunks, and then start the live content spreading. The bootstrap server maintains a centralized record of all system peers. When a new peer joins the channel, it contacts the bootstrap (Fig. 2, step 1), which, in turn, returns a subset of system peers (Fig. 2, step 2). The new peer tries to establish partnerships with these clients (Fig. 2, step 3). Peers are free to establish and break partnerships dynamically.

SopCast peers establish partnerships and exchange data among themselves in order to watch the live streaming content. Each peer has a chunk map that indicates the data they have available. Periodically, peers exchange theirs chunk maps among theirs partners. Then, they can request each other for the desired chunks. A peer may download chunks from several partners simultaneously.

3. SOPCAST DATA CRAWLING

All traces we provide have been collected using the PlanetLab testbed [9]. During our experiments, we setup a group of PlanetLab machines to act as regular SopCast peers and data crawlers. These crawlers are spread worldwide, and they are usually hosted in servers at university campuses. PlanetLab nodes currently run a Linux distribution that is largely based on Fedora Core 8.

During our experiments, we have used the largest number of crawlers available, which normally has varied between 350 and 450 stable PlanetLab peers. Our crawling strategy is close to previous works [2, 4–6, 8, 10]. We highlight two main phases of our crawling methodology, namely, (1) *setup configuration* and; (2) *data crawling*.

At the beginning of the setup configuration phase, all crawlers update their local software, including Wireshark (http://www.wireshark.org.), which is the network monitoring application we use to capture the network traffic during the second phase. We have also checked clock synchronization among PlanetLab nodes. More precisely, crawlers synchronize their local times according to a given server, using Network Time Protocol NTP [11]. We used the standard configuration of the PlanetLab nodes for NTP servers. If this configuration was not available, we used public NTP servers such as ntp.ubuntu.com. This procedure makes the local time differences among the data crawlers quite small (about 1 second).

During the data crawling phase, all crawlers join the same (target) SopCast channel. Joining times are normally distributed over a given initial period T, and the date collected during this period is included in our logs. Our crawlers capture all traffic generated by SopCast. We consider UDP/TCP packets exchanged through ports configured by the SopCast application.

At the end of the data crawling phase, we download each crawler log file to a centralized server. In our characterization, we merged all logs to one single file to analyze system peer behavior and SopCast protocol. This approach is used in order to rebuild the overlay topology. However, the logs we provide are not merged, thus researches may analyze each crawler data in detail.

Table 1 presents our data log format. It contains each packet's timestamp (at a 1-second granularity), packet's source and destination, and packet's size information.

Table 1: Log Format.

Time	Source IP	Dest. IP	size of data (in bytes)

We provide logs for two types of SopCast channels: data from *public available channel* and data from *private channels*. The logs come with metadata, which describes if it was collected from a public or a private channel, the video streaming rate, and the date and time from the crawling period. Metadata information from each log can be retried from our website: `netlab.ice.ufjf.br/traces`.

For instance, considering the *public available channel* case, we have collected data from CCTV on SopCast. CCTV is very a popular TV channel in China, and it transmits higher quality video (around 600 kbps) in comparison to the majority of the SopCast channels. We also provide logs from a specialized sports channel in Chinese language. Usual program examples include sports news, live matches and sports documentaries. Moreover, we present logs collected during the transmission of two major sports events for the Brazilian audience, i.e., matches of the final rounds of the Brazilian football championship. During these transmissions, the live content was broadcasted at medium to low rates for SopCast standards (around 250 kbps).

Clearly we are not able to provide a full overview of the SopCast topology when we are collecting a public available channel. First, we are not aware of the channel real population. Second, as our crawlers acts as SopCast

standard clients, we do not control their partnerships. To circumvent this, we try to use the maximum number of crawlers as possible. In fact, during our experiments, we notice that the number of SopCast clients we discover increases with the number of crawlers.

For example, we conduct an experiment using 421 crawlers during a sports event. Table 2 shows the average results for the ratio between the number of simultaneous clients discovered by k crawlers, and the number of simultaneous clients discovered by the 421 crawlers.

As shown in Table 2, using at most 70 clients (as opposed to 421) leads to a loss of at least 11%, on average, in the number of simultaneous clients we can discover. In this crawling session, the benefit of having more than 200 crawlers is only marginal.

Table 2: Ratio between the number of simultaneous peers discovered using k crawlers and 421 crawlers.

$k = 2$	$k = 10$	$k = 50$	$k = 70$	$k = 100$	$k = 200$
13.5%	42.1%	85.5%	88.9%	93%	98%

For the *private channel* crawling approach, we control all peers in the live streaming session. Thus, we are able to rebuild a complete view of the overlay network. Recall that, on public available channels, we can not track data exchanges and partnerships between non-crawlers.

To build our private SopCast channel, we set up a server to encode and transmit the video. The server is a computer located on our campus network. Our experiments on controlled environment usually last for 1 hour. Most of the videos we transmit on private channels have between 250 to 500 kbps.

We experiment a private channel with, or without peers churn. In the former case, SopCast peers (crawlers) start mimicking the behavior of real peers. They dynamically leave and rejoin the channel, following the behavior of real SopCast clients that we have modeled in a previous work [2].

For example, a peer p_i joins the channel and remains connected during an ON period of time. After that, it leaves the channel and remains idle during an OFF period of time and rejoins the channel. This process repeats until the end of the experiment. We model the ON time as a Weibull distribution and the OFF time as an Exponential distribution, according to Table 3.

Table 3: ON-OFF parameters.

ON time - Weibull: $p_X(x) = \alpha\beta x^{\beta-0} e^{-\alpha x^{\beta}} I_{(0,\infty)}(x)$; mean=23.59; std. dev.=34.99; $\alpha = 2.032$ and $\beta = 0.233$.
OFF time - Exponential: $p_X(x) = \lambda e^{-\lambda x}$; mean=18.49; std. dev=16.17 and $\lambda = 0.054$.

As it can be noticed, the logs we have collected provide a powerful tool for characterizing, modeling and mimicking clients of P2P streaming live transmission applications.

4. KEY TRACES CHARACTERISTICS

In this section, we briefly report and discuss some knowledge that can be extracted from the SopCast traces we have collected. The results we present are just a small sample of the kind of information that can be processed from them. The following characterization may be used as a start point to research groups that intend to develop or to simulate P2P live streaming systems similar to SopCast. As we have pointed out before, the need for a SopCast characterization is a consequence of the proprietary and, therefore unknown, mechanisms adopted by such system.

We consider five experiments using a private channel. In each experiment, we have merged all log files to rebuild the SopCast overlay network as a sequence of snapshots. Using snapshots allows us to understand the dynamic behavior of the P2P live streaming application, bringing a finer granularity on the characterization itself. Sliding window snapshots are five-seconds long, and each experiment contains $3,295$ snapshots.

The experimental overlay topology is applied for defining the mathematical model used as base of the characterization. SopCast dynamic overlay is mathematically represented by a family of graphs $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ indexed over time; $\mathcal{V}_t$ is the set of nodes and $\mathcal{E}_t$ is the set of links. Parameter t spans over $\{T_0 \leq t \leq T_N\}$. Let Δ be the interval between two consecutive *snapshots*. Each graph $\mathcal{G}_t$ aggregates the system information over δ seconds. Without loss of generality, let T_0 and T_1 be two consecutive *snapshots*, with $T_1 = T_0 + \Delta$. In this case, $\mathcal{G}_{T_0}$ aggregates information over $[T_0, T_0 + \delta - 1]$ seconds; $\mathcal{G}_{T_1}$, instead, aggregates information over $[T_1, T_1 + \delta - 1]$ seconds. In other words, we are aggregating information using a sliding window mechanism, with a δ-seconds duration. Characterization analysis is performed between two consecutive *snapshots*.

Let us focus on how links are established among nodes. Let u and v be two nodes on the overlay. If node u has sent at least one chunk of video to node v (and *vice-versa*) during the analyzed δ seconds, then the link (u, v) exists. We define data exchange as the exchange of a full payload network package between peers. As we focus on the content distribution process, a chunk of video is distinguished from a signaling packet based on its size: packets with more than 400 bytes are classified as a chunk of video [6]. Links are weighted by the total of traffic exchanged between any pair of nodes, during δ seconds. Given the application dynamics, it is clear that a given link can exist in *snapshot* T_i, but may not in *snapshot* T_{i+1}.

We characterize the properties of each graph $\mathcal{G}_t$ using complex network metrics, and analyze how these metrics change when computed over graphs representing successive snapshots of our private video channel. We define the metrics we have analyzed as follows.

1. In/Out-Degree:

   The in-degree of node v, $d_{in}(v)$, is the total number of incoming links. In the same way, the out-degree of node v, $d_{out}(v)$ is the total number of outcoming links. Then, the degree of node v, $d(v)$, is given by the sum of $d_{in}(v)$ and $d_{out}(v)$, representing the total number of partners of the corresponding peer. The mean degree, d_m, of a $\mathcal{G}_t$ is given by:

$$d_m(\mathcal{G}_t) = \sum_{\forall v \in \mathcal{V}_t} d(v)/|\mathcal{V}_t|.$$

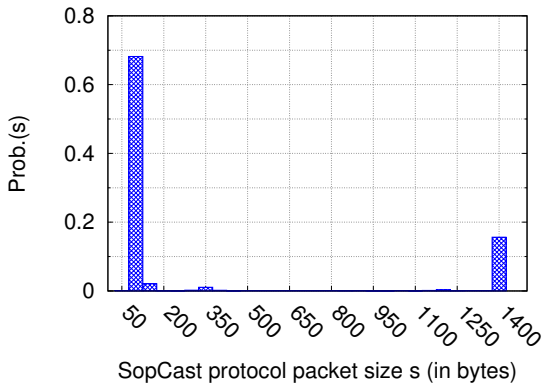

Figure 3: Packet Sizes Probability Dist. Function.

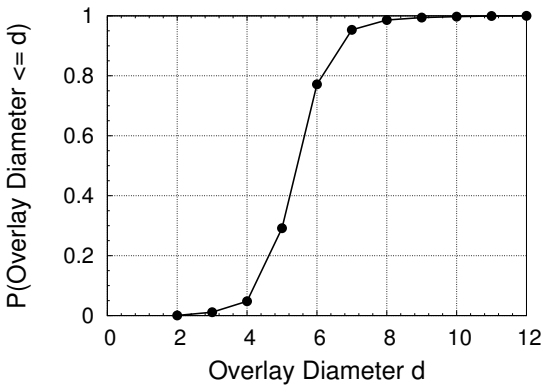

Figure 4: P2P Overlay Diameter.

2. Diameter: Let $l(u, v)$ be the shortest path between nodes u and v. The diameter is defined as the longest shortest path between any two nodes in the network:

$$\textbf{Diameter}(\mathcal{G}_t) = max_{\forall(u,v)\in\mathcal{V}_t} l(u, v).$$

This network property provides an idea of the dispersion of $\mathcal{G}_t$. Then, it is possible to infer if a given information spreads quickly over the network. For P2P streaming live applications, small diameter may indicate, for the majority of peers, a small latency in the video visualization,

Finally, we perform the system characterization from two perspectives: *P2P overlay view*, i.e., analyzing the global behavior of the whole network; and *client view*, i.e., analyzing the individual peer behavior.

4.1 P2P Overlay Metrics

In what follows, we analyze the SopCast overlay characteristics related to our experiments.

First, let us focus on the packets exchanged between any pair of peers. Figure 3 shows the probability function for the SopCast network messages size. We can see that almost 75% of the packets have 50 bytes. Almost 20% have a size of $1,400$ bytes. The large number of small packets is mainly related to the algorithms used to discover new good partners, and to maintain the overlay itself.

Additionally, Figure 3 supports the heuristics for packet classification used in this paper, and proposed in [6]. We have almost a bimodal probability function in two peaks: packets with 50 bytes and packets with $1,400$ bytes. Then small packets that are in major number are classified as signaling messages. Larger packets, however, are classified as video messages. Finally, the message size probability function indicates a high protocol overhead.

We also investigate the SopCast overlay diameter. A small overlay diameter may indicate that information spreads fast over the P2P network. Small diameter is a desirable characteristic of P2P live streaming applications: nodes expect to watch the video with a latency as small as possible. As we can see on Figure 4, the SopCast overlay diameter varies from very low values, as 2, to large values, as 12. However, the SopCast overlay presents a mean diameter as low as 6 hops in almost 90% of the time.

Larger SopCast networks may have larger diameters. However, the major number of SopCast channels is not that large. According to our experience, channels tend to have users in the order of hundreds. We have just observed crowded channels during special events, such as the decisive football matches. The experiments we have conducted represent the most common SopCast channel.

We complete our first set of results by considering the low path size distribution between any pair of peers. Figure 5 presents the cumulative distribution function of end-points shortest path size. In the SopCast overlay network, more than 96% of peers' end-to-end connections present only 3 hops. Both low overlay diameter and low size end-to-end connections may indicate a fast streaming chunk dissemination.

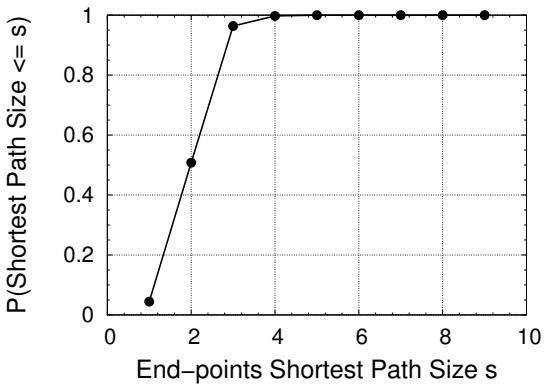

Figure 5: Mean Shortest Distance Between a Pair of Nodes.

4.2 SopCast Client Metrics

We have also characterized the SopCast client behavior. First, we quantify the aliveness of SopCast application, and turn our attention to the interval time of two consecutive data exchanging (video and signaling data), for any pair of nodes. Figure 6 reports the interval time distribution. As we can see, 96% of data exchanging occurs in an interval time less than 1 second.

Figure 7 presents the cumulative distribution function of SopCast peers degree. Peers in SopCast presents 14.85 mean peer degree (whit a high C.V. of 0.34). Moreover, there is a non-negligible number of peers, which present

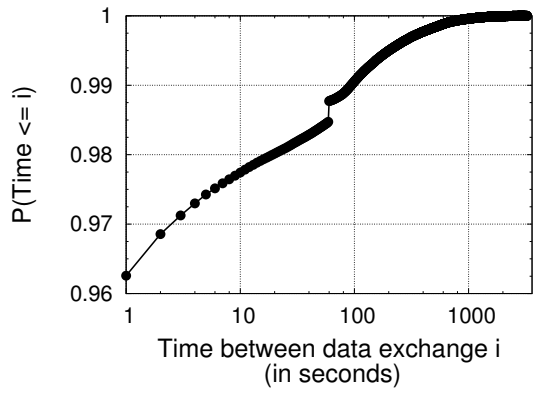

Figure 6: Time Between Peers Communication.

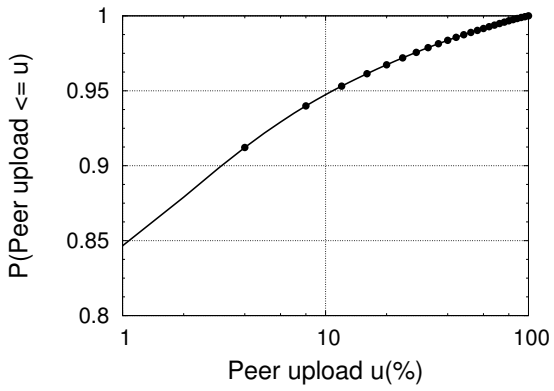

Figure 8: Peers' Upload Distribution.

more than 20 partners during a snapshot. From this result, we may infer that even if a peer exchanges signaling messages with a large number of nodes, the video data exchanging is performed with a small subset of the overall contact nodes. Finally, we note that there is not a considerable difference among the three degree curves. We may conjecture that the degree is a hard coded property of the SopCast protocol. Even if a peer does not contribute considerable to the P2P system, it tries to establish a fixed number of partnerships. Once a partnership is established, a peer eventually uploads/downloads (not necessary at a balanced ratio) data from its partner.

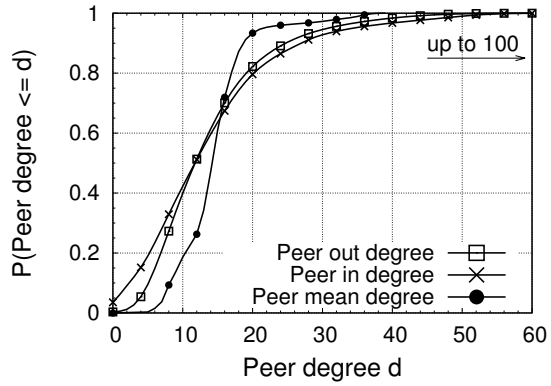

Figure 7: Peers Degree Distribution.

Finally, Figure 8 presents the peer upload characteristics. In a SopCast channel, a low number of peers upload the larger portion of the streaming. In other words, less than 10% of peers' maintain more than 90% of system streaming traffic. Based on this result, SopCast-like applications may implement reward mechanisms to favor these key nodes. Consequently, application overall quality can be improved.

5. CONCLUSIONS

In this work we provide, to the best of our knowledge, the first SopCast dataset collection available for research purposes. Our SopCast traces have been collected over a wide number of channels and scenarios. Moreover, the datasets we present have already been used in important works from our team, which reinforces their relevance.

We believe that our datasets, and also the characterization we present, can be used as a substrate to the development of new streaming media services. Researchers may use our traces to test their multimedia and P2P systems, as they will be able to recover the client behavior from an important commercial application.

The most recent set of datasets and its description are available at: `http://netlab.ice.ufjf.br/traces`.

As future work, we plan to do develop a tool to generate P2P live streaming synthetic workloads. We also plan to update and extend our dataset adding data from different P2P live streaming as TVAnts and PPlive. Moreover, we also intend to add data from storage video services.

6. ACKNOWLEDGMENTS

This work was partially supported by the Brazilian agencies FAPEMIG, CAPES, and CNPq.

7. REFERENCES

[1] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith Ross. A Measurement Study of a Large-Scale P2P IPTV System. In *IEEE Transactions on Multimedia*, volume 9(8), pages 1672–1687, 2007.

[2] Alex Borges, Pedro Gomes, José Nacif, Rodrigo Mantini, Jussara M. Almeida, and Sérgio Campos. Characterizing SopCast Client Behavior. *Computer Communications*, 35(8):1004–1016, 2012.

[3] Thomas Silverston, Olivier Fourmaux, Alessio Botta, Alberto Dainotti, Antonio Pescapé, Giorgio Ventre, and Kavé Salamatian. Traffic analysis of Peer-to-Peer IPTV Communities. *Computer Networks*, 53(4), 2009.

[4] Long H Vu, Indranil Gupta, Jin Liang, and Klara Nahrstedt. Measurement and Modeling of a Large-scale Overlay for Multimedia Streaming. In *Proc. Int'l Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pages 1–7, 2007.

[5] Benny Fallica, Yue Lu, Fernando Kuipers, Rob Kooij, and Piet Van Mieghem. On the Quality of Experience of SopCast. In *Proc. Int'l Conference on Next Generation Mobile Applications, Services and Technologies*, pages 501–506, 2008.

[6] Siyu Tang, Yue Lu, Javier Martín Hernández, Fernando Kuipers, and Piet Mieghem. Topology

Dynamics in a P2PTV Network. In *Proc. of the 8th International IFIP-TC 6 Networking Conference (NETWORKING'09)*, pages 326–337. Springer-Verlag, 2009.

[7] Sopcast. http://www.sopcast.com, outubro 2010.

[8] Kenia C. Gonçalves, Alex Borges Vieira, Jussara Almeida, Ana Paula C. da Silva, Humberto Marques-Neto and Sergio Vale Aguiar Campos . Characterizing Dynamic Properties of the SopCast Overlay Network. In *Proc. of 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP'11)*, pages 319–326. IEEE, 2012.

[9] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, jul 2003.

[10] Glauber Gonçalves, Anna Guimarães, Italo Cunha, Alex Vieira, and Jussara Almeida. Using Centrality Metrics to Predict Peer Cooperation in Live Streaming Applications. In *Proc. of 11th International IFIP Networking Conference (NETWORKING'12)*, pages 84–96, 2012.

[11] The Network Time Protocol. `www.ntp.org/`, last accessed at June 2011.