Peer-to-Peer Networking and Applications manuscript No.
(will be inserted by the editor)

Black-box analysis of Internet P2P applications

Dario Rossi · Elisa Sottile · Paolo Veglia

the date of receipt and acceptance should be inserted later

Abstract After P2P file-sharing and VoIP telephony applications, VoD and live-streaming P2P applications have finally gained a large Internet audience as well. In this work, we define a framework for the comparison of these applications, based on the measurement and analysis of the traffic they generate.

In order for the framework to be descriptive for all P2P applications, we first define a *minimum set* of observables of interest: such features either pertain to different layers of the protocol stack (from network up to the application), or convey cross-layer information (such as the degree of awareness, at overlay layer, of properties characterizing the underlying physical network).

The framework is compact (as it allows to represent all the above information at once), general (as is can be extended to consider features different from the one reported in this work), and flexible in both space and time (as it allows different levels of spatial aggregation, and also to represent the temporal evolution of the quantities of interest). Using the minimum feature set, we analyze some of the most popular P2P application nowadays, highlighting their main similarities and differences. We then apply the framework, using also different features and metrics, to two interesting case study: namely, the detection of malfunctioning or misbehaving peers, and a fine-grained analysis of P2P network-awareness and friendliness.

Keywords Traffic monitoring · Traffic characterization · Kiviat charts · Network awareness · Anomaly detection

Dario Rossi · Elisa Sottile · Paolo Veglia
Telecom ParisTech, Paris, France.
E-mail: firstname.lastname@enst.fr

1 Introduction

The population of Internet P2P applications follows a Darwinian evolution: soon after its birth, any new application offering new and exciting services, is either destined to enjoy fame and success, or to face oblivion and death. As a consequence, the offer of P2P services now spans a very wide spectrum [1–9]: besides the ever-present file-sharing applications as BitTorrent [1] and eMule [2], we use P2P application such as Skype [3] to call our friends with VoIP; for entertainment purposes, we rely on P2P-VoD and live TV applications such as Joost[1] [4], TVAnts [5], SopCast [6] and PPLive [7]; moreover, even operating system [8] and applications [9] are moving toward P2P distribution of their updates.

Despite the services proposed are different, the transport layer patterns of the traffic generated by such P2P applications share some similarities. Indeed, all P2P applications have to perform similar tasks (e.g., network discovery, queries, refresh of contact lists) irrespectively of the service they implement. Moreover, considering file-sharing and live-streaming applications, similarities are also present in the way the content is diffused (e.g., such as by spreading chunks of data over meshed overlays in BitTorrent and PPLive), though the actual content, as well as the inner algorithms for its selection, may differ (e.g., rarest chunks are selected first in BitTorrent file-sharing, while peers of streaming applications such as PPLive need to select chunks that are closer to their play-out deadline first).

Yet, each P2P application differs from the others not only for what concerns the service offered, but also from many design aspects. For instance, P2P applications differ in their architecture (e.g., unstructured, hierarchical or struc-

[1] Since October 2008 Joost is no more using P2P to deliver video content, but it was using P2P media delivery during the trace collection period.

tured), in their connectivity degree and the topology of their overlay graph, in the mechanism employed to prevent free-riding (if any), in their peer selection algorithms, in the size of the chunks used for content diffusion, in their degree of awareness of the underlying IP network, etc.

The problem thus arise of how one can represent, in a furthermore visually intuitive and compact way, the above similarities and differences. Previous research already deeply studied different P2P systems [10–32], pointing out several features to characterize important aspects of such applications. At the same time, P2P overlays have, with few exceptions [19, 24–27], been studied in isolation: therefore, what the scientific community still lacks is a mean to contrast and relatively weight such similarities and differences. Moreover, as P2P systems keeps evolving, this comparison has likely to be continuously done, as the relevance of the results may otherwise quickly become outdated. As many successful commercial applications are also closed and proprietary, a *black-box approach* is therefore needed, so that the methodology is widely applicable while avoiding at the same time the overhead of reverse engineering.

This is precisely the aim of *Sherlock* [33], a framework to *Sketch Hallmark Elements to Recap and Look-into Overlays with Charts of Kiviat*, which is able to compactly describe the traffic generated by currently deployed Internet P2P applications. As Sherlock Holmes rightly says [34], "*It is a capital mistake to theorize before you have all the evidence. It biases the judgment.*" It is not by chance that the framework has been named after the popular detective, as its primary goal is to *collect* and *present* as much evidence as possible. Collection of the evidence is based on a careful selection of the features to investigate, that convey information either pertaining to a single-layer of the protocol stack, or cross-layer information that involves several layer at the same time. Presentation of evidence leverages on the use of Kiviat charts [35], which allow to represent a great amount of possibly very heterogeneous information, in a furthermore very compact and visually intuitive way.

While Sherlock identifies a *minimum set* of features that are relevant for a wide range of P2P applications, nevertheless we argue that bounding the analysis to a single set of properties may result in a too constrained viewpoint: as such, the framework has been designed to be extremely flexible. Sherlock can be customized in both the selection of the relevant *features* (i.e., the traffic properties of interest) and of the *metrics* to be adopted for their representation (e.g., scalar vs vectorial values, etc.). Moreover, Sherlock is flexible in both the *spatial* level of granularity (e.g., from individual endpoints to endpoint aggregates), as well as in the *temporal* observation interval (e.g., long-term averages versus instantaneous snapshots).

This work extends our earlier effort [33] in several directions. First, we consider a larger dataset. Specifically, using the SopCast P2P-TV application, we explore a wider range of channels featuring different content (e.g., from football matches to news and movies, spanning also over different languages). Second, we consider a larger number of features, exemplifying how the framework can be customized to fit specific analysis needs. Third, we present two use cases of the framework: on the one hand (i) we show how Sherlock can help revealing pathological situations (e.g., malfunctioning or mis-behaving peers); on the other hand, (ii) we show how Sherlock can be used to study specific aspects of P2P applications, considering a fine-grained analysis of network awareness. Notice that, in particular, for the latter use case we implemented the Sherlock framework as an open-source software [36], available for the research community at [37].

The remainder of this paper is organized as follows. After overviewing the related effort in Sec. 2, we preliminary describe in Sec. 3 the dataset of applications considered in this work (namely, BitTorrent, eDonkey, Skype, Joost, TVAnts, SopCast, PPLive). We then introduce the Sherlock framework in Sec. 4, by applying in Sec. 5 the minimum set of features to the study of popular P2P applications. Then, further use cases of Sherlock are reported in Sec. 6 (which focuses on the analysis of pathological behavior) and Sec. 7 (which instead address network-awareness and friendliness issues), prior that Sec. 8 concludes the paper.

2 Related Work

As a consequence of P2P widespread adoption, the research activities related to P2P traffic measurement, such as characterization and classification, acquired importance [10–32].

File-sharing, being the first class of applications exploiting the P2P paradigm, has been studied for a relatively long time [10–19]: as a result, many details concerning the query process [11], user churn [12, 13] and files popularity [14] are available. In more details, researchers studied proprietary applications such as KaZaa [10], unstructured systems such as BitTorrent [15] and Gnutella [16], and Distributed Hash Tables (DHTs) such as Kademlia [17, 18]. Work comparing different protocols also exists, such as [19], which considers eDonkey, BitTorrent, FastTrack and WinMX.

More recently, proprietary P2P applications offering Internet telephony, video-conferencing and video-streaming services, have enjoyed an enormous success. This has motivated further research, and there exist already valuable work that focused on VoIP applications such as Skype [20, 21] and of P2P-TV applications such as PPlive [22] and Coolstreaming [23]. Again, work comparing different protocols also exists, such as [24] which considers PPlive, SopCast and TVAnts.

As far as methodology is concerned, the above work can be roughly divided into two classes. The first approach is to

use active crawlers, which allow to gather very detailed information from the whole network. This is however a daunting task (especially for proprietary systems, in which case a partial reverse engineering of the application is required) which practically limits the investigation to a specific system. A second set of work adopts a black-box approach, measuring and analyzing the traffic generated by the application. Our work fits in the latter class, whose advantage is to be applicable to a more general extent, though this tradeoffs with the level of details of the information at our disposal for the analysis.

With this respect, works of the latter class closest to our are [19, 24–27]: [19, 24] focus only on P2P applications, whereas [25–27] are more general but still very relevant. Nevertheless, the purpose of the above works, the features adopted in the investigation and their presentation differ from our approach. Authors in [19] and [24] compare different applications, but limitedly focus on a single P2P service (i.e., filesharing and IPTV respectively). The aim of [25, 26] is instead traffic classification, while [27] targets end-host profiling. Thus, [25, 26] consider P2P as a single class of application, which we instead decompose further, discriminating among individual applications.

As far as *features* are concerned, [26,27] consider mainly transport-layer characteristics (e.g., number of hosts contacted, on which ports), while [19, 24, 25] additionally take into account network-layer information (e.g., packet size and interarrival times). Except [19], which only marginally addresses the geographical breakdown of contacted peers, none of the above work considers cross-layer characteristics (e.g., such as IP proximity of overlay hosts). Our framerwork instead takes into account all the above aspects.

As far as the *representation* is instead concerned, simple yet powerful "graphlets" are proposed in [26, 27], that allow to abstract different transport-layer behaviors and compactly present them in a descriptive graph. Authors in [19, 24] instead characterize the different applications by means of cumulative distribution function and scatter plots of different features. Kiviat representation has been used in [25], which inspired our work, and whose differences will be highlighted in more details in the following sections.

Finally, concerning the two use-cases considered in this work, further relevant work can be identified [28–32]. Concerning the undesirable behavior of P2P applications, recent work includes for instance [28, 29], whereas network-awareness issues have been studied in the context of P2P-TV by [30–32], either considering a single [30] or several [31, 32] applications.

3 P2P Applications Dataset

The application we consider in this work are listed in Tab. 1: more precisely, we select BitTorrent and eDonkey as examples of P2P file-sharing; Skype as an example of P2P VoIP; Joost as an example of P2P VoD; SopCast, TVAnts and PPLive as examples of live streaming P2P TV. We note that all the above application largely prefer UDP[2], at the transport layer, to which we restrict our attention in the following.

To gather traffic of the above applications, we rely on both passive and active methodologies. Passive methodology implies to sniff traffic from operational network: traffic traces are then representative of real-world usage, and this methodology should thus be the preferred. In this case however, a reliable classification engine is needed to isolate the traffic generated by each P2P application, which is known to be a non-trivial problem – especially for new P2P applications offering VoIP, VoD and IPTV services. Active methodology requires instead to deploy probes in the network running the applications of choice: since probe peers are known, there is no need for traffic classification capabilities. Rather, in this case special care must be taken in order to ensure that the gathered traces are representative of real world traffic.

Given the above tradeoff, the dataset we consider in this work has been gathered using three different methodologies: (i) a purely passive approach which classifies P2P traffic from unmodified operational networks; (ii) an active experimental approach, in which we actively deploy unmodified P2P applications in several vantage point in the Internet, and passively monitor their traffic; (iii) an active probing approach, in which we augment the previous active methodology by further probing the peers contacted by unmodified P2P applications, by means of a custom software tool.

Specifically, we employ a passive methodology (indicated with "P" in Tab. 1) to gather eDonkey and Skype traffic from real networks. We resort to Deep Packet Inspection (DPI) capabilities to gather eDonkey traffic [39, 40], whereas we exploit [41] to classify Skype traffic: both classification engines have been implemented in Tstat [42], an open-source flow-level logger available at [43]. For further details concerning the classification mechanisms, we refer the reader to [44, 45].

As far as the active approach (indicated with "A" in Tab. 1) is concerned, we rely on an Internet-scale testbed: in other words, we deploy unmodified probes in different networks and passively capture packet-level traces of the traffic they generate. In order to gather results that are representative of a large number of real scenarios and usage, our P2P-TV and P2P-VoD probes are scattered in 7 Autonomous Systems of 4 European Countries, having either high-speed or DSL/Cable Internet access. Besides, we notice that beyond access technology and geographical probe position, there

[2] We point out that since December 2008 [38], file transfer in BitTorrent moved to uTP, a closed-loop congestion control protocol implemented at the application layer and running over UDP at the transport layer.

Table 1 Summary of applications analyzed in this study. Traces in the dataset have been collected with either passive (P) or active (A, T) methodologies.

Application	Service Offered	Probe Peers	External Peers	Packets [$\cdot 10^6$]	Bytes [$\cdot 10^9$]
BitTorrent	file-sharing	A,7	47,561	30.81	4.74
eDonkey	file-sharing	P,20	2,410,136	14.37	2.02
Skype	VoIP	P,15	153,755	70.96	18.77
Joost	VoD	A,37	25,481	6.97	6.87
TVAnts	live TV	A,38	13,274	9.95	5.19
SopCast	live TV	A,38	54,588	33.87	12.20
SopCast (T)	live TV	T,19	33,961	29.72	15.34
PPLive	live TV	A,44	2,159,522	158.98	77.70
PPLive (U)	live TV	A,44	189,844	18.22	9.00
Total	-	262	5,088,122	373.85	151.83

are other factors affecting P2P applications: in the case of PPLive, we therefore consider both popular and unpopular channels (the latter indicated with "U" in Tab. 1), in order to provide a larger dataset for the analysis. We also point out that some applications are more represented in the testbed (i.e., some tests involve a larger number of probe peers than others) whereas other are less well represented: this is especially true in the case of BitTorrent, in reason of its very recent evolution which limited the number of experiments we were able to perform. For further details concerning the Internet-scale testbed, we refer the reader to [32].

Finally, while the A and P datasets were already considered in earlier version of this work [33], we further perform additional experiments in the case of SopCast live-TV streaming, exploring a wider range of channels featuring different content (e.g., from football matches to news and movies, considering also different languages). In this case (indicated with "T" in Tab. 1), we perform experiments with the P2PGauge black-box software tool, that we specifically designed to couple passive traffic observation with active probing of the peers contacted by unmodified P2P applications. The P2PGauge software tool implements the Sherlock framework, and is available as open-source software at [37]. As we will focus on this dataset for the network-awareness case study, we defer further details concerning the methodology and the gathered dataset in Sec. 7.

Finally, we stress that the overall size of the considered dataset is significant, since our 262 probes contacted about 5 millions external peers, exchanging with them about 151 GBytes of data in 373 millions of packets. For reference purpose, the dataset size of closest works to ours amount to 200 hosts in [27], 20 thousand flows in [25], "several thousands"[3] hosts in [19], about 19 GBytes of data in [24] and 3 TBytes in [26].

3.1 P2P Applications at a Glance: Traffic Patterns

P2P application typically use a single *end-point*, identified by their IP address and transport layer port pair (IP, P), over which they multiplex signaling and service traffic. In order to show, at a glance, similarity and differences of the P2P applications listed in Tab. 1, let us depict in Fig. 1 the activity of a few end-point samples.

Each plot in Fig. 1 concerns a single probe X per application, chosen as the most active probe in our dataset, of which we depict one hour worth of traffic. Time runs on the x-axis, while the y-axis represent an arbitrary identifier for external peers P contacted, starting at 0 and incremented by one unit for each new peer contacted. Each dot in the picture corresponds to a packet in the trace: packets sent from X to P have a positive identifier $\mathrm{ID}(X, P)$, whereas packets received from X and sent by P have a negative identifier $\mathrm{ID}(P, X) = -\mathrm{ID}(X, P)$.

Intuitively, this representation tells us a wealth of information concerning peers activity. For instance, at any given time, the range of the y-values corresponds to the portion of the overlay discovered by peer X. The fact that the y range grows over time for most applications implies that network discovery is carried out during the whole peer lifetime: notice indeed that some peers are contacted only once, by the transmission of a single packet, to which (most of the times) some kind of acknowledgment follows. Moreover, the slope of the curve identified by the maximum ID is related to the rate and intensity of the network discovery task: indeed, the steeper the slope, the higher the network probing rate.

These properties are remarkably different across the considered applications. For instance, notice that plots are ordered (left to right, top to bottom) according to the number of peers contacted during the one hour interval. Such number varies widely across applications: for instance, Joost contact the least number of peers (100); the number increases for TVAnts (250) and SopCast (500), raises to about 1,000 for BitTorrent and Skype, exceeds 10,000 contacts for eDonkey and reaches up to 45,000 contacts for PPLive (15,000 in case of unpopular channel).

At the same time, the largest part of data exchange happens with peers that are contacted on a regular basis: in the activity plot, points that fall below the network discovery line state that the same peer is contacted several times during X lifetime. Indeed, horizontal lines are visible on the plots, which correspond to stable and regular contacts, which are possibly carried on during the whole peer X lifetime. The number of such lines again varies widely across applications, though their actual number is difficult to grasp from the plot: in the case of Joost VoD services, only a few stable contacts are noticeable, whose duration furthermore extends across the whole hour. In case of TVAnts, again a few stable contacts are clearly visible, but their duration is shorter. Conversely, SopCast transmissions are more scattered among all contacts, alternating short on/off periods of silence and communication with many peers. Skype signaling pattern is very regular though very complex: no VoIP

[3] Due to NDA, authors in [19] only disclose *relative* amounts.

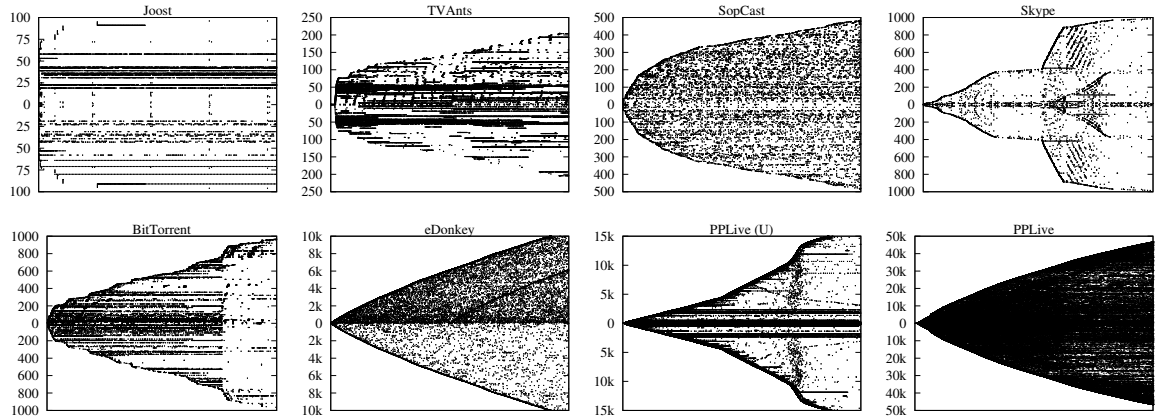

Fig. 1 P2P applications at a glance: Traffic patterns, conveying network and transport layer information and their temporal evolution

call was ongoing during the experiment, and the probe alternates quiescent times where no traffic is exchanged, to intense phases of network discovery corresponding to steep increases of the maximum ID. Notice also that the behavior of the application may be heavily influenced by properties of the service, as in the case of PPLive: in the case of unpopular channel, one can notice several lines, whose number is difficult to quantify but in any case much smaller than the number of peers in the overlay; conversely, lines are no longer visible in the much more scattered PPLive popular channel case[4]. At a first glance also, almost all patterns are roughly symmetrical with respect to the x-axis, with the exception of eDonkey and Joost endpoints: yet, it is hard to precisely quantify the symmetry level, e.g., to state whether the endpoints send to their contacts as many packets (and bytes) as they receive.

Therefore, despite the above representation convey a number of useful information, is it clear than it hides much more than what it shows. Moreover, while the activity plot is a very handy tool to represent a *single* application instance, it does not generalize well to represent a multitude of endpoints, nor it can present a more comprehensive view of the traffic. Indeed, apart from macroscopic quantities and differences, the activity plot fails to capture important aspects of the peers exchanges (such as the amount of peers falling into the same AS and the amount of bytes exchanged with them, whether peers use random or fixed ports, whether data exchanges are symmetric due to tit-for-tat, etc.), which are essential in order to provide a full-relief characterization and comparison of P2P applications.

[4] Notice that, the pattern of PPLive in the popular channel case (i.e., bottom right plot) has been aggressively subsampled for the sake of readability, as otherwise no points would have been visible but a very dense black cloud: specifically, out of the total $6.4 \cdot 10^6$ packets exchanged by that particular peer, only about $2.5 \cdot 10^5$ have been used in the visualization. Yet, we point out that sampling has been carefully performed to preserve the pattern visual structure: only 1 out of 10 probes are represented, and at most 1 packet every 5 seconds has been reported for other peers.

4 Framework Definition

In reason of the above observations, we built the Sherlock framework with the following design goals:

- **Expressiveness and readability:** represent possibly many features at the same time in a visually compact, intuitive and readable way;
- **Feature-flexibility:** capture key P2P features which are intrinsically different in nature (e.g., packet size and inter-arrival time, connectivity degree, geographical peer location, etc.), possibly adapting the choice of features to the specific aspect under investigation;
- **Metrics-flexibility:** represent different metrics of any given feature, both scalar (e.g., mean, variance, coefficient of variation) or vectorial (e.g., empirical probability mass function, etc.);
- **Spatial-flexibility:** zoom at different levels of granularity, considering peers either individually or aggregated (e.g., set of peers in the same sub-network, AS, country, etc.);
- **Temporal-flexibility:** express long-term averages as well as temporal snapshots of the system behavior (e.g., since the beginning of the peer lifetime or during an arbitrary time-window).

In what follows, we describe Sherlock by decoupling the *choice* of the features that we use to characterize the P2P applications from their *representation*, whose detailed description will be addressed later in Sec. 5. We point out that, for the sake of clarity, we denote as "features" the properties that we are interested in observing (e.g., packet size, RTT, etc.), while we indicate with "metric" the way in which features are expressed (e.g., scalar average, vectorial probability mass function, etc.). Sherlock representation is based on

Kiviat graphs [35], a very simple but expressive means of representing heterogeneous information in a compact and flexible way. Introduced in the 70s to characterize CPUs workload, Kiviat graphs have been used in networking research by [25], which considers different classes of application (e.g., Web, interactive, VoIP, etc.) and represent some of their noteworthy characteristics by means of Kiviat graphs for the purpose of traffic classification.

Inspired by [25], our work differentiates from it in many aspects. In our case, we target the characterization of P2P traffic, rather than its identification, and we consider individual P2P applications (as opposite to coarse application classes). Our work also differs in the choice of the observables, which are in our case tailored for P2P applications. Moreover, our methodology is flexible in space, as it applies to endpoints and endpoint aggregates (as opposite to flow-level only). Finally, our methodology is also flexible in both the choice of the observable metrics and features, as we will show later on in Sec. 5.2.3 and Sec. 7 respectively.

4.1 Features Definition: Hallmark of P2P Traffic

From a high level point of view, the usefulness of the framework, as well as the depth of the insights produced by its use, largely depend on the choice of the features to be represented. Since we want the framework to produce readable results, this implies that we need to limit the amount of information to display. Moreover, since we want the framework to be applicable to any P2P application without requiring reverse-engineering, we need to individuate features that can be measured by a purely black-box approach.

Focusing on P2P traffic, we can define a number of interesting features, pertaining to different layers, such as:

- **Network**: e.g., packet size and inter-arrival, bitrate, etc.
- **Transport**: e.g., randomness of used ports, symmetry of the exchanges, preferred transport layer protocol, etc.
- **Application**: e.g., overlay degree and stability, network probing and discovery rate, overlay topology, etc.
- **Cross-layer**: e.g., awareness of IP-underlay properties at P2P-overlay layer, etc.

In this work, we devise a *minimum set* of features able to convey telling information concerning a wide range of P2P systems: in the reminder of this section, we will highlight the principle of our choice. At the same time we point out that, in reason of its flexibility, the Sherlock framework can profitably be applied with features other than the minimum set (useful whenever one wants to focus, e.g., on a specific aspect of P2P applications, or on a specific layer of the protocol stack). As an example, we therefore provide a set of features and metrics tailored to investigate the network-awareness issue in Sec. 7.

Before introducing the minimum set of features, we also need to outline an important remark concerning the range of values taken by a metric $x(F)$, independently thus from the semantic of the feature F. At first sight, it may seems that metrics that can be represented with a pre-determined fixed range (such as the unity interval $[0, 1] \in \mathbb{R}$) should be preferred. The advantage of fixed-range[5] metrics is that their representation is easier (as their range is known in advance) and moreover results are directly comparable (e.g., across different applications, different endpoints of a same application, etc.). Yet, in some case such metrics hide a useful *absolute* information (e.g., the magnitude of the normalization factor), that could assist the interpretation of the results (e.g., as in the case of packets inter-arrival times and size, application bitrate, number of peers contacted, network discovery rate, etc.). In this case, interpretation of the metric will be easier, though the selection of the range for its representation can be more difficult in reason of its variability.

4.2 Network-layer Features

Network layer features characterize P2P traffic at packet level. We argue that packet size correlates with the type of activity carried on (e.g., data/video transfer will likely used bigger packet sizes with respect to signaling activity, network discovery, keep-alive, etc.), and is thus a telling observable. Similarly, packet inter-arrival time (IAT) also conveys useful information: for instance, the mean inter-arrival time is correlated with the level of activity of a given end-point, while its variation is related to the burstiness of the arrival process.

Directionality plays an important role in this case, as very important differences may arise in the received versus transmitted traffic. For instance, as early noticeable in Fig. 1, eDonkey traffic is not symmetric (in the endpoint shown in the figure, the intensity of outgoing traffic is much higher, and thus IAT is much smaller with respect to the incoming traffic direction). Similarly, we may expect BitTorrent traffic to be mainly outgoing during seeding, and more balanced otherwise (i.e., due to tit-for-tat). We also expect downlink P2P-TV traffic to be steady (i.e., roughly equal to the stream rate) whereas the uplink may be much more variable (i.e., depending on the number of peers to which the same chunk is re-distributed [22]). As such, packet size and interarrival times need to be separately measured for incoming and outgoing traffic. Finally, it is unnecessary to explicitly consider the IP bitrate, as the same information can be gathered by the joint examination of the packet size and IAT features.

[5] Notice that range of values can either implicitly extend over the $[0, 1]$ range (e.g., as in case of a percentage, a breakdown, etc.) or be mapped to $[0, 1]$ (e.g., by normalization as in $\hat{x}_i = x_i / \max_i x_i$).

4.3 Transport-layer Features

Transport layer features concern flows rather than individual packets: we consider two different features to evaluate the port space usage and the symmetry of the traffic flows.

Concerning transport layer ports, it is well known that some P2P application initially used fixed port ranges for all their exchanges (e.g., port 4662 for eDonkey and port range 6880-6889 for BitTorrent), whereas this changed with newer applications that employ a random port (e.g., Skype, PPLive), which is possibly changed across sessions but is typically chosen only once at installation. It is thus interesting to test whether the external peers contacted are more likely to use few ports chosen in a given range (e.g., hard-coded in the application) or pseudo-random ports chosen independently by each peer. We discriminate between these two rough behaviors, by evaluating the fairness F_{port} of the port range utilization. Focusing on an endpoint X, let denote with n_i the number of peers in its neighborhood set $\mathcal{N}$ that employs port number i. We then define the fairness as

$$F_{port} = \frac{(\sum_i n_i)^2}{N \sum_i n_i^2} \tag{1}$$

where $N{=}\sum_i n_i{=}card(\mathcal{N})$ is the total number of contacted peers. Intuitively, F_{port} is close to 1 as long as peers use different ports, whereas it equals $1/N$ whenever all peers use the same port. Notice that peers are counted exactly once, irrespectively of the amount of traffic they exchange.

Moreover, we wish to assess whether exchanges among peers are symmetrical in terms of the volume of packet and bytes exchanged, or whether a direction is prevalent. Intuitively, packets and byte symmetry reflect rather different design choices. On the one hand, applications using a per-packet acknowledgement policy over UDP will be highly symmetrical counting traffic packet-wise. On the other hand, byte-wise symmetry will only show up when the amount of data transferred is comparable for both directions (e.g., due to tit-for-tat). More formally, consider a single flow between peers X and Y, and denote with $P(X,Y)$ and $B(X,Y)$ the amount of packets and bytes sent from X to Y, and with $P(Y,X)$ and $B(Y,X)$ the amount in the reverse direction. We then define the packet-wise Sym_P and byte-wise Sym_P symmetry indexes as:

$$\text{Sym}_P = \frac{P(X,Y)}{P(X,Y)+P(Y,X)} \tag{2}$$

$$\text{Sym}_B = \frac{B(X,Y)}{B(X,Y)+B(Y,X)} \tag{3}$$

Intuitively, these variables are equal to 0.5 when the same number of packets (Sym_P) or bytes (Sym_B) flow between the peers, while both indexes tends to 1 (or 0) when all the traffic is outgoing from (or incoming to) peer X.

Notice also that other interesting features pertaining to the transport layer include the quantification of the probing (i.e., single-packet flows sent out by peers to perform overlay network discovery) and signaling overhead. These two traffic components are usually separated from the rest of the "service" traffic (i.e., video, data, voice, etc.) by means of threshold-based heuristics (i.e., requiring service flows size to exceed a given threshold, possibly coupled to a threshold on the size of individual packets). Yet, as the precise value of these threshold differ across applications [19, 22] we prefer to adopt a conservative approach and leave these features out of the minimum set.

4.4 Application-layer Features

Application layer features concern the overlay graph: as topology inference requires active crawling of the P2P system, we resort to simpler features to characterize the overlay graph, such as its degree, contact stability and peer discovery rate.

Without loss of generality, let us consider windows of length ΔT. Let $\mathcal{P}_k$ be the set of peers whom peer X exchanged packets with during the k-th time window – i.e., considering only packets exchanged during the interval $[(k-1)\Delta T, k\Delta T]$. Similarly, denote with $\mathcal{N}_k$ the set of all peers discovered from time 0 until time $k\Delta T$. Formally, we have:

$$\mathcal{P}_k = \{p : P(X,p) + P(p,X) > 0\} \tag{4}$$

$$\mathcal{N}_k = \cup_{i=1}^{k} \mathcal{P}_i \tag{5}$$

Notice that, for the sake of simplicity, we define in this case a-directional features. Again, we follow a conservative approach and do not further differentiate the peer type (e.g., signaling versus data contributor peers) which is usually done by requiring a minimum amount of bytes and packets exchanged [22, 24].

We then define features to count the instantaneous degree $P_{\Delta T}$ of the endpoint, the number of peers discovered in the last time-window P_{new} and the number of stable peers P_{same} that were contacted in the previous time-window and that are still contacted in the current one. Formally, we have:

$$P_{\Delta T} = card(\mathcal{P}_k) \tag{6}$$

$$P_{new} = card(\mathcal{P}_k \backslash \mathcal{N}_{k-1}) \tag{7}$$

$$P_{same} = card(\mathcal{P}_k \cap \mathcal{P}_{k-1}) \tag{8}$$

Clearly, these indexes will reflect different kind of activities (e.g., an idle Skype peer versus a peer sending a message to all its buddies to notify them of a status change). Moreover, these indexes will also change during the application lifetime (e.g., as network discovery rate may be more intense at startup), therefore it will be interesting to assess their temporal evolution as well.

4.5 Cross-layer Features

Finally, cross-layer features represent the awareness that the P2P overlay has of the underlying IP network properties,

such as IP host proximity of overlay peers. Peers proximity can be expressed in a number of way, as for instance using the RTT delay or IP hop-counts distance among peers. Proximity can also be expressed as the fact that two peers belong to the same Autonomous System (AS) or that they are located in the same geographical Country (CC).

In Sec. 7 we will explore a wide range of cross-layer features, in order to assess, at a fine-graine, the level of IP network-awareness embedded in a P2P application. As such, for the time being we are rather interested in considering a single, simple, compact indicator of network-awareness. To this purpose, we point out that by passive measurement of UDP traffic is difficult to infer RTT latency, since reverse engineering is needed to match data packets with the corresponding application-layer acknowledgements. Conversely, the IP hop-count distance is easier to measure, but far less meaningful than RTT to express network awareness. Finally, AS preference is a relevant feature, that is however unable to capture proximity methods implemented by means of RTT measurement at the application layer. We argue that CC feature can instead convey useful information concerning both AS and RTT: indeed, two peers that are in the same AS are also in the same CC, while RTT of two peers that are in the same CC is likely smaller that of faraway peers.

We thus select the CC feature and geolocalize peers IP addresses by means of an open database [47], and evaluate the percentage CC_P of peers that belong to the same Country over the total number of contacted peers (and the percentage of bytes CC_B exchanged with them). Intuitively, CC_P and CC_B will reflect different aspects depending on the application, so that their interpretation will not necessarily be the same across application. For instance, in the case of an interactive service as Skype, CC features will be affected by both the location of the overlay super-peers as well as of the location of Skype buddies. In case of content to be diffused (as in file-sharing and live TV streaming) geolocation will rather reflect the preferred location to download content, which is possibly affected by both proximity-aware peer selection (e.g., download preferentially from closest peers) as well as by the content type (e.g., as the popularity of movies/music/etc. may be bound to Country borders).

5 Experimental Analysis

5.1 Framework Expressiveness

Fig. 2 reports the Kiviat representation of all dataset, using the same application order than Fig. 1. A Kiviat chart consists of several axis represented in the same planar space. Each axis reports a different feature, and in Fig. 2 we represent the minimum set of transport-layer (F_{port}, Sym_B, Sym_P), application-layer ($P_{\Delta T}$, P_{same}, P_{new}) and cross-layer (CC_B, CC_P) features.

Table 2 Tabular representation of Sherlock data: mean values of the minimum feature set

Feature	Joost	TVAnts	SopCast	Skype
CC_B	2.86	35.30	6.58	71.54
CC_P	7.29	6.19	2.93	4.18
$\mathcal{P}_{\Delta T}$	15.09	23.41	55.32	1.93
$\mathcal{P}_{same}$	11.36	21.41	44.92	0.86
$\mathcal{P}_{new}$	0.72	0.38	1.72	0.16
Sym_P	0.08	0.52	0.50	0.45
Sym_B	0.03	0.50	0.32	0.40
$\mathcal{F}_{port}$	0.12	0.16	0.81	1.00

Feature	BitTorrent	eDonkey	PPLive(U)	PPLive
CC_B	0.48	0.18	19.58	3.34
CC_P	1.18	2.03	2.37	0.07
$\mathcal{P}_{\Delta T}$	21.70	31.57	26.78	362.05
$\mathcal{P}_{same}$	15.10	1.73	23.37	215.40
$\mathcal{P}_{new}$	1.67	5.32	0.85	47.73
Sym_P	0.51	0.64	0.54	0.51
Sym_B	0.43	0.68	0.44	0.81
$\mathcal{F}_{port}$	0.98	0.10	0.81	0.85

Focusing on a single application, for each feature we report the mean value μ over all peers in our dataset for that application: by joining the mean values of different features with a black thick line, we obtain a closed shape – the Kiviat chart. To show the variability of applications behavior among different peers, we use thin lines to represent the standard deviation σ of the features, and depict them relatively to the average (i.e., thin lines represent $\mu \pm \sigma$) and we shade the area between the curves for the sake of readability. For each feature, we report the maximum range value under the feature label of each axis directly in the graph (the same range is used for all applications except in the bottom right plot, corresponding to the popular channel case of PPLive). Notice that the closed shapes are remarkably different across applications, allowing us to quickly compare the P2P systems. To better highlight the visual expressiveness of Kiviat charts, we report in Tab. 2 the mean value of the considered features in a tabular format: comparing all the different applications at once is in this case clearly harder, even though Tab. 2 conveys less information (i.e., average value only) with respect to Fig. 2 (i.e., both average and standard deviation values).

Several interesting observations can gathered from Fig. 2. For instance, considering transport layer characteristics, one can notice that only Skype, BitTorrent, SopCast and PPLive employs random ports ($F_{port} \rightarrow 1$), while Joost, TVAnts and eDonkey seems to have preferred ports. Almost all applications send roughly as many packets as they receive ($\text{Sym}_P \simeq 0.5$), which suggests a per-packet acknowledgement policy, with the exception of Joost ($\text{Sym}_P < 1/10$) and eDonkey ($\text{Sym}_P > 0.65$). Exchanges are instead rather unbalanced when it comes to the amount of bytes transferred: in this case, only BitTorrent, TVAnts and the unpopular channel of

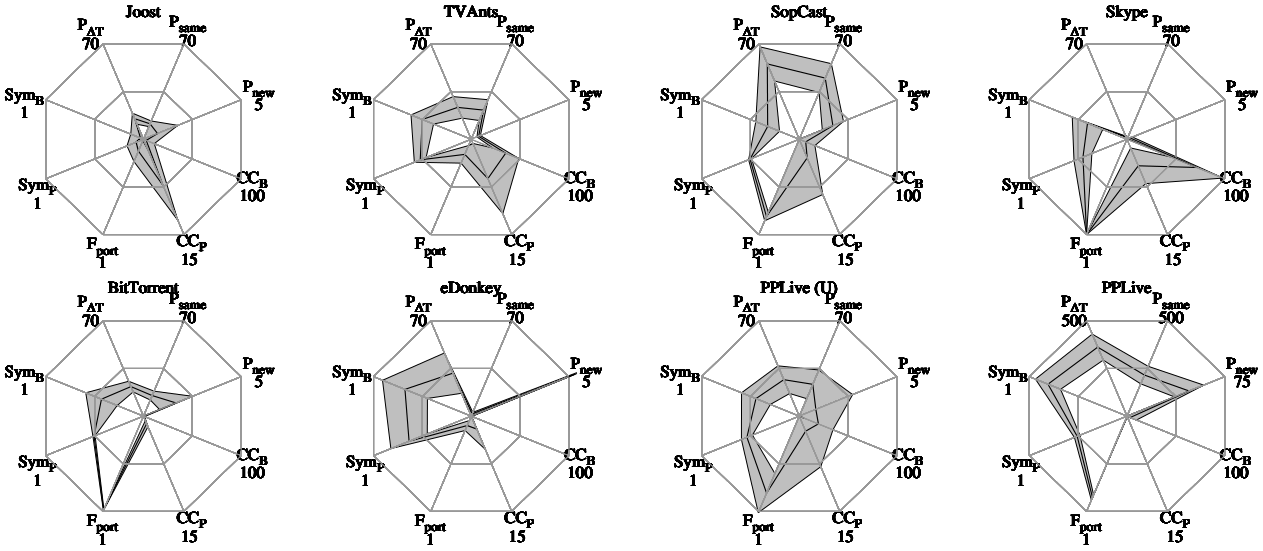

Fig. 2 P2P applications at a glance: Kiviat representation of transport, application and cross-layer information: each axis reports a specific feature (notice that ranges differs for PPLive case). Thick line joins the average over all dataset probes, thinner lines and gray shading are used to represent the standard deviation relatively to the average.

PPLive happen to be fairly symmetrical ($\text{Sym}_B \simeq 0.5$). Conversely, traffic is mostly incoming for Joost (i.e., implying that not many peers are asking our probes for video content) and mostly outgoing in the popular channel of PPLive (i.e., meaning that many peers download video chunks from our probes).

As far as application-layer features are concerned, we observe rather different behaviors, starting from the number of peers contacted during a $\Delta T = 5\,\text{s}$ window. While Joost and Skype contact very few peers (low $P_{\Delta T}$) during the same time window, PPLive, SopCast and eDonkey instead keeps a large number of contact open at the same time. Yet, we can notice important differences: while in the case of PPLive and SopCast, about half of the peers were already contacted in the previous windows ($P_{same}/P_{\Delta T} \simeq 0.5$), in the case of eDonkey contacts are much less stable ($P_{same}/P_{\Delta T} \to 0$). Probing rate ($P_{new}$) varies widely across applications and overlay size: consider for instance that PPLive discover about 50 new peers every ΔT round in the popular channel case, while this number drops by more than an order of magnitude in the unpopular channel case. Network discovery process is also quite active for eDonkey ($P_{new} \simeq 5$), SopCast and BitTorrent, while it is slow, on average, for Joost, TVAnts and Skype.

Finally, as far as cross-layer features are concerned, we can observe that Joost, TVAnts and SopCast discover a fair amount of peers located in the same Country (mean CC_P varies from 3% to 7%): at the same time, only TVAnts peers successfully confine a significant amount of data exchange within country borders ($CC_B = 35\%$), whereas proximity-aware data exchange drops for Joost and SopCast ($CC_B < 5\%$). A different phenomenon happens in the case of Skype, which sends most of the traffic ($CC_B > 70\%$) to peers in the same Country, even if they constitute only the $CC_P = 4\%$ of the peer population. Since no call were made, traffic is mostly constituted by signaling, hinting to a proximity-aware super-peer selection (possibly coupled to the fact that the buddy list contains many people living in the same country). Conversely, as Skype free services are used to phone faraway people, we can expect that the amount of VoIP traffic sent during a call would outweigh the geolocalized signaling traffic (thereby decreasing CC_B significantly). Finally, geolocalization is modest for BitTorrent, eDonkey and the popular channel case of PPLive (while it is non-negligible in case of unpopular PPLive channel).

It is worth stressing that behavior can significantly differ between peers of a single application, such as for a Skype peer making a call vs an idle peer, or in the popular vs unpopular channel case of PPLive. In the latter case, channel popularity affects the overlay size, which in turns massively reflects on the application-layer statistics: in the unpopular channel case, $P_{\Delta T}$, P_{same} and P_{new} decrease by about one order of magnitude. The reduced overlay size has clearly no effect on transport-layer statistics such as the packet-wise symmetry Sym_P and port usage F_{port}. At the same time, the lower the channel popularity, the lower the number of peers looking for the content, which explains the reduction of Sym_B. Also the increase in the cross-layer statistics CC_P and CC_B follows popularity reduction: since all of our probes watched the same channel at the same time, they stand out (i.e., CC_P increases) as a consequence of the overlay size reduction. Also, as the channel is not popular,

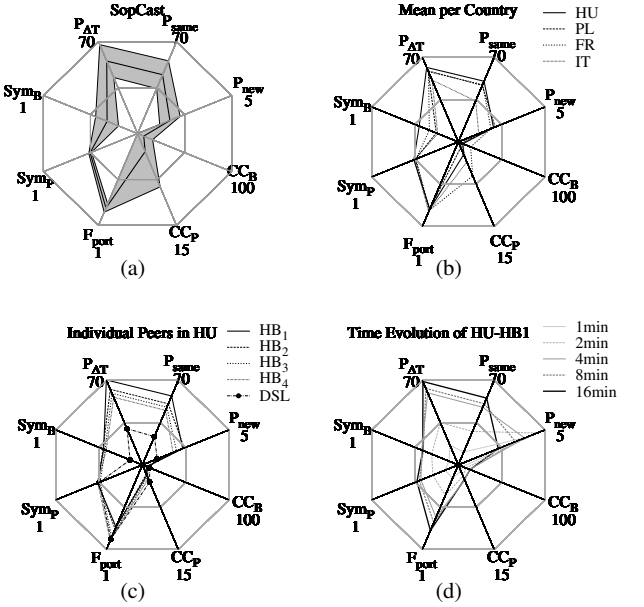

Fig. 3 Spatial and temporal flexibility: Kiviat representation at different granularities for SopCast application: (a) mean and standard deviation over all peers in the dataset, (b) mean over all peers belonging to the same Country, (c) individual peers in a single Country, and (d) temporal evolution of a single peer

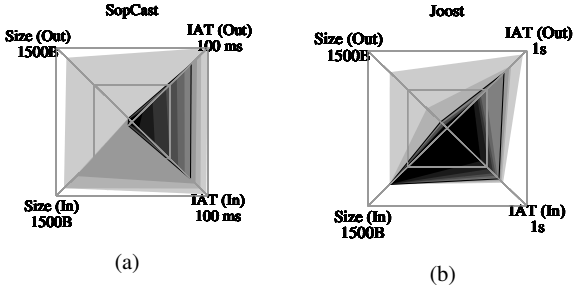

Fig. 4 Metrics flexibility: Kiviat representation of network-layer statistics: percentiles of the packet size and IAT distributions, for outgoing and incoming traffic directions, for the SopCast (a) and PPLive (b) applications

the content can be found only at a fewer number of peers, raising the impact of content diffusion proximity measured by CC_B.

5.2 Framework Flexibility

As previously stated, flexibility is among the primary Sherlock goals: in this section, we show that the framework is flexible for what concern spatial and temporal aggregation, as well as for what concerns the metrics used in the representation. We do not explicitly address feature flexibility here, since in Sec. 7 we will define a set of custom properties that are specifically targeted for the analysis of network awareness and friendliness.

5.2.1 Spatial Flexibility

Focusing on a single application, namely SopCast, we now show Sherlock flexibility by adopting different levels of granularity in our observation. At the highest level, we have a single aggregate, constituted by all SopCast peers in our dataset, of which we plot the mean and standard deviation in Fig. 3-(a). At a finer granularity, we can consider instead different subsets of probes: for example, each line in Fig. 3-(b) reports the mean over all dataset probes belonging to the same Country, while we avoid representing the standard deviation for the sake of readability. From Fig. 3-(b), it can be seen that while some features (e.g., such as packet-wise symmetry and fairness of port usage) have rather similar values irrespectively of the network where probes are located in, some other features (e.g., such as byte-wise symmetry, geolocalization and network discovery) instead may vary significantly across network environment.

This follows from the fact that some features are directly tied to design decisions and are not influenced by network conditions (e.g., per-packet acknowledgement policy, hard-coded port, hard-coded number of neighbor peers, etc.). Conversely, other features may instead be heavily influenced by the network and overlay conditions (such as for instance access technology, resource popularity, etc.). As such, the number of probes that need to be observed, in order to gather results that are representative of the full range of possible application behaviors, may change depending on the feature under observation. This is a very important point, which is still open and that motivates a large and continuous measurement campaign.

At an even finer level of granularity, Fig. 3-(c) plots the Kiviat of a few individual SopCast peers, among those located in the Hungarian country represented by a solid black line in Fig. 3-(b). It is easy to spot different behaviors, such as the ADSL peer (represented with lines and points), which contacts about half of the peers with respect to high-bandwidth HB peers (low $\mathrm{P}_{\Delta T}$, P_{same} and P_{new}), and that mostly receives traffic (low Sym_B) due to its uplink/downlink capacity asymmetry.

5.2.2 Temporal Flexibility

Temporal flexibility of the framework is testified by Fig. 3-(d), which depicts the temporal evolution of the feature set considering the HB1 peer represented with a solid black line in Fig. 3-(c). Several Kiviat graphs are overlaid in Fig. 3-(d), each of which represents the mean value of the observables at time $T=\{1, 2, 4, 8, 16\}$ minutes after the beginning of the peer activity. Notice that the mean is computed over the interval $[0, T]$, so that values represented in subsequent intervals can be thought as a moving average. Colors are darker

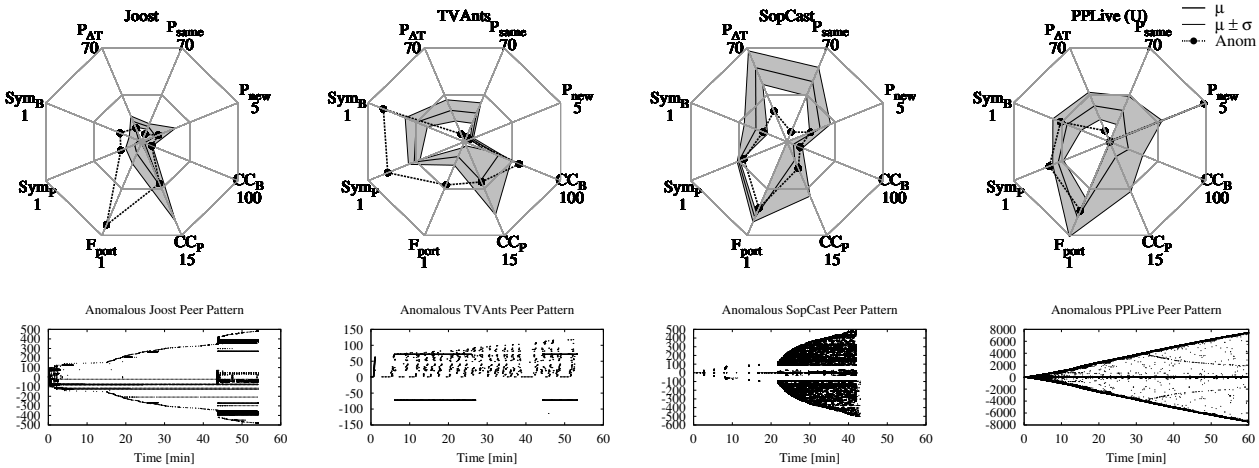

Fig. 5 Misbehaving and malfunctioning applications: Kiviat representation of average and anomalous endpoint behavior (top) and corresponding traffic level pattern of the anomalous endpoint (bottom) for P2P-TV and P2P-VoD applications.

for recent intervals, fading lighter toward the past: a clear transient can be seen for all features when $T<4$, which then progressively stabilizes for $T\geq 4$. During the transient phase, the number of new peers contacted every $\Delta T=5$ seconds is larger than in steady state, hinting to more aggressive network discovery at startup.

5.2.3 Metric Flexibility

Irrespectively of the specific feature considered, Sherlock still offer freedom in its representation: indeed, Kiviat charts not only cope with scalar values, but also allow a richer vectorial representation. For instance, Fig. 4 depicts the distributions of the packet size and Inter-Arrival Time (IAT) network-layer statistics for two P2P-TV applications, namely SopCast and PPLive.

More precisely, Fig. 4 reports a few representative percentiles (namely, from the dark 10-th, to the light 90-th in step of 10), where for readability a thick black line indicates the median (or 50-th percentile). IAT axes use a logarithmic scale, ranging from 1 μs to 100 ms, whereas packet size use a linear scale from 0 to 1500 Bytes. The plots discriminates incoming versus outgoing traffic directions: notice that statistics are computed at the network-layer, thus not making any distinction among flows.

From Fig. 4, one can gather that SopCast IAT is more symmetric than that of PPLive, where IAT of outgoing traffic is smaller due to the very high number of serviced peer. Considering packet size, it can be seen that small (signaling) packets dominate SopCast traffic, with a larger portion of big video packets in the incoming direction. In the case of PPLive, incoming traffic is instead constituted by small application-layer acknowledgements, gathered in reply to big outgoing video packets distributed to a significant number of peers.

6 Case Study: Anomalous Endpoint

As a further example of use, we show how the Sherlock framework can assist the identification of pathological situations. By pathological, we mean situation in which the Kiviat chart of a specific endpoint *significantly* differs with respect to the average chart gathered over *all* endpoints of that application (i.e., including the anomalous ones as well). We point out that the precise definition of criteria (e.g., how many features? on which metric? above/below what threshold? etc.) corresponding to an actually pathological situations, is an interesting area for research that however falls outside the scope of this work. As a consequence, our aim is not to propose and evaluate the performance of a technique able to detect such anomalous behavior. Rather, we aim at contrasting pathological and normal behaviors by means of Kiviat charts, assisting the analysis of the root cause for such misbehavior.

Considering the set of P2P-TV and P2P-VoD applications, we isolate a few anomalous peers, of which we depict with a dotted line and solid black points the individual Kiviat chart in the top plot of Fig. 5, along with the aggregated Kiviat charts (i.e., the mean and standard deviation shown early in Fig. 2) for reference purposes. For the sake of completeness, bottom plots of Fig. 5 report instead the transport pattern, that have instead to be contrasted with those reported Fig. 1. While the transport pattern quickly conveys the idea of some misbhehavior (e.g., asymmetry in Joost and TVAnts, delayed start and sudden stop in SopCast, very sparse pattern in PPLive), it would be instead rather hard to define a *normal* average pattern, which can instead easily be done with Kiviat charts. Indeed, by overlaying Kiviats corresponding to different levels of granularity in a single plot, it is easy to spot individual peers behaviors that significantly deviates with respect to the average one (i.e., the solid dots

helps in highlighting that several features of the anomalous peers fall well outside the standard deviation boundaries).

In case of Joost, the content is provided by a smaller amount of peers, and after a rather long period (i.e., $t \in [0, 50]$ min) during which the peer is mainly probing the network and receiving content, the peers starts sending video/signaling as well (i.e., higher Sym_P and Sym_B than usual). Notice that, oddly, the contacted peers use in this case rather different port numbers as F_{port} suggests: this may be, partly, consequence of a slightly more aggressive peer discovery rate F_{new}, which increases the number of peers discovered (with respect to the number of Joost servers), and the chances that a random port is used as well.

In case of TVAnts, high Sym_P and Sym_B values correspond to mostly outgoing traffic, hinting to the fact that the video stream was not correctly received. Moreover, the peer contacts a lower number of peers than the usual one $P_{\Delta T}$, but its discovery rate P_{new} is not higher than the usual one (i.e., the peer does not try to recover the lack of content by aggressively probing new overlay nodes). In this case, it seems as though a significant signaling phase is conducted with few users, although the communication pattern is sporadic (i.e., low P_{same}).

In case of SopCast, the peer first exhibits a difficult startup ($[0, 20)$ min), then exhibits a normal behavior ($[20, 40)$ min), to fail suddenly later on ($[40, 60]$ min). Even in this very hard case, where thus anomalous and normal behaviors mix over time, an aggregated Kiviat chart over the whole interval still allows to pinpoint some differences (e.g., significant for $P_{\Delta T}$, P_{same} but almost negligible for Sym_P, F_{port}).

Finally, in case of PPLive notice again that the number of contacts is extremely low $P_{\Delta T}$ and unstable (i.e., practically no peers P_{same} are contacted over two consecutive windows) as early noticed for TVAnts. However, differently from the TVAnts case, the number of new peers contacted is significantly larger than the average P_{new} (actually, in this case we have clipped the P_{new} value which would have otherwise slightly fallen outside the maximum range). In in this case, it seems that the anomalous peer was mainly performing network discovery, without however being able to find the needed content.

7 Case Study: Network Awareness

In this section, we apply the Sherlock visualization tool to the analysis of a more specific aspect of P2P applications: namely, their network awareness and friendliness level – in other words, whether their peer selection and content diffusion algorithms are aware of peer location in the underlying network. To this purpose, we define a new set of cross-layer *features* able to express network awareness, as well as new *metrics* to compactly represent them.

Considering the P2P-TV applications branch, previous work focused on the definition of black-box methodologies to assess network awareness of P2P-TV endpoints as [31, 32]. Authors in [31] focus on path-wise properties by means of an *active testbed* where they enforce artificial bandwidth limitations, packet loss and delay, and examine P2P-TV reaction to adverse network conditions. In our previous work [32], we instead investigated peer-wise features in by adopting a purely *passive approach*: by inferring from measurement the main properties of content exchange, we assessed which parameters mostly influence the download/upload preference of P2P-TV application.

In this work, we instead follow an *hybrid* methodology, that merges both [31,32] approaches in a software tool, named P2PGauge, that exploits passive and active techniques to gather full-relief results. The software is available as open-source at [37] and implements a traffic analyzer based on the Sherlock framework. In more details, narrowing the scope of our investigation to cross-layer properties, we can further define two different categories. On the one hand, there are *path-wise features* (such as RTT delay, IP hop count, bottleneck bandwidth, etc.) which are determined by the conditions on the path between two peers in the overlay. On the other hand, there are *peer-wise features* (such as Autonomous Systems, geographical location, /16 IP prefix, etc.) that only depend on properties of a single peer. As we will describe in the following, the software exploits a mix of active and passive methodology to gather path-wise and peer-wise information respectively.

It is necessary to point out that the P2PGauge tool exploits active probing of peers contacted by unmodified P2P clients: whilst the tool is able to process offline traces (e.g., the "P" and "A" databases in Tab. 1), active probing should be better performed *simultaneously* to the running P2P application, as otherwise contacted peers may go offline (and thus be no longer available for later probing, compromising the accuracy of the dataset). Therefore, using P2PGauge as monitoring and analyzer tool, we performed a new set of 1-hour long experiments running the SopCast application (i.e., the "T" dataset of Tab. 1). In these experiments, a single probe peers in France is used to join different channels at different hours, exploring thus a wider spectrum of content locality and channel popularity. Moreover, care has been taken in order to consider *local* content (e.g., European Champions League football matches, or matches of the French Ligue-1) as well as *foreign* content (e.g., news and movies in foreign languages). Beside the availability of a larger number of metrics, there is another important difference between the "A" dataset considered in the previous section and the "T" dataset of the current one. Indeed, in this case each experiment is carried out in isolation, while in previous experiments all peers watched the same channel at the same time. Thus, the previous dataset was possibly *biased* by the pres-

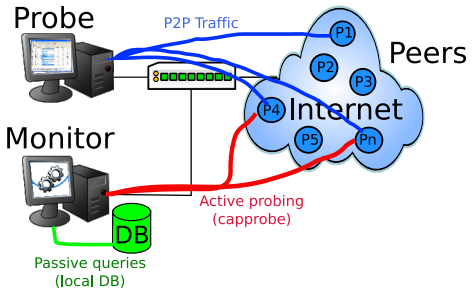

Fig. 6 P2PGauge analysis process

ence of several high-bandwidth peers, located in Europe, that were moreover sometimes co-located within the LAN of a single institution. As a consequence, unless specific care is taken, there is the possibility that a self-induced artifact increases the observed geolocalization [32]. The fact that each observation is carried out independently, guarantees instead that such bias does not affect the new dataset, on which we focus on the following.

In the remainder of this section, we briefly explain the analysis process and the new set of features and metrics, which we then apply to the study of SopCast network awareness. Although P2PGauge takes into account both *timescales* (e.g., short term snapshot vs long-term averages) and *traffic directionality* issues (i.e., meaning that it is possible to either separately analyze the download/upload application behavior), in the following we limitedly consider the long-term, unidirectional, aggregated traffic volume for the sake of simplicity.

7.1 Analysis Process

We describe the analysis process with the help of Fig. 6. In our experiments, an unmodified SopCast client runs on the probe machine, whose traffic is sniffed by the P2PGauge tool running on the monitor machine. P2PGauge analyzes the traffic generated by SopCast and collects statistics about (i) peer-wise features by passive analysis and (ii) path-wise features by sending active probes toward peers contacted by the monitored SopCast client.

Prior to delve into the features and metrics selection, let us stress an important implication of this choice. As far as passive methodology is concerned, P2PGauge gathers peer-wise features by means of a local database [47] (e.g., geolocalization and AS number, etc.) or through simple inference and analysis (e.g., IP prefix length, throughput, hop-count, etc.). Passive analysis cannot interfere with the observed P2P application traffic, but may be rather limited by database access speed: since the database API supports more than 40,000 queries per second, this clearly does not constitute a bottleneck.

However, the tool also performs *active measurements* to gather path-wise properties, thus possibly interfering with the observed P2P traffic: as such, active path-wise measurement should be limited as much as possible. Notice indeed that, although measurements are performed by a dedicated machine, monitor and probe machines share the same access link. Consider for instance the issue of path capacity estimation: expensive active-path probing techniques (such as bandwidth measurement by means of packet trains) are not suitable for our purposes, and we rather need light-weight measurement technique (such as those based on packet-pair dispersion). In reason of this observation, we resort to CapProbe [46] to actively estimate the bottleneck capacity, the RTT delay and the IP time-to-live (from which we can infer the IP hop path distance). For each peer, we perform $N = 100$ measurements by sending pairs of back-to-back ICMP packets, and each pair is spaced by $\Delta T = 0.5$ seconds.

To limit the number of probes during intense network discovery phase, we further upper-bound the number of concurrently active path-probing processes at $C = 50$. Although the amount of active-probing traffic is limited to $R = 2C/\Delta T = 200$ packets per second, performing active experiments for the whole peer population may be a prohibitive task. Furthermore, concurrent experiments may have mutual influence, thus we would like to reduce their occurrence. To this extent, we recall that a large number of peers is only contacted once (i.e., during the network discovery phase), but is not contacted later on – thus is not involved in the content exchange. While such peers may constitute a significant percentage of the peer population (e.g., in case of PPLive), they are nevertheless irrelevant as far as the traffic volume is concerned. As we are interested in the bulk of the traffic volume, we thus limit active measurements only to peers that actively contribute to the video stream. Specifically, we consider only peers who send at least two packets in a time window ΔT. This simple heuristic still allows to focus on the bulk of the traffic volume (e.g., above 95% for the worst case application, namely PPLive), while significantly limiting the bias induced by active probing traffic. Notice that this heuristic is robust and applies also to other classes of P2P services such as filesharing.

7.2 Features Definition

The choice of the features pertaining network awareness has already been preliminary discussed in Sec. 4. We point out that the discussion was derived from a purely passive viewpoint, while in some cases it may be possible to measure the same feature (e.g., IP TTL, RTT, etc.) with either methodology. Yet, as early noticed, passive measurement can be less reliable than active ones: e.g., in case of RTT, the difficulty lies in matching data packets with the correspond-

ing application-layer acknowledgement. We therefore follow a conservative approach, and adopt on the most accurate methodology for each feature. More precisely, we exploit *passive analysis* to infer AS, CC and NET properties, while we use *active probing* to measure the CAP, RTT and HOP features, which are described as follows:

Autonomous System (AS) and *Country Code (CC)*: For these peer-wise properties, we rely on same public database [47], used early to gather cross-layer metrics, which enables us to map public IP addresses to Country Codes (CC) or Autonomous System (AS) numbers.

Network prefix (NET): Namely, the length of the bitwise prefix match between the monitored peers IP address and the IP addresses of the peers it contacts. This feature gives a raw estimation of peers distance in the IP address space: when two peers are in the same subnetwork, they likely share a longer prefix than faraway peers.

Path capacity (CAP): We measure the bottleneck capacity along the path between two peers with CapProbe [46], a packet-pair technique that infers capacity based on the dispersion of the acknowledgement packets on the backward path. Bottleneck capacity is measured over $N = 100$ packet-pairs measurements.

Round Trip time (RTT): RTT measurements are directly available as a side effect of Capacity probing. Indeed, CapProbe sends $N = 100$ packet-pairs, from which we gather the same amount of RTT samples.

IP hopcount (HOP): The IP hop-count distance corresponds to the number of layer-3 nodes traversed by an IP packet. Usually, we infer as in [32] this value from the TTL field in the IP header of the CapProbe packets. However, we found that, in some cases, this value is mangled by non-standard networking devices: in case P2PGauge notices such anomalous behaviour, it falls back on the more reliable (but longer and more costly) path discovery operation by means of the common Traceroute tool.

7.3 Metric Definition

P2PGauge acquires a great number of informations: namely, the amount of sent and received traffic, along with the path-wise and peer-wise features early described is stored for each remote peer contacted. This raw information has thus to be processed in order to be displayed on a Kiviat chart. At the same time, a careful selection of display metrics should be made, in order not to loose too much information in the data processing. In this section, we present two out of the four metrics implemented in the P2PGauge software.

Preferential Partition (PP)
As the simplest and most intuitive metrics, we ressort to the preferential partition (PP) metric defined in [32]. For each feature F, the set $\mathcal{N}_k$ of peers contacted until time $T = k\Delta T$, defined early in (5), is split in two disjoint groups $\mathcal{N}_k = \mathcal{N}_k^{close(F)} \cup \mathcal{N}_k^{far(F)}$, so that peers that are "close" to the monitored peer X in terms of the feature F are grouped altogether.

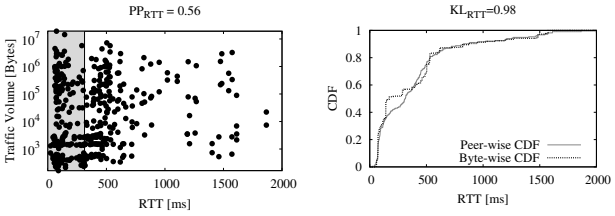

Fig. 7 Network-awareness metrics: (a) Preferential partitioning PP_{RTT} and (b) Kullback-Leibner divergence KL_{RTT} of the RTT feature.

Specifically, we use the following rules to partition the sets. We consider peers falling in the same AS and CC of the monitored peer to be part of the close peer set. As far as the NET feature is concerned, we use a fixed threshold of 16 bits, above which we consider peers to be close. Finally, for the RTT, HOP and CAP features we use a relative threshold, equal to the median value computed over all peers: namely, peer whose RTT and HOP values are below the median threshold are considered to be close, while peers having a bottleneck capacity CAP higher than the threshold are included in the preferential set.

Based on this simple partitions, we now quantify the preference level by evaluating the percentage of bytes that the monitored peer X has exchanged with peers belonging to the preferential set $\mathcal{N}_k^{close(F)}$, as:

$$PP_F = \frac{\sum_{Y \in \mathcal{N}_k^{close(F)}} B(X,Y)}{\sum_{Y \in \mathcal{N}_k} B(X,Y)} \tag{9}$$

Notice that, given this definition, the PP_{CC} metric is perfectly equivalent to the CC_B metric early defined in Sec. 4.5. Considering the RTT feature, Fig. 7-(a) exemplify the preferential partition as a gray shaded zone: in the scatter plot, each (x, y) point corresponds to the amount y of bytes exchanged with a peer having a given RTT equal to x. In the case of figure, about 56% of the data is exchanged with the 50% of peers that constitutes the preferential set (notice that, since we used the median RTT as threshold, the population size is equal for both sets), hinting thus toward a slight preference for peers that are close in IP-latency terms.

Kullback-Leibler (KL)
As a second metric, we consider the Kullback-Leibler (KL) divergence (10), which is a known measure of the distance between two probability distribution functions (pdf) p and b:

$$KL(p\|b) = \sum_{x \in X} p(x) \log \frac{p(x)}{b(x)} \tag{10}$$

We use the KL divergence to measure difference between the *peer-wise* and the *byte-wise* pdf of a given feature F. In other words, we evaluate the pdf of F, either counting each peer once, or by taking into account the volume of traffic that remote peers have exchanged with the monitored peer. The KL divergence tells us whether the two distribution matches (KL$\simeq$0), or whether some discrepancies arises instead (KL$>$0). Notice that, as opposite to before, a large KL value cannot be directly read as *preference* indicator: rather, it merely pinpoint the existence of a *bias* between the number of peers exhibiting a given value for a feature F, and the amount of bytes exchanged with those peers. For instance, a large KL$_{AS}$ value does not mean that a large amount of bytes is exchanged with peers falling in the *same* AS, but rather expresses the fact that *some* AS possibly contributes for a significant portion of the traffic, inducing a distortion in the byte-wise pdf with respect to the peer-wise one. In other words, high KL values correspond to high bias, which however do not necessarily translate into higher awareness.

An example of the KL$_{RTT}$ metric is reported in Fig. 7-(b) considering the same dataset depicted in Fig. 7-(a). In this case, dashed and continuous lines are used to represent the byte-wise and peer-wise RTT cumulative distribution functions respectively. In the case of figure, notice that the two curves do not overlap, which is especially visible for RTT$\in [200, 300]$ ms, and that yield to a value of KL$_{RTT} = 0.98$. This means that there is a group of peers, whose RTT is about [200,300] ms, that contribute more data than others: notice indeed that such a couple of highly-contributing peers is clearly visible in Fig. 7-(a) in the same RTT range.

7.4 Experimental Results

We now adopt a Kiviat representation of the cross-layer feature set, expressed using the PP and KL metrics, for the SopCast application. Fig. 8 reports the Kiviat charts, arranged in such a way that features gathered by passive inference (i.e., AS, CC and NET) are represented on the three top axis, whereas features involving active probing (i.e., CAP, RTT and HOP) are represented on the three bottom axis. Preferential partition metric PP and Kullback-Leibner divergence KL are reported on the left Fig. 8-(a) and right (b) plots respectively. Notice also that axis extend until a maximum value of 1.0 (2.0) for the PP (KL) metrics.

Kiviat reports, as usual, the mean and standard deviation over all the peers in the novel SopCast dataset. Let us consider the preferential partition metric first, which is depicted in Fig. 8-(a). It is easy to notice that, despite experiments include content that is very popular in EU (e.g., Champions League matches) and possibly also very local (e.g., French Ligue-1 matches), nevertheless SopCast managed to find a few peers that were located in the same network ($PP_{NET} \simeq 0\%$), AS or CC ($PP_{AS} \simeq 1.6\%$ and $PP_{CC} \simeq 4.5\%$) boundaries.

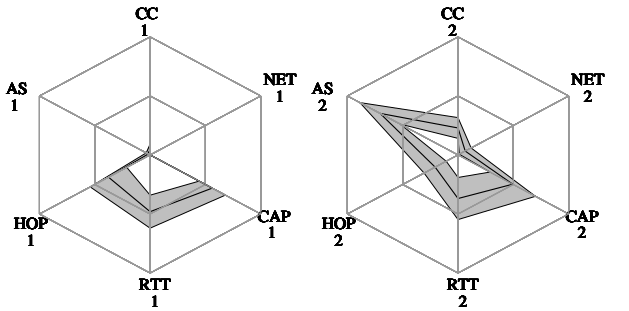

Fig. 8 Network-awareness representation: Kiviat charts of (a) Preferential partitioning and (b) Kullback-Leibner divergence on the new SopCast dataset. Features gathered with passive measurement are displayed on top axis (AS, CC, NET), features requiring active measurement on the bottom axis (HOP, RTT, CAP).

As the percentage of bytes exchanged with peers in the same country actually *diminishes* with respect to the one early observed on Fig. 2 ($CC_B = PP_{CC} \simeq 6.5\%$), this suggests that the slightly higher geolocation previously observed in the "P" dataset, could possibly have artificially induced by other preferences: for instance, the simple greedy choice of high-capacity peers, that in the case of the Fig. 2 dataset were also incidentally located in the same AS. And indeed, this is corroborated by the capacity feature ($PP_{CAP} > 50\%$), which shows a slight preference for higher bandwidth peers. On the contrary, no such preference is shown for close peers, as only about half of the overal traffic volume is exchanged with peers close in terms of RTT latency ($PP_{RTT} \simeq 50\%$), hinting toward no locality preference. Similarly, the fact that $PP_{HOP} < 50\%$ confirms that slightly longer IP paths may be taken to find those high-capacity peers.

Let then consider the Kullback Leibner plot of Fig. 8-(b). In this case, we recall that a larger KL value expresses a larger bias, but not necessarily larger awareness. In this case, a large bias is exhibited for the capacity KL_{CAP} metrics, corroborating in this case the hypothesis of a greedy selection policy. An even larger bias is visible for KL_{AS}, which in this case corresponds to an unbalanced traffic distribution. In this case, a few ASes act as main contributors: however, such ASes differ from the monitored peer AS, and their occurrence may rather be the result of other peer-selection policies (e.g., possibly due to the presence of high capacity peers in such ASes). Overall, we can conclude that current popular P2P-TV applications such as SopCast, have not yet considered network-awareness issues.

8 Conclusions

This paper presented Sherlock, a framework for the characterization of P2P applications based on a black-box mea-

surement and analysis of the traffic they generate, coupled to an expressive data representation exploiting Kiviat graphs. We used Sherlock to analyze a number of file-sharing, VoIP, VoD and live-streaming P2P applications that are popular nowadays, further presenting two case studies, namely P2P anomaly detection and P2P network awareness.

As emerges from the results, Sherlock has a number of desirable properties, which makes it a valuable tool for P2P traffic analysis. First of all, it allows a very compact representation of rather heterogeneous features and metrics, which can be furthermore easily customized as we shown. Moreover, the representation is flexible in the space domain, which is suited to express not only individual peers behavior, but also generalizes well to express the aggregated peer behavior (e.g., mean) and its variability (e.g., standard deviation). The representation is also flexible in the time domain, which allows to observe not only the long-term behavior of P2P applications, but the temporal system evolution as well. Finally, Sherlock is generally applicable, in virtue of its black-box approach, which is important in reason of both the varying popularity of Internet applications and the closeness of popular P2P applications.

Acknowledgment

This work has been funded by Celtic TRANS, a project of the Eureka cluster.

References

1. BitTorrent, http://www.bittorrent.com/
2. eMule, http://www.emule-project.net/
3. Skype, http://www.skype.com/
4. Joost, http://www.joost.com/
5. TVAnts, http://www.tvants.com/
6. SOPCast, http://www.sopcast.com/
7. PPLive, http://www.pplive.com/
8. apt-p2p, http://www.camrdale.org/apt-p2p/
9. World of Warcraft, Blizzard Downloader, http://www.worldofwarcraft.com/info/faq/blizzarddownloader.html/
10. K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M.Levy, J. Zahorjan, "Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload." *In ACM Symposium of Operating Systems Principles (SOSP'03)*, Bolton Landing, NY, USA, October 2003.
11. A. Klemm, C. Lindemann, M. K. Vernon, O. P. Waldhorst. "Characterizing the query behavior in peer-to-peer file sharing systems," *In ACM Internet Measurement Conference (IMC'04)*, Italy, October 2004.
12. R.J. Dunn, J. Zahorjan, S.D. Gribble, H.M. Levy, "Presence-based availability and P2P systems," *In IEEE P2P'05*, Konstanz, Germany, August 2005
13. D. Stutzbach, R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," *In ACM Internet Measurement Conference (IMC'06)*, Brazil, October 2006.
14. D. Stutzbach, S. Zhao, R. Rejaie, "Characterizing Files in the Gnutella Network," *ACM/SPIE Multimedia Systems Journal*, Vol. 1, No. 13, pp. 35–50, March 2007.
15. M. Izal, G. Urvoy-Keller, E. W. Biersack, P.A. Felber, A.L. Garcs-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime" *In Passive and Active Measurement (PAM'04)*, Antibes, France, April 2004
16. W. Acosta, S. Chandra "Trace Driven Analysis of the Long Term Evolution of Gnutella Peer-to-Peer Traffic," *In Passive and Active Measurement (PAM'07)* Louvain-la-neuve, Belgium, April 2007
17. M. Steiner, T. En-Najjary, E. W. Biersack, "A Global View of KAD," *In ACM Internet Measurement Conference (IMC'07)* San Diego, CA, USA, October 2007
18. J. Falkner, M. Piatek, J.P. John, A. Krishnamurthy, T. Anderson, "Profiling a Million User DHT," *In ACM Internet Measurement Conference (IMC'07)* San Diego, CA, USA, October 2007
19. L. Plissonneau, J-L. Costeux, Patrick Brown "Analysis of Peer-to-Peer Traffic on ADSL," *In Passive and Active Measurement (PAM'05)* Boston, MA, April 2005
20. S. Guha, N. Daswani, R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," *In IPTPS'06,* Santa Barbara, CA, USA, February 2006
21. D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, D. Rossi "Detailed analysis of Skype traffic," *IEEE Transactions on Multimedia*, Vol.11, No.1, January 2009.
22. X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, Vol.9, No.8, December 2007.
23. Bo Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, X. Zhang "Inside the New Coolstreaming: Principles, Measurements and Performance Implications," *In IEEE INFOCOM'08* Phoenix, AZ, USA, April 2008
24. T. Silverston, O. Fourmaux, "Measuring P2P IPTV Systems," *In ACM NOSSDAV'07*, Urbana-Champaign, IL, USA, June 2007.
25. A. McGregor, M. Hall, P. Lorier, and J. Brunskill "Flow Clustering Using Machine Learning Techniques" *In Passive and Active Measurement (PAM'04)*, Antibes, France, April 2004
26. T. Karagiannis, K. Papagiannaki, M. Faloutsos "BLINC: multilevel traffic classification in the dark," *In ACM SIGCOMM'05*, Philadelphia, Pennsylvania, USA, August 2005
27. T. Karagiannis, K. Papagiannaki, N. Taft, M. Faloutsos "Profiling the End Host," *In Passive and Active Measurement (PAM'07)* Louvain-la-neuve, Belgium, April 2007
28. T.Z.J. Fu, Y. Hu, X. Shi, D.M. Chiu, J.C.S. Lui "PBS: Periodic Behavioral Spectrum of P2P Application," *In Passive and Active Measurement (PAM'09)* Korea, April 2009
29. R. Torres, M. Hajjat, S. Rao, M. Mellia, M. Munafo' "Inferring Undesirable Behavior from P2P Traffic Analysis," *In ACM SIGMETRICS'09*, Seattle, WA, USA, June 2009
30. C. Wu, B. Li, S.Zhao, "Exploring large-scale peer-to-peer live streaming topologies," In *IEEE Transactions on Multimedia Computing, Communications and Applications*, Vol. 4, No. 3, 2008
31. E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, M. Meo, "P2P-TV systems under adverse network conditions: a measurement study," *IEEE Infocom'09,* Rio de Janeiro, Brazil, Apr. 2009
32. D. Ciullo, M.A.Garcia, A. Horvat, E. Leonardi, M. Mellia, D. Rossi, M. Telek and P. Veglia. " Network Awareness of P2P Live Streaming Applications: a Measurement Study," *IEEE Transactions on Multimedia,* to appear.
33. D. Rossi and E. Sottile, "Sherlock: A framework for P2P traffic analysis," In *IEEE P2P'09*, Seattle, WA, USA, September 2009.
34. A. C. Doyle, "A Study in Scarlet", 1888
35. K. Kolence, P. Kiviat, "Software Unit Profiles and Kiviat Figures," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 2, No. 3, September 1973.
36. D. Rossi, E. Sottile, S. Valenti and P. Veglia, "Gauging the network friendliness of P2P applications", *In ACM SIGCOMM, Demo Session,* Barcelona, Spain, August 2009.

37. P2PGauge software, available online at http://www.infres.enst.fr/~drossi/P2PGauge
38. Post in thread "μTorrent 1.9 alpha", http://forum.utorrent.com/viewtopic.php?pid=377209#p377209
39. "IPP2P home page", http://www.ipp2p.org/.
40. Y. Kulbak, D. Bickson, "The eMule protocol specification," *Technical Report Leibniz Center TR-2005-03*, School of Computer Science and Engineering, The Hebrew University, 2005.
41. D.Bonfiglio, M.Mellia, M.Meo, D.Rossi, P.Tofanelli, "Revealing Skype Traffic: when Randomness Plays with You," *ACM SIGCOMM*, Kyoto, JP, August 2007.
42. M.Mellia, R.Lo Cigno, F.Neri, "Measuring IP and TCP behavior on edge nodes with Tstat,"
43. Tstat web page, http://tstat.tlc.polito.it/
44. S. Valenti, D. Rossi, M. Meo, M.Mellia and P. Bermolen, "Accurate and Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets," *In Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537,* May 2009.
45. A. Finamore, M. Mellia, M. Meo and D. Rossi, "KISS: Stochastic Packet Inspection," *In Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537,* May 2009.
46. R. Kapoor, L. J. Chen, Li Lao, M. Gerla, and M. Y. Sanadidi. "CapProbe: A Simple and Accurate Capacity Estimation Technique," *In ACM SIGCOMM'04*, Portland, USA, 2004
47. MaxMind GeoLite, http://www.maxmind.com/