Sherlock: A framework for P2P traffic analyis

Dario Rossi, Elisa Sottile

TELECOM ParisTech
Paris, France
firstname.lastname@enst.fr

***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 the observable of interest: such metrics 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 metrics 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). Based on this framework, we analyze some of the most popular P2P application nowadays, highlighting their main similarities and differences.

***Keywords*-traffic monitoring; traffic characterization; kiviat charts;**

I. Introduction

The Internet today is populated with a rather large number of P2P applications. The offer of services now spans a very wide spectrum: besides the ever-present file-sharing applications, we use P2P application to call our friends with VoIP; for entertainment purposes, we use P2P based VoD and live TV applications; moreover, even operating system and application 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 structured), 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 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 of P2P systems. Previous research [1]–[20] already deeply studied different systems, pointing out several metrics to characterize important aspects of such applications. At the same time, P2P overlays have, with few exceptions [10], [15]–[18], been studied in isolation: therefore, what the scientific community still lacks is a mean to contrast and relatively weight similarities and differences of those systems. Moreover, as P2P systems continuously evolve, 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 this work, which proposes *Sherlock*, a framework to *Sketch Hallmark Elements to Recap and Look-into Overlays with Charts of Kiviat*, that we designed to compactly describe the traffic generated by currently deployed Internet P2P applications. As Sherlock Holmes rightly says [21], "*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 metrics 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 [22], which allow to represent a great amount of possibly very heterogeneous information, in a furthermore very compact and visually intuitive way. The framework is extremely flexible, as it allows to focus at different levels of granularity (e.g., from individual endpoints to endpoint aggregates), to consider different timescales (e.g., long-term average versus instantaneous snapshots) and as its use is not limited to the metric that we describe in this work.

II. Related Work

As a consequence of P2P widespread adoption, the research activities related to P2P traffic measurement, such as characterization and classification, acquired importance [1]–[20].

File-sharing, being the first example of new application exploiting the P2P paradigm has been studied for a relatively long time [1], [1]–[10]: as a result, many details concerning the query process [2], user churn [3], [4] and files popularity [5] are available. In more details, researchers studied proprietary applications such as KaZaa [1], unstructured systems such as BitTorrent [6] and Gnutella [7], and Distributed Hash Tables (DHTs) such as Kademlia [8], [9]. Work comparing different protocols also exists, such as [10], 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 [11], [12] and of P2P-TV applications such as PPlive [13] and Coolstreaming [14]. Again, work comparing different protocols also exists, such as [15] 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 [10], [15]–[18]: [10], [15] focus only on P2P applications, whereas [16]–[18] are more general but still very relevant. Nevertheless, the purpose of the above works, the metrics adopted in the investigation and their presentation are different from our approach. In more details, authors in [10] and [15] compare different applications, but limitedly focus on a single P2P service (i.e., filesharing and IPTV respectively). The aim of [16], [17] is instead traffic classification, while [18] targets end-host profiling. Thus, [16], [17] consider P2P as a single class of application, which we instead decompose further, discriminating among individual applications.

As far as metrics are concerned, [17], [18] consider mainly transport-layer characteristics (e.g., number of hosts contacted, on which ports), while [10], [15], [16] additionally take into account network-layer information (e.g., packet size and interarrival times). Except [10], 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 hosts participating in the overlay). Our framerwork instead takes into account all the above aspects.

Finally, as far as the representation is concerned, simple yet powerful "graphlets" are proposed in [17], [18], that allow to abstract different transport-layer behaviors and compactly present them in a descriptive graph. Authors in [10], [15] instead characterize the different applications by means of cumulative distribution function and scatter plots of different metrics. Kiviat representation has been used in [16], which inspired our work, and whose differences will be highlighted in more details in the following sections.

As a last note, we stress that related effort is ongoing to characterize the periodic behavioral spectrum [19] and undesirable behavior of [20] applications, which can also be partly assessed using the framework proposed in this paper.

III. P2P Applications Dataset

The application we consider in this work are listed in Tab. I: more precisely, we select BitTorrent and eDonkey as examples of P2P file-sharing; Skype as an example of P2P VoIP; Joost[1] 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 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.

In this work, we employ a passive methodology to gather eDonkey and Skype traffic from real networks (indicated

[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.

with “P” in Tab. I), whereas we deploy an active Internet-scale testbed for the other applications (indicated with “A” in Tab. I). We resort to Deep Packet Inspection (DPI) capabilities to gather eDonkey traffic [24], [25], whereas we exploit [26] to classify Skype traffic: both classification engines have been implemented in Tstat [27], an open-source flow-level logger available at [28].

As far as the other P2P application are 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 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. I), 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[2] which limited the number of experiments we were able to perform.

For further details concerning the application dataset we investigate in this work, the classification mechanism and the Internet-scale testbed, we refer the reader to [29], [30]. Finally, we stress that overall size of the considered dataset is still significant, since our 243 probes contacted about 5 millions external peers, exchanging with them about 136 GBytes of data in 344 millions of packets. For reference purpose, the dataset size of closest works to ours amount to 200 hosts in [18], 20 thousand flows in [16], “several thousands”[3] hosts in [10], about 19 GBytes of data in [15] and 3 TBytes in [17].

Table I
SUMMARY OF APPLICATIONS ANALYZED IN THIS STUDY. TRACES IN THE DATASET HAVE BEEN COLLECTED WITH EITHER PASSIVE (P) OR ACTIVE (A) 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
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	-	*243*	*5,054,161*	*344.13*	*136.49*

A. 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. I, 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 $\text{ID}(X, P)$, whereas packets received from X and send by P have a negative identifier $\text{ID}(P, X) = -\text{ID}(X, P)$.

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

[2]Since December 2008 [23], the official BitTorrent client is no longer open-source and implements an application-layer congestion-control protocol over UDP.

[3]Due to NDA, authors in [10] only disclose *relative* amounts.

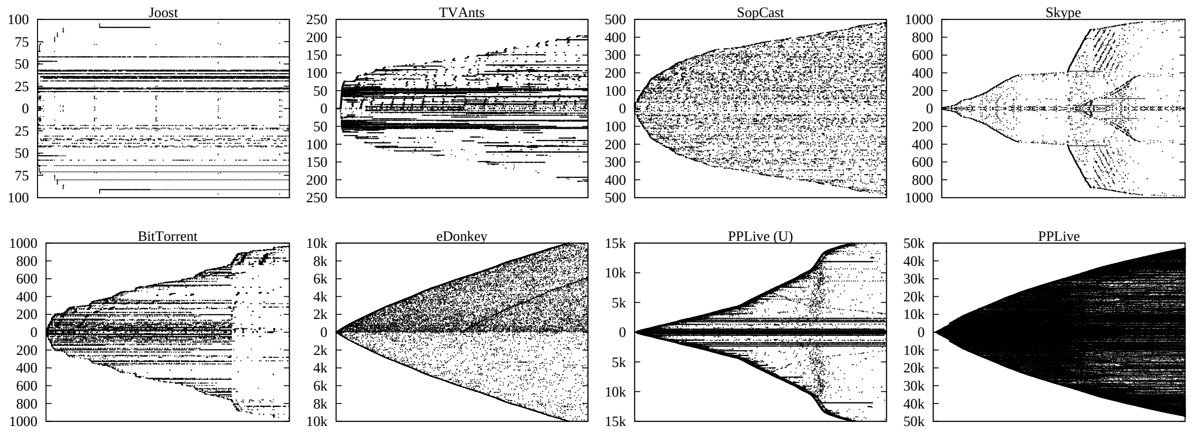


Figure 1. P2P application patterns, conveying network and transport layer information and their temporal evolution

contacts are clearly visible, but their duration is shorter. Conversely, SopCast contacts 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 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. 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 end-points, 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.

IV. Framework Definition

In reason of the above observations, we built the Sherlock framework with design goals such as the ability to:

- express possibly many properties at the same time in a visually compact, intuitive and readable way;
- capture key P2P properties which are intrinsically different in nature (e.g., packet size and inter-arrival time, connectivity degree, geographical peer location, etc.);
- zoom at different levels of granularity, considering peers either individually or aggregated;
- represent long-term averages as well as temporal snapshots of the system behavior.

In the following, we describe Sherlock by decoupling the *choice* of the metrics that we use to characterize the P2P applications from their *representation*, whose detailed description will be addressed later in Sec. V. Sherlock representation is based on Kiviat graphs [22], 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 [16], 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 [16], 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 as in [16]. Our work also differs in the choice of the observables, which are in our case tailored for P2P applications. Finally, out methodology is flexible, as it applies to endpoints and endpoint aggregates, as opposite to flows only as in [16].

A. Metric 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 metrics to be represented. Since we want our framework to produce readable results, this implies that we need to limit the amount of information to display. Moreover, since we want out framework to be applicable to any P2P application without requiring reverse-engineering, we need to individuate metrics that can be measured by a purely black-box approach.

Focusing on P2P traffic, we can define a number of interesting metrics, 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 metrics 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 could profitably be applied with metrics other than the ones considered in this work – which could possibly be useful whenever one wants to focus, e.g., on a specific aspect of P2P applications, or on a specific layer of the protocol stack.

Before introducing the minimum set of metrics we will focus on for the reminder of this paper, we also need to outline an important remark concerning the range of values taken by a metric x (independently thus from the semantic of x). 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[4] 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.

B. Network-layer metrics

Network layer metrics characterize P2P traffic at packet level. We argue that packet size is correlated 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 [13]).

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 metrics.

C. Transport-layer metrics

Transport layer metrics concern flows rather than individual packets: we consider two different metrics 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 $F_{port} = (\sum_i n_i)^2/(N \sum_i n_i^2)$, 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

[4]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 $\hat{x}_i = x_i / \max_i x_i$).

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 symmetry index as $\text{Sym}_P = P(X,Y)/[P(X,Y)+P(Y,X)]$ and the byte-wise one as $\text{Sym}_B = B(X,Y)/[B(X,Y)+B(Y,X)]$. 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 metrics 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 [10], [13] we prefer to leave these metrics out of the framework for the time being.

D. Application-layer metrics

Application layer metrics concern the overlay graph: as topology inference requires active crawling of the P2P system, we resort to simpler metrics 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 $\mathcal{P}_k = \{p : P(X,p) + P(p,X) > 0\}$ 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 $\mathcal{N}_k = \cup_{i=1}^{k} \mathcal{P}_i$ the set of all peers discovered from time 0 until time $k\Delta T$. Notice that, for the sake of simplicity, we define in this case a-directional metrics, and do not further differentiate between type of peers (e.g., discrimination between signaling versus data contributor peers is usually done by requiring a minimum amount of bytes and packets exchanged [13], [15]).

We thus define metrics to count the instantaneous degree $P_{\Delta T} = card(\mathcal{P}_k)$ of the endpoint, the number of peers discovered in the last time-window $P_{new} = card(\mathcal{P}_k \setminus \mathcal{N}_{k-1})$ and the number of stable peers $P_{same} = card(\mathcal{P}_k \cap \mathcal{P}_{k-1})$ that were contacted in the previous time-window and that are still contacted in the current one. Clearly, these indexes will reflect different kind of activities (as 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.

E. Cross-layer metrics

Finally, cross-layer metric 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).

We stress 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 metric, that is however unable to capture proximity methods implemented by means of RTT measurement at the application layer. We argue that CC metric 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 geolocalize peers IP addresses, 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 metrics 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).

V. Experimental Analysis

A. P2P Applications at a Glance: Kiviat Representation

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 metric, and in Fig. 2 we consider for the time being only transport-layer

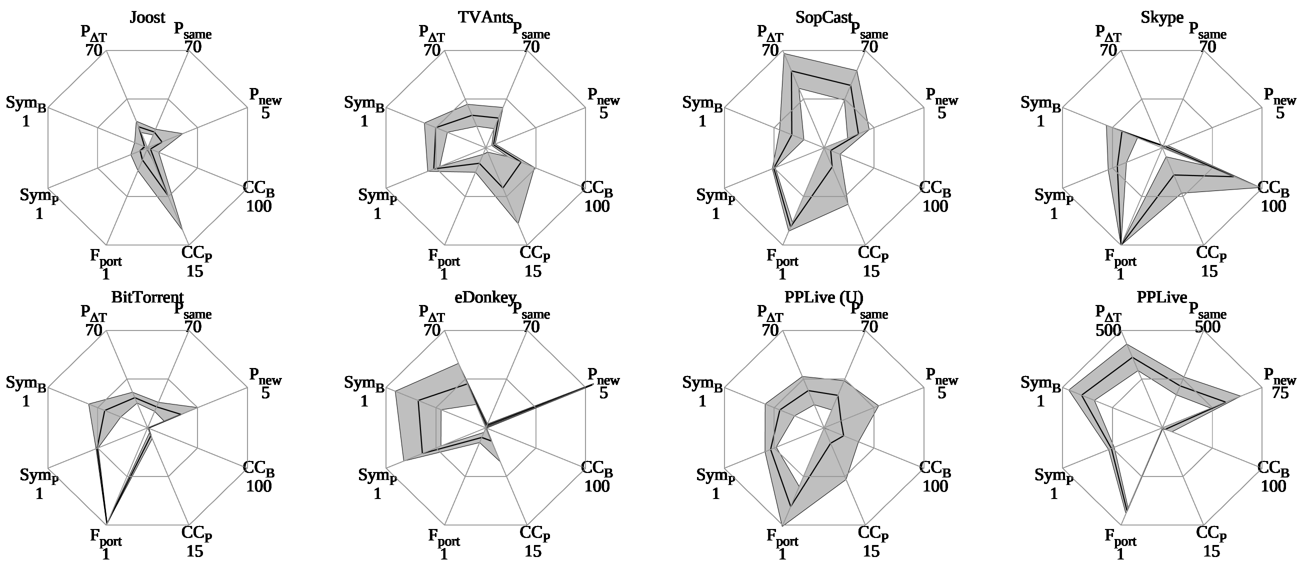


Figure 2. Kiviat representation of transport, application and cross-layer information: each axis reports a specific metric (notice that ranges differs for PPLive case). Thick line joins the average over all dataset probes, thinner lines and gray shading are used to represent the standard deviation relatively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

(F_{port},Sym$_B$,Sym$_P$), application-layer ($P_{\Delta T}$,P_{same},P_{new}) and cross-layer (CC_B, CC_P) information. Focusing on a single application, for each metric we report the mean value μ over all peers in our dataset for that application: by joining the mean values of different metrics with a black thick line, we obtain a closed shape.

To show the variability of applications behavior among different peers, we use thin lines to represent the standard deviation σ of the metrics, 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 metric, we report the maximum range value under the metric 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.

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 (Sym$_P \simeq 0.5$), which suggests a per-packet acknowledgement policy, with the exception of Joost (Sym$_P < 1/10$) and eDonkey (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 PPLive happen to be fairly symmetrical (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 metrics are concerned, we observe rather different behaviors, starting from the number of peers contacted during a $\Delta T = 5\,$s window. While Joost and Skype contact very few peers during the same time window ($P_{\Delta T}$), 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 ($P_{same}/P_{\Delta T} \rightarrow 0$). 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 metrics 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

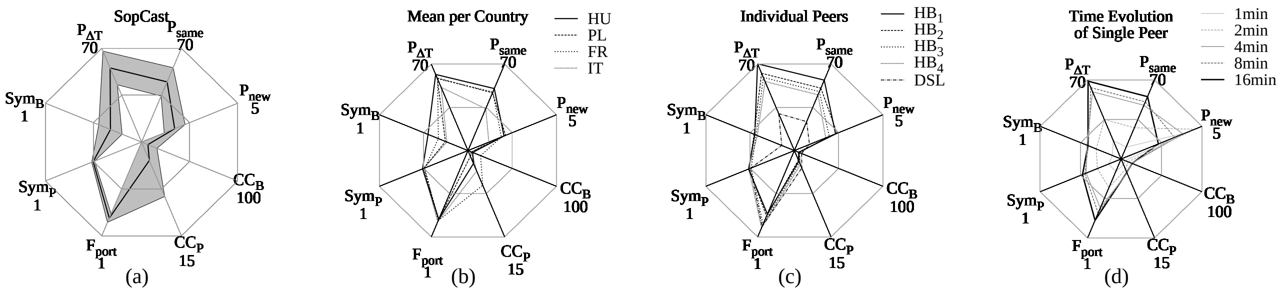


Figure 3. Kiviat representation at different granularities for SopCast application: (a) mean and standard deviation over all peers in the dataset, (b) mean over all peers belonging to the same Country, (c) individual peers in a single Country, and (d) temporal evolution of a single peer

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 to 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 endpoint behavior can be significantly different 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, the 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: our probes stand out (i.e., CC_P increases) as a consequence of the reduction of the overlay size with respect to the popular channel case; moreover, since the channel is not popular, the content can be found only at a fewer number of peers, which again raise the impact of proximity of the content diffusion measured by CC_B.

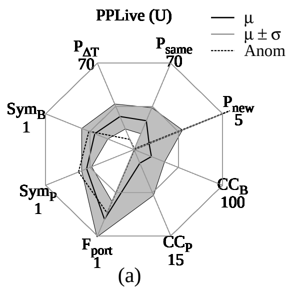
B. Granularity

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 of aggregation, we have a single aggregate, constituted of 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 metrics (e.g., such as packet-wise symmetry and faireness of port usage) have rather similar values irrespectively of the network where probes are located in, some other metrics (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 metrics can be directly tied to design decisions (e.g., per-packet acknowledgement policy, hard-coded port, hard-coded number of neighbor peers, etc.) that are not influenced by network conditions, as opposite as other metrics (which are instead influenced by network conditions, access technology, ressource popularity, etc.). As such, the number of probes that need to be observed, in order to gather results that are representive of the full range of possible application behaviors, may change depending on the metrics under observation. This is a very important point, which is still open and that we leave as an interesting future work.

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

To further prove Sherlock flexibility, we depict in Fig. 3-(d) the temporal evolution of the solid black line peer of Fig. 3-(c). Fig. 3-(d) depicts several Kiviat graphs, 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 for recent intervals, fading lighter toward the past: a clear transient can be seen for all metrics when T<2, which then stabilizes for T≥4. During the transient phase, the number of new peers



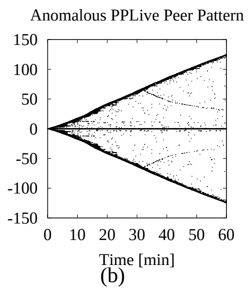


Figure 4. Anomalous peer: mixed Kiviat representation of the whole PPLive unpopular channel dataset and of the anomalous peer (a) and packet level pattern of the anomalous peer (b)

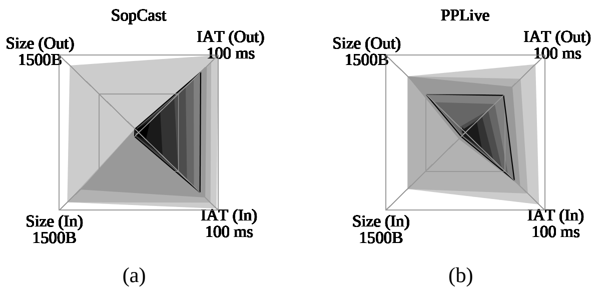


Figure 5. Kiviat representation of network-layer statistics: percentiles of the packet size and IAT distributions, for outgoing and incoming traffic directions, for the SopCast (a) and PPLive (b) application

contacted every $\Delta T\!=\!5$ seconds is larger than in steady state, hinting to more aggressive network discovery at startup.

Different levels of granularity can be mixed in a single plot: for instance, Fig. 4-(a) report both the high-level aggregated view of the PPLive unpopular-channel experiment (mean and standard deviation), as well as a specific peer x (mean only, thick dotted line). Mixed representation allows to contrast the behavior of individual peers to a reference benchmark, which can be useful, e.g., to spot misbehaving peers from an aggregate. For example, from Fig. 4-(a) is easy to gather that the Kiviat of the individual peer x largely differ from the Kiviat of the aggregate (which includes the individual peer as well): this hints to a suspicious behavior of x, possibly due to mal-functioning. Indeed, notice that the number of contacts is extremely low, that such contacts are not stable (i.e., practically no peers are contacted over two consecutive windows) and that the number of new peers contacted is larger than the average. In other words, it seems as peer x was mainly performing network discovery, without being able to find the needed content. For completeness, we report in Fig. 4-(b) the activity pattern of the anomalous peer, which has to be contrasted to the one reported earlier in Fig. 1, and that confirms the intuition gathered from the Kiviat graph.

C. *Metrics*

Kiviat charts not only cope with scalar values, but also allow to represent vectorial metrics: for instance, Fig. 5 depicts the distributions of the packet size and Inter-Arrival Time (IAT) network-layer statistics for SopCast and PPLive. More precisely, Fig. 5 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 50-th. 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. 5, 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 constituted by small application-layer acknowledgements, gathered in reply to big outgoing video packets distributed to a significant number of peers.

VI. Conclusions

This paper presented Sherlock, a framework for the characterization of P2P applications based on a black-box measurement 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. 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 metrics, which can be easily extended to metrics other than the ones considered in this work. Moreover, the representation is flexible, as it allows different levels of spatial aggregation and to observe temporal evolution of P2P systems 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] 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.

[2] 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.

[3] 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

[4] D. Stutzbach, R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," *In ACM Internet Measurement Conference (IMC'06)*, Brazil, October 2006.

[5] 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.

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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

[12] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, D. Rossi "Tracking Down Skype Traffic," *In IEEE INFOCOM'08* Phoenix, AZ, USA, April 2008

[13] 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.

[14] 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

[15] T. Silverston, O. Fourmaux, "Measuring P2P IPTV Systems," *In ACM NOSSDAV'07*, Urbana-Champaign, IL, USA, June 2007.

[16] 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

[17] T. Karagiannis, K. Papagiannaki, M. Faloutsos "BLINC: multilevel traffic classification in the dark," *In ACM SIGCOMM'05*, Philadelphia, Pennsylvania, USA, August 2005

[18] 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

[19] 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

[20] 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

[21] A. C. Doyle, "A Study in Scarlet", 1888

[22] K. Kolence, P. Kiviat, "Software Unit Profiles and Kiviat Figures," *ACM Performance Evaluation Review*, Vol. 2, No. 3, 1973

[23] Post in thread "μTorrent 1.9 alpha", http://forum.utorrent.com/viewtopic.php?pid=377209#p377209

[24] "IPP2P home page", http://www.ipp2p.org/.

[25] 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.

[26] D.Bonfiglio, M.Mellia, M.Meo, D.Rossi, P.Tofanelli, "Revealing Skype Traffic: when Randomness Plays with You," *ACM SIGCOMM*, Kyoto, JP, August 2007.

[27] M.Mellia, R.Lo Cigno, F.Neri, "Measuring IP and TCP behavior on edge nodes with Tstat,"

[28] Tstat web page, http://tstat.tlc.polito.it/

[29] 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.

[30] 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.