INTERNET TRAFFIC AND TOPOLOGY CHARACTERISTICS FROM A NATIONAL ISP PERSPECTIVE

by

Artan Salihu

B.S. in Electrical Engineering – Telecommunications, University of Prishtina, 2011

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Master of Science in Telecommunications

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This thesis was presented

by

Artan Salihu

It was defended on

April 28, 2016

and approved by

Kostas Pelechrinis, PhD, Associate Professor

Martin Weiss, PhD, Professor & Associate Dean

Prashant Krishnamurthy, PhD, Associate Professor

Thesis Advisor: Kostas Pelechrinis, PhD, Associate Professor

INTERNET TRAFFIC AND TOPOLOGY CHARACTERISTICS FROM A NATIONAL ISP PERSPECTIVE

Artan Salihu, MST

University of Pittsburgh, 2016

Measurement is the first step in predicting the growth of Internet. They can reveal information about traffic and topology characteristics of the Internet. Understanding traffic and topology characteristics are vital for evaluating the performance of networking protocols, creating accurate models for simulation and helping service providers to better utilize their resources. Using the data collected from a National Internet Service Provider in Kosovo, PTK, we report on traffic measurements and analyze some of the most important characteristics of Internet traffic such as self-similarity and long-range dependence. Also, we reveal information about the topology structure of Internet at IP level, from the perspective of our data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to express gratitude to my thesis advisor, Dr. Kostas Pelechrinis, and to my program advisor Dr. Martin Weiss, and to Dr. Prashant Krishnamurthy for their help and offered invaluable assistance, support and guidance throughout this work and during my studies.

I would like to thank Telecom of Kosovo and people who agreed, encouraged and helped me to collect data. Without them, this work would have not been possible. My parents and Erza have been a constant source of love and support, without whose encouragement; I would have not pursued my masters degree.

1.0 INTRODUCTION

Since its origin in *ARPANET*, Internet has continued to grow at fast speeds, become more complex and harder to be analyzed and understood. Analyzing and understanding its behavior is vital to developing more efficient protocols and better utilize resources. This requires of having accurate models which reflect the actual behavior of Internet. And, as we cannot replicate and study Internet as a whole, instead we rely on thorough analysis of sample network measurements taken from different parts of Internet. In general, measurements and analysis of Internet are done at two different levels: traffic level and topology level.

Over the last two decades, studies about Internet traffic have been gradually concentrated into two school of thoughts. One school of thought refers to Internet traffic as smooth and such to be easy modelled and understood using traditional *Teletraffic* models introduced by Erlang (e.g. Poisson or Markovian) [1]. Others, view Internet traffic as bursty in many or all timescales and with long-memory [2] [3]. A more realistic approach is those who have shown that both can coexist, depending the scale of the observation [4]. Intuitively, we can think of these differences in terms of how they understand traffic characteristics [5] and, for example, how this affects traffic engineering for ensuring quality of service. If traffic was bursty, then in order to manage the inevitable peaks that exceed the planned capacity, very sophisticated buffers and packet scheduling would be required. On the other hand, if aggregated traffic is smooth, guaranteeing QoS would be only a function of long-term capacity planning because there would be no queue buildups. All this debate started with the seminal work of Leland et al [2] who showed that traffic is *bursty* in different timescales and it cannot be described using few parameters by Poisson processes. Internet and typical voice calls used in circuit switching have dramatically different statistical characteristics from each other. Internet sessions tend to be much more variable and longer in duration than voice calls. Therefore, they introduced the *self-similar processes* as a notion to better understand and model data traffic.

Likewise, understating and modelling Internet topology was *enigmatic* for a very long time and it was believed that can be described and modelled using random graphs [6], [7]. Internet topology, as any other type of complex network, consist of nodes connected to each other and placement of these nodes was considered to be random and such to be well described by Poisson [8]. Despite the random placement of the links, it was believed that most of the nodes will have approximately the same number of links and it would be extremely rare to find nodes that have considerably more or fewer links deviated from the mean. As the probability that a node is connected to k other nodes decreases exponentially for large k random nodes are also

referred to as exponential. The discovery of Faloutsos brother in 1999 [4] that Internet topology obeys power-law distribution, invalidated all previous efforts in modelling Internet topology. Findings of authors in [4] had significant implications in designing efficient protocols and hard to be ignored. Later, authors in [8], referred to these types of networks as *scale-free* and explain the reasons behind by mechanisms of growth and preferential attachment. As new nodes appear, they tend to connect to more connected nodes and thus these *popular* nodes acquire more links over the time than their less connected neighbors. Consequently, we have some very highly connected nodes and many other low connected.

1.1 MOTIVATION

Long-range dependence of Internet traffic has been mostly attributed to file transfer sizes. Authors in [9] showed that causes of self-similarity are associated with heavy-tailed distribution of file transfers. Their conclusion is based on empirical data of WWW traffic collected at local area network Web server (NCSA Mosaic). Likewise, authors in [10] confirm the heavy-tailed distributions of file sizes directly affect the degree of self-similarity (and long-range dependence - LRD) and this phenomenon is likely to continue to happen even under network constrains (such as bottlenecks). But, Internet traffic over the past five years has drastically changed in terms of applications. As reported by Cisco [11], it is Internet video which is the dominant application and drives the growth instead of file-sharing. In addition, as the structure of Internet traffic is affected by demographic properties (user “think-time”) and as authors in [4] suggest that

different backbones might show different traffic structure, motivate us to revisit notions of self-similarity, long-range dependence and power-laws of Internet traffic.

The increased attention to privacy and data protection in one hand and the openness and flexibility of Internet on the other, gives the privileges to individual network administrators to use different policies and security protection mechanisms. This raises a major problem when using *active measurements* (usually used to map Internet at router-level) in order to obtain information about the topology of Internet. Measurements based on *traceroute* are biased due to number of firewalls, missing links and routing policies by individual administrators. Studies in [12] and [13] have shown that current techniques used to conduct measurements at AS level miss around 30% of links. In addition, a study in [14] shows that even if underlying node degree distribution was exponential, traceroute measurements resulted in power-law distribution. This motivates us to use the collected traffic traces, as slightly different approach, in order to study and reveal information about the Internet topology characteristics and its underlying strucuture.

1.2 APPROACH

Most of the recent and past studies of network traffic and topology patterns and characteristics rely on measurements reported by Center of Applied Internet Data Analysis (CAIDA) [20]. CAIDA datasets are a great resource but they contain anonymized passive traffic traces from monitors on specific Internet backbone-links belonging to more than one service provider which generate very high traffic volumes. In contrast, our goal is to have a broad picture of network traffic characteristics from single Internet service provider that shares all main properties, yet is smaller in scale and represents a different demographic part of the world.

In this thesis work, traffic traces collected from a Service Provider, PTK, are analyzed. Traffic flows are captured using *nfcapd* (a Netflow capture daemon) where the machine collecting the data listens in a 1 Gbps link that connects two Internet Border Gateway routers that are responsible for routing the whole traffic at inter domain level as well as intra domain, as shown in Figure 1. Intuitively, this means that flow collector located at PTK captures all conversations between machines at PTK network and outside it. We have been collecting data

for a period of approximately three months, 28 December 2015 to 31 March 2016. Traces used in this study are taken between December 29th 2015 00:00:00 and January 4th 2016 23:59:00. They include 31,602,250 flows and generate $14 \cdot 10^{14}$ packets.

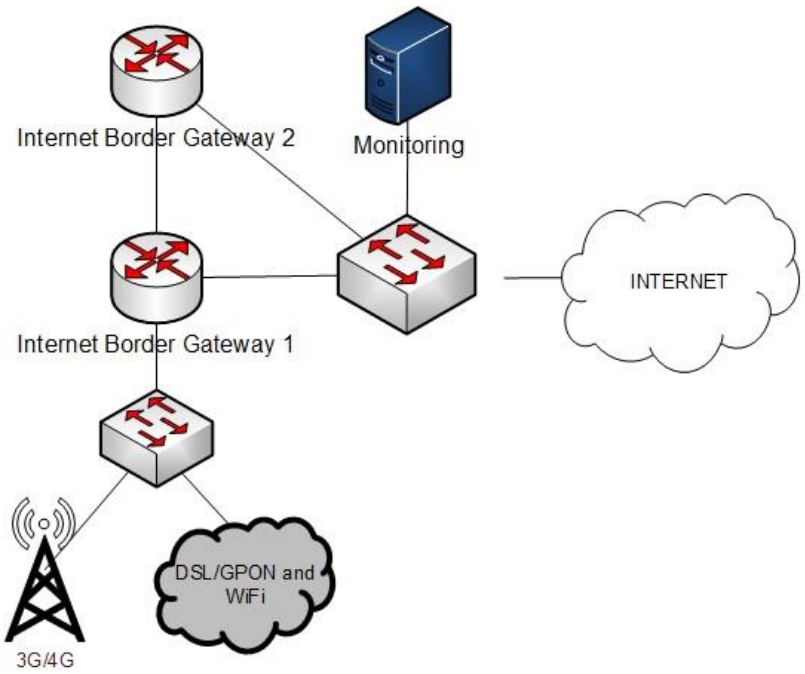

Figure 1. The network at PTK

1.3 GOAL

The goal of this thesis work is to report on traffic measurements and characteristics of Internet traffic over two time scales, 24 hours and 7 days, in terms of traffic volume and packet sizes and time-series analysis. Finally, it aims to provide analysis of the topology structure, using graph metrics, based on the information about the traffic generated by any two communicating hosts. We focus on answering two main questions:

- *Is network data traffic still self-similar?*

- *How does Internet topology look like from the perspective of Internet traffic?*

1.4 PRIVACY

The data collected from PTK contains sensitive information about the network. Such information is not allowed to be disclosed by any party. Because of that, all the results in this thesis work will not show any actual IP or host address.

1.5 STRUCTURE OF THE DOCUMENT

The work is organized as follows. Section 2 provides definitions, metrics and a description of methods used to estimate and quantify variables of our interest for Internet traffic and topology, together with relevant background information. In Section 3, the collected dataset is described and general characteristics of these traces are shown. The section begins with a more detailed explanation of network under study, PTK, and the data collection phases and then shows, for instance, the packet size distribution, the composition of IP traffic from a data snapshot and most popular protocols. Later, Section 4 shows experiments together with the results. Byte counts and packet counts are examined with respect to self-similarity and long-range dependence in section 4.1. Then, in section 4.2, communicating hosts are extracted and information about underlying degree distribution are shown together with other topology characteristics such as network diameter and assortativity. Finally, Section 5 summarizes and concludes the results and shows the future work.

2.0 BACKGROUND

This Section starts with a literature review and related work which lead to definitions and metrics that are used to quantify underlying structure of Internet traffic and topology. In Section 2.1 articles related to Internet traffic characteristics reviewed are given in a chronological order and then similarly in Section 2.2 reviewed work on the structure of Internet topology is shown. In Section 2.3 the mathematics used in these articles are presented, leading up to definitions and methods to quantify self-similarity and long-memory properties of Internet traffic followed by the metrics and definitions to compute node degree, assortativity coefficient and network diameter as chosen metrics for Internet topology.

2.1 INTERNET TRAFFIC

First step in understanding network traffic behavior and the main characteristics is the collection of traffic data. It was Leland et al [2] who first collected an extensive amount of traffic traces from Ethernet LAN's (also known as Bellcore Data) over a three-year period (1989 to 1992) and showed some widespread patterns in data traffic, which until then were not known. More specifically, they showed that data traffic is characterized by "burstiness" on many time-scales. Findings of authors in [2] were considered striking for research community as that invalidated formal models that were used for telephone traffic analysis, such as Poisson models. Poisson based models were considered the most widely used *Teletraffic* models in the context of circuit switching traffic introduced by Erlang. Telephone systems were understood as homogenous systems where the notion of "generic" behavior, "typical" user and average statistics were sufficient in adequately describing and model of Teletraffic [15]. In addition, a traffic that is characterized by Poisson would be an ideal case to model and control data networks. It would

reduce complexity of design for routers and quality of service mechanisms. But, Poisson model has a very basic limitation; it cannot capture traffic "burstiness". For example, if we consider a Poisson process with arrival rate $\lambda = 20$ then timescale for "burstiness" is $\frac{1}{\lambda} = 50ms$.

Probability that a traffic burst occurs over other timescales decreases exponentially [16]. This was true for circuit-switching traffic but traffic bursts in data networks is much more variable and happens at different time scales. This large variability in time (also in space) causes for the traffic to exhibit repeated statistical properties in different timescales, commonly referred to *fractal* or *self-similar* behavior. Up until then, *fractals* were mostly used to describe geometry of irregular shape objects by mathematicians (e.g. the famous Koch snow-flake) and brought to attention of statisticians by Mandelbrot [17] as a notion of *self-similarity* in time-series.

After the evidence of self-similarity in Ethernet traffic, it was Paxson and Floyd [3], who evaluated 24 traces collected from a Wide Area Network (between 1993 and 1995, mostly at the Lawrence Berkeley National Laboratory and at Digital's Western Research Lab) and observed that Poisson models underestimate the burstiness of data traffic, especially at time scales below hundreds of milliseconds. At that time, they showed that only packet arrivals for TELNET and FTP control sessions with fixed rate of transmission can be modeled by Poisson, while other WAN arrival processes could be better described and modeled by self-similar processes. More specifically, they showed that inter-arrival times generated by a single connection best fit a Pareto distribution.

With the rapid raise of WWW in late nineties, findings of [2] and [3] became even more evident and were confirmed in [9] and [10]. Authors in [9] shade some more light about the causes of self-similarity and they associate it with heavy-tailed distribution of file transfers. Their conclusion is based on empirical data of WWW traffic collected at local area network Web server (NCSA Mosaic). Likewise, authors in [10] confirm the heavy-tailed distributions of file sizes directly affect the degree of self-similarity (or long-range dependence - LRD) and this phenomenon is likely to continue to happen even under network constrains (such as bottlenecks), topology changes or distribution of file request inter-arrival times. They show that transport layer protocol (TCP and its versions) plays the main role in preservation of a such relationship.

In contrast to previous findings, authors in [18] argue that with the increase of number of connections between different pairs of sources and destinations, the notion of nonstationarity

must be considered as fundamental characteristic of data traffic and is more important than evidence of LRD and heavy-tailed distributions that characterize a self-similar time series. Using data collected at Bell Labs LAN link, they show that nonstationarity affects traffic variables through time. As the rate of connections increases, LRD becomes weaker (traffic will be less bursty) while packet inter-arrivals have a tendency toward Poisson and packet sizes toward independence. They suggest that Poisson assumptions would be even more evident in Core network, where the number of aggregated sources is much higher.

More, a study in [19] analysis main variables of internet traffic (packet and byte counts) for a period of time between 2002 and 2003. They observed that LRD became significantly weaker for packet counts time series of 2003 compared to those in 2002. This observation is attributed to peer-to-peer file sharing application which caused high variability at some time scales. In contrast, for byte counts time-series they observed no difference. Another important finding and contradicting to [18] was that LRD was not affected by time of the day, implying the strength of self-similarity or LRD does not change as number of active sources change. It is important to note that this study was conducted using measurements taken in a university environment.

Finally, Karagiannis et al. [4] show undoubtedly a more complete picture of network traffic in Wide Area Networks. Using three different types of datasets collected during 2002-2003 from CAIDA monitors in a link that belonged to MFN (Metro Media Fiber Network, a former US Tier 1 ISP), they showed that both Poisson and LRD can coexist in traffic aggregated at Core network, depending the scale of observation. More specifically, authors analyzed two common variables of network traffic: byte counts and inter-arrival times. They found that packet size arrivals and inter-arrival time series at small timescales (sub-seconds) can be well described and modeled by Poisson. Beyond that traffic exhibits dominant characteristics of nonstationary and in addition shows self-similar properties. They attribute this finding to the evolution of Internet and backup the arguments showing that results are consistent from theoretical results for large-scale aggregations of renewal processes. In addition, they suggest that as Internet will continue to grow in number of applications and sources, traffic might become even more *regular* and we can abandon sophisticated traffic models instead of simple models such as those based on Poisson assumptions. Finally, they suggest that aggregated traffic from different backbone links might appear different.

2.2 INTERNET TOPOLOGY

While Internet traffic got the most of attention during nineties, Paxson and Floyd continued to challenge research community about many aspects of Internet behaviour. In the article [20], they discuss the challenges of simulating Internet topology due to the lack of understanding the behaviour of it. Internet topology is very complex and more importantly it changes drastically over the time (due to new connected networks, failure of some nodes, nature of routing protocols to balance the load in different paths, routing policies, etc). The architecture of Internet Protocol (IP) allows different technologies to converge and different networks to be administered based on different policies, yet to communicate seamleassly and allow many types of applications to be built on top of it. This flexibility comes together with the tradeoff of not having a clear picture of what is going on in the Internet. Hence, this challenges research community in designing efficient and scalable protocols and creating more robust and realistic models that can predict the behaviour of it.

In 1999, it was Faloutsos brothers [21] who discovered that Internet has some regular shape. Despite its appearance as random, they showed that Internet obeys simple rules too. After they analyzed three datasets collected between November 1997 to December 1998 based on *traceroute*, they found that Internet is a scale-free network which can be modeled and understood by power-law degree distributions. It was considered as important as the work of Leland et al [2] and raised debates among reseach community. Internet was not excpected to show scale-free features as this invalidated another long believed theory based on classical random graphs, where the seminal work of Erdos and Renyi showed that networks grow randomly by adding new nodes and connections have an exponential distribution (well discribed by Poisson). Up until then, network protocols were designed for random graphs which perform poorly in scale-free networks.

They studied internet topology at two granularities: router level (where each node is represented by a router) and inter-domain level (AS) where each domain represents a node and

each edge is connection between two nodes. And, metrics they identified to describe the properties of Internet topology as a graph are degree-rank power-law and eigenvalue-rank power-law.

A major problem with such conclusions about Internet topology is the way measurements are conducted as they miss around 30% of the links at AS level [12] [13]. Research that is based on *passive measurements* usually rely on BGP routing tables which contain information about links from one AS to its neighors. Other types of datasets include those from RouteViews Project [22] and those obtained from Routing Information Service of RIPE [23]. On the other side, *active measurements* (usually used to map Internet at router-level) based on *traceroute* are even more biased due to number of firewalls, missing links and routing policies by individual administrators. An example of famous active measurements resource is *skitter* tool developed by CAIDA [24].

In order to better describe the structure of Internet, power-laws and degree distributions are not enough. Different networks might have the same degree distribution but totally different structural properties [25]. Thus, it is important to look beyond the degree distribution in order to have more acurately models which reflect the real properties of a network. To do so, authors use different metrics such as the correlation between node degrees to see wether high degree nodes tend to connect with high degree nodes or to low degree nodes (assortativity coefficient) [26] and how often a node or link (edge) can be found on the shortest path [27].

In this thesis work, the focus is to show if the network traffic has changed and show network topology characteristics from a different perspective but using the same methodology and metrics used by the work of authors mentioned above. Therefore, in Section 2.3 mathematics of work mentioned are given and implementation of methods is shown.

2.3 DEFINITIONS

This Section shows the mathematics behind the definitions and notions found on the articles reviewed above that are required to understand and quantify metrics of interest for this study.

A random variable or a stochastic variable is discrete when it takes only countable number of individual values and it is continuous if it assumes that all values within a specific interval according to a density function $f(x)$.

Cumulative Distribution Function or CDF of a stochastic variable X shows the probability that X is less than or equal to a given value x:

$$F(x) = P(X \leq x)$$

Probability Distribution Function or PDF of X is the derivative of CDF and is given as:

$$f(x) = \frac{d}{dx} F(X)$$

When X is a discrete, we use PDF of probability function $p(k) = P(X = k)$. This tells us the probability that a values of X is equal to x. Some of the mentioned distributions in the Section 2.1 are:

Exponential Distribution:

$$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0 \text{ and mean is } \frac{1}{\lambda}$$

Poisson Distribution:

$$p(k) = P(X = k) = \lambda^k \frac{e^{-\lambda}}{k!} \text{ for } k = 0,1,2, \ldots \text{ and mean is } \lambda$$

Pareto Distribution:

$$F(X) = P(X \leq x) = 1 - (\frac{\alpha}{x})^{\beta} \text{ where } \alpha, \beta \geq 0, x \geq \alpha$$

Pareto Distribution is the most popular *heavy-tailed* distribution where for $\beta \leq 2$, the distribution has infinite variance and for $\beta \leq 1$ it has infinite mean.

Heavy-tailed distributions are used to understand underlying distribution of network traffic as observations of self-similarity and long-range dependence have been strongly related to the presence of heavy tailed distributions. As shown in [16], the probability that an event will persist in the future increases with the period of observation of such event. For example, the longer we observe a connection, the more certain we can be that it will repeat or continue in the future.

In addition, network topology inferred from network traffic, both at router level and AS level are shown to have a node degree distribution that decays much slower than exponential [21]. A probability distribution of a function is given as:

$$P(X \geq x) \sim C x^{-\beta} \text{ for } x \to \infty, \beta \geq 0$$

where β is the tail-index and C is a positive constant.

To evaluate the presence of heavy tails, we plot probability density function for the variable of our interest in a log-log plot and observe for the presence of linear behavior in the tail of the data.

In order to estimate the value of shape parameter β we use either least-square method or Maximum-likelihood estimation (MLE). Least-square regression is computed using *polifit* function in Matlab and MLE is computed using the method and code in [28]. Furthermore, to check the goodness of fit, in addition to visual inspection where we plot our data together with best fitted distribution, we calculate coefficient of determination r^2, plot the residuals, and test the null hypothesis against generated synthetic data.

In contrast to cumulative distribution function, index of central tendency is used to describe the measured data using a single meaningful value. The most widely used index is the mean or expected value:

$$E(X) = \bar{X} = \sum_i x_i p(x_i)$$

Measures or indices of dispersion are used in addition to those of central tendency in order to have a more accurate and realistic conclusion typically when two measures are contrasted. They show the variability in a data. The ones that are mostly used are:

Variance:

$$Var(X) = \sigma^2 = E[x^2] - (E[x])^2$$

Standard Deviation:

$$S(X) = \sigma = \sqrt{Var(X)}$$

Coefficient of Variation:

$$R(X) = \frac{S(X)}{E(X)}$$

Now, let $\{X_n\}$ be a time discrete stochastic process which is created by periodic sampling across a series of fixed length and is a collection of random variables that represent the evolution of some system.

Such stochastic process is said to be *strictly stationary* if:

$$\{X_n\} \text{ and } \{X_{n+k}\}$$

share the same distribution. That implies that a shifted process by k is equivalent to its original process with regard to finite dimensional distributions:

$$X =_d X_k.$$

Strict stationarity is too restrictive and instead we are more interested in second-order or also known as covariance stationary stochastic processes. That is, covariance only depends between two time periods. For example, let's say we have a time series of 5-hour byte counts. So,

covariance of byte counts between hour 1 and hour 2 should be the same as covariance of values between hour 3 and 4 (lay of 1 hour).

Thus, we characterize the dependence between two values of stochastic processes at different times by evaluating the Autocorrelation Function (ACF). ACF of a time series at lag k and mean $\bar{X}$ lies between -1 and 1 and is given by:

$$r(k) = \frac{Cov(X_n, X_{n+k})}{Var(X)} = \frac{E[(X_n - \bar{X})\,(X_{n+k} - \bar{X})]}{Var(X)}$$

As a result, if packet size arrivals or byte counts are uncorrelated, all ACF values for the calculated lags will lie around zero within 95% confidence interval.

In contrast, if there are no zero correlation values in $r(k)$, then we say that a stationary stochastic process X_n is *long-range dependent* (LRD). That implies that the sum of all ACF values does not converge:

$$\sum_{k=1}^{\infty} r(k) = \infty$$

There are couple of definitions of self-similarity in literature, but we follow the sources mentioned in the Section 2.1 and the most standard one states that a phenomenon is considered self-similar if the properties of an object are preserved irrespective the scale in time or space. A stochastic continuous time process $X(t)$ is self-similar with self-similar parameter H and scaling factor α if:

$$X(t) =_d a^{-H} X(at), for\ \forall\ t \geq 0, \forall\ a > 0\ and\ 0 < H < 1$$

This implies that $X(t)$ and its version scaled in time $X(at)$, after normalized by a^{-H}, follow the same distribution. Finally, a stochastic time-series is said to be second-order self-similar if the autocorrelation function, ACF, preserves the same structure regardless the aggregation in time.

Intuitively, self-similarity refers to scaling behavior of the distribution of a time-series while LRD observes the behavior of tail of the ACF of assumed stationary time-series.

Self-similarity and long-range dependence can be quantified using only one parameter, *Hurst parameter H*. For $H > 0.5$ a second-order self-similar time-series is said to be long-range dependent. Such process demonstrates a very slowly decaying ACF and the curve under the slope has an area of infinity. In distribution terms, ACF has an exponential form of:

$$r(k) \sim k^{-\beta} \text{ as } k \to \infty, \text{ where } 0 < \beta < 1$$

and

$$H = 1 - \frac{\beta}{2}$$

As $H \to 1$, the degree of self-similarity and long-range dependence are higher. For $0 < H \leq 0.5$, usuaylly means that a time-series is short-range dependent and ACF decays exponentially fast. For H=0.5 it would be a smooth Poisson traffic model. And, for values of $H \leq 0$ or $H \geq 1$, the result does not imply any useful information regarding the model.

2.3.1 Estimating Hurst Parameter

Hurst parameter can only be estimated and not calculated using the definitions. Thus, in order to test a time-series for self-similarity and long-range dependence, different properties of a finite time-series composed of traffic values such as slow decaying variance of partial sums or spectral density should be investigated. To do so, there are several Hurst estimate methods both in time domain and frequency domain. The reason behind existence of more than one estimator is the asymptotic nature of Hurst parameter itself [29]. We have used two estimators in time domain and two in frequency domain. In time domain, we have chosen to use *rescaled-range statistics* (R/S) and *Variance-Time* methods whereas in frequency domain we have chosen *Periodogram* and *Whittle* methods. Each estimator looks at different properties of time series and below we have described the four used ones. The relative accuracy of these estimators is discussed in [2].

Methods for estimating Hurst parameter are implemented in Matlab using the reference code in [30] and after compared with scientific tool, *Selfis*, developed by Karaginis et al [29], it is observed that there are slight differences in results. Because Selfis is a more reliable tool and is widely used by research community, results that will be shown in this work are calculated with it.

2.3.1.1 R/S Method

For a time-series X_n, sample mean $\bar{X}(n)$ and standard deviation S, R/S statistics is the range of partial sums of standard deviations from the mean, rescaled by S. In our case, if we denote by X_t the number of bytes at time t and if

$$Y_t = \sum_{i=1}^{j} X_i$$

to be the cumulative of byte counts up to time *j*, then R/S statistics is given by:

$$R/S = \frac{R(t,k)}{S(t,k)} \quad \text{where}$$

$$R(t,k) = \max_{0 \le i \le k} \left[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t) \right] - \min_{0 \le i \le k} \left[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t) \right]$$

and,

$$S(t,k) = \sqrt{\frac{1}{k} \sum_{i=t+1}^{t+k} (X_i - \bar{X}_{t,k})^2}$$

One can estimate Hurst directly from a log-log plot of R/S versus the number of observations of an aggregated time series. This gives a straight line with a slope estimating Hurst parameter. The slope of a line fitted to the values in a graph is calculated by a least square method. In practice, R/S cannot be calculated for every value of t and k and instead an equal

number of non-overlapping blocks or intervals (lags) k are chosen. Because the least square method is applied to log-log plot, it is chosen to use the logarithmically spaced values of k. The dilemma in using R/S method is how to choose the adequate number of t and k values and the low and up cut-off values to include in calculation. This can be one reason that Matlab code and Java based tools, Selfis, show different results.

2.3.1.2 Variance Method

Let X^m be an aggregated time-series which is derived from non-overlapping blocks of size m of original time-series. That is, in one chosen interval or bin of let's say 1 second we have 100 observations of byte counts, then $\bar{X}^1$ is the value of byte counts during 1 second interval of 100 observations. After calculating the non-overlapping blocks and the variance, $Var(X^m)$, of the aggregated values in those blocks, the variance is plotted against the block size in a log-log plot. If the formed line, has a slope between $-1 < \beta < 0$ (applied by a least square method), it indicates for self-similarity and Hurst parameter, H, can be calculated as $H = 1 + \beta/2$. Dilemma in computing Hurst with this method is how to choose the minimum and maximum cut-off block sizes to include in calculations.

2.3.1.3 Periodogram Method

In order to investigate the shape of spectral density properties and the distribution of frequencies, Periodogram is calculated using the definition below:

$$I(v) = \frac{1}{2\pi N}\left|\sum_{j=1}^{N} X(j)e^{ijv}\right|^2$$

where v is the frequency, N is the number of terms in a time-series, and X is the time-series. For a time-series to be long-range dependent, periodogram should be proportional to $|v|^{1-2H}$. This

implies that LRD is demonstrated in spectral density which is characterized by a power-law near the origin. The slope formed in the log-log plot of the periodogram of the time-series forms a straight line with $slope = \beta - 1 = 1 - H$. In practice, only the lowest 10-20% of approximately

$\frac{N}{2}$ frequencies are used to estimate H.

2.3.1.4 Whittle

Whittle does not provide us with a graphical approach but is considered one of the most stable estimator and estimating H together with 95% confidence interval is given by $H \pm 1.96 \cdot S_H$.

This method is based on the minimization of likelihood function applied to the periodogram of the time-series. Because it requires to provide with an underlying stochastic process such Fractional Gaussian noise (FGN) or Fractional ARIMA and is relatively complex to implement, calculations are strictly relied on the scientific tool, Selfis.

2.3.2 Properties of Topology

The structure of Internet topology is built upon connections between hosts, routers or Autonomous Systems. Properties of Internet topology at different granularities can be analyzed and quantified using the metrics and notions of graph theory.

The unit properties of a graph are number of nodes (vertices) n and number of links (edges) m

(Figure 2). In our case, we have considered undirected graph which means that nodes connected together have all the links bidirectional. Formally, a graph G with vertex set V and edge set E is given by $G = (V, E)$.

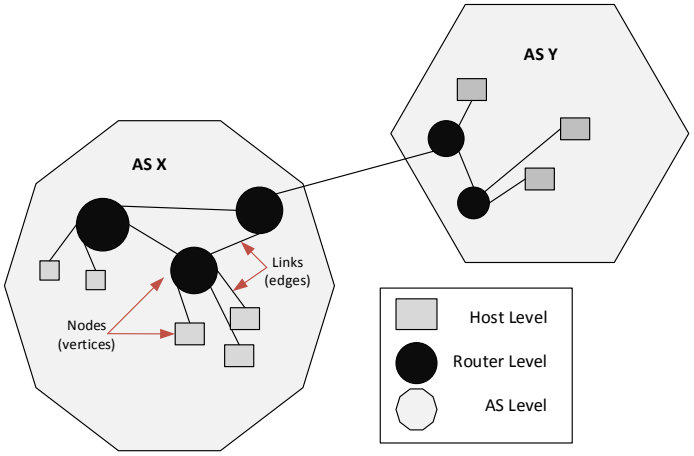

Figure 2. Unit Properties of graph at different granularities

The Internet topology, from the perspective of collected data traffic, is studied at IP granularity (Router level), where one unique IP address is considered one router or one node. IP addresses are extracted from participating hosts in a session (flow) and the number of unique established connections that one IP has within interval t is the number of links or edges this IP address has acquired during this interval. Internet topology is studied at two timescales: 24 hours and 7 days.

To quantify underlying structure of the topology, it is chosen to use some of the most popular metrics used in literature which are mentioned in Section 2.2. Such metrics include centrality metrics, correlation between node degrees and diameter of the graph.

Centrality metrics can be classified into metrics based on the number of connections (degree) and metrics based on the shortest paths in the network. Degree centrality and eigenvector centrality are categorized based on the degree while betweeness centrality based on the shortest path. Degree centrality and degree distribution are many times considered illuminating as they only describe a network topology based on the number of connections a node has. But, network elements that are more visited are clearly more important for end users and thus are more critical for network topology and its properties. The node degree distribution is the probability that a randomly selected node is k degree:

$$p(k) = \frac{n(k)}{n}$$

where n(k) is the number of nodes n that have degree k. As mentioned in Section 2.2, degree distribution is shown to obey power-laws. Intuitively, this means that majority of nodes have low degree but small number of nodes, also known as hubs, have high degree.

Beyond centrality metrics, assortativity mixing by degree, r, is used to investigate the correlation between nodes. Assortativity mixing shows the preference of a high-degree node to attach to other high-degree ones. For evaluating r on a network under study, one can calculate Pearson correlation coefficient as stated in [26]:

$$r = \frac{M^{-1}\sum_i j_i k_i - \left[M^{-1}\sum_i \frac{1}{2}(j_i + k_i)\right]^2}{M^{-1}\sum_i \frac{1}{2}({j_i}^2 + {k_i}^2) - \left[M^{-1}\sum_i \frac{1}{2}(j_i + k_i)\right]^2}$$

where j_i, k_i are the degrees of the nodes at the ends of ith edge, with $i = 1 \ldots M$. This and other graph metrics are calculated using *iGraph-R* which are based on the implementation from [26]. The assortativity coefficient, r, can take values between $-1 < r < 1$. For negative a network is considered disassortatively mixed by degree and vice versa. Internet topology is known to be disassortatively mixed and shows that nodes with high degree are more connected to nodes with low degree. Networks that are *disassortative* are more vulnerable to attacks.

Finally, the diameter d of a graph is the maximum number of links a *message* has to travel between any pair of nodes. That is, d is the greatest distance between any pair of vertices or, $d = \max_{v \in V} \in (v)$. To find the diameter of a graph, first the shortest path between each pair of nodes are calculated and then the greatest length of any of these paths is the diameter of the graph. Surprisingly, many networks have been shown to have a very small diameter, regardless

the huge number of nodes and links. This is known as *small-world* phenomenon. There have been contradictions whether diameter increases over time with increased number of nodes in a graph or not [31], and still empirical studies show contradicting results.

3.0 DATA COLLECTION AND ANALYSIS

In the previous Section metrics and defined methods to quantify characteristics of the network traffic and topological structure were described. In this Section, the network under study is shown. In addition, traffic measurements and general characteristics of the collected data are stated.

The dataset of the network traffic used in this paper is from a National Tier 1 Service Provider, Post and Telecom of Kosovo (PTK), which serves more than one million broadband users and is considered the biggest one in the country. PTK offers fixed and mobile converged services. As most providers, it is also in the final transition phase from a telephony service provider towards integrated quad play service provider.

PTK backbone is comprised of seven points of presence or aggregation sites (PoP) distributed in geographical areas of the country. Recent technical developments are leading towards convergence between fixed and mobile networks, especially in regards to broadband services, and PTK is an example of this trend. Two Internet Border Gateways enable communication to other upstream providers in global Internet. In this regard, we analyze traffic both at IP level and autonomous system level (ASN) and by upstream traffic we assume the traffic generated by any source IP address that falls in a range of allocated networks for PTK toward other providers (global Internet) and with downstream traffic, the traffic that has as a destination one of these IP addresses that falls in this range. An overview of the data measurement infrastructure is depicted in Figure 3 and is broadly divided into three main components: Monitoring, Collection and Analysis.

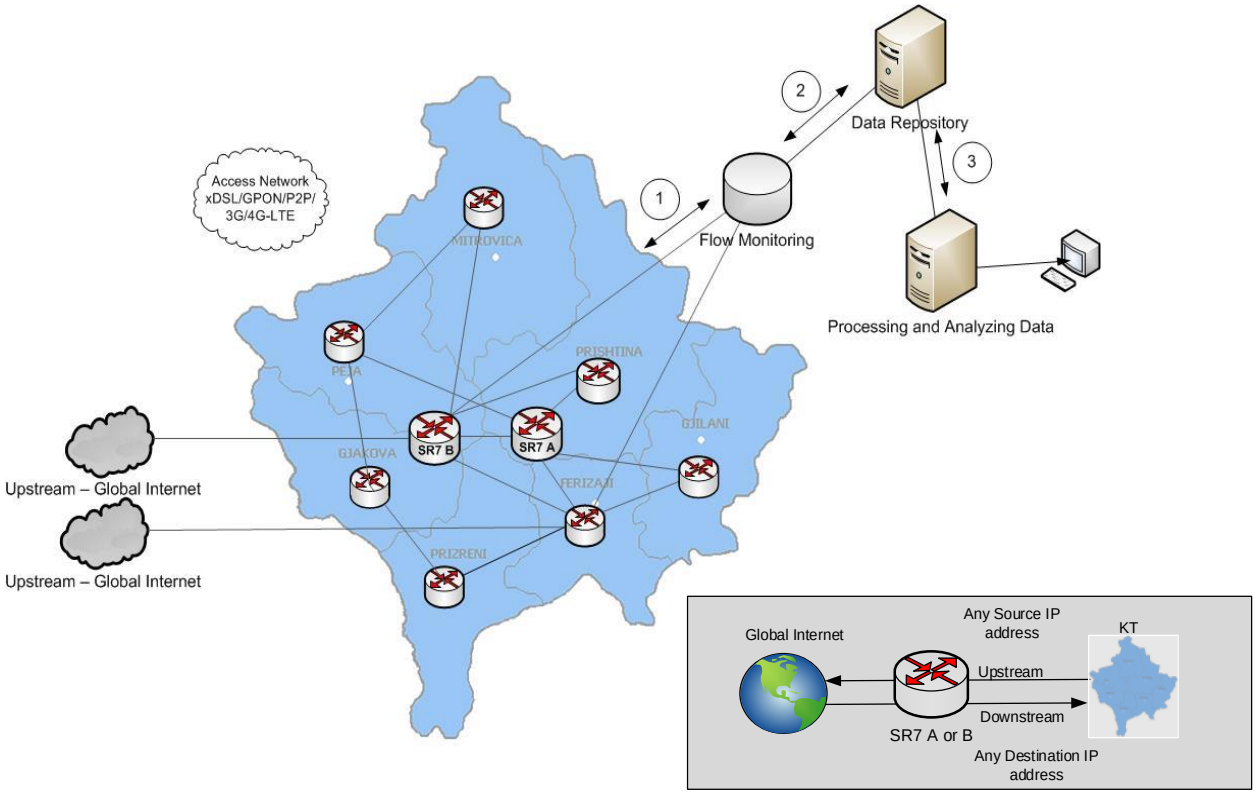

Figure 3. PTK Network and measurement infrastructure to collect data from all regions in Kosovo. b) Netflow definition for upstream/downstream traffic

3.1 MONITORING

First, IP Flow feature was enabled in two Internet Border Gateways (IBGs). In our case, NetFlow protocol was already configured. NetFlow is one of the most used passive network monitoring tools and was first developed and used by Cisco Operating Systems which later became a standard for other manufactures [32]. Lately, NetFlow has been widely used by network operators and found application in network capacity planning.

A general definition of an IP Flow is a quintet or a five-tuple made up of source and destination IP address, source and destination port number and the protocol. From one flow, information about traffic volume in bytes and packets can be extracted, source and destination IP addresses, type of service, port numbers, protocol type, throughput in terms of packets per

second and other details found in a IP packet header. In Table 1 a summary information is depicted that can be found in a typical field of a NetFlow. From the table below, not all the fields are used for the purpose of this paper. Basically, input/output interfaces, type of service and next destination address did not contribute to this study at all.

Table 1. Netflow fields and their description

Field	Description	Bytes
%ts	Unix start time of the flow	4
%te	Unix end time of the flow	4
%sa	IPv4 source address where flow is originated from	4
%da	IPv4 destination address where flow is destined to	4
%sp	IPv4 source port where is originated from	2
%dp	IPv4 destination port where flow is destined to	2
%pr	Protocols up to Layer 4: UDP, TCP, ICMP, ESP, etc.	1
%flg	TCP flag	1
%tos	IP type of service	1
%ipkt	Number of packets in one session or flow	4
%ibyt	Number of bytes for one session or flow	4
%in	Interface number where incoming flow is processed by	4
%out	Interface number where outgoing flow is processed by	4
%smk	Source subnet prefix based on BGP router's table	1
%dmk	Destination subnet prefix based on BGP router's table	1
%sas	Autonomous System where flow was originated from	4
%das	Autonomous System where flow is destined to	4
%nh	Next destination address	4
Total	Total Byte per Typical Flow	53

In terms of access technologies, before we started collecting the data, we took a snapshot from a Cacti RRD tool (tool used for traffic monitoring by PTK) to get an idea of traffic utilization based on technologies. DSL&GPON comprise 94% of traffic (both upstream and downstream), 3G/LTE has around 4.69% and WiFi has only 0.57%. Also, Figure 4 shows the traffic load during seven days in PTK toward upstream providers a) and b), while, c) to e) shows the YouTube traffic. In total, downstream network utilization is around 11.5 GB from which 5.5

GB are only YouTube video. This implies that traffic in PTK network is more than 50% of Internet video.

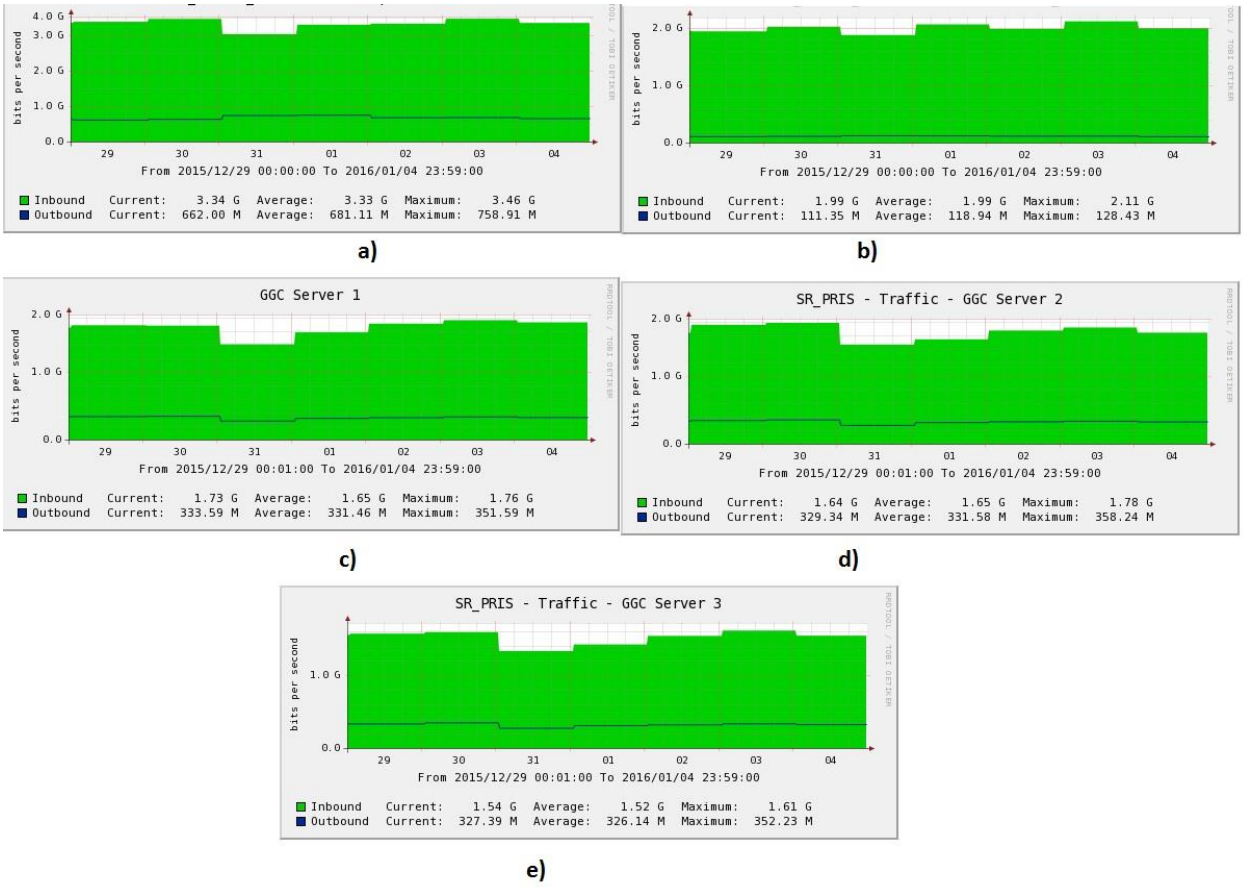

Figure 4. Traffic load in/out at gateways a) and b) and three Google Cache Servers located at PTK c)-e)

3.2 DATA COLLECTION

Second component is collecting and exporting IP flows using *nfcapd*, a netflow capture daemon which is configured in the local shared machine at PTK as in [33]. All these series of packets that share same quintet, or a flow, is saved on a IBGs either until buffer/cache is filled or every five-minute interval. Whenever system clock hits five minutes interval or router IBG cash if filled, a file in a format of *nfcapd.<datastamp>* is created, e.g. *nfcapd.201512280845* contains data from

Dec 28th 2015 08:45 onward. Data are stored in three folders (Source 1, Source 2 and Source 3) where each folder contains subfolders with all dump files for respective day, Figure 5. Based on a 5 min time interval, this results in 288 files per day. In Source 1 there are 92 subfolders (28 December 2015 to 31 March 2016) and each subfolder has 288 *nfcapd* files, which corresponds to the interval of 5 minutes each (24*12=288). Same for Source 2 and Source 3. All the files stored in local folders can be read with *nfdump* using its syntax and *boolean* expressions to filter the data. For example, the command:

nfdump -r nfcapd.201512291520 "proto tcp"

selects all the flows in the file *nfcapd.201512291520* and prints out ones that use TCP protocol and other information like

…

Date first seen Duration Proto Src IP Addr:Port Dst IP Addr:Port Packets Bytes Flows

2015-12-29 09:23:09.240 4.750 TCP 24.46.103.210:12733 -> 178.175.37.237:60651 2000 3.0 M 1

2015-12-29 09:23:02.230 11.780 TCP 213.163.125.246:43058 -> 8.254.105.254:80 2000 104000 1

…

showing the timestamp, duration of the session is seconds, protocol (only TCP because we used the expression "proto tcp"), source IP address and the port number, destination IP address and port number, number of packets sent during this session, bytes and the flows. Usually, number of flows is 1 unless the connections that share the same parameters (src IP/port, dst IP/port and Protocol) are aggregated. Aggregation can be accomplished by using the *-a* option after the filename.

Collectors specifications are: RAM 18 Gb, CPU Intel Xeon 2 GHz, HDD 100 Gb, GE NIC. From the collector, data is crawled and transferred to a personal FTP server and then to University of Pittsburgh networking lab machine during night time using secure connection (SSH2).

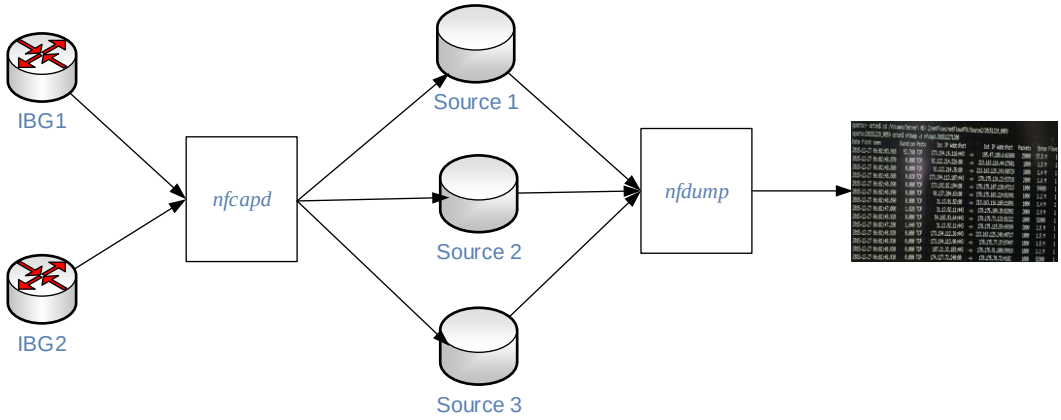

Figure 5. Data Collection using nfcapd and sources for processing netflows

3.3 DATA ANALYSIS

In addition to *nfdump* which can compute simple statistics, data are processed using custom *Python* scripts in order to read large amount of flows in an efficient way and to store them in a human-readable format. In order to compute statistical analysis of the data, software such as Matlab, Selfis, R, Gephi and associated developed packages for them are used to get the most accurate results.

Besides the question whether the Internet traffic shows self-similar and long-memory nature it is also important to know general characteristics of the data set such as distribution of packet sizes, traffic volume for inter and intra domain and what protocols are present and their proportion. In this Section, main characteristics of the dataset that are used for the purpose of this study are shown in Table 2. One can see an overview of the processed daily data that are used for analysis. In total we have worked with 31,602,250 flows, when aggregated. They generated $1{,}4752 \cdot 10^{11}$ packets during seven-day interval. In addition, it shows the average number of packets per second, average bytes per packet as well as the number of unique connections during busy hours (11 AM to 3PM).

Table 2. Characteristics of Dataset used in this study

Days	Number of Flows	Number of Packets	Average Packets per second	Average Bytes per packets	Unique Connections during busy hour (Upstream/Downstream)
Day1	9,467,477	25,259,807,842	1947494	1068	160911 / 1864484
Day2	4,261,459	20,526,364,391	2174049	1054	167047 / 1945073
Day3	2,929,059	16,161,126,666	1496402	986	334513 / 3585167
Day4	4,164,621	24,641,601,294	2085229	1058	227875 / 2705763
Day5	4,150,023	23,951,432,184	2094708	1064	212447 / 2277633
Day6	2,799,257	14,815,754,013	1371831	1065	183080 / 2122497
Day7	3,830,354	22,163,825,619	2052211	1043	235895 / 1705700
	31,602,250	147,519,912,009	13221924	1047	

Figure 6 below shows the number of packets that arrive each minute for 24 hours for the local traffic (intra-domain). In order to get information about the number of packets from the *dump* files, timestamp and filter only the local traffic, custom python script is executed for all files that should belong to a 24 h interval. To filter only local traffic, knowledge about subnetworks from IP-Plan obtained by PTK is used. In order to concatenate all files processed by *python* or *nfdump*, the classic unix tool, *awk*, is used all the times. For example, the script below concatenates all the files that share the same header information and creates a new file.

```
...
awk '
FNR==1 && NR!=1 { while (/^<header>/) getline; }
1 {print}
' *.csv >bytesDownstream.csv
...
```

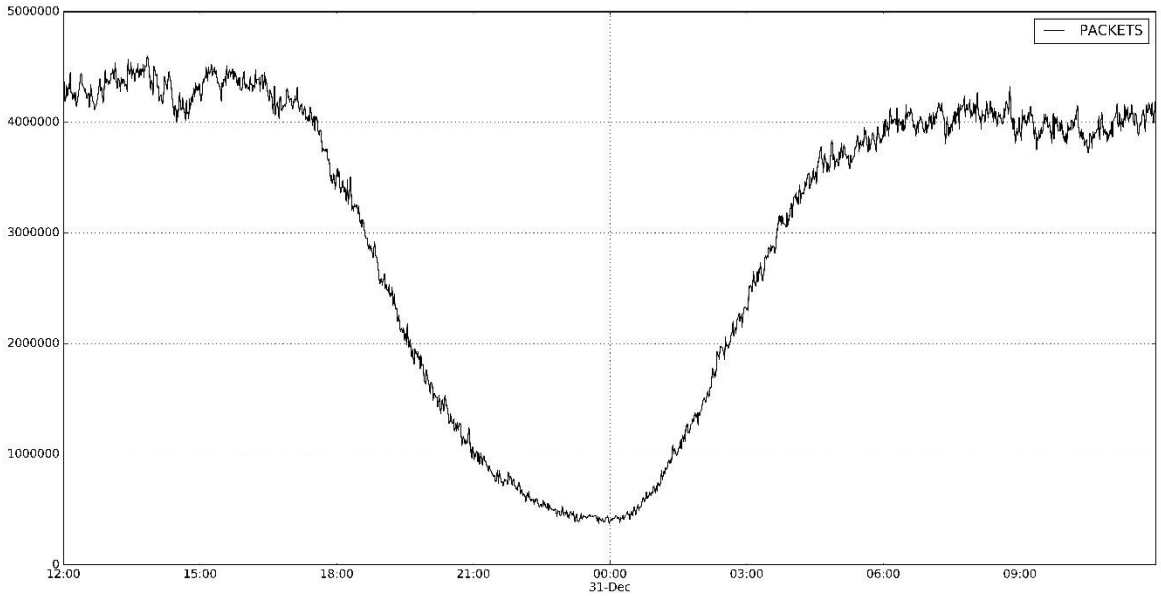

Figure 6. Local Packet Traffic

There is a maximum of 4598000 packets in one minute at 13:51 and minimum of 372000 packets at 23:59. The maximum number of bytes is 667001000 bytes at 08:47 which surprisingly is not contained by the maximum number of packets during this interval but corresponds to 4321000 packets. This suggests that despite the fewer packet arrivals at 08:47, they are greater in size. The minimum number of bytes is 45597000 at 23:47. The local byte traffic during this interval is shown in Figure 7.

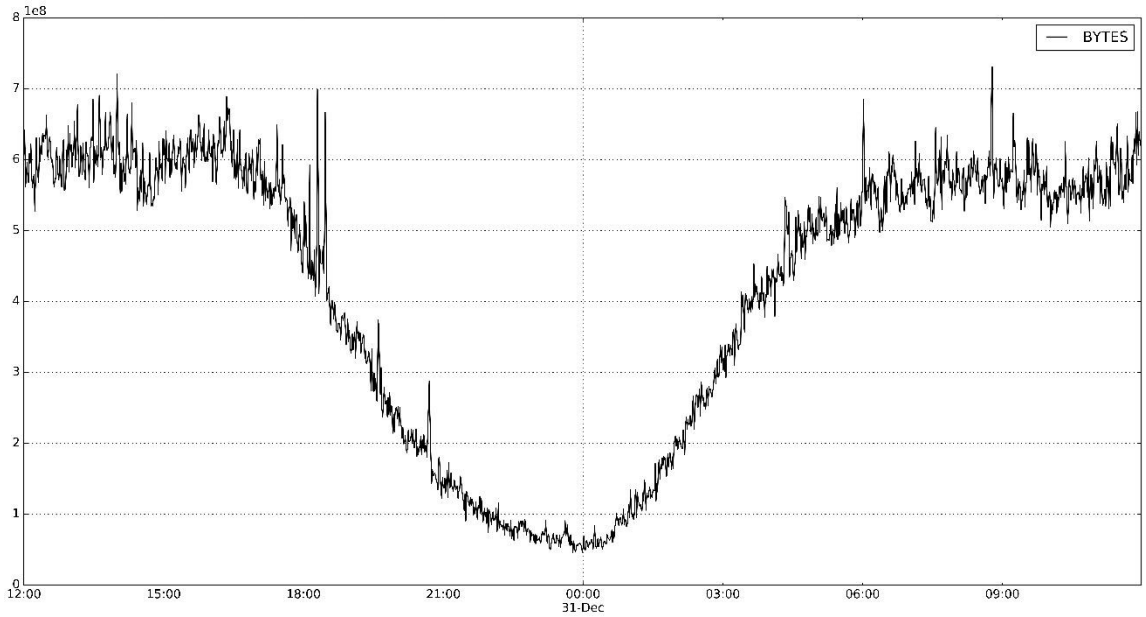

Figure 7. Local Byte Traffic

While the local traffic does not exceed more than 0.6 GB in one minute, downstream traffic toward users within PTK network reaches a peak of 68.85 GB in one minute which corresponds to time interval around 16:19 and minimum of 0.045 GB at 23:47. One can observe that the peak interval is not the same for local and international downstream traffic. In addition, if graphs for local traffic and downstream international traffic are compared (Fig 6 and 7 versus Fig 8 and 9), it can be observed that international downstream traffic is much smoother than local traffic when aggregated at one minute. This implies that at large-scale aggregations of renewal processes where the number of sources and network traffic load increases, traffic tends toward Poisson as shown in [4]. In contrast to [2], which suggest that with the increase load of traffic, also the long-memory properties become even more evident, one should expect the opposite where the LRD strength, H parameter, should be lower during busy hours for downstream traffic.

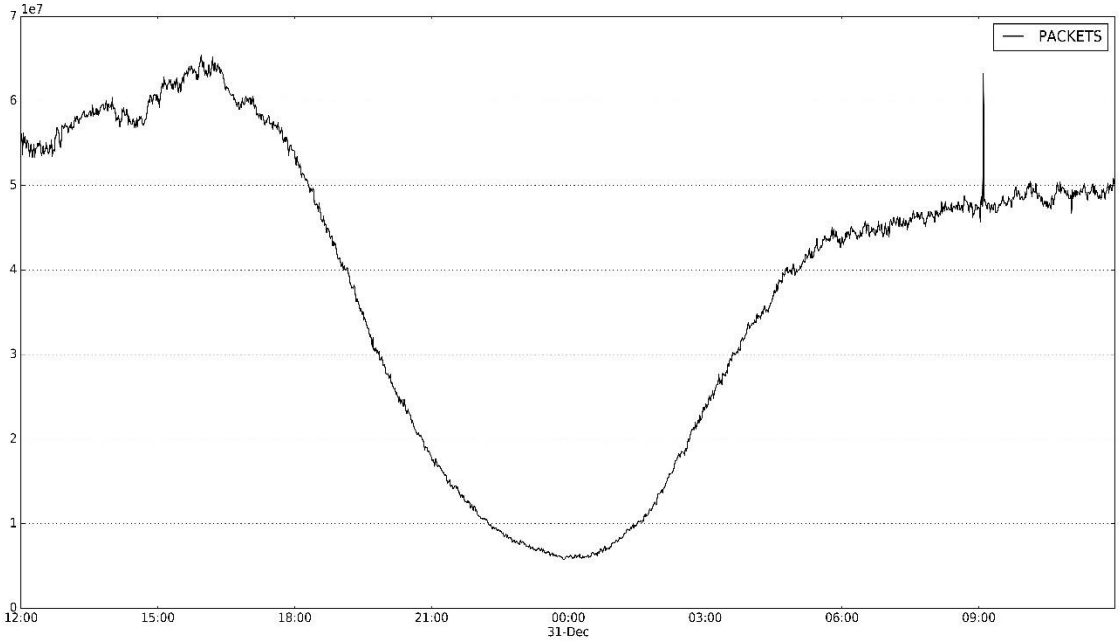

Figure 8. International Downstream Packets

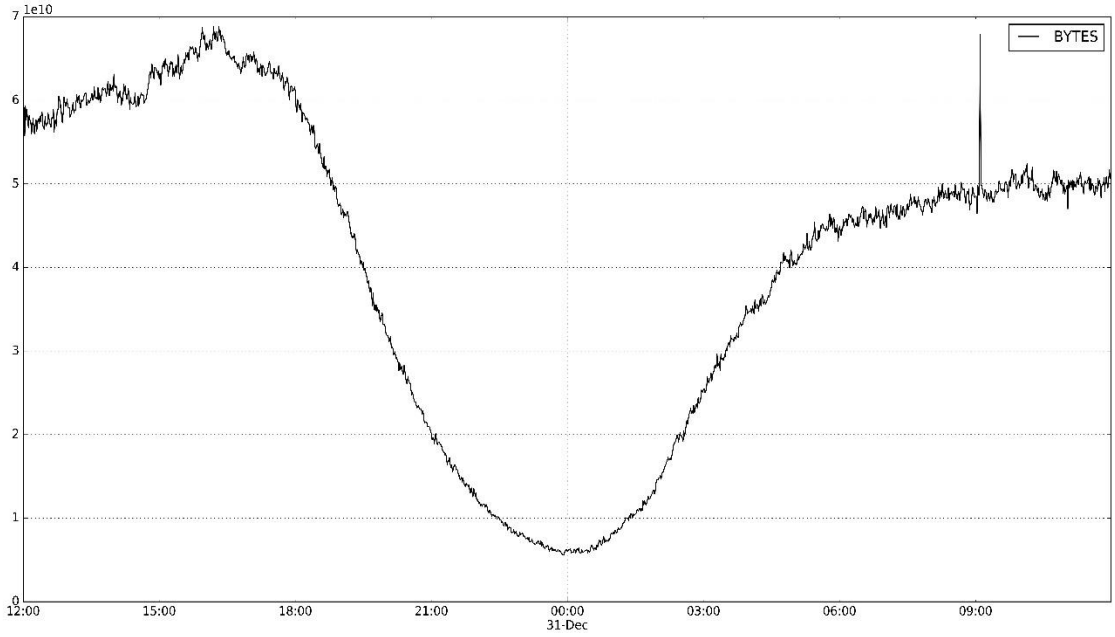

Figure 9. International Downstream Bytes

Another way of expressing the load in a more common unit is by counting or computing bits per second or packets per second within interval of the interest. To do so, there is a simple way by utilizing the nfdump and filtering syntax. For example, to see the average bps and pps respectively during a 2-hour interval, one can use the following command:

nfdump -M Folder1:Folder2:Folder3 -R . -t 2015/12/29.15:00:00-2015/12/29.17:00:00

After it reads all the raw data, at the end of print it shows a summary of the processed flows:

…
Summary: total flows: 2232523, total bytes: 4813154636000, total packets: 4372031000, avg bps: 5347214353, avg pps: 607143, avg bpp: 1100
Time window: 2015-12-29 14:57:31 - 2015-12-29 17:03:32
…

One can read the statistics about bits and packets per second during this interval. In this case, a 2-hour interval was taken into consideration which corresponds to the peak hours on December 29 2015. It shows that average bits in one second is 5347214353 (around 5.3 Gbps) and in average 607143 packets in one second.

3.3.1 Protocols

The table below shows statistics about the proportion IP protocols which appears in the data collected during a one-day interval. This, also can be done easily using the nfdump syntax which can create a lot of top N statistics about protocols, destination or source IP addresses and others, ordered by any available field. In this case, the Table 3 shows six present protocols ordered by the volume (bytes) which is obtained using the syntax:

nfdump -M Source1/20151229_0005:Surce2/20151229_0005:Source3/2015:1229_0005 -R . -s proto/bytes

TCP continues to be the transport layer protocol that dominates the traffic with approximately 93.3% of the bytes and 82% of the packets. Nearly 6.5% of bytes and 17.8% of packets in one-day interval are UDP. TCP together with UDP comprise 99.9% of bytes and 99.8% of packets in PTK network. Note that other IP protocols, appear with negligible probability. ESP and GRE, two IP encapsulation protocols that are mostly used in applications such as Virtual Private Networks, comprise nearly 0.1% of the total traffic. Despite the fact that organizations have been planning to move to IPv6 and its importance in new applications with

Internet of Things, only 24 MB per day are generated by machines running IPv6 in PTK network.

Table 3. IP Protocols Proportion

Proto	Bytes (%)	Packets (%)	pps	bps	bpp
TCP	48.7 T(93.3)	42.6 G(82.0)	492493	4.5 G	1144
UDP	3.5 T(6.6)	9.2 G(17.8)	106691	320.1 M	375
ESP	33.0 G(0.1)	70.4 M(0.1)	814	3.1 M	468
ICMP	5.4 G(0.0)	55.1 M(0.1)	637	502554	98
GRE	4.2 G(0.0)	8.9 M(0.0)	103	387946	469
IPv6	24.2 M(0.0)	53000(0.0)	1	3692	457

3.3.2 Packet size

In the Figure 10, the frequency of packet sizes is given from a dataset extracted between 15:00 and 18:00 on December 29 2015. The smallest and largest packet sizes are 21 bytes and 1500 bytes respectively. The ten most frequent packet sizes are 1430, 1450, 32 52, 1472, 1492, 21, 40, 1480 and 1500. The smallest packet size of 21 bytes corresponds to the smallest possible data sent from the application layer (1 byte) which has to be encapsulated in a header for lower layers. The data (for example a keystroke of 1-byte character) is encapsulated in a TCP segment which adds 20 bytes making the smallest possible packet size of 20+1=21 bytes. Similar, the 1500-byte packet size corresponds to the most common maximum transmission unit (MTU) set by IP technologies, such as Ethernet, that a layer can pass onwards without handling fragmentation.

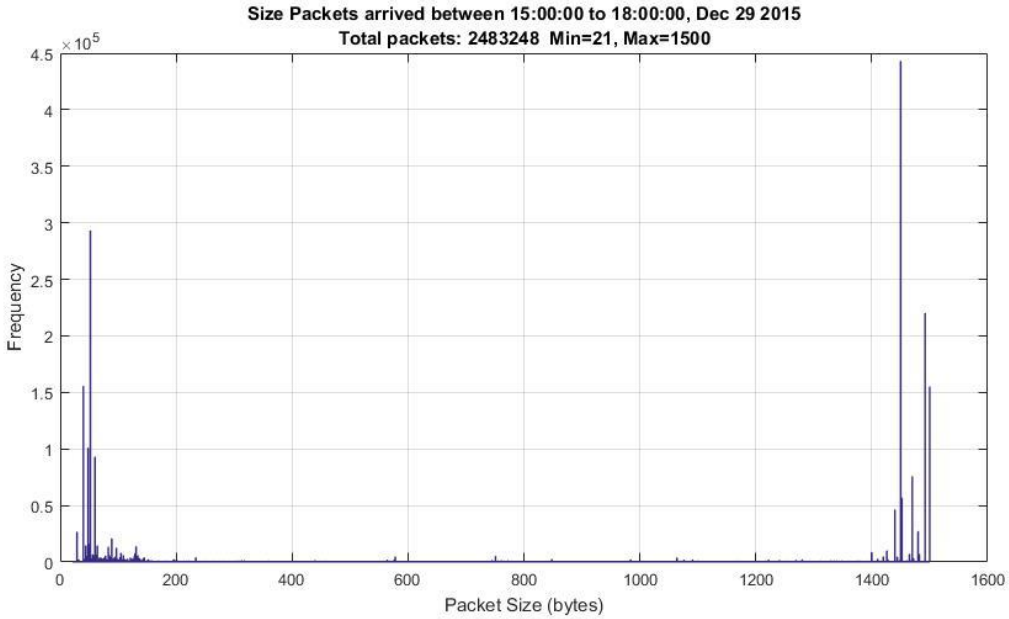

Figure 10. Frequency of Packet Sizes in PTK Network

Cumulative distribution of packet sizes is shown in Figure 11. It shows that 30% of packets are small and less than 100 bytes, most of the packet sizes are distributed between 100 and 1400 and almost 30% of packets are between 1430 and 1500.

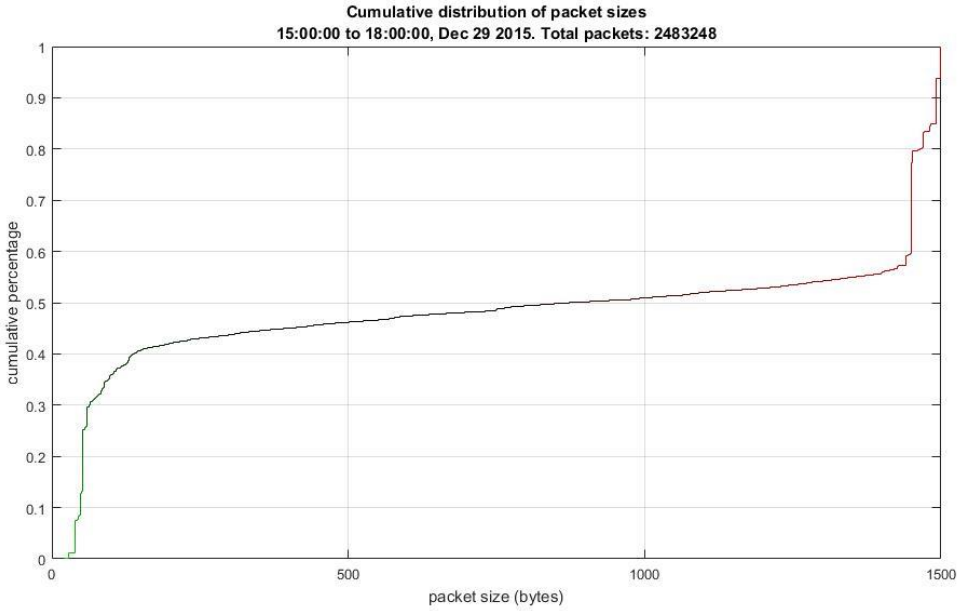

Figure 11. Empirical Cumulative Distribution of Packet Sizes

4.0 EXPERIMENTS AND RESULTS

In this Section we show experiments and results conducted from dataset illustrated in Section 3. It begins by showing whether sample data in PTK network is self-similar and long-range dependent and then the properties of network topology inferred from the same dataset with focus on node degree distribution at IP level and other statistical parameters such as assortativity and network diameter. Finally, other experiments conducted during this study are presented, where packet size arrival independence and temporal disitrubution of top 10 IP addresses and Port numbers is shown.

4.1 IS PTK NETWORK TRAFFIC SELF-SIMILAR AND LONG-RANGE DEPENDENT?

In this section results of investigated traffic captured in PTK network traces are shown with respect to self-similarity and long-range dependence. The definition of self-similarity and LRD are based on the assumptions that the traffic, or the time-series of byte counts (or packet counts), is stationary. This means that traffic that arrives at 1 second interval has the same mean and variance and co-variance only depends on the lag, irrespective the time of the day traffic is investigated. This would have not been true if we take into consideration intervals of 24-hours due to periodicity and daily trends as the number of users and their behavior is different at different time. The fact that stationarity is an elusive property of network traffic and it is very difficult to distinguish between time-series that is stationary and long-range dependent with the one that is non-stationary and short-range dependent, as a common rule is to examine traces in intervals as short as possible. In this case, traces in 30 min interval are investigated both at low hour and busy hour for seven days that arrive in bins of 1 second.

4.1.1 Pictorial view of Periodicity and Self-Similarity

Before checking for self-similarity and LRD characteristics, we show the most well-known property of traffic time-series, daily periodicity, together with the so called fractal-behavior of network traffic. Figure 12 depicts some simple plots of the traffic volume (bytes per time unit) in different timescales for the upstream traffic generated by all the sources. In Figure 12a) we can observe the length of traffic bursts is present in even time scale of 10 s where traffic consists of "bursty" sub periods separated by less "bursty" sub periods as was shown in LAN traffic in [2]. Then we consider one-day traffic in Figure 12c) and focus on a "randomly chosen" period Figure 12b). The graph shares similar characteristics with Figure 12d) where we "zoomed in" by factor of 10 (1 s granularity). This is known as fractal-like behavior of network traffic or self-similarity.

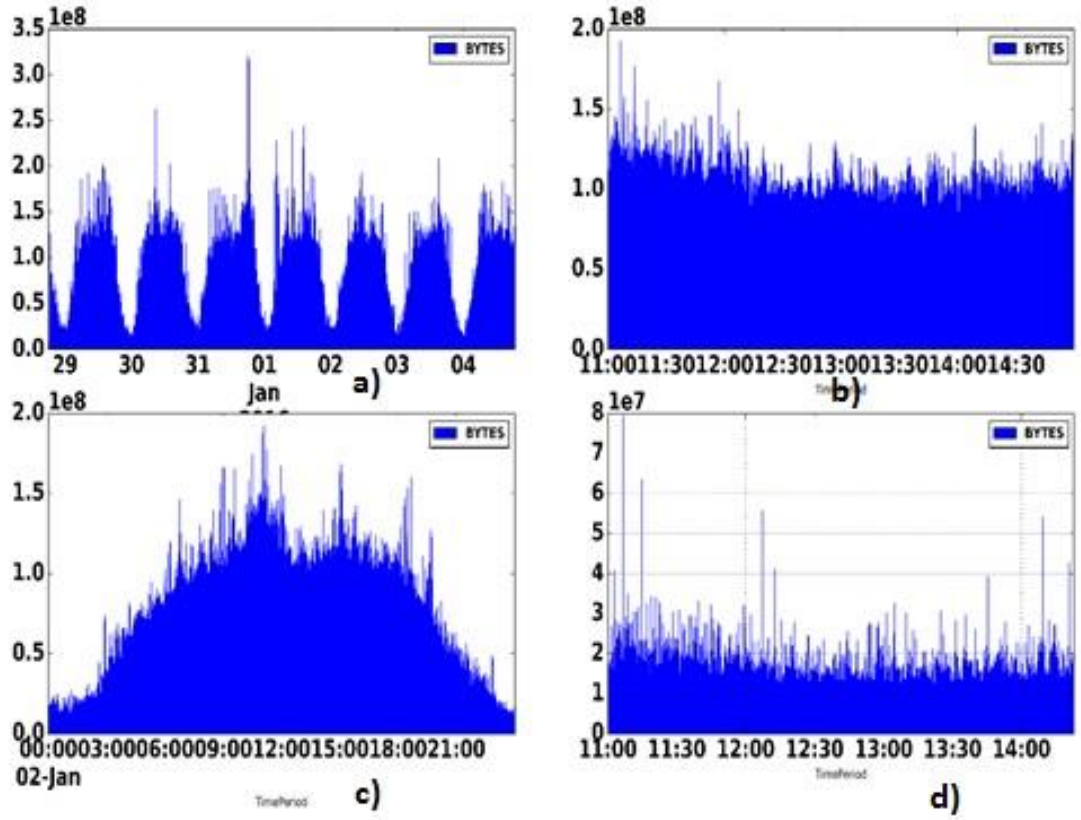

Figure 12. Pictorial view of Periodicity and Self-similarity - a) weekly and c) daily periodicity; b) a random interval when d) zoomed by factor of 10

4.1.2 Self-similarity and Long-range dependence

In this Section, we show the results of seven-days traces for downstream traffic using four types of Hurst estimators, two in frequency domain and two in time domain. For each day we have 1,800 aggregated measurement points for byte counts and packet counts in interval of 30 min during busy hour (15:30:00 to 15:59:59) and low hour (23:30:00 to 23:59:59) at 1 s granularity. Results show that Hurst parameter varies between 0.4 and 0.75 for other estimators except the R/S statistics computed by Selfis. For low hours, Hurst is slightly higher than for busy hours. Again, suggesting that with the increase load and number of users, traffic becomes smoother and

short-range dependent. Results of computed Hurst for all the traces are shown in the Table 5 (in Appendix A). Next, daily results for Hurst estimation are presented for:

a) Busy-Hour Traffic (15:30:00 to 15:59:59) and
b) Low-Hour Traffic (11:30:00 to 11:59:59)

Busy-Hour: Periodogram shows that data fit for the slope at (-4.988) and thus it suggests that time-series is self-similar but short-range dependent, Figure 13. Likewise, Time-variance plot in Figure 14 shows a line with the slope of almost (-1), at around -1.048, and $H = 1 + \frac{slope}{2} \cong 0.5244$. The plot of R/S method shows a slope of ~0.53 which is clearly close to the line with slope $\frac{1}{2}$, Figure 15. Finally, Whittle estimator shows $H = 0.5$ together with confidence interval $[0.461 - 0.538]$ with 90% confidence level. Whittle estimator is the only one which does not provide us with graphical approach, but it has provided us with the most stable results at all timescales. Results for seven days are shown in Figure 16, from where it can be observed that all estimators show very similar results each day, suggesting that traffic in PTK network is short range dependent in large-time scales for all samples both for byte and packet counts.

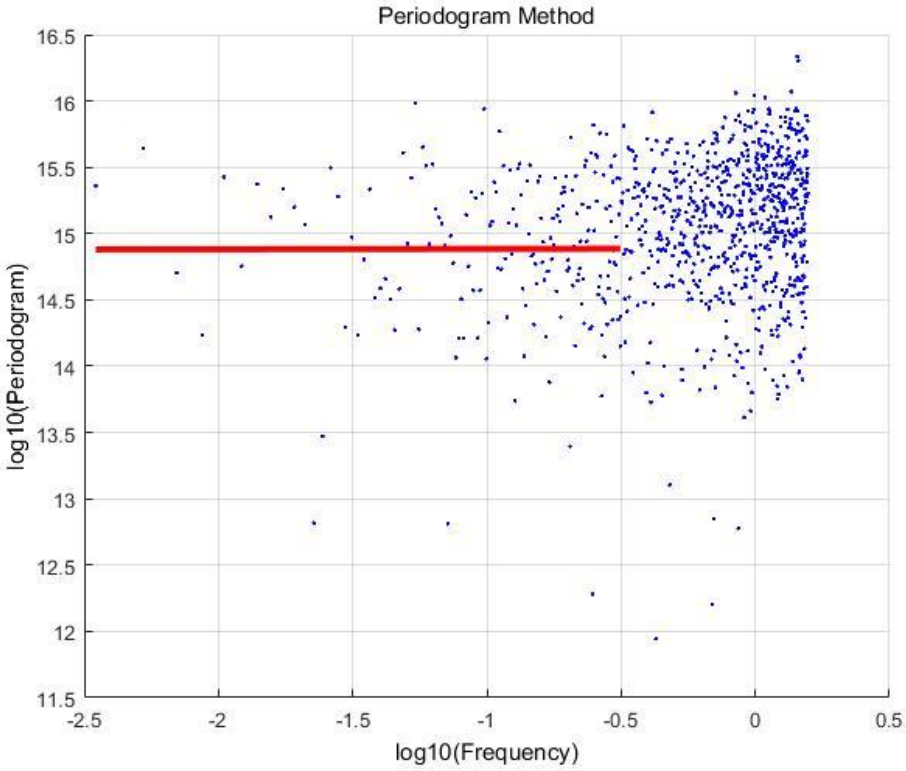

Figure 13. Periodogram for Busy Hour Byte Series

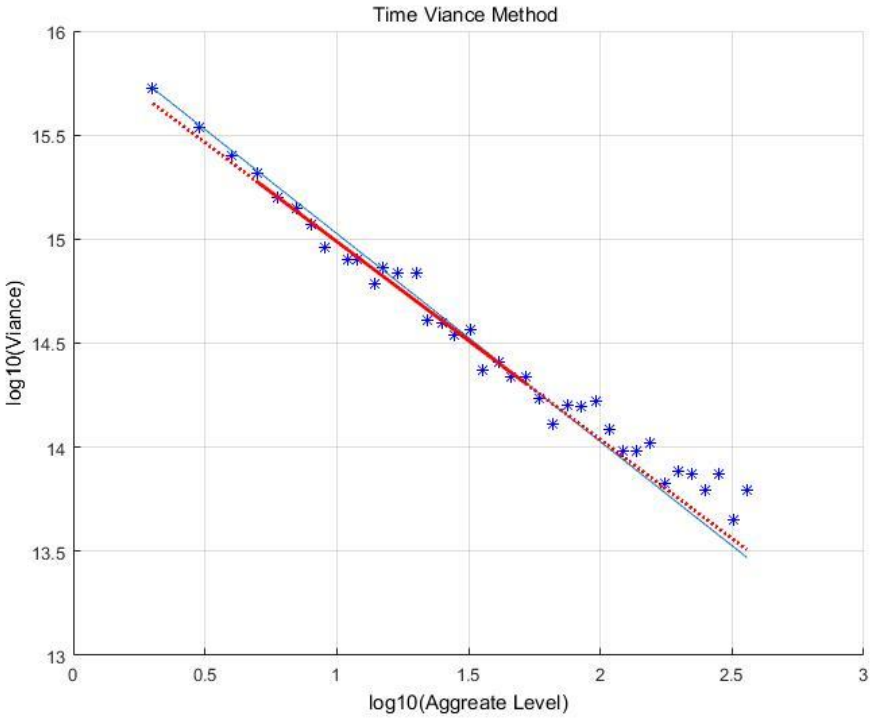

Figure 14. Time Variance for Busy Hour Byte Series

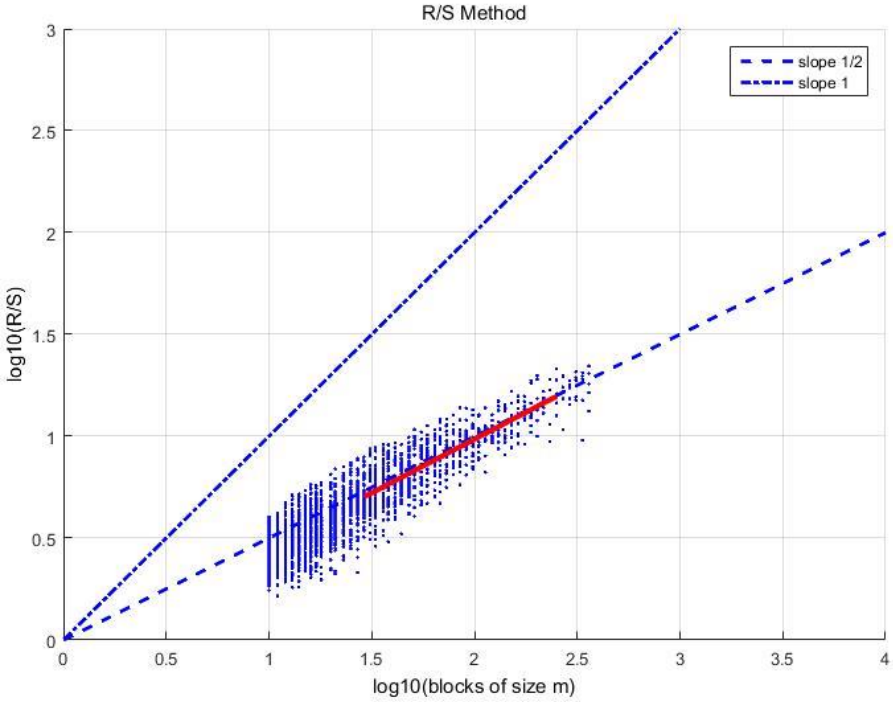

Figure 15. R/S for Busy Hour Byte Series

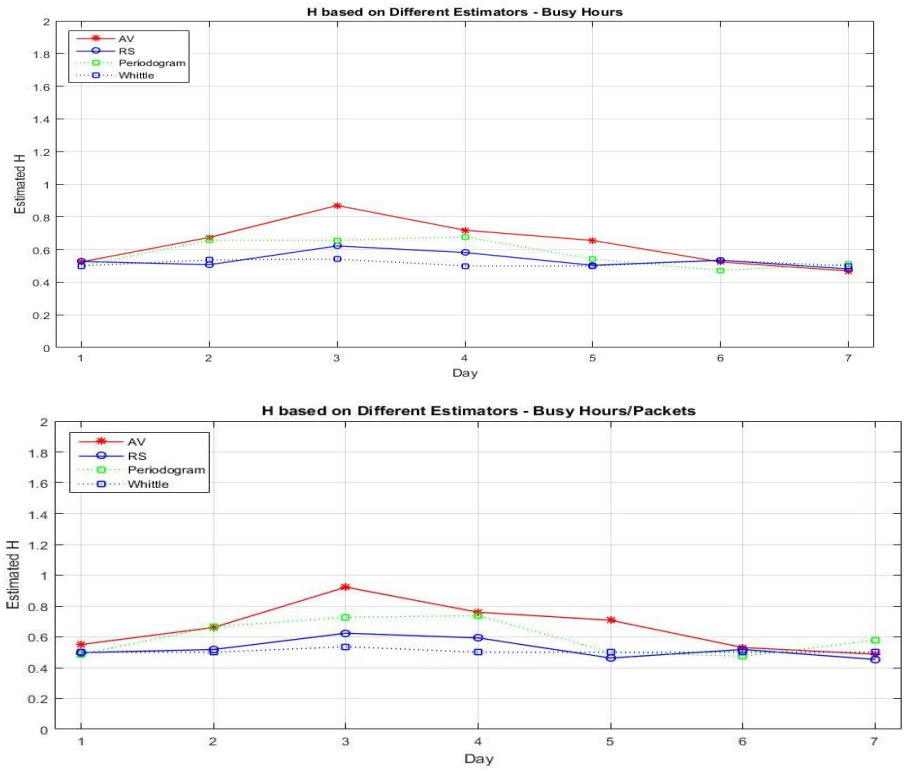

Figure 16. Hurst based on different estimators in busy hour for seven days a) Byte b) Packet counts

Low-Hour: Figure 17 shows the variation of Hurst parameter during seven days for byte and packet counts during low-hour. In contrast to busy hour, Hurst is slightly higher (around 0.02). Despite the ample evidence that Hurst parameter for time-series shows values H>0.75, we witness that overall PTK network traffic for investigated traces is short-range dependent. In addition, if results rely only on Whittle estimator, it can be observed that Hurst is within interval bounds of 0.461 and 0.573. This implies that these traces can be well approximated by classical models such as Poisson, because a H=0.5 is a pure Brownian Motion which shares the same independent and memoryless properties of Poisson.

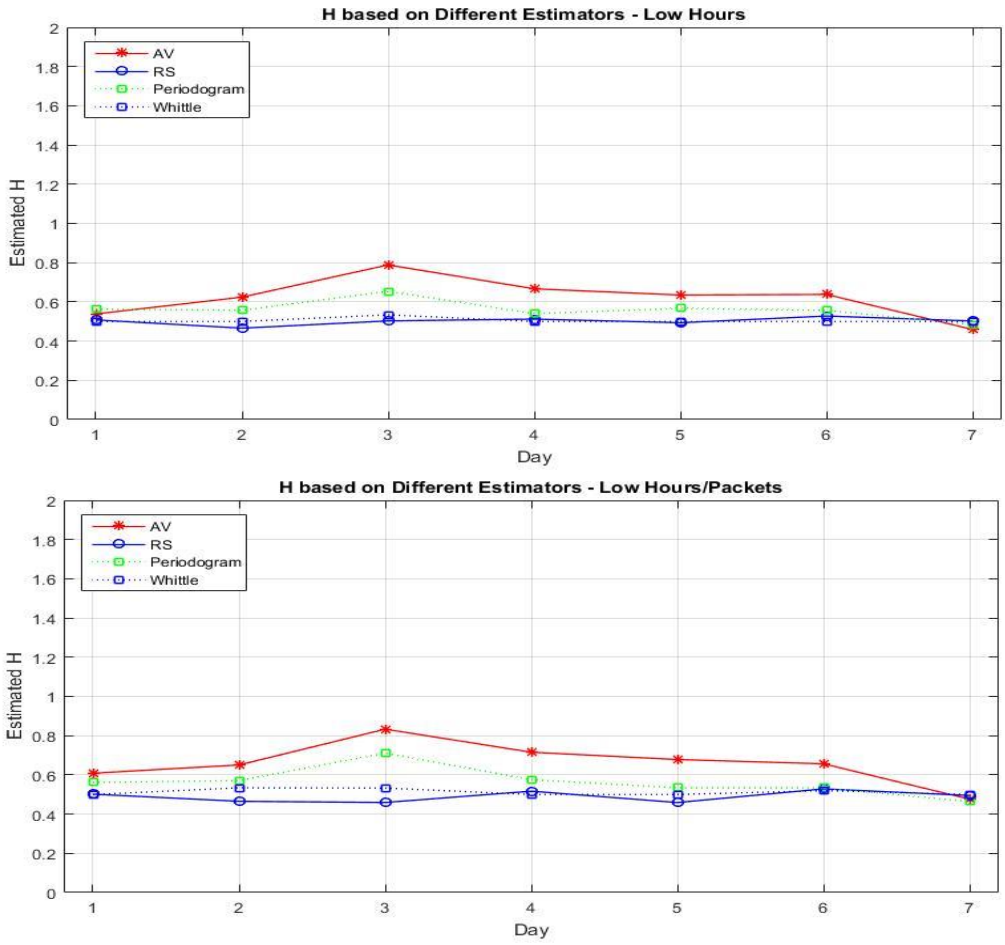

Figure 17. Hurst based on different estimators in low hour for seven days a) Byte b) Packet counts

4.2 TOPOLOGY CHARACTERISTICS FROM THE NETWORK TRAFFIC

In this Section results for underlying structure of network topology are shown based on the network traffic traces that are observed for 7 days. The structure of network topology is investigated with regards to node degree, network diameter and correlation between nodes. To calculate the degree of each node, we have extracted all pairs of IP addresses (source-destination pairs) from the communicating hosts using nfdump syntax. Because of an extensive amount of raw data during, in order to get information about all communicating hosts within one day, flows that last less than 1000 ms and those that transfer less than 100 bytes are filtered out. This has no

or negligible effect on number of IPs as the same IPs are shown in other sessions as well. Command used to filter source and destination IPs is done using nfdump syntax as:

nfdump -M Folder1:Folder2:Folder3 -R . -a -t 2015/12/29.00:00:00-2015/12/29.23:59:59 'duration > 10000 and bytes>100' -o "fmt:%sa,%da" >Day1.csv

which filters flows during 29 December 2015 and dumps into a csv format file only source and destination IP addresses. To get information about ASes, instead of extracting IP addresses, one can mine information from the AS field of a netflow (*%sas* and *%das* for source AS and destination AS respectively). Same procedure as per IP addresses follows with regards to labeling ASes and inferring information about connections between peers.

It is important to mention that source-destination IP addresses can interchange their positions. For example, one connection originated by IP_1 can be considered as IP_1_to_IP_2 and a connection originated from IP_2 can be considered as IP_2_to_IP_1. After that, a *python* script is utilized to replace IP addresses with a numbering label instead of IP address format. For example, IP address 10.10.10.10 is replaced with a number, let's say 1. Finally, *Matlab* was used to calculate the number of unique IP addresses and the number of connections each IP address has (undirected graph where one pair is considered one connection, despite which IP is source or destination of the connection).

4.2.1 Node Degree Distribution

This experiment shows the observed power-laws of the Internet topology inferred from daily traffic traces. Scaling parameter obeys the power-law behavior and lies between $2 < \alpha < 2.06$ for seven day traces.

Initial observation is done by using a log-log plot of PDF of node the degree k as can be seen in Figure 8 (left shows the histogram and right the PDF in log-log scale). After the initial observation, data is the power-law distribution is inspected visually for a fit of α and x_{min} calculated based on MLE.

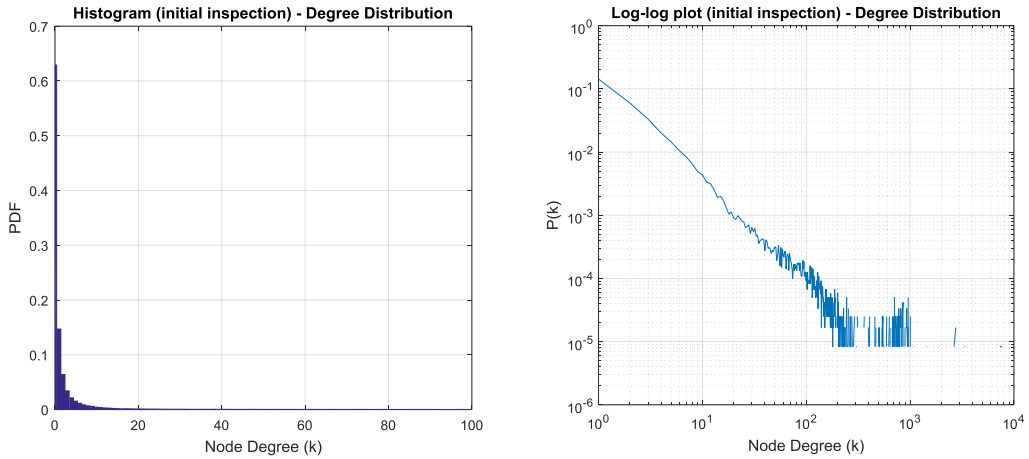

Figure 18. Initial inspection of power-law behaviour of Node degree

Visual inspection: Figure 19 presents the results graphically for node degree distribution for seven-day traces together with respective values of estimated α. Blue line represents our data points and red one the distribution fit using MLE, given our data. In addition, Figure 19 (last down-right) shows the stability of parameter α. Parameter x_{min} for all datasets is $x_{min} = 1$.

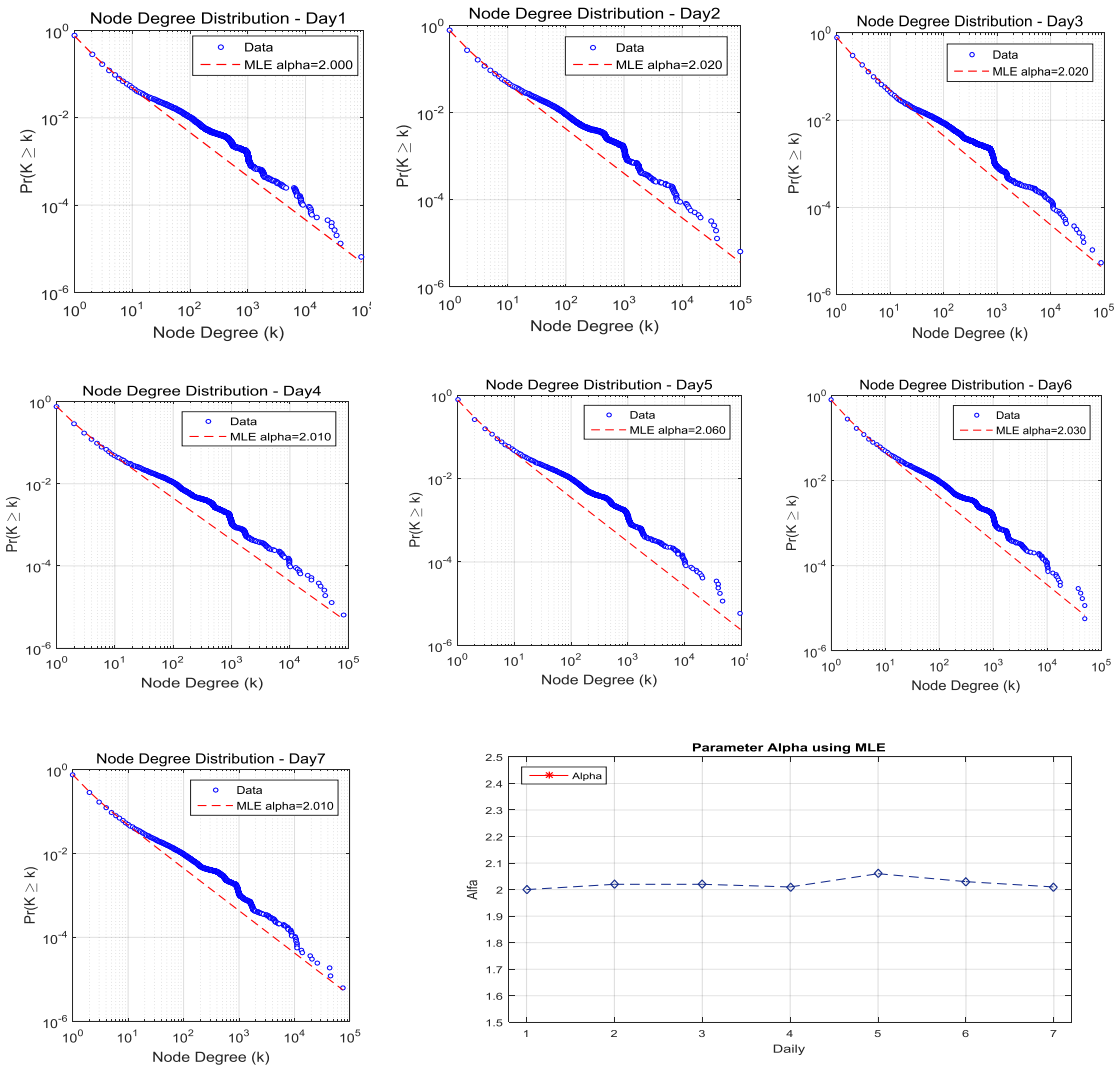

Figure 19. Node Degree distribution and exponential parameter alpha using MLE

4.2.2 Other Topology Characteristics

In the table below, a summary of measured metrics for seven days is given where n is the number of nodes, m is number of links, c(min) is the minimum degree, c(max) maximum degree, c(avg) is average degree, D is the farthest node distance, C is Global clustering coefficient and r is the assortativity coefficient.

Maximum number of connections a node has (node degree) is 136,447. This type of node when identified using online *WHOIS* databases, is associated to social media networks (Facebook for example).

Diameter of the network varies between 17 to 26 and despite the contradictions that it increases or shrinks with the number of nodes in a network, we do not see any pattern here. For example, for traces in Day5, that has both fewer number of nodes and links than traces for Day6, diameter is higher. On the other hand, if we compare traces for Day2 and Day3 we have the opposite. Diameter of traces for Day2 has fewer nodes but same value, 22.

Assortativity coefficient has values between -0.23 to -0.198 and shows similar that characterize Internet structure from other results. Meaning that low-degree nodes tend to connect more with high-degree nodes and the vice versa.

Table 4. Main Topology Characteristics

Day	n	m	c(min)	c(max)	c(avg)	D	r
1	151868	1566942	1	93180	20.63558	17	-0.22804
2	157417	1560825	1	99529	19.83045	22	-0.23314
3	188180	1859187	1	86866	19.75967	22	-0.20455
4	155321	1688403	1	83962	21.74082	23	-0.23454
5	171495	1745819	1	96424	20.36	26	-0.237074
6	175167	1841772	1	136447	21.02876	20	-0.215368
7	164153	1722727	1	132863	20.98928	20	-0.19837

4.2.3 Topology Characteristics at AS Level

At AS level, collected traffic data revealed information for BGP direct neighbor's ISP has. This implies that netflow data shows BGP peering configuration and number of connections between AS of PTK and others. We identified that PTK has 4 direct BGP peers, from which 3 are other

National Internet Service Providers to which PTK is connected through Internet Exchange Point (IXP) and the other is INIT7, a transit provider to reach other parts of global network.

Information about other ASes through which traffic flows cannot be revealed and basically netflow identifies those as ASN zero. Because of this limitation, metrics for characterizing topological structure at AS-level does not give any useful information. Despite that, we calculated node degree and assortativity to understand how are the four identified domains connected with domain of PTK. As expected, the top domain is the transit domain, INIT7, with degree in average of 933,498 for 7 day traces. Others, the local peers, have a degree in average of 311,558, 146,464 and 40,133 respectively. Degree, in this case means the number of unique connections each AS has within one day. Assortative at AS level for seven day traces is in average -0.105.

4.3 OTHER EXPERIMENTS

In addition to self-similar nature of internet traffic and main characteristics of topology, below are other experiments that are very common in literature and articles reviewed in Section 2. First one is the independence of packet size arrivals in order to show the validity of Poisson assumptions in smaller time-scales. Second experiment aims to check the rank distribution of top 10 IP and Port numbers for a period of 7 days.

4.3.1 Independence of Packet Size Arrivals

In this experiment we show the distribution of packet sizes of 1,000,000 consecutive packet arrivals from a snapshot of our dataset. Results show that packet size arrivals appear to be independent and consistent with findings in [4]. The independence of packet size arrivals is validated using three types of tests:

a) The autocorrelation function (ACF)
b) Box-Ljung Q-test
c) Visual inspection using scatter plot

Autocorrelation function: Figure 20 presents the autocorrelation function calculated for 200 lags of 1,000,000 packet arrivals for the packet size series. Packet sizes series consist of the packets as individual flows arrive, without aggregation. There is a trivial correlation for small lags, but it is close to zero with confidence level higher than 95%.

Box-Ljung Statistic: The Box-Ljung Q-test statistic is used to test for correlation of packet size series. The Box-Ljung statistic is defined as:

$$Q_k = n(n+2)\sum_{i=1}^{k}\frac{r_i^2}{n-i},$$

where r_i is the autocorrelation function for lags $1 \leq i \leq k$ and n the length of the series. The null hypothesis for this test is that the first k autocorrelations are jointly zero. Using Matlab, we computed the test for lags k 1 up to 200 and null hypothesis is rejected for any of k values with confidence level 95%. This indicates that packet size series can be considered independent and identically distributed.

Visual inspection: In Figure 21 we visually examined packet size series with itself (x_k, x_{k+1}). Scatter plots are a graphical representation of relationship between two quantitative variables or can be used to test independence of a one variable when plotted with shifted version of itself. In X-axis we have the size of packets at time k and in Y-axis we have the packet size at $k+1$. Because the plot is symmetric and no relationship can be observed, we can conclude that packet size series are independent.

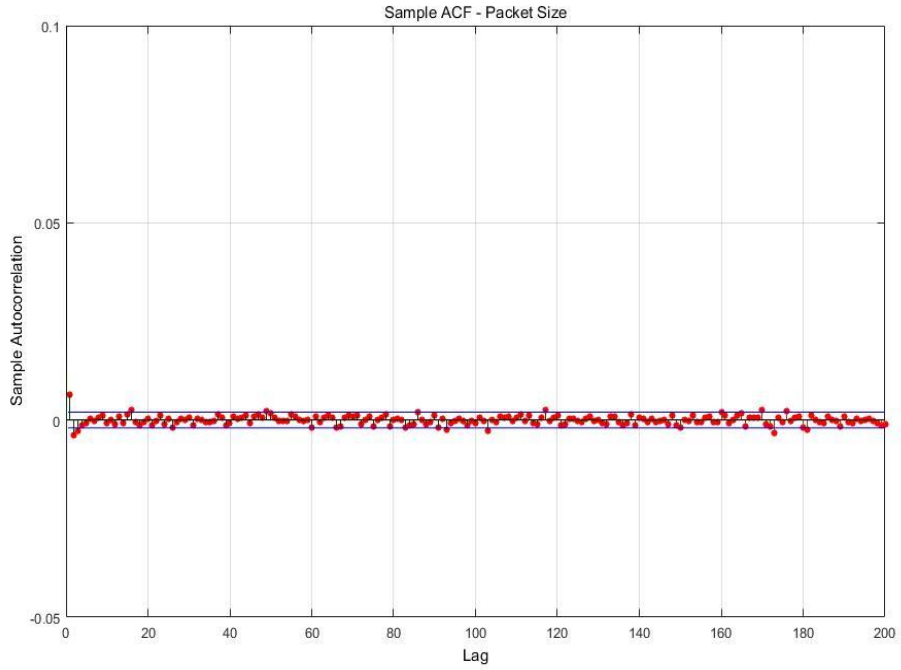

Figure 20. Sample ACF of packet sizes. Correlation Coefficients are within 95% C.I.

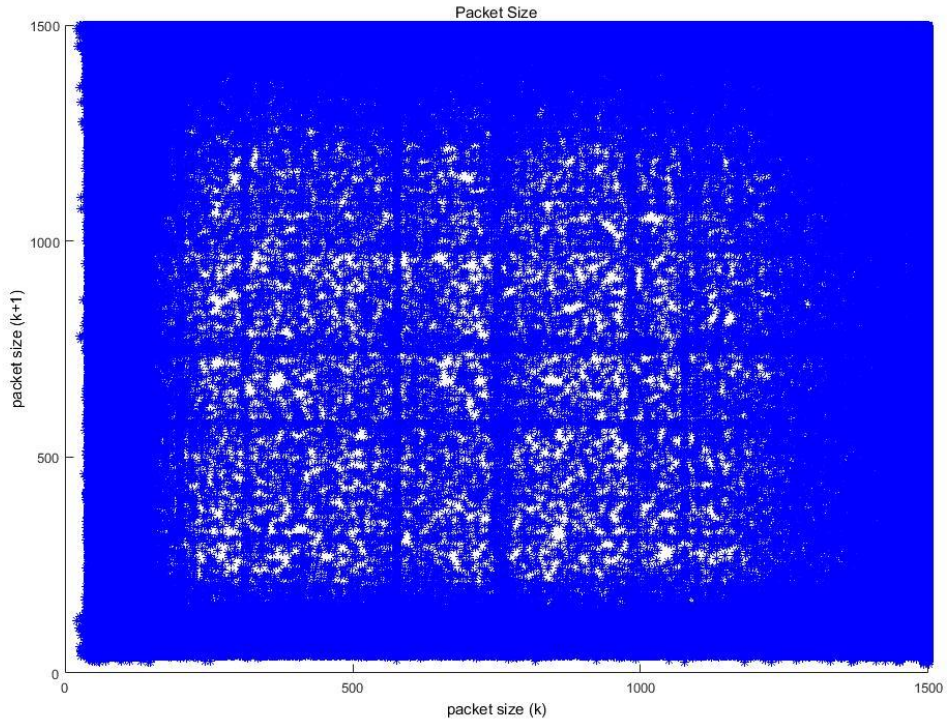

Figure 21. Scatter plot of 1,000,000 consecutive packet sizes.

4.3.2 Distribution of IP addresses and Port Numbers

This experiment aims to check for temporal properties of the dataset under study. We show that distribution of frequencies for port numbers and IP addresses do not change over time. To do so, we evaluate daily measurements for most active destination IP addresses and port numbers for the upstream traffic and calculate parameter α where data can fit an exponential form of $y \sim Cx^{-\alpha}$. We show that the number of days have very little impact in the distribution of most visited port numbers and IP addresses. Certainly, our calculations are for only seven days, but we have daily observations from a very rich number of sources and we do not expect that it changes if we calculate for longer period. Instead variation is more related to daily periodicity rather than number of days itself. It is important to mention that Network Address Translation (NAT) affects the accuracy of the results as it cover-ups IP addresses and Port numbers.

Parameter α during 7-day observations for most visited port numbers is $\alpha = 1.9420 \pm 0.0592$ with 90% confidence level, using t-test. And, α for most visited IP addresses is $\alpha = 0.6961 \pm 0.0627$. It would be interesting to see for spatial diversity and variation of parameter α. A study in [34] claims that distribution of port numbers and IP addresses varies from place to place. Data collected from this study considers university institutions across a country with similar properties (such as number of students, faculties and staff).

In addition, we have shown the fitted data to the exponential distribution of the form discussed above and have checked the accuracy using:

a) Visual inspection of distribution fit

b) Residuals plot

c) Coefficient of determination r^2.

Visual inspection of distribution fit: Figure 22 and Figure 23 shows the best-fit parameter together with least square in logarithmic form and cumulative distribution. In log-log plot we can observe a piecewise linear line, which is an indication of heavy-tailed distribution.

Residuals plot: In order to show that most active port numbers and IP addresses can be modeled using exponential distribution of a form $y \sim Cx^{-\alpha}$, we can determine residuals. Residuals are plotted in Figure 24 as a result of the difference between observed value of the response variable and the value of the response value predicted by linear regression. For negative values, prediction is said to be lower than observed values and for positive it is higher than observed values.

Coefficient of determination r^2*:* Coefficient of determination measures the proportion of variation explained by the independent variable in a regression model and is given by $r^2 = \frac{S_{\hat{y}}^2}{S_y^2}$ where $S_{\hat{y}}^2$ is the variance of the predicted values and S_y^2 is the variance of the calculated values from the measurements. Larger the r^2 the better model can explain variations and can take values from zero to one, $0 \leq r^2 \leq 1$. For one day measurements, coefficient of determination is $r^2 = 0.9657$ for port numbers $r^2 = 0.9235$ and for IP addresses.

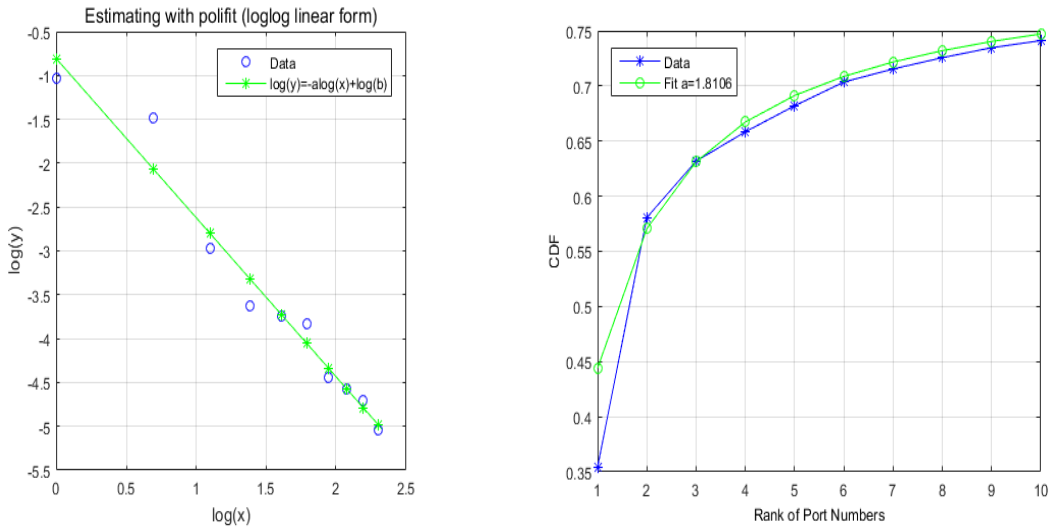

Figure 22. Cumulative distribution function of most popular port numbers

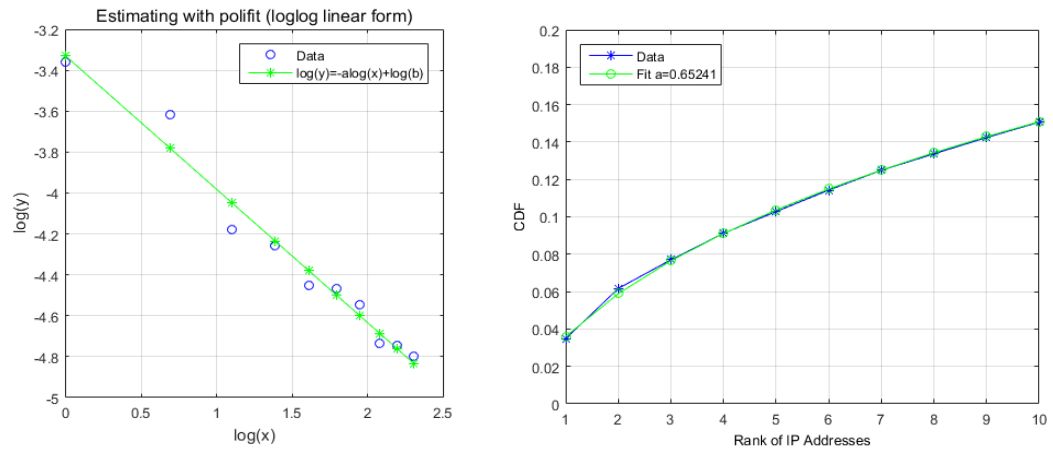

Figure 23. Cumulative distribution function of most popular IP addresses for upstream traffic in one day.

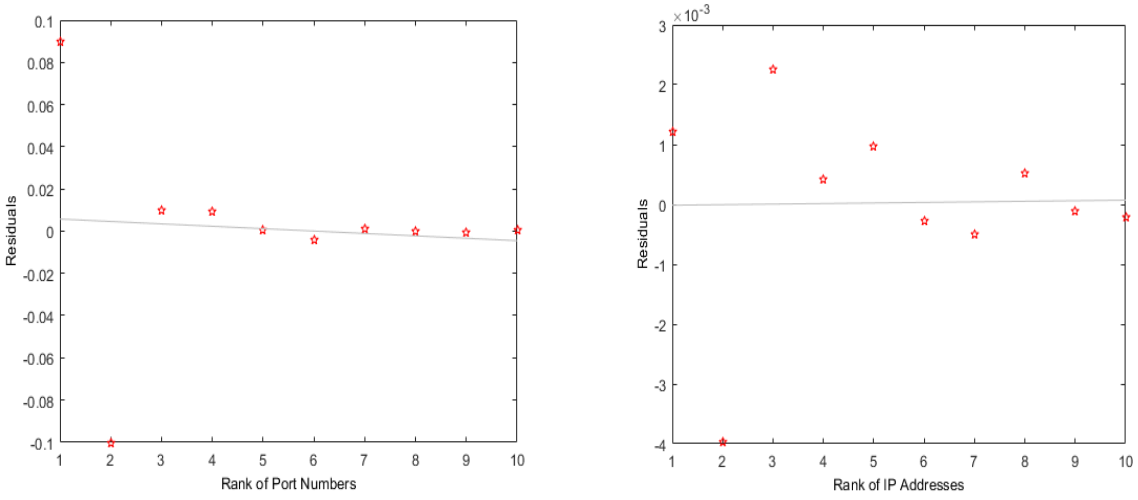

Figure 24. Residuals from data of one-day port numbers and IP addresses

5.0 CONCLUSIONS

In this thesis work, network traffic at PTK was captured and analyzed. It is shown that Internet traffic is characterized with daily periodicity and despite the aggregation in large time-scales (10 s and 1 s), it preserves the *burstiness*.

The downstream traffic at PTK is shown to be self-similar but not long-range dependent with Hurst parameter estimated to $H \cong 0.53$. Results suggest that the changes in overall network traffic have pushed the Internet in the general direction of better-behaved traffic models (i.e., the Poisson assumption) or at least not in the direction of sophisticated models. The value of Hurst depends on estimation method, bin size of measurements and the time measurements are taken. All estimators are easily tricked by stationarity and instead shorter intervals should be taken into consideration (seconds or minutes and not hours). In addition, the number of samples should be increased while aggregating the time series into non-overlapping blocks at orders of *milliseconds*. Also, as congestion is more problematic and evident in access network, results are not valid in that case and the analysis of traffic LRD properties are more important in that part of network. Most of backbones are overprovisioned and thus effect of traffic *busrtiness* in backbone and core is much less of interest.

Then, we have used graph metrics to characterize topological properties interfered from network traffic at router (IP) and AS level. At IP level, degree distribution obeys *power-law* distribution with shaping parameter $2 < \alpha < 2.06$. Topology of PTK network inferred from traffic traces shows dissasortative behavior with assortativity coefficient between -0.23 to -0.198. And, diameter of the network varies between 17 to 26 and despite the contradictions that it increases or shrinks with the number of nodes in a network, results showed that there is no pattern from seven-day traces. At AS level, information about BGP peering configuration of an

ISP can be inferred from Netflow and their importance can be identified based on the number of connections.

Traffic was also investigated with respect to which packet sizes and protocols are more present and their share. It is shown that most of the packets are small and TCP is still the largest transport protocol.

Finally, it is shown that packet size arrivals at milliseconds granularity show no correlation between each other and thus indicating that they can be considered independent and identically distributed. And, top 10 IP addresses and Port numbers are stable and do not change during the course of 7 days in terms of distribution.

5.1 FUTURE WORK

As becoming more familiar with the techniques of mining network traffic data and concepts of network traffic in general, I consider this work as a first step toward a more rigorous analysis, including implications of self-similarity and metrics to describe graphs.

First thing that should be done is to find an accurate technique to extract inter-arrival times and test if they are mutually independent and identically distributed. Thus, to validate hypothesis that Poisson can coexist together with LRD. Second, in this regard is to extract data at finer granularities and shorter periods, where stationarity is less illusive.

As the structure of Internet traffic has changed, in terms of applications, it is very important to study the Internet video in particular and to show whether the session lengths (ON times) and video files sizes in Internet are the same with respect to distribution.

Other work that is in progress includes:

1. Detecting network anomalies using data mining techniques such as PARAFAC analysis. In particular, port-scanning attack is shown to be easily detected and isolated using multidimensional information as shown in [35].
2. Network topology visualization from where we can identify the role of the nodes in a network, observe the sparseness of nodes and growth together with preferential attachment phenomena over the time. In the Figure 25, it has been extracted communicating nodes from Netflows for 15 minute interval and using Gephi software an

undirected graph is constructed. It shows that only few nodes are highly connected, while the majority have couple of edges. This is a visual way of observing a power-law behavior of placement of nodes in a network.

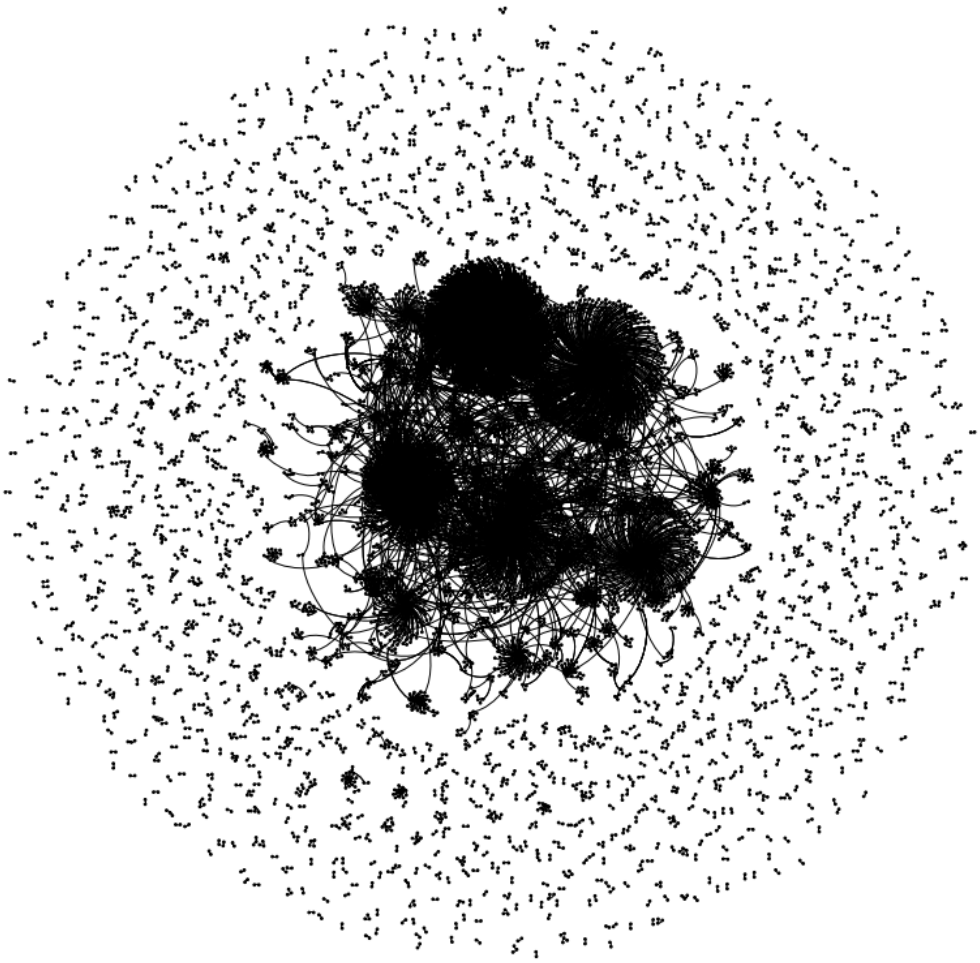

Figure 25. Network topology visualization sample

APPENDIX A

CODE AND FILES FOR EXPERIMENTS

This Appendix shows the codes that are used to conduct the experiments in this paper. In addition, together with the codes, it shows how the files are (pre)processed and what are the filenames of the processed raw data in order to use them for later purposes.

1. Packets and Bytes arrived at each minute for 24 h interval (Figure 5, 6, 7, 8)

```
#This script extracts packets and bytes that arrive at each minute together with Timestamp. It searches through all
#files in folders and for each folder generates a file with a name #folder+Experiment1And2.csv. Required packages
#before executing this script are also included.
import numpy
import pandas as pd
from pandas import Series, DataFrame
import pynfdump
import os
import sys
from datetime import date
import subprocess

folders = [name for name in os.listdir(".") if os.path.isdir(name)]
for folder in folders:
        py2 = open(folder+'Experiment1And2.csv','w')
        d=pynfdump.Dumper()
        d.set_where(dirfiles=folder) #folders that one assumes flows of that interval are found in.
#For figures 7 and 8, where only downstream traffic is queried, the below line becomes:
# records=d.search(query="dst NET 213.163.125.0/19 or dst NET 178.175.0.0/17 or dst NET 185.47.188.0/22 and
#not (src NET 213.163.125.0/19 or src NET  178.175.0.0/17  or src NET 185.47.188.0/22)")
        records=d.search(query="src NET 213.163.125.0/19 or src NET 178.175.0.0/17 or src NET
185.47.188.0/22 and (dst NET 213.163.125.0/19 or dst NET  178.175.0.0/17  or dst NET 185.47.188.0/22)")
        tStart=[]
        bytes=[]
        packets=[]
        for r in records:
                tStart.append(r['first'])
                bytes.append(r['bytes'])
```

```
            packets.append(r['packets'])
        bytesSeries = pd.DataFrame({'BYTES' : bytes, 'tStart':tStart, 'PACKETS':packets})
        bytesSeries['tStart'] = pd.to_datetime(bytesSeries['tStart'], format='%Y-%m-%d %H:%M:%S.%f')
        data = bytesSeries.groupby(['tStart'])
        data2 = data.sum()
        data3=data2.resample("1T", how='sum')
        data3.to_csv(py2, sep=',', encoding='utf-8')
        py2.close()
```

```
#This script plots only the 24h interval specified as '2015-12-31 12:00:00':'2015-12-31 12:59:00'.
#The same is used for byte and packet counts
import pandas as pd
import dateutil
import csv
import re
from pandas import DataFrame, Series
import numpy as np
import matplotlib.pyplot as plt

data20 = pd.read_csv("Local Traffic Bytes and Packets.csv") #Careful with the file name.
data20[tStart] =  pd.to_datetime(data20[tStart], format='%Y-%m-%d %H:%M:%S')
data30 = DataFrame(data20)
data40 = data30.groupby([tStart])
data50 = data40.sum()
data60 = DataFrame(data50[BYTES]) #change to <PACKETS> to plot the packet counts
data70=data60['2015-12-31 12:00:00':'2015-12-31 12:59:00']
Data70.plot(colormap='winter', label='Series', style='k')
plt.grid()
manager = plt.get_current_fig_manager()
manager.window.showMaximized()
plt.savefig('ByteCounts1MBin.jpeg, format=jpeg, dpi=800) # Change the Name to <PacketCounts1MBin>
plt.show()
```

```
#Compute min, max for Byte and Packet counts
min(data70['PACKETS'])
max(data70[' PACKETS'])
min(data70['BYTES'])
max(data70[' BYTES'])
```

2. Protocols proportion (Figure 7 and 8)

```
# To simply show the present protocols ordered by volume (bytes) in a data set, we used the nfdump syntax
nfdump -M Source1/20151229_0005:Surce2/20151229_0005:Source3/2015:1229_0005 -R . -s proto/bytes
```

3. Packet size (Figure 9 and 10)

```
#To get the packet size arrivals for one day, December 29 2015.
nfdump -M Day1:D1P1:D1P2:D1S1P1:D1S1P2:D2P1:D2P2:D2P3:D2S1P1:D2S1P2:D2S1P3 -R . -a -t
2015/12/29.15:00:00-2015/12/29.18:00:00 -o "fmt:%ts,%bpp" | sort > PacketSizesDec29BusyHours.csv
```

```
#Matlab code to see the frequency of packet sizes and to relative frequency plot
unique_k = unique(Y);
hist(Y,unique(unique_k));
min(Y);
max(Y);
ylabel('cumulative percentage');
xlabel('packet size (bytes)');
title({'Size of Packets arrived between 15:00:00 to 18:00:00, Dec 29 2015. Total packets: 2483248 Min=21,
Max=1500'});
grid on;
#Matlab code to plot the empirical cumulative distribution, ecdf
ecdf(Y);
ylabel('cumulative percentage');
xlabel('packet size (bytes)');
title({'Cumulative distribution of packet sizes'; '15:00:00 to 18:00:00, Dec 29 2015. Total packets: 2483248'});
grid on;
```

```
#10 most frequent Packet sizes in the dataset
load PacketSizesDec29BusyHoursOnlyPackets.csv;
Y=PacketSizesDec29BusyHoursOnlyPackets;
X = unique(Y);
for j = 1:length(X)
   Z =X(j);
x = strmatch(Z, Y, 'exact');
ind(j) = length(x);
end
Packets = sort(ind,'descend');
% get the top ten packet sizes
for n = 1:10
   [pack, inde] = find(ind==Packets(n))
   X(inde)
end
```

4. Periodicity and Self-similarity (Figure 11)

```
#Extract the byte time-series from raw data in 10s granularity for upstream traffic that is generated by
#users in two subnets of PTK: NET 178.175.0.0/19 NET 213.163.96.0/19
folders = [name for name in os.listdir(".") if os.path.isdir(name)]
for folder in folders:
        py2 = open(folder+'.csv','w')
        d=pynfdump.Dumper()
        #ip = "178.175.27.127" and "17.154.66.73"
        d.set_where(dirfiles=folder)
        records=d.search(query='src NET 178.175.0.0/19 or src NET 213.163.96.0/19') ## these are two networks
#of PTK. Confidential***
        bytes=[]
        time=[]
        for r in records:
                bytes.append(r['bytes'])
                time.append(r['first'])
        bytesSeries = pd.DataFrame({'BYTES' : bytes, 'TimePeriod' : time})
```

```
        bytesSeries['TimePeriod'] =  pd.to_datetime(bytesSeries['TimePeriod'], format='%Y-%m-%d
%H:%M:%S')
        data = bytesSeries.groupby('TimePeriod')
        data2 = data.sum()
        data3=data2.resample("10s", how='sum') #aggregate in 10 s bins.
        data3.to_csv(py2, sep=',', encoding='utf-8')
        py2.close()
```

```
#Pot 11b) and c).
data20 = pd.read_csv("PBUWeek1.csv") #open the file that was a result of concatenated files in previous script
data20['TimePeriod'] =  pd.to_datetime(data20['TimePeriod'], format='%Y-%m-%d %H:%M:%S')
data30 = DataFrame(data20)
data40 = data30.groupby(['TimePeriod'])
data50 = data40.sum()
data60 = DataFrame(data50)
#Resample now inevery 10s
data70=data60['2016-01-02 00:00:00':'2016-01-02 23:59:50'] #c)
data70=data70.resample("10s", how='sum')
data71=data70['2016-01-02 11:00:00':'2016-01-02 14:59:50'] #b)
data71=data71.resample("10s", how='mean')
data71=DataFrame(data71)
data72=DataFrame(data71, index=data70.index)

#One day bytes in 1s granularity extracted using the same script as previous, only sampling done in 1s.
data200 = pd.read_csv("Day5Bytes.csv")
data200['TimePeriod'] =  pd.to_datetime(data200['TimePeriod'], format='%Y-%m-%d %H:%M:%S.%f')
data300 = DataFrame(data200)
data500 = data300.groupby(['TimePeriod'])
data500 = data500.sum()
data600 = DataFrame(data500)
data650 = data600['2016-01-02 11:00:00':'2016-01-02 14:59:50'] #d)
#Create Figure and plot.
fig, axes = plt.subplots(nrows=2, ncols=2)
data75.plot(ax=axes[0,0],colormap='winter', label='Series',kind='area')
data70.plot(ax=axes[1,0],colormap='winter', label='Series',kind='area')
data71.plot(ax=axes[0,1],colormap='winter', label='Series',kind='area')
data650.plot(ax=axes[1,1],colormap='winter', label='Series',kind='area')
plt.grid()
manager = plt.get_current_fig_manager()
manager.window.showMaximized()
plt.savefig('FracatBehaviour.jpeg', format='jpeg', dpi=800)
plt.show()
```

5. Time-series for calculating Hurst Parameter and Methods

```
import numpy
import pandas as pd
from pandas import Series, DataFrame
import pynfdump
import os
import sys
from datetime import date
import subprocess

folders = [name for name in os.listdir(".") if os.path.isdir(name)]
```

```
for folder in folders:
        py2 = open(folder+'LRD.csv','w')
        d=pynfdump.Dumper()
        d.set_where(dirfiles=folder) #folders that one assumes flows of that interval are found in.
records=d.search(query="dst NET 213.163.125.0/19 or dst NET 178.175.0.0/17 or dst NET 185.47.188.0/22")
        tStart=[]
        bytes=[]
        packets=[]
        for r in records:
                tStart.append(r['first'])
                bytes.append(r['bytes'])
                packets.append(r['packets'])
        bytesSeries = pd.DataFrame({'BYTES' : bytes, 'tStart':tStart, 'PACKETS':packets})
        bytesSeries['tStart'] = pd.to_datetime(bytesSeries['tStart'], format='%Y-%m-%d %H:%M:%S.%f')
        data = bytesSeries.groupby(['tStart'])
        data2 = data.sum()
        data3=data2.resample("1S", how='sum') #Bins of 1 seconds.
        data3.to_csv(py2, sep=',', encoding='utf-8')
        py2.close()
```

#Compute (Estimate) Hurst parameter using the code below (code in [30]) or directly from Selfis Tool
#(download link in http://alumni.cs.ucr.edu/~tkarag/Selfis/Selfis.html)
#Files that need/are uploaded containing time-series of bytes and packets.
#All these files are extracted from the main file that contains a 7 day time-series of byte and packet counts:
 # BytesAndPackets_1Second_Downstream.csv

```
%Files loaded for Bytes during Busy hour!
load Busy_Bytes_Dec29_LRD.csv;
a=Busy_Bytes_Dec29_LRD';
load Busy_Bytes_Dec30_LRD.csv;
b=Busy_Bytes_Dec30_LRD';
load Busy_Bytes_Dec31_LRD.csv;
c=Busy_Bytes_Dec31_LRD';
load Busy_Bytes_Jan1_LRD.csv;
d=Busy_Bytes_Jan1_LRD';
load Busy_Bytes_Jan2_LRD.csv;
e=Busy_Bytes_Jan2_LRD';
load Busy_Bytes_Jan3_LRD.csv;
f=Busy_Bytes_Jan3_LRD';
load Busy_Bytes_Jan4_LRD.csv;
g=Busy_Bytes_Jan4_LRD';
%Files loaded for Bytes during Low hour!
load Low_Bytes_Dec29_LRD.csv;
aa=Low_Bytes_Dec29_LRD';
load Low_Bytes_Dec30_LRD.csv;
bb=Low_Bytes_Dec30_LRD';
load Low_Bytes_Dec31_LRD.csv;
cc=Low_Bytes_Dec31_LRD';
load Low_Bytes_Jan1_LRD.csv;
dd=Low_Bytes_Jan1_LRD';
load Low_Bytes_Jan2_LRD.csv;
ee=Low_Bytes_Jan2_LRD';
load Low_Bytes_Jan3_LRD.csv;
ff=Low_Bytes_Jan3_LRD';
load Low_Bytes_Jan4_LRD.csv;
```

```
gg=Low_Bytes_Jan4_LRD';
%Files loaded for Packets during Busy hour!
load Busy_Packets_Dec29_LRD.csv;
pa=Busy_Packets_Dec29_LRD';
load Busy_Packets_Dec30_LRD.csv;
pb=Busy_Packets_Dec30_LRD';
load Busy_Packets_Dec31_LRD.csv;
pc=Busy_Packets_Dec31_LRD';
load Busy_Packets_Jan1_LRD.csv;
pd=Busy_Packets_Jan1_LRD';
load Busy_Packets_Jan2_LRD.csv;
pe=Busy_Packets_Jan2_LRD';
load Busy_Packets_Jan3_LRD.csv;
pf=Busy_Packets_Jan3_LRD';
load Busy_Packets_Jan4_LRD.csv;
pg=Busy_Packets_Jan4_LRD';
%Files loaded for Packets during Low hour!
load Low_Packets_Dec29_LRD.csv;
paa=Low_Packets_Dec29_LRD';
load Low_Packets_Dec30_LRD.csv;
pbb=Low_Packets_Dec30_LRD';
load Low_Packets_Dec31_LRD.csv;
pcc=Low_Packets_Dec31_LRD';
load Low_Packets_Jan1_LRD.csv;
pdd=Low_Packets_Jan1_LRD';
load Low_Packets_Jan2_LRD.csv;
pee=Low_Packets_Jan2_LRD';
load Low_Packets_Jan3_LRD.csv;
pff=Low_Packets_Jan3_LRD';
load Low_Packets_Jan4_LRD.csv;
pgg=Low_Packets_Jan4_LRD';
```

```
#Time Variance Mathod
function H = aggvar(sequence,isplot)
%
% Time variance method
%
% Inputs:
%     sequence: the input sequence for estimate
%     isplot: whether display the plot. without a plot if isplot equal to 0
% Outputs:
%     H: the estimated hurst coeffeient of the input sequence
if nargin == 1
   isplot = 0;
end
N = length(sequence);
mlarge = floor(N/5);
M = [floor(logspace(0,log10(mlarge),50))];
M = unique(M(M>1));
n = length(M);
cut_min = ceil(n/10);
cut_max = floor(6*n/10);
V = zeros(1,n);
for i = 1:n
   m = M(i);
   k = floor(N/m);
```

```
    matrix_sequence = reshape(sequence(1:m*k),m,k);
    V(i) = var(sum(matrix_sequence,1)/m);
end
x = log10(M);
y = log10(V);
y1 = -x+y(1)+x(1);
X = x(cut_min:cut_max);
Y = y(cut_min:cut_max);
p1 = polyfit(X,Y,1);
Yfit = polyval(p1,X);
yfit = polyval(p1,x);
beta = -(Yfit(end)-Yfit(1))/(X(end)-X(1));
H = 1-beta/2;
if isplot ~= 0
    figure,hold on;
    plot(x,y,'b*');
    h = plot(x,y1);
    plot(X,Yfit,'r-','LineWidth',2);
    plot(x(1:cut_min),yfit(1:cut_min),'r:','LineWidth',2);
    plot(x(cut_max:end),yfit(cut_max:end),'r:','LineWidth',2);
    xlabel('log10(Aggreate Level)','FontSize',10,'FontWeight','normal','Color','k');
    ylabel('log10(Viance)','FontSize',10,'FontWeight','normal','Color','k');
    T=title('Time Viance Method  - Day Six');
    set(T,'FontSize',12,'FontWeight','normal');
    %str = {'Hurst Estimated: ', num2str(H)};
    %text(1,16,str)
    grid on;
end
```

```
function H = RS(sequence,isplot)
%
% R/S method.
%
% Inputs:
%     sequence: the input sequence for estimate
%     isplot: whether display the plot. without a plot if isplot equal to 0
% Outputs:
%     H: the estimated hurst coeffeient of the input sequence
if nargin == 1
    isplot = 0;
end
N = length(sequence);
dlarge = floor(N/5);
dsmall = max(10,log10(N)^2);
D = floor(logspace(log10(dsmall),log10(dlarge),50));
D = unique(D);
n = length(D);
x = zeros(1,n);
y = zeros(1,n);
R = cell(1,n);
S = cell(1,n);
for i = 1:n
    d = D(i);
    m = floor(N/d);
    R{i} = zeros(1,m);
    S{i} = zeros(1,m);
```

```
    matrix_sequence = reshape(sequence(1:d*m),d,m);
    Z1 = cumsum(matrix_sequence);
    Z2 = cumsum(repmat(mean(matrix_sequence),d,1));
    R{i} = (max(Z1-Z2)-min(Z1-Z2));
    S{i} = std(matrix_sequence);
    if min(R{i})==0 || min(S{i}) ==0
       continue;
    end
    x(i) = log10(d);
    y(i) = mean(log10(R{i}./S{i}));
end
% fit a line with middle part of sequence
index = x~=0;
x = x(index);
y = y(index);
n2 = length(x);
cut_min = ceil(3*n2/10);
cut_max = floor(9*n2/10);
X = x(cut_min:cut_max);
Y = y(cut_min:cut_max);
p1 = polyfit(X,Y,1);
Yfit = polyval(p1,X);
H = (Yfit(end)-Yfit(1))/(X(end)-X(1));

if isplot ~= 0
    figure,hold on;
    bound = ceil(log10(N));
    axis([0 bound 0 0.75*bound]);
   temp = (1:n).*index;
    index = temp(index);
    for i = 1:n2
       plot(x(i),log10(R{index(i)}./S{index(i)}),'b.');
    end
    x = linspace(0,bound,10);
    y1 = 0.5*x;
    y2 = x;
    h1 = plot(x,y1,'b--','LineWidth',2);
    h2 = plot(x,y2,'b-.','LineWidth',2);
    plot(X,Yfit,'r-','LineWidth',3);
    legend([h1,h2],'slope 1/2','slope 1',4)
    xlabel('log10(blocks of size m)','FontSize',10,'FontWeight','normal','Color','k')
    ylabel('log10(R/S)','FontSize',10,'FontWeight','normal','Color','k')
    T=title('R/S Method - One Week');
    set(T,'FontSize',12,'FontWeight','normal');
    %text(1,3,str)
end
```

```
function H = per(sequence,isplot)
%
%  periodogram
% Inputs:
%     sequence: the input sequence for estimate
%     isplot: whether display the plot. without a plot if isplot equal to 0
% Outputs:
%     H: the estimated hurst coeffeient of the input sequence
 if nargin == 1
```

```
    isplot = 0;
end
n = length(sequence);
Xk = fft(sequence);
P_origin = abs(Xk).^2/(2*pi*n);
P = P_origin(1:floor(n/2)+1);
x = log10((pi/n)*[2:floor(0.5*n)]);
y = log10(P(2:floor(0.5*n)));
% Use the lowest 20% part of periodogram
X = x(1:floor(length(x)/5));
Y = y(1:floor(length(y)/5));
p1 = polyfit(X,Y,1);
Yfit = polyval(p1,X);
H = (1-(Yfit(end)-Yfit(1))/(X(end)-X(1)))/2;
if isplot ~= 0
    figure,clf,hold on;
    plot(x,y,'b.');
    plot(X,Yfit,'r-','LineWidth',3);
    xlabel('log10(Frequency)','FontSize',10,'FontWeight','normal','Color','k')
    ylabel('log10(Periodogram)','FontSize',10,'FontWeight','normal','Color','k')
    T=title('Periodogram Method - Norma Hour Day Five');
    set(T,'FontSize',12,'FontWeight','normal');
    %str = {'Hurst Estimated: 1.122'}; //here is written parameter estimated by SELFIS (for more acurate res)
    %text(-0.5,17,str)
    grid on
end
```

```
#Figure 16 and 17, using the data from Table 5 for vectors AV, RS, PG, WT.
%Comparing Hurst Parameters (Busy Hours)
AV=[]
RS=[]
PG=[]
WT=[]
plot(AV,'-r*','LineWidth',1,'MarkerSize',6)
hold on
plot(RS,'-bo','LineWidth',1,'MarkerSize',6)
hold on
plot(PG,':gs','LineWidth',1,'MarkerSize',6)
hold on
plot(WT,':bs','LineWidth',1,'MarkerSize',6)
ylim([0 2])
xlim([0.8,7.2])
grid on
legend('AV','RS','Periodogram','Whittle','Location','northwest')
ylabel('Estimated H'); xlabel('Daily'); title('H based on Different Estimators - Busy Hours')
```

Table 5. Hurst computed (up) using Selfis tool and (down) using Matlab codes

SELFIS Java TOOL

		Busy Hour					Low Hour				
		Aggregate Variance	R/S	Periodogram	Whittle and C.I. @ 95 C.L		Aggregate Variance	R/S	Periodogram	Whittle and C.I. @ 95 C.L	
Day1	Bytes	0.515	0.156	0.549	0.5	0.461-0.538	0.654	0.37	0.596	0.5	0.461-0.538
	Packets	0.615	0.138	0.499	0.5	0.461-0.538	0.702	0.305	0.578	0.5	0.461-0.538
Day2	Bytes	0.663	0.179	0.587	0.5	0.461-0.538	0.698	0.348	0.555	0.536	0.498-0.575
	Packets	0.658	0.141	0.598	0.5	0.461-0.538	0.721	0.303	0.535	0.534	0.495-0.573
Day3	Bytes	0.769	0.259	0.492	0.534	0.496-0.573	0.783	0.312	0.605	0.541	0.502-0.580
	Packets	0.803	0.204	0.534	0.536	0.497-0.575	0.695	0.389	0.509	0.533	0.495-0.572
Day4	Bytes	0.438	0.207	0.571	0.5	0.461-0.538	0.612	0.335	0.485	0.5	0.461-0.538
	Packets	0.512	0.142	0.536	0.5	0.461-0.538	0.629	0.259	0.373	0.5	0.461-0.538
Day5	Bytes	0.583	0.122	0.48	0.5	0.461-0.538	0.623	0.342	0.445	0.5	0.461-0.538
	Packets	0.581	0.063	0.412	0.5	0.461-0.538	0.659	0.251	0.426	0.5	0.461-0.538
Day6	Bytes	0.572	0.14	0.507	0.5	0.461-0.538	0.695	0.389	0.509	0.533	0.495-0.572
	Packets	0.572	0.14	0.507	0.5	0.461-0.538	0.684	0.307	0.474	0.52	0.481-0.558
Day7	Bytes	0.552	0.149	0.503	0.5	0.461-0.538	0.282	0.389	0.386	0.5	0.461-0.538
	Packets	0.591	0.121	0.504	0.5	0.461-0.538	0.488	0.325	0.388	0.5	0.461-0.538

Matlab Tool

		Busy Hour			Low Hour		
		Aggregate Variance	R/S	Periodogram	Aggregate Variance	R/S	Periodogram
Day1	Bytes	0.524	0.5272	0.4988	0.5386	0.508	0.5625
	Packets	0.5504	0.4976	0.487	0.6087	0.5016	0.5617
Day2	Bytes	0.6739	0.5069	0.6572	0.6244	0.4661	0.5575
	Packets	0.6609	0.5179	0.6646	0.6506	0.4654	0.5709
Day3	Bytes	0.8697	0.6218	0.6559	0.7885	0.5031	0.6539
	Packets	0.9227	0.6234	0.7273	0.8331	0.4593	0.7112
Day4	Bytes	0.7185	0.5817	0.6781	0.6667	0.5122	0.5401
	Packets	0.7597	0.5933	0.7373	0.7157	0.517	0.5756
Day5	Bytes	0.6554	0.5033	0.5414	0.635	0.4938	0.5674
	Packets	0.7078	0.4626	0.5006	0.6783	0.459	0.5345
Day6	Bytes	0.5236	0.5347	0.4723	0.638	0.528	0.5569
	Packets	0.5306	0.518	0.4737	0.6566	0.5288	0.5341
Day7	Bytes	0.4687	0.4811	0.5111	0.4593	0.5025	0.4837
	Packets	0.4875	0.4533	0.5781	0.4762	0.4949	0.464

6. Topology Characteristics

#Extracting only IPs from the Netflows with nfdump
nfdump -M Folder1:Folder2:Folder3 -R . -a -t 2015/12/29.00:00:00-2015/12/29.23:59:59 'duration > 10000 and bytes>100' -o "fmt:%sa,%da" >*FILENAME*.csv

#Files processed with IP addresses as Nodes and Edges
graphDay1.csv
graphDay2.csv
graphDay3.csv
graphDay4.csv
graphDay5.csv
graphDay6.csv
graphDay7.csv

#Convert IPs into Labels (numbers)
import pandas as pd
import csv

```
import re
from pandas import DataFrame, Series
import numpy as np
import sys
data2 = pd.read_csv("graphDay1.csv")
data3 = DataFrame(data2)
f = open('graphDay1Cleaned.csv','w')
IPs={};
IP_ctr=1;
for index, row in data3.iterrows():
    if not row['Source'] in IPs:
        IPs[row['Source']] = IP_ctr
        IP_ctr += 1
    IP_source_as_cts = IPs[row['Source']]
    if not row['Target'] in IPs:
        IPs[row['Target']] = IP_ctr
        IP_ctr += 1
    IP_target_as_cts = IPs[row['Target']]

    print >>f, ("{0},{1}".format(IP_source_as_cts, IP_target_as_cts))
f.close()
```

```
#Files created after above script that contain only labels are named and loaded in Matlab as:
load graphDay1Cleaned.csv
load graphDay2Cleaned.csv
load graphDay3Cleaned.csv
load graphDay4Cleaned.csv
load graphDay5Cleaned.csv
load graphDay6Cleaned.csv
load graphDay7Cleaned.csv
```

```
#Calculate Node Degree
edges = graphDay1Cleaned;%name of the file.
adjacentList={};
for i=1:1:max(max(edges,1))
  a=find(edges(:,1)==i);
  adjacentList{i,1}=edges(a,2)';
end
%Node Degrees
for i=1:1:size(adjacentList,1)
    Nodes_degrees_List(i,1) = size(adjacentList{i},2);
end
```

```
#Initial Inspection plot
uniquek = unique(Nodes_degrees_List);
for k=1:1:length(uniquek)
    b =find(Nodes_degrees_List==unique(k));
    k_occur(k,1) = length(b);  %frequency
    end
p_of_k = k_occur/sum(k_occur);
figure(1)
bar(uniquek,p_of_k)
title('Probability Degree Distribution - Day1');  %name careful
```

```
xlabel('Node Degree (k)');
ylabel('P(k)');
xlim([0 100]);
loglog(uniquek,p_of_k)
title('Day1');
xlabel('Node Degree (k)');
ylabel('P(k)');
grid on
```

```
#Calculate alpha and xmin
#Codes are taken from http://tuvalu.santafe.edu/~aaronc/powerlaws/ as described in [28]
%    Execute:
%          [alpha, xmin, L] = plfit(x);
%
%    The output 'alpha' is the maximum likelihood estimate of the scaling
%    exponent, 'xmin' is the estimate of the lower bound of the power-law
%    behavior, and L is the log-likelihood of the data x>=xmin under the
%    fitted power law.
vec    = [];
sample  = [];
xminx   = [];
limit   = [];
finite  = false;
nosmall = false;
nowarn  = false;
% parse command-line parameters; trap for bad input
i=1;
while i<=length(varargin),
  argok = 1;
  if ischar(varargin{i}),
    switch varargin{i},
       case 'range',       vec     = varargin{i+1}; i = i + 1;
       case 'sample',      sample  = varargin{i+1}; i = i + 1;
       case 'limit',       limit   = varargin{i+1}; i = i + 1;
       case 'xmin',        xminx   = varargin{i+1}; i = i + 1;
       case 'finite',      finite  = true;
       case 'nowarn',      nowarn  = true;
       case 'nosmall',     nosmall = true;
       otherwise, argok=0;
    end
  end
  if ~argok,
    disp(['(PLFIT) Ignoring invalid argument #' num2str(i+1)]);
  end
  i = i+1;
end
if ~isempty(vec) && (~isvector(vec) || min(vec)<=1),
          fprintf('(PLFIT) Error: "range" argument must contain a vector; using default.\n');
    vec = [];
end;
if ~isempty(sample) && (~isscalar(sample) || sample<2),
          fprintf('(PLFIT) Error: "sample" argument must be a positive integer > 1; using default.\n');
```

```
   sample = [];
end;
if ~isempty(limit) && (~isscalar(limit) || limit<min(x)),
        fprintf('(PLFIT) Error: "limit" argument must be a positive value >= 1; using default.\n');
   limit = [];
end;
if ~isempty(xminx) && (~isscalar(xminx) || xminx>=max(x)),
        fprintf('(PLFIT) Error: "xmin" argument must be a positive value < max(x); using default behavior.\n');
   xminx = [];
end;
% reshape input vector
x = reshape(x,numel(x),1);
% select method (discrete or continuous) for fitting
if     isempty(setdiff(x,floor(x))), f_dattype = 'INTS';
elseif isreal(x),    f_dattype = 'REAL';
else                 f_dattype = 'UNKN';
end;
if strcmp(f_dattype,'INTS') && min(x) > 1000 && length(x)>100,
   f_dattype = 'REAL';
end;
% estimate xmin and alpha, accordingly
switch f_dattype,
   case 'REAL',
      xmins = unique(x);
      xmins = xmins(1:end-1);
      if ~isempty(xminx),
         xmins = xmins(find(xmins>=xminx,1,'first'));
      end;
      if ~isempty(limit),
         xmins(xmins>limit) = [];
      end;
      if ~isempty(sample),
         xmins = xmins(unique(round(linspace(1,length(xmins),sample))));
      end;
      dat   = zeros(size(xmins));
      z     = sort(x);
      for xm=1:length(xmins)
         xmin = xmins(xm);
         z    = z(z>=xmin);
         n    = length(z);
         % estimate alpha using direct MLE
         a    = n ./ sum( log(z./xmin) );
         if nosmall,
            if (a-1)/sqrt(n) > 0.1
               dat(xm:end) = [];
               xm = length(xmins)+1;
               break;
            end;
         end;
         % compute KS statistic
         cx   = (0:n-1)'./n;
         cf   = 1-(xmin./z).^a;
```

```
        dat(xm) = max( abs(cf-cx) );
    end;
    D    = min(dat);
    xmin  = xmins(find(dat<=D,1,'first'));
    z    = x(x>=xmin);
    n    = length(z);
    alpha = 1 + n ./ sum( log(z./xmin) );
    if finite, alpha = alpha*(n-1)/n+1/n; end; % finite-size correction
    if n < 50 && ~finite && ~nowarn,
        fprintf('(PLFIT) Warning: finite-size bias may be present.\n');
    end;
    L = n*log((alpha-1)/xmin) - alpha.*sum(log(z./xmin));
case 'INTS',
    if isempty(vec),
        vec  = (1.50:0.01:3.50);    % covers range of most practical
    end;                            % scaling parameters
    zvec = zeta(vec);
    xmins = unique(x);
    xmins = xmins(1:end-1);
    if ~isempty(xminx),
        xmins = xmins(find(xmins>=xminx,1,'first'));
    end;
    if ~isempty(limit),
        limit = round(limit);
        xmins(xmins>limit) = [];
    end;
    if ~isempty(sample),
        xmins = xmins(unique(round(linspace(1,length(xmins),sample))));
    end;
    if isempty(xmins)
        fprintf('(PLFIT) Error: x must contain at least two unique values.\n');
        alpha = NaN; xmin = x(1); D = NaN;
        return;
    end;
    xmax   = max(x);
    dat    = zeros(length(xmins),2);
    z      = x;
    fcatch = 0;
    for xm=1:length(xmins)
        xmin = xmins(xm);
        z    = z(z>=xmin);
        n    = length(z);
        % estimate alpha via direct maximization of likelihood function
        if fcatch==0
            try
                % vectorized version of numerical calculation
                zdiff = sum( repmat((1:xmin-1)',1,length(vec)).^-repmat(vec,xmin-1,1) ,1);
                L = -vec.*sum(log(z)) - n.*log(zvec - zdiff);
            catch
                % catch: force loop to default to iterative version for
                % remainder of the search
                fcatch = 1;
```

```
            end;
        end;
        if fcatch==1
            % force iterative calculation (more memory efficient, but
            % can be slower)
            L       = -Inf*ones(size(vec));
            slogz   = sum(log(z));
            xminvec = (1:xmin-1);
            for k=1:length(vec)
                L(k) = -vec(k)*slogz - n*log(zvec(k) - sum(xminvec.^-vec(k)));
            end
        end;
        [Y,I] = max(L);
        % compute KS statistic
        fit = cumsum((((xmin:xmax).^-vec(I))))./ (zvec(I) - sum((1:xmin-1).^-vec(I))));
        cdi = cumsum(hist(z,xmin:xmax)./n);
        dat(xm,:) = [max(abs( fit - cdi )) vec(I)];
    end
    % select the index for the minimum value of D
    [D,I] = min(dat(:,1));
    xmin  = xmins(I);
    z     = x(x>=xmin);
    n     = length(z);
    alpha = dat(I,2);
    if finite, alpha = alpha*(n-1)/n+1/n; end; % finite-size correction
    if n < 50 && ~finite && ~nowarn,
        fprintf('(PLFIT) Warning: finite-size bias may be present.\n');
    end;
    L     = -alpha*sum(log(z)) - n*log(zvec(find(vec<=alpha,1,'last')) - sum((1:xmin-1).^-alpha));
  otherwise,
    fprintf('(PLFIT) Error: x must contain only reals or only integers.\n');
    alpha = [];
    xmin  = [];
    L     = [];
    return;
end;
```

```
#Distribution Plots (Fig 19)
%    PLPLOT(x, xmin, alpha) plots (on log axes) the data contained in x
%    and a power-law distribution of the form p(x) ~ x^-alpha for
%    x >= xmin.
function h=plplot(x, xmin, alpha)
% reshape input vector
x = reshape(x,numel(x),1);
% initialize storage for output handles
h = zeros(2,1);
% select method (discrete or continuous) for plotting
if     isempty(setdiff(x,floor(x))), f_dattype = 'INTS';
elseif isreal(x),    f_dattype = 'REAL';
else                 f_dattype = 'UNKN';
end;
```

```
if strcmp(f_dattype,'INTS') && min(x) > 50,
   f_dattype = 'REAL';
end;
% estimate xmin and alpha, accordingly
switch f_dattype,
   case 'REAL',
      n = length(x);
      c = [sort(x) (n:-1:1)'./n];
      q = sort(x(x>=xmin));
      cf = [q (q./xmin).^(1-alpha)];
      cf(:,2) = cf(:,2) .* c(find(c(:,1)>=xmin,1,'first'),2);
      figure;
      h(1) = loglog(c(:,1),c(:,2),'bo','MarkerSize',3,'MarkerFaceColor',[1 1 1]); hold on;
      h(2) = loglog(cf(:,1),cf(:,2),'k--','LineWidth',2); hold off;
      xr  = [10.^floor(log10(min(x))) 10.^ceil(log10(max(x)))];
      xrt = (round(log10(xr(1))):2:round(log10(xr(2))));
      if length(xrt)<4, xrt = (round(log10(xr(1))):1:round(log10(xr(2)))); end;
      yr  = [10.^floor(log10(1/n)) 1];
      yrt = (round(log10(yr(1))):2:round(log10(yr(2))));
      if length(yrt)<4, yrt = (round(log10(yr(1))):1:round(log10(yr(2)))); end;
      set(gca,'XLim',xr,'XTick',10.^xrt);
      set(gca,'YLim',yr,'YTick',10.^yrt,'FontSize',16);
      ylabel('Pr(X \geq x)','FontSize',16);
      xlabel('x','FontSize',16)
      legend([xr,yr],'slope 1/2','slope 1')
   case 'INTS',
      n = length(x);
      q = unique(x);
      c = hist(x,q)'./n;
      c = [[q; q(end)+1] 1-[0; cumsum(c)]]; c(c(:,2)<10^-10,:) = [];
      cf = ((xmin:q(end))'.^-alpha)./(zeta(alpha) - sum((1:xmin-1).^-alpha));
      cf = [(xmin:q(end)+1)' 1-[0; cumsum(cf)]];
      cf(:,2) = cf(:,2) .* c(c(:,1)==xmin,2);

      figure;
      h(1) = loglog(c(:,1),c(:,2),'bo','MarkerSize',4,'MarkerFaceColor',[1 1 1]); hold on;
      h(2) = loglog(cf(:,1),cf(:,2),'--r','LineWidth',1); hold off;
      xr  = [10.^floor(log10(1)) 10.^ceil(log10(max(x)))];
      xrt = (round(log10(xr(1))):2:round(log10(xr(2))));
      if length(xrt)<4, xrt = (round(log10(xr(1))):1:round(log10(xr(2)))); end;
      yr  = [10.^floor(log10(1/n)) 1];
      yrt = (round(log10(yr(1))):2:round(log10(yr(2))));
      if length(yrt)<4, yrt = (round(log10(yr(1))):1:round(log10(yr(2)))); end;
      legend('Data',sprintf('MLE alpha=%.3f',alpha));
      axis equal square
      set(gca,'XLim',xr,'XTick',10.^xrt);
      set(gca,'YLim',yr,'YTick',10.^yrt,'FontSize',12);
      ylabel('Pr(K \geq k)','FontSize',12);
      xlabel('Node Degree (k)','FontSize',12)
      T=title('Node Degree Distribution - Day7');
      set(T,'FontSize',12,'FontWeight','normal');
      grid on;
```

```
   otherwise,
      fprintf('(PLPLOT) Error: x must contain only reals or only integers.\n');
      h = [];
      return;
end;
```

```
#Compare for 7 Days Alpha
%Comparing Hurst Parameters (Days of the Week)
Alpha=[2.00, 2.0200, 2.0200, 2.0100, 2.0600, 2.0300, 2.0100];
plot(Alpha,'-r*','LineWidth',1,'MarkerSize',6)
ylim([1.5 2.5])
xlim([0.8,7.2])
grid on
legend('Alpha','Location','northwest')
ylabel('Alfa'); xlabel('Daily'); title('Parameter Alpha using MLE')
```

#R Code to calculate Assortativity and Node Diamater (Table 4).

```
#In this workspace also other metrics are calculated such as:
#Betweeness Centrality
#Cluster Coefficitent
#Egent Vector Centrality
library(igraph)
dat=read.csv(file.choose(),header=TRUE)
el=as.matrix(dat) # data into a two-column matrix format
el[,1]=as.character(el[,1])
el[,2]=as.character(el[,2])

g=graph.edgelist(el,directed=FALSE) # turns the edgelist into a 'graph object'


Node_Degree=degree(g)
dSorted=sort.int(Node_Degree, decreasing = TRUE)
Average_Node_Degree=mean(degree(g))
Min_Node_Degree=min(degree(g))
Max_Node_Degree=max(degree(g))
#Diameter
Graph_Diameter=diameter(g,directed = FALSE,unconnected = TRUE)#if directed=True it returns +1 larger than
max number of vertices
Farthest_Node_Diameter=farthest.nodes(g, directed = FALSE, unconnected = TRUE)
#Assortativity
Assortativity_Degree=assortativity.degree(g)
Betweenes_Centrality = betweenness(g)
bSorted=sort(Betweenes_Centrality,decreasing = TRUE)
write.table(Betweenes_Centrality, "C:/Users/telecom/Desktop/artan/Day2_Betweenes_Centrality.csv", sep="\t")
#Cluster Coeff (also called Transitivity) for local (each vercex) and global
Global_Cluster_Coeff=transitivity(g,type=c("global"))
Local_Cluster_Coeff=transitivity(g,type=c("local"))
cSorted=sort.int(Local_Cluster_Coeff, decreasing = FALSE)
#Egent Vector Centrality
Egent_Vector_Centrality=evcent(g)
```

```
#Same code for ASes, Files that are pre and po-processed are named:
```

ASD1.csv	ASD1Cleaned.csv
ASD2.csv	ASD2Cleaned.csv
ASD3.csv	ASD3Cleaned.csv
ASD4.csv	ASD4Cleaned.csv
ASD5.csv	ASD5Cleaned.csv
ASD6.csv	ASD6Cleaned.csv
ASD7.csv	ASD7Cleaned.csv

7. Independence of Packet Size Arrivals

```
#Processed Files
% load PacketSizeApart;
% load PacketSizeApart1;
% Apart=PacketSizeApart;
% Apart1=PacketSizeApart1;
```

```
%% SAMPLE ACF for Packet Size Series
[ACF,lags,bounds] = autocorr(PacketSIzeAPART,[],2);
autocorr(PacketSIzeAPART,200); ylim([-0.05 0.1]);
ylabel('Sample Autocorrelation','FontSize',10,'FontWeight','normal','Color','k');
xlabel('Lag','FontSize',10,'FontWeight','normal','Color','k');
T=title('Sample ACF - Packet Size');
set(T,'FontSize',12,'FontWeight','normal');
```

```
%% Scatter Plot of 1million packet size arrivals
%%subplot(2,1,1)
scatter(Apart,Apart1,'*b','LineWidth',0.01);
ylabel('packet size (k+1)','FontSize',10,'FontWeight','normal','Color','k');
xlabel('packet size (k)','FontSize',10,'FontWeight','normal','Color','k');
T=title('Scatter Plot - Packet Size');
set(T,'FontSize',12,'FontWeight','normal');
```

```
%% BOX-JUNG STATISTIC TEST
mA=mean(Apart); %
mB=mean(Apart1);
sA=std(Apart);
sB=std(Apart1);
X1=(Apart-mA)/sA;
X2=(Apart1-mB)/sB;
X=X1.*X2;
n=length(Apart);
r=(1/(n-1))*sum(X) %%Autocorrelation value 0.0065

%%Box-Ljung using lbtest Matlab Function
i=1:200;
[h,pValue] = lbqtest(Apart,'lags',i,'alpha',0.05);
if h==0
   sprintf('There is not enough evidence to reject the null hypythesis')
end
```

8. Distribution of 10 top IP addresses and Port Numbers (Figure 22, 23 Coef of Determination and Figure 24)

```
#Extracting only IPs from the Netflows with nfdump
nfdump -M Folder1:Folder2:Folder3 -R . -a -t 2015/12/29.00:00:00-2015/12/29.23:59:59 'duration > 10000 and bytes>100' -o "fmt: %da" >FILENAME.csv
```

#Files processed with IP addresses

Files pre-processed	Files of IPs labeled	Files for Port numbers (no need to convert into int.)
IP1.csv	IP1C.csv	Ports1.csv
IP2.csv	IP2C.csv	Ports2.csv
IP3.csv	IP3C.csv	Ports3.csv
IP4.csv	IP4C.csv	Ports4.csv
IP5.csv	IP5C.csv	Ports5.csv
IP6.csv	IP6C.csv	Ports6.csv
IP7.csv	IP7C.csv	Ports7.csv

```
#Convert IPs into Labels (numbers)
import pandas as pd
import csv
import re
from pandas import DataFrame, Series
import numpy as np
import sys
data2 = pd.read_csv("graphDay1.csv")
data3 = DataFrame(data2)
f = open('graphDay1Cleaned.csv','w')
IPs={};
IP_ctr=1;
for index, row in data3.iterrows():
    if not row['Source'] in IPs:
        IPs[row['Source']] = IP_ctr
        IP_ctr += 1
    IP_source_as_cts = IPs[row['Source']]
    print >>f, ("{0}".format(IP_source_as_cts))
f.close()
```

```
%% Powerlaw and CDF of port and IP addresses
clear all;
clf;
load Ports7.csv;
Y=Ports7; clear Ports7.csv; %careful with filenames.
a=unique(Y);
out=[a, histc(Y(:),a)];
out;
Frequencies=out(:,2);
sorted_Frequencies=sort(Frequencies,'descend');
Prob=sorted_Frequencies/sum(sorted_Frequencies);
x=1:10 %Top 10 most frequent.
y=Prob(1:10)';
logx=log(x);
logy=log(y);
```

```
p=polyfit(logx,logy,1);
plot(logx(1:10),logy(1:10),'bo');
axis equal square
grid
xlabel('log(x)');
ylabel('log(y)');
k=p(1);
loga=p(2);
a=exp(loga);
hold on; plot(logx,k*logx+loga,'*-g')
legend('Data',sprintf('log(y)=-alog(x)+log(b)'));
T=title('Estimating with polifit (loglog linear form)');
set(T,'FontSize',12,'FontWeight','normal');
%%Plot CDF from PDF
alfa=-k;
figure
P=cumsum(y);
plot(x,P,'*-b');
hold on;
%%For y2=0.457*x.^-2.155. Careful with equation, depends from values
%calculated from polofit for slope and intercept.
y3=0.444*x.^(-1.811);
%y2=0.457*(x.^(-2.155));
P_fit=cumsum(y3);
plot(x,P_fit,'-og');
legend('Data',['Fit a=', num2str(alfa)],'Location','NorthWest');
grid on;
hold off;
ylabel('CDF','FontSize',10,'FontWeight','normal','Color','k');
xlabel('Rank of Port Numbers','FontSize',10,'FontWeight','normal','Color','k');

figure
plot(x,y,'bo');
xlabel('x');
ylabel('y');
axis equal square
grid
hold on; plot(x,a*x.^k,'*-g')
legend('Data',sprintf('y=%.3f{}x^{%.3f}',a,k));
hold off;
T=title('Estimating with polifit (loglog linear form)');
set(T,'FontSize',12,'FontWeight','normal');
```

```
%%Residual scatter plot
Residuals=y3-y;
scatter(x,Residuals,'rp');
lsline;
ylabel('Residuals','FontSize',10,'FontWeight','normal','Color','k');
xlabel('Rank of Port Numbers','FontSize',10,'FontWeight','normal','Color','k');
box on;
variance_Ports=(var(y));
variance_Ports_estimated=(var(y3));
```

r_Square=(variance_Ports_estimated).^2/(variance_Ports).^2;

#Alpha for seven days and Coeff of Determ for port Numbers

2.15	1.914	1.816	1.884	1.914	2.025	1.891

A=[2.15, 1.914, 1.816, 1.884, 1.914, 2.025, 1.891]
mean_A=mean(A)
1.415*std(A)/(sqrt(7))%1.45 from T-test table for 90% C.L.

#7 DAYS t-TEST 90%, Mean +- t[1-alfa/2;n-1]S/sqrt(n)
A=[0.652, 0.646, 0.652, 0.621, 0.6708,0.959, 0.672]
mean_A=mean(A);
1.415*std(A)/sqrt(7);

BIBLIOGRAPHY

[1] J. Cao, W. S. Cleveland, D. Lin and D. X. Sun, "Internet Traffic Trends Toward Poisson and Independendent as the Load Increases," in *Nonlinear Estimation and Classification*, New York, Springer, 2003, pp. 83-109.

[2] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson, "On the Self-Similar nature," *IEEE/ACM Transactions o Networking,* 1994.

[3] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *ACM Transactions on Networking,* 1995.

[4] T. Karagiannis, M. Molle and M. Faloutsos, "A Nonstationary Poisson View of Internet Traffic," *IEEE INFOCOM,* 2004.

[5] T. Telkamp, A. Maghbouleh, V. Sharma and S. Gordon, "Internet Traffic is not Self-Similar at Timescales Relevant to Quality of Service," 2004.

[6] B. Bollobas, Random Graphs, London: Academic.

[7] P. Erdos and A. Renyi, *Science,* vol. 5, pp. 17-61.

[8] A. Barabasi and E. Bonabea, "Scale-Free Networks," *Rev. Mod. Phys,* vol. 74, p. 67–97, 2002.

[9] M. Crovella and A. Bestavros, "Self-similarity in WWW traffic: evidence and possible causes," *IEEE/ACM Trans. on Networking 5 ,* p. 835– 846, 1997.

[10] K. Park, G. Kim and M. Crovella, "On the relationship between file sizes, transport protocols, and self-similar network traffic," *Tech. Rep. 1996-016,* 1996.

[11] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper," Cisco, 2015.

[12] R. Cohen and D. Raz, "The Internet dark matter – on the missing links in the AS connectivity map," *IEEE INFOCOM,* p. 1–12, 2006.

[13] H. Chang, R. Govindan, S. Jamin, S. Shenker and W. Willinger, "Towards capturing representative AS-level Internet topologies," *Computer Networks Journal 44 ,* p. 737–755, 2004.

[14] A. Lakhina, J. W. Byers, M. Crovella and P. Xie, "Sampling biases in IP topology measurements," *INFOCOM/IEEE,* pp. 332 - 341, 2003.

[15] W. W. a. V. Paxson, V. Paxson and W. Willinger, "Where Mathematics meet the Internet," *American Mathematical Society,* vol. 45, 1998.

[16] K. Park and W. Willinger, "Self-similar network traffic: An overview," 2000. [Online]. Available: https://www.cs.purdue.edu/nsl/intro-ss-chap.pdf. [Accessed 4 2016].

[17] B. Mandelbrot, "The Fractal Geometry of Nature," *Freeman,* 1983.

[18] J. Cao, W. S. Cleveland, D. Lin, D. X. Sun and M. Hill, "On the Nonstationarity of Internet Traffic," *Springer,* 2002.

[19] C. Park, F. Hernandez-Campos, J. S. Marron and F. D. Smith, "Long-Range Dependence in a Changing Internet Traffic Mix," *Computer Networks,* pp. 401-422, 2005.

[20] V. Paxson and S. Floyd, "Why we don't know how to simulate the Internet," *Proceedings of the 29th conference on Winter simulation,* pp. 1037-1044, 1997.

[21] M. Faloutsos, P. Faloutsos and C. Faloutsos, " On power–law relationships of the Internet topology," *ACM SIGCOMM - Computer Communication Rev 29,* pp. 251-261, 1999.

[22] "The Route Views project," University of Oregon, Eugene, [Online]. Available: http://www. routeviews.org/.

[23] "RIPE, Routing information service," RIPE, [Online]. Available: http://www.ripe.net.

[24] The Cooperative Association for Internet Data Analysis - CAIDA, [Online]. Available: http://www.caida.org.

[25] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," *ACM SIGCOMM,* p. 147–159, 2002.

[26] M. E. J. Newman, "Assortative mixing in networks," *Phys. Rev. Lett. 89, 208701,* 2002.

[27] P. Mahadevan, D. Krioukov, M. Fomenkov and B. Huffaker, "The Internet AS-Level Topology: Three Data Sources and One Definitive Metric," *ACM SIGCOMM Computer Communication Review,* vol. 36 , no. 1, pp. 17-26 , 2006.

[28] A. Clauset, C. R. Shalizi and M. E. J. Newman, "Power-law distributions in empirical data," *ACM,* vol. 51, no. 4, pp. 661-703, 2009.

[29] T. Karagiannis, M. Faloutsos and M. Molle, "A User-Friendly Self-Similarity Analysis Tool," *Special Section on Tools and Technologies for Networking Research and Education, ACM SIGCOMM Computer Communication Review,* 2003.

[30] C. Chen, "http://www.mathworks.com/," 11 Mar 2008. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/19148-hurst-parameter-estimate. [Accessed Mar 2016].

[31] J. Leskovec, J. Kleinberg and C. Faloutsos, "Graphs over Time: Densification Laws, Shrinking," *ACM SIGKDD international conference on Knowledge discovery in data mining,* pp. 177-187 , 2005.

[32] "www.cisco.com," [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/netflow/configuration/xe-3s/asr1000/nf-xe-3s-asr1000-book.pdf.

[33] "manpages.ubuntu.com," [Online]. Available: http://manpages.ubuntu.com/manpages/gutsy/man1/nfdump.1.html.

[34] J. L. Garcia-Dorado, J. A. Hernandez, J. Aracil, J. E. d. Vergara, F. J. Monserrat and T. P. d. M. E. Robles, "On the duration and spatial characteristics of internet traffic measurement experiments," *IEEE/ACM Communications Magazine,* vol. 46, no. 11, 2008 .

[35] E. Papalexakis and N. D. S. C. Faloutsos, "ParCube: Sparse Parallelizable Tensor Decompositions," in *Lecture Notes in Computer Science*, vol. 7523 , Springer Berlin Heidelberg, 2012, pp. 521-536.

[36] P. Erdos and A. Renyi, *Science,* vol. 5, pp. 17-61.

[37] A. L. Barabasi and R. Albert, "Statistical mechanics of complex networks," *Science,* no. 286, pp. 17-61.

[38] R. Jain, "Selection of Techniques and Metrics," in *The Art of Computer Systems Performance Analysis*, New York, John Wiley & Sons, 1992, pp. 30-40.

[39] D. J. Watts and S H Strogatz, "Collective dynamics of ‘small-world’ networks," *Nature ,* no. 393, p. 440–442, 1998.

[40] M. Newman, "Measures and metrics," in *Networks: An Introduction*, Published to Oxford Scholarship Online, 2010.

[41] U. Brandes, "A Faster Algorithm for Betweenness Centrality".

[42] V. E. Krebs, "“Mapping Networks of Terrorist Cells”".