Performance Metrics, Configuration Strategies and Traffic Identification for Group Network Application

FU, Zhengjia

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Information Engineering

September 2008

Abstract of thesis entitled:

Performance Metrics, Configuration Strategies and Traffic Identification for Group Network Application

Submitted by FU, Zhengjia

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in September 2008

This thesis consists of two parts of the work. The first part is to design performance metrics and configuration strategies for network voice conference and the second part is to propose a novel method of identifying P2P applications based on behavioral-signatures.

For the first part of the work, since there is an increasing number of group-based multimedia applications over the Internet, for example, voice conference or multi-player games. For these applications, it is often necessary to select a strategy to distribute the multimedia streams or mixing the multimedia stream data so as to provide better quality of service (QoS) guarantees. However, there is no appropriate metrics to evaluate the QoS of a group multimedia session, despite abundant literature on how to evaluate the QoS for two-party communication (e.g. MOS, E-Model). In the first part of this thesis, we propose a new measure which is called the *group mean opinion score* (GMOS). To leverage on existing work, our definition of GMOS is based on two-party MOS, hence, it can be estimated via measurement of network parameters and fitting these data into the E-Model. We conduct large scale experiments using the latest SKYPE conference software. We first calibrate the GMOS based on the sub-

jective scores of our experiments, then for individual conference sessions, we check whether our approach can pick a server configuration strategy to achieve the best GMOS. The study shows our proposed methodology is very promising and the potential of applying to other group-based applications.

On the other hand, due to the significant increase of peer-to-peer (P2P) traffic in the past few years, network operators in the ISPs and enterprise networks want to be able to identify P2P applications and their associated traffic so as to perform a better traffic engineering and capacity planning. In the second part of this thesis, we propose a novel approach to identify P2P applications, as well as pinpointing a specific P2P software from packet traces based on *behavioral signatures*. The main advantage of the proposed approach is that we do not need to examine the packet payloads, but instead, we only need to check whether a specific periodic signature of a given P2P application is available in the traffic. We provide justifications as to why P2P traffic has some inherent periodic signatures, and show how to extract the signature for a given P2P software. The proposed methodology is effective not only in identifying P2P applications but discovering specific P2P softwares such as BitTorrent, Emule, (P2P content distribution applications), PPLive and PPStream (P2P video streaming applications),...etc. We apply our approach to real traffic traces and show its effectiveness in P2P software identification.

中文摘要

本论文主要由两个部分的工作组成。第一部分的工作是针对网络语音会议设计服务质量（QoS）的评估参数和参数设置的策略方案。第二部分的工作是提出了一种创新的基于行为特征的识别对等网（P2P）技术的网络应用的方法。

对于第一部分的工作，目前在英特网中，同时有多方参与的网络多媒体应用正在不断的增长，例如，多人网络语音会议及多参与者同时进行的网络游戏。对于这些应用来说，为了对其提供的服务质量（QoS）有较好的保证，通常选择一种策略来有效的分发、传递流媒体或者混合流媒体数据是非常有必要的。尽管之前已经有大量的工作来研究如何评估“两方”参与的网络语音通信的服务质量（例如，MOS 和 E-Model），然而，却没有适合于评估多方参与的网络语音会议的服务质量的评估参数。在第一部分工作中，我们提出了一种新的评估参数叫做“平均多方主观评价分数”（GMOS）。为了利用已有的研究结果，我们对 GMOS 的定义是基于“平均主观评价分数”（MOS）的，也就是说，GMOS 是可以通过对网络通信参数的测量并将测量得到的数据结果通过运用 E-Model 的拟合而估计出来的。我们使用最近非常流行的 SKYPE 软件的网络语音会议功能，进行了大量的实验。我们先是基于通过对我们的语音实验的平均主观打分来调整 GMOS 评估参数的准确性，然后我们进行多次单独的网络语音会议的实验，来检验我们所提出的选择语音会议发起者（Leader）的方法，是否可以使这次网络语音会议的“平均多方主观评价分数”达到最高。从检验结果来看，我们提出的方法是非常有效的，并且从应用角度出发，这个方法也能够被运用到其他的同时有多方参与的网络应用中去。

对于第二部分工作，由于在过去的几年中，基于对等网（P2P）技术的网络应用所产生的网络数据流量正在非常迅速的增长。英特网服务提供商（ISP）以及公司网络的网络运营者非常希望具备能够识别 P2P 网络应用软件以及它们产生的相应的网络流量的能力，从而来实行更好的流量工程管理及带宽分配方案。在第二部分工作中，我们提出一种新颖的基于行为特征的通过抓取和分析网络数据包的识别 P2P 应用的，并能指出是哪一种特殊的 P2P 的软件的方法。我们所提出的方法的主要优势在于我们不需要数据包负载数据的信息，而只是通过检测网络数据流量中是否存在给定的 P2P 应用所具有的特殊的周期性特征来进行识别。对于 P2P 应用的网络流量为何具有内在的周期性特征的原因我们给出了定性的分析，并且还展示了如何获取一种给定的 P2P 应用软件的周期性特征的方法。我们所提出的方法不仅可以有效地识别出流量中是否存在 P2P 应用，而且可以判别出它是属于哪一种特定的，诸如 BitTorrent、Emule（P2P 文件分发和传播软件）以及 PPLive、PPstream（P2P 视频播放软件）等等，的 P2P 应用软件。我们将我们提出的方法对实际的网络数据流量进行分析，验证了它对于 P2P 应用软件识别的有效性。

Acknowledgement

I would like to thank my supervisors Prof. Dah-Ming Chiu and Prof. John C. S. Lui for their great help in my research work and thesis writing. I would also thank Miss Yan Hu and Mr. Xingang Shi for their help in the second part of the work. In addition, I would thank Mr. Alan Lam, the technique staff in IE department of CUHK for his help of data trace collection for the experiment validation in the second part of the work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Summary

In this chapter, we give the introduction of both two parts of the work respectively. We first describe the back ground of the work. We then discuss the contribution of our work and finally the organization of this thesis.

In the thesis, we conduct two parts of work. The first part is to design performance metrics and configuration strategies for group network communication and the second part is to propose a novel method of identifying P2P applications based on behavioral-signatures.

For two-party multimedia applications, there is extensive literature on how to characterize, measure and model the performance of such applications [17, 23, 18, 20, 21, 57, 43, 46, 22, 55]. In particular, the performance of two-party multimedia applications can be evaluated at a subjective level, using *mean opinion score* (MOS [17]). Various factors that affect performance are identified, such as the type of codecs used and other measurable network conditions like loss rate and round-trip time. Through extensive measurement studies and comparison with MOS data, a model is developed to predict MOS based on measurable pa-

rameters. One well establish model is the E-Model [23].

More recently, multi-party multimedia applications have also become popular. One example is the support for conferencing in the recently released version of SKYPE [1]. However, there is no standard metric to characterize the performance of a conferencing application (or multi-party application). The motivation for us to seek a performance metric is due to the consideration of how to configure a communication and mixing strategy for a conferencing application. In a peer-to-peer implementation of group communication [33], a strategy is needed so as to select a particular peer as the server. Unless there is a performance metric to compare the difference of using different peers as servers, it is not clear how the choice can be systematically made.

Motivated by this problem, we explore how to define a *group-based* MOS metric for all parties - we call this the *group mean opinion score* (GMOS). Following the same framework of the two-party paradigm, one needs to have both a subjective measure as well as a connection to physically measurable parameters, such as network delay and loss rates between different peers. In the end, this new framework should allow us to make configuration decisions that is expected to yield the best subjective evaluation. Parallel to the E-model case, we also develop a mapping between the subjective GMOS and the measurable parameters as well as the configuration decisions.

In the first part of the thesis, we propose a GMOS equation based on MOS between bilateral communications at the beginning. This GMOS model includes a parameter that can be used to calibrate the model for specific applications and users. We conduct SKYPE conferencing experiments to see whether the model can be consistently applied to multiple experiments with the same application and user population. The result gives us some idea of how to calibrate the model parameter, α.

Secondly, we develop a *two-step mapping method* (TSMM) to predict GMOS based on measurable network parameters and the server selection decision. The first step is to measure network parameters (delay and loss) and apply E-Model to find out MOSes for each bilateral session. The second step is to use our calibrated GMOS model to predict the subjective evaluation for different leader (server) selections. Finally, we compare our predication to actual scores given by users. Our conclusion is that our GMOS model and the leader selection approach is able to produce good decisions.

At the same time, there is also a significant increase in peer-to-peer applications running over the Internet and enterprise IP networks during the past few years. These applications include P2P content distribution applications like Bit-torrent[34], Bit-comet[2] and Emule[3], and P2P streaming applications like PPLive[4], PPStream[5], Sopcast[6] and so on. P2P applications account for a large share of the total traffic in the Internet.

Network operators of both the Internet and enterprise networks want to be able to identify the P2P applications and their associated traffic in order to do traffic engineering, capacity planning, and manage the cost of the network. For example, some enterprises may want to rate-limit or block P2P traffic, in order to ensure good performance for more critical applications. Broadband ISPs may want to limit P2P traffic to reduce the cost charged by upstream ISPs.

There are several existing approaches to identify P2P traffic. The simplest method, based on packets' destination port, was effective in the early days when most P2P applications used default and fixed port numbers to communicate between peers. However, the current trend is for peers to dynamically find each other (via a tracker or supernode) so that it is not necessary for peers to have fixed port number (and IP addresses). Indeed, it is found that most P2P traffic nowadays do not use fixed port

numbers, making the port-based method ineffective.

Payload signature-based identification method is more reliable to identify P2P applications, and is adopted by commercial products. It checks packet payload to find application-specific signatures. However, it has several drawbacks: (a) such *deep packet inspection* usually requires hardware support and is expensive; (b) there are privacy and legal concerns in allowing traffic monitors to look at payload information; (c) some P2P applications may do payload encryption to make this method ineffective.

More recently, a novel approach to traffic classification and application detection is proposed, and it is based on traffic behavior rather than payload. One such proposal is BLINC [45], which uses flow-level information generated by standard tools such as Netflow [7]. Based on behavioral analysis, one can discover what type of flows (TCP or UDP), how many flows comes in/out of a host, and to how many other hosts these flows connect to, or the volume of traffic in these flows. BLINC tries to classify applications into different categories, such as WEB, DNS, MAIL, ATTACK or P2P. Note, however, BLINC does not try to detect specific applications (e.g. BitTorrent or PPStream) as in the case of payload signature based methods.

In the second part of this thesis, we propose a "*behavioral signature*" based approach for application detection, hence also traffic classification. A behavioral signature can be associated with a flow, or with a host. For example, a host is observed to send a packet of certain size to ten or more other hosts every 10 seconds. This can be considered a behavioral signature for asserting some application that host is running. The idea is that such behavioral signatures can be developed for specific applications of interest, and then later used to detect user behavior, or isolate certain type of traffic in a large traffic trace. Note, this is different than payload-based signatures, and hence it does not

have the drawbacks of the payload signature based approach. This is a type of behavioral based traffic analysis method, but different than tools like BLINC which does not try to identify specific applications.

The main body of this second part is actually about applying this idea to detecting specific P2P applications. In fact, all the behavior signature developed and studied in this thesis are based on the following idea - P2P applications, by their distributed nature, exhibit certain "*periodic group communication behavior*". Yet, there are several different forms of such behavior, with different periodicity, which can be used to detect different applications. If applications do have somewhat different periodicity, there are standard tools to convert the *time-domain* packet sequences into *frequency-domain* patterns ready to be recognized mechanically.

The outline of the thesis is as follows. In Chapter 2, we describe the first part of the work, design of performance metrics and configuration strategies for network voice conference. The second part of the work, the method of identifying P2P applications based on behavioral-signature is discussed in Chapter 3. We then discuss the related work in Chapter 4 and finally comes the conclusion.

□ **End of chapter.**

Chapter 2

Design for group network communication

Summary

In this chapter, we describe the first part of the work, design of performance metrics and configuration strategies for network voice conference. In Section 2.1, we present the definition of GMOS and results from our experiments to support this measure. In Section 2.2, we illustrate various topologies used to support the voice conference and propose the *leader selection strategy* (LSS) based on the end system mixing topology. We then describe our experimental settings and measurements in Section 2.3. In Section 2.4, we present the *two-step mapping method* (TSMM) and analyze the effectiveness of this proposal. We discuss the applications of our proposed methodology in section 2.5.

2.1 Performance metrics of network Voice Conference: $GMOS$

When users participate in a voice conference session, they will hear more than one speaker's voice. The voice quality of different speakers will vary depending on the heterogeneous network conditions between the listener and the speakers. Assume that the session has N participants where participant i is denoted as $\mathcal{P}_i$, $i \in \{1, \ldots, N\}$. $\mathcal{P}_i$ will provide $N-1$ MOS scores [17] for other participants and these MOS scores represent the audio quality of these participants from $\mathcal{P}_i$'s perspective. Additionally, $\mathcal{P}_i$ has a score to reflect the overall quality of the conference session, and this is our proposed group mean opinion score (GMOS). We propose to use GMOS to relate the MOS scores of other participants as well as a subjective measure on the group audio quality. Formally, the GMOS of $\mathcal{P}_i$ is:

$$\begin{aligned} &GMOS_i\left(MOS_i(1), \ldots, MOS_i(N), \alpha\right) = \\ &AVE + \alpha(AVE - MIN)U(-\alpha) + \alpha(MAX - AVE)U(\alpha), \end{aligned} \tag{2.1}$$

where $MOS_i(k)$ is the MOS score set by $\mathcal{P}_i$ for $\mathcal{P}_k$, and

$$\begin{aligned} AVE &= \frac{\sum_{k=1}^{N-1} MOS_i(k)}{N-1}, \\ MAX &= \max\left\{MOS_i(1), \ldots, MOS_i(N)\right\}, \\ MIN &= \min\left\{MOS_i(1), \ldots, MOS_i(N)\right\}, \\ \alpha &\in [-1, 1], \\ U(x) &= \begin{cases} 1 & x > 0, \\ 0 & x \leq 0. \end{cases} \end{aligned}$$

Note that participant $\mathcal{P}_i$ can use the parameter α to control his subjectivity on the quality of the group communication. For example, when $\mathcal{P}_i$ sets $\alpha = -1$ (or $\alpha = 1$), the $GMOS_i$ will be

equal to MIN (or MAX). When $\mathcal{P}_i$ sets $\alpha = 0$, $GMOS_i$ will be equal to AVE. In other words, if $\mathcal{P}_i$ feels that the conference quality is defined by the minimum (or maximum) MOS of other participants, $\mathcal{P}_i$ will set α to -1 (or 1). If $\mathcal{P}_i$ feels that the conference quality is defined by the average of $N-1$ MOS scores, he will set $\alpha = 0$. In practice, different values of α represent different subjective view of $\mathcal{P}_i$ on the overall quality of the group conference.

Table 2.1: MOS, GMOS and Meanings

MOS	GMOS	Meaning
5	5.0	Excellent
4	4.0	Good
3	3.0	Fair
2	2.0	Poor
1	1.0	Bad

Based on the ITU-T P.800 recommendation, the MOS is an integer between 1 to 5. Table 2.1 provides the physical meaning of each value of MOS [17, 19]. For GMOS, we believe it is appropriate to represent it also by a number between 1.0 and 5.0, but we relax the integer constraint. By making it a real number with one decimal place has enough resolution to reflect QoS measure. Note that we need more resolution that GMOS is affected by more parameters than MOS. Other than the resolution, the physical meaning of GMOS is very similar to MOS (refer to Table 2.1). To evaluate the effectiveness of the proposed GMOS, we first test the GMOS formula using some real life experiments. The detail of the experiment settings will be described in a later section. Here we just show the figures and discuss the meaning of the results. To carry out the experiment, we invited some people to have audio conferences via SKYPE [1]. All the voice contents were recorded using standard recording software. A total of 18 sets of experiments were carried out during January

of 2007. Out of the 18 conferences, three were 3-person conferences, ten were 4-person conferences and five were 5-person conferences. Subsequently, we invited 25 subjects to listen to and give subjective scores to these 18 records. Every time they finished one record, they first gave MOSes to all the speakers who appeared in that experiment (one MOS for each speaker), and then they gave a subjective GMOS (between $1.0 - 5.0$) to express their satisfaction on the overall quality of the voice conference record. Consider a 4-person conference as an example. Each subject would have given five scores, four of which are integer MOSes towards the quality of individual participants and one for the subjective GMOS score, which reflects his opinion on the overall quality of the voice conference.

On collecting all these scores, we calculate the α value through the MOSes and GMOS according to Eq (2.1). The total number of the MOSes and GMOS pairs are 437 including all the three types of conference: 3-person, 4-person and 5-person. Based on Eq. (2.1), we can determine 392 α values. Figure 2.1 is the frequency distribution of the computed α.

We make the following observation.

a) In our GMOS model, we implicitly assume that GMOS should be a number between the minimum and the maximum of the MOSes. As we have explained before, when $\alpha \in [-1, 0)$, this implies that the user concerns more about the worst case performance, or they are "pessimistic" about the overall conference quality. Consequently, the GMOS is below the average and it is between the AVE and MIN value of the MOSes. When $\alpha \in (0, 1]$, it implies that the user concerns more about the best case performance, or they are "optimistic" about the overall conference quality. The results of the experiment reveal that in reality, a certain percentage of people are "very pessimistic" or "very optimistic" , which means that their GMOS scores will either

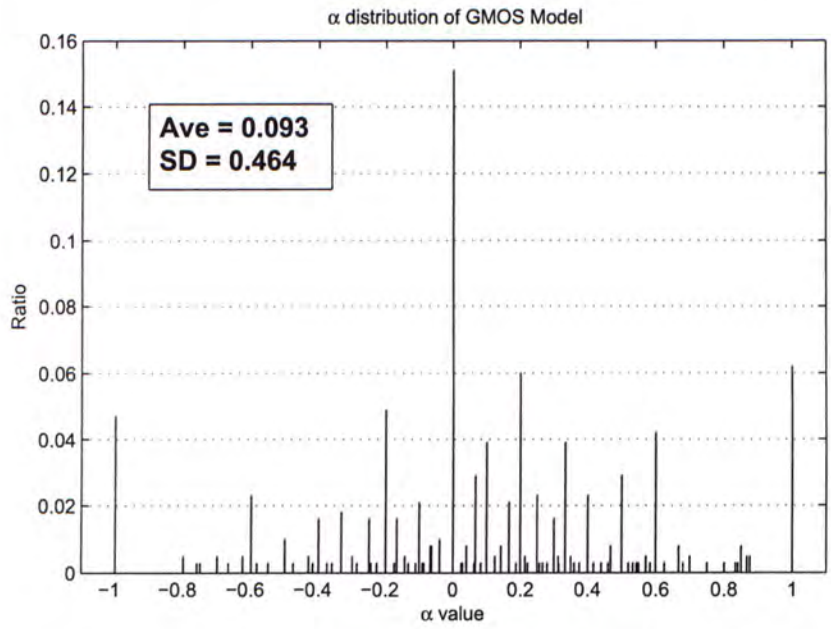


Figure 2.1: α distribution

be smaller than the minimum or larger than the maximum value of the MOSes. In our experiment, this proportion is $(437 - 392)/437$, or $\approx 10\%$.

b) From Figure 2.1, one can observe that a large proportion of subjects think that GMOS should be the average (15%), or very close to the average of MOSes (summing up the ratio of $\alpha \in [-0.2, 0.2] \approx 50\%$). As a result, we set the default value of α to be 0.

c) The average value of all the 392 α samples is 0.093, and it is larger than 0 (the default value). Statistically, it indicates that these subjects are more "optimistic" on average. Another possible interpretation is that the value of α is application dependent. The average value of $\alpha = 0.093$, obtained by an experiment using SKYPE, might be specifically applicable to SKYPE, or audio conference only, but

not necessarily for other group-based multimedia applications.

2.2 Conference Leader Selection strategies

In this section, we make use of the GMOS model and take a closer look at the QoS issues of voice conference application. In particular, we seek to answer the following question: is it possible to improve the overall quality of a voice conference session via some configuration strategies?

Several types of topologies supporting multi-party voice conference are discussed in [48], e.g., end system mixing, conference server mixing, full mesh and combination of conference servers and full mesh. Figure 2.2 illustrates four types of the topologies [48].

We will mainly focus on the "*end system mixing*" topology illustrated in Figure 2.2. Following are some justifications of choosing this type of topology:

- It is the simplest and easiest to implement among the four topologies.
- It does not require a dedicated server to hold the conference.
- It requires far less bandwidth than the full mesh topology.
- In an overlay peer-to-peer network scenario, the end system mixing topology is more effective and suitable as compared with the other three topologies.

Note that the main disadvantage of the end system topology is the heavy loading on the leader or the media stream mixer (the "A" node in the top-right of Figure 2.2). Based on the end system mixing topology, one of the conference participants should be the conference leader who is in charge of establishing and

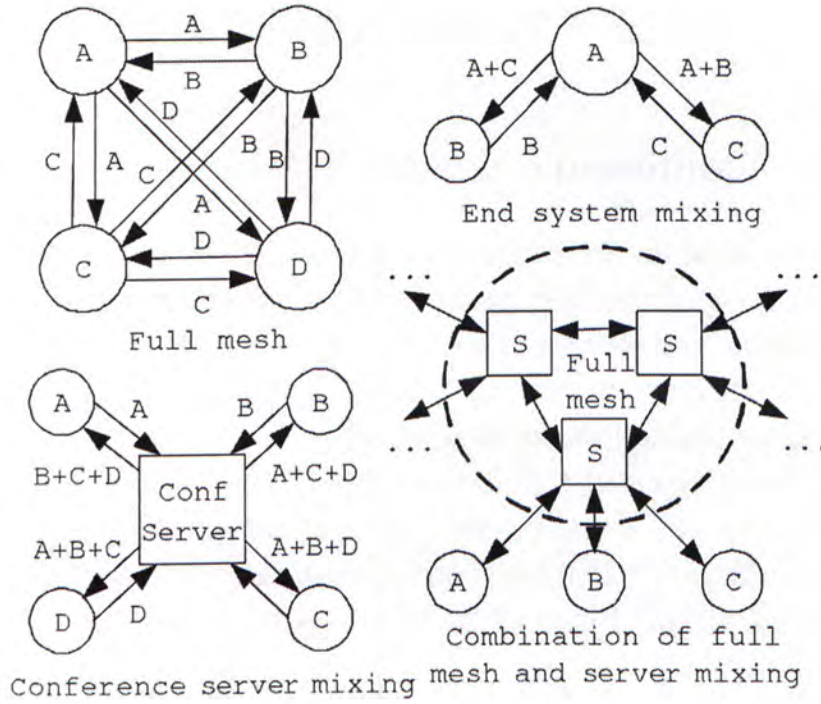


Figure 2.2: Illustration of four types of voice conference topologies [48]

starting the conference, inviting and adding participants to the conference and also ending it. Also, the computation of mixing and forwarding media streams is carried out by this leader. It implies that the leader performs many essential functions and it will greatly affect the overall quality of voice conference.

In Section 2.1, we mentioned about the voice conference experiments using SKYPE [1] and the three types of experiments: 3-person, 4-person and 5-person conference call. We use Ethereal [8] to collect all packets from computers of all participants including the leader. The traffic from measurement indicates that the topology of SKYPE conference is indeed an end system mixing topology. Figure 2.3 shows three types of topologies of our conference experiments. The leader is the one that determines the overall QoS and there is no traffic between non-leader

participants, i.e., their traffic has to be relayed via the leader.

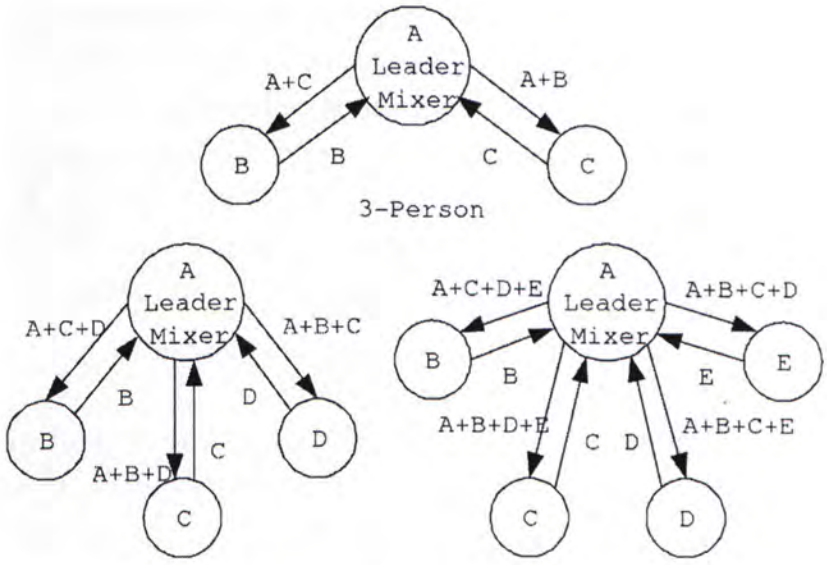


Figure 2.3: Topology of 3-person, 4-person and 5-person SKYPE conference

Since every participant will have a GMOS to show their subjective view on the overall quality of the voice conference, we need to categorize GMOS into conference leader's GMOS (denoted as GMOS_L) and non-leader participants' GMOS (denoted as GMOS_M). Furthermore, we assume that the leader's GMOS_L, which is the subjective assessment towards each non-leader participant from the leader's perspective, is the representation of the subjective evaluation of all non-leader participants towards the overall quality of the voice conference. In other words, this implies that the leader's opinion can represent other non-leader participants' opinion on evaluating the overall conference quality.

Based on the assumption and discussions above, we propose the *leader selection strategy*(LSS) of properly selecting the conference leader to improve the overall quality of the voice conference. Suppose there are N participants in a voice conference. Each of them being the leader once in turn, they will get a

$\text{GMOS}_{L_i}, (i = 1, 2, \ldots, N)$, and finally a total of N GMOS_L. Each of the $\text{GMOS}_{L_i}, (i = 1, 2, \ldots, N)$ represents the overall quality of the N voice conference under the condition that participant i being the conference leader. The *leader selection strategy*(LSS) is to select participant i whose GMOS_{L_i} is the largest among all the $\text{GMOS}_{L_i}, (i = 1, 2, \ldots, N)$ to be the conference leader. Also, participant i, after being selected as the conference leader, his/her GMOS_{L_i} should satisfy the following equation:

$$GMOS_{L_i} = \arg \max_{k=1,2,\ldots,N} GMOS_{L_k}. \tag{2.2}$$

Asking participants to provide GMOS on the overall quality of voice conference is a subjective test and it is difficult to obtain the GMOS before or during the conference. Instead, we first estimate the MOS from the network traffics (e.g., packet loss rate, jitter, codec,..etc) during a voice session. We also need to estimate GMOS. We propose the *two-step mapping method*(TSMM) to estimate the leader and participants' GMOSes. This will be describe in Section 2.4.

2.3 Experiment Description

Our experiments are separated into two parts. In the first part, the aim is to validate the GMOS(MOSes, α) and to determine the default and the average values of α. These experimental results were shown in Section 2.1. In the second part of the experiment, it is to verify the effectiveness of the *leader selection strategy*(LSS) proposed in Section 2.2.

In both parts of the experiments, the network setting is represented in Figure 2.4. All computers in the experiments are equipped with Intel P4 CPUs and at least 512M Memory with 10M/100M Ethernet network card. Three of these computers are located in Shanghai China, accessing the Internet via ADSL of Shanghai Telecomm while the other computer is in Shanghai

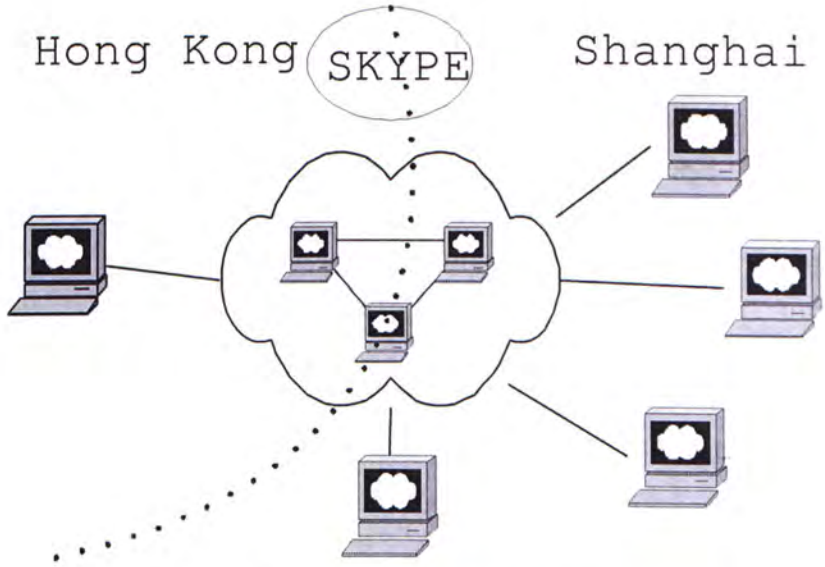


Figure 2.4: Network setting of SKYPE conference

assessing the Internet through the campus LAN in the Shanghai Jiao Tong University. The computer in Hong Kong is connected through the campus LAN in the Chinese University of Hong Kong. These computers are installed with SKYPE [1] (Version 2.5), professional audio recording and processing software Audition [9] (Version 1.5), and measurement tool Ethereal [8] (Version 0.99).

In the first part of the experiment, we asked people to have voice conference using SKYPE [1] and every participants used Audition [9] to record voice of the communication session. The duration of each conference was around 30 seconds per person, e.g., if there were four persons in the conference, it would last for about two minutes. The reason why we set the length of recording longer than that of the voice session described in [17] is that we consider the subjects who are invited to listen to the records and give the MOSes (to each individual speaker) and GMOS (to the whole conference) need more time to distinguish different speakers' voices.

In the second part of the experiments, we not only asked participants to have voice conference and performing the recording work, but also organized them to change the conference leader in turn. For instance, when we are going to have a 3-person conference, e.g., Alice, Bob and Cathy. We will ask them to do three experiments in which each of them being the conference leader once in turn. Lastly, the group of N experiments for our second part were carried out in short duration so that these experiments could be considered as operating under the same traffic condition and the results could be more accurate.

In order to evaluate the *two-step mapping method*(TSMM) and further validate the *leader selection strategy*(LSS) proposed in section 2.2, we need to measure some network parameters during the voice conference. Through packet trace by Ethereal [8], one can obtain the statistics such as *bit rate*, *jitter*, *loss rate* (under random loss model) and *loss rate* and *state transfer probability* (under 2-state Markov loss model [59]). We perform the Ping-like measurements between the leader and each non-leader participant separately during the conference to obtain the *Round Trip Time*(RTT) so that we can estimate the one way delay from the measured RTT. These measured and calculated statistics are inputs to the E-Model [23]. The frequency of the Ping-like measurement is set to 1 Hz, which is low enough not to affect the normal traffic generated by SKYPE and this is performed at the leader's computer to each non-leader participant's computer.

2.4 Data analysis and results

The results of the first part of the experiments are for GMOS model and parameter α, and we described them in Section 2.1 already. Here, we concentrate on the second part of the experiments: to evaluate the *two-step mapping method*(TSMM) and

validate the *leader selection strategy*(LSS) that we presented in previous section.

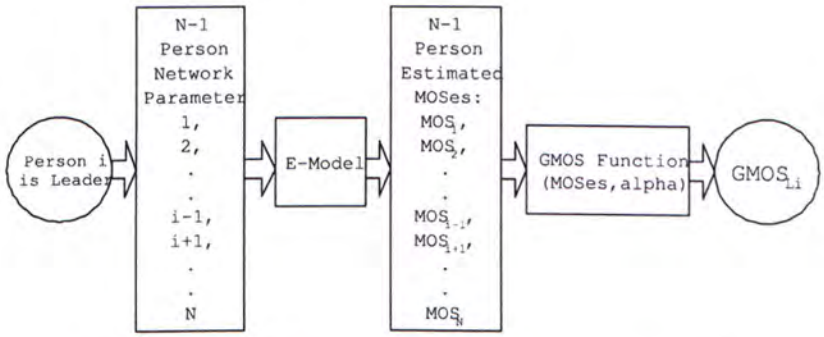


Figure 2.5: Architecture of the *two-step mapping method*

The *two-step mapping method*(TSMM) is used to calculate the estimated GMOS with inputs of network parameters between the leader and non-leader participants. Figure 2.5 depicts the architecture of the *two-step mapping method*(TSMM).

The first step of the *two-step mapping method*(TSMM) is to map the measured network parameters to estimate MOS. For this step, we apply the E-Model[23] since it is widely used for two-party communication[58, 59, 60, 52, 40, 49, 16].

For the E-model, the inputs are different classes of impairments which will affect the output R-value. The R-value is a real number between 1 to 100, which represents the quality of the 2-party communication. Some classes of the impairments are related to network condition such as the *effective equipment impairment factor* ($I_{e,eff}$), which comprises the effect of voice codec and packet loss, and the *delayed impairment factor* (I_d). Other impairments are not related to network conditions, for example, R_0, the initial value representing the *signal-to-noise ratio*(SNR) quality and I_s, which accounts for the effect of simultaneous problem etc. Based on the proposal in[30], we take default values for all impairments parameters defined in[23] except for $I_{e,eff}$ and I_d since they are related to network conditions.

The E-Model is then reduced to:

$$R = 93.2 - I_{e,eff} - I_d. \tag{2.3}$$

Once we know the value of R, we can find the MOS value via:

$$MOS = \begin{cases} 1 & \text{if } R < 0 \\ 4.5 & \text{if } R > 100 \\ 1 + 0.035R & \\ +7 * 10^{-6} R(R-60)(100-R) & \text{if } 0 \leq R \leq 100. \end{cases} \tag{2.4}$$

The remaining issue is to estimate $I_{e,eff}$ and I_d. As $I_{e,eff}$ is defined to comprise the effect of voice codec and packet loss [23, 59], we need to find the codec SKYPE uses before estimating $I_{e,eff}$. From [26, 63, 38], one will find that SKYPE uses iLBC [10, 24] or iSAC [11] codec, both of which are the products of GlobalIPSound [12]. Authors in[58] propose a method to calculate $I_{e,eff}$, which is suitable for us because it proposes how to set the constants in the formula when the speech codec is iLBC [10, 24]. In summary, $I_{e,eff}$ is

$$I_{e,eff} = I_e + (95 - I_e)\frac{Ppl}{Ppl + Bpl}. \tag{2.5}$$

In Equation (2.5), I_e and Bpl are two constants derived by authors in[58]. Table 2.2 shows part of the settings of the constants of different codec [59]. Here, Bpl is the packet loss robustness factor, i.e. the larger the Bpl, the smaller the effect of packet loss on the audio session. Ppl is the packet loss percentage which is expressed as

$$Ppl = ppl * 100\% \tag{2.6}$$

and ppl is the packet loss probability. For the parameter ppl, one can estimate it as an independent loss probability under the assumption that packet loss is a "random loss". Alternatively,

Table 2.2: Settings of I_e and Bpl for different codecs [59]

Codec	Tp[ms]	PLC	I_e	Bpl[%]
G.711	10	Sil.Ins	0	4.3
G.711	10	App.I	0	25.1
G.729A	20	Native	11	19.0
iLBC	30	Native	11	32.0

one can use a 2-state Markov loss model (or Gilbert model)[59] to model the bursty loss scenario. For the 2-state Markov loss model, it has two parameters p and q, with p being the transition probability from good state to loss state, while q is the transition probability from loss state to good state, and $pc = 1 - q$ is the conditional loss probability.

The model to calculate $I_{e,eff}$ under the 2-state Markov loss model is also proposed in [59]:

$$\begin{cases} I_{e,eff} & = I_e + (95 - I_e)\frac{Ppl}{\frac{Ppl}{BurstR}+Bpl}, \\ BurstR & = \frac{1-pc}{1-ppl}. \end{cases} \tag{2.7}$$

The $BurstR$ is the burst ratio of loss. If $BurstR < 1$, it means that when the previous packet was lost, it has a lower probability of losing the current packet. When $BurstR = 1$, it represents "random loss" and when $BurstR > 1$, it implies bursty loss.

We obtain the values of ppl, pc and $BrustR$ by analyzing the traffic data. We observe that the cases that $BurstR > 1$ occur with low frequency, and the measured $BrustR$ is slightly larger than 1, which implies that the packet loss scenarios during our experiments were mainly from random losses. Therefore, we apply Equation (2.5) to obtain the $I_{e,eff}$.

To estimate the impairment I_d, we use two approaches. One is letting I_d be the default value defined by [23] and ignoring end-to-end delay. The other one is to apply the formula with inputs RTT, which was obtained by our ping-like measurements.

We calculate the impairment I_d via:

$$\begin{aligned} I_d &= 0.024d + 0.11(d - 177.3)V(d - 177.3), \\ d &\approx RTT/2, \\ V(x) &= \begin{cases} 0 & x < 0, \\ 1 & x \geq 0. \end{cases} \end{aligned} \tag{2.8}$$

Since the delay condition will have different effects on systems implemented by different buffer strategies, and the mechanism that SKYPE uses to deal with end-to-end packet delay is unknown, we use both approaches to estimate I_d:

$$\begin{cases} R = 93.2 - [I_e + (95 - I_e)\frac{Ppl}{Ppl+Bpl}] - I_d(\text{default},) & \text{(a)} \\ \\ R = 93.2 - [I_e + (95 - I_e)\frac{Ppl}{Ppl+Bpl}] \\ \qquad -[0.024d + 0.11(d - 177.3)V(d - 177.3)]. & \text{(b)} \end{cases}$$

After obtaining the R-value using function (a) and (b) listed above, one can derive two different MOS values of the same pair of network parameters by applying Equation (2.4) with two different R values from (a) and (b). We call the process of mapping network parameters to MOS value as the first-step mapping function.

The second step of the *two-step mapping method*(TSMM) is to apply the estimated MOSes obtained from first-step mapping function into Equation (2.1) so as to get the estimated GMOS. Since $\alpha = 0$ is the default value and $\alpha = 0.093$ is the average of all 392 samples obtained from our experiments. We experiment with these two values in the second-step mapping function. As a result, we have four types of the combination of applying the *two-step mapping method*(TSMM). Table 2.3 summarizes these four types. Note that these four types are actually objective methods to estimate the MOSes and GMOS given by various

Table 2.3: Summary of 4 types of the *two-step mapping method* (TSMM)

Method	R fitting model	α
M1	(a)	0.093
M2	(b)	0.093
M3	(a)	0
M4	(b)	0

participants. So in order to evaluate and analyze the performance of the *two-step mapping method*(TSMM) and the *leader selection strategies*(LSS), we invite participants to listen to the audio clips which were recorded from the conference leader's computer and to provide MOSes and GMOS scores just like what have been done in the first part of the experiments. The results are shown in Figure 2.6 to Figure 2.8.

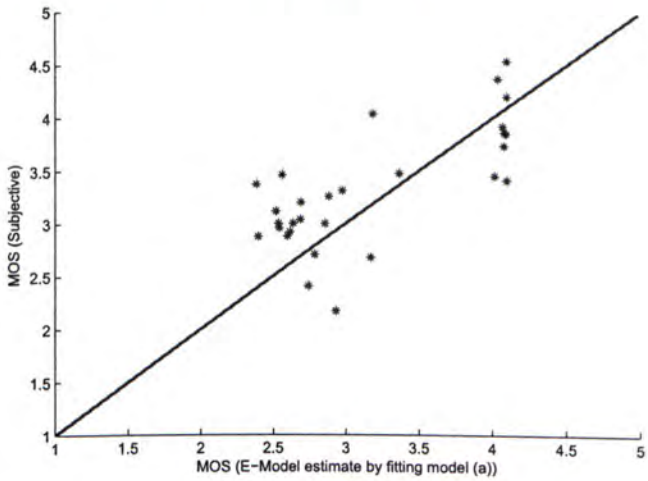


Figure 2.6: Mapping from MOS (E-Model) of first-step mapping model (a) to MOS (Subjective)

Figure 2.6 shows the estimated MOS by the first-step mapping function (a) with the MOS which is actually the average value of all the integer MOSes (1 to 5) scored by subjects. Figure 2.7 shows the estimated MOSes by the first-step mapping

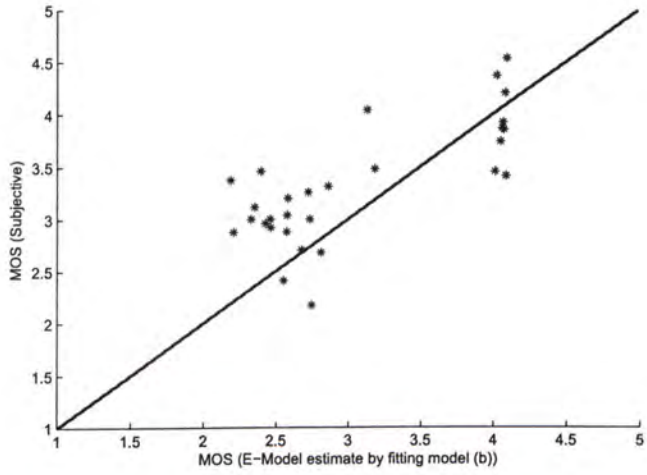


Figure 2.7: Mapping from MOS (E-Model) of first-step mapping model (b) to MOS (Subjective)

function (b) with subjective MOSes. Here, we use the following measure to analyze the mapping results:

$$\text{Average}(\Delta y) = \frac{\Sigma_{i=1}^{N} |\hat{y}_i - y_i|}{N} \tag{2.9}$$

In Equation (2.9), y_i is the value of mapping results and $\hat{y}_i$ is the expected value with the same x_i. For results in Figure 2.6 and Figure 2.7, the expected values are on the line of $y = x$ because the mapping is from MOS (E-Model [23], objective) to MOS (Subjective). Through Equation (2.9), we derive the average(Δy) of fitting function (a) is 0.4282 and average(Δy) of fitting function (b) is 0.4755.

Figure 2.8 illustrates the performance of these four types of *two-state mapping method*(TSMM). We apply the average(Δy) in Equation (2.9) again. Here, y_i is the result obtained by applying the four approaches of the *two-step mapping method*(TSMM) and $\hat{y}_i$ is the subjective GMOS. One can observe from Table 2.4 that the difference between GMOS scored by subjects and that obtained by four approaches of *two-step mapping method*(TSMM)

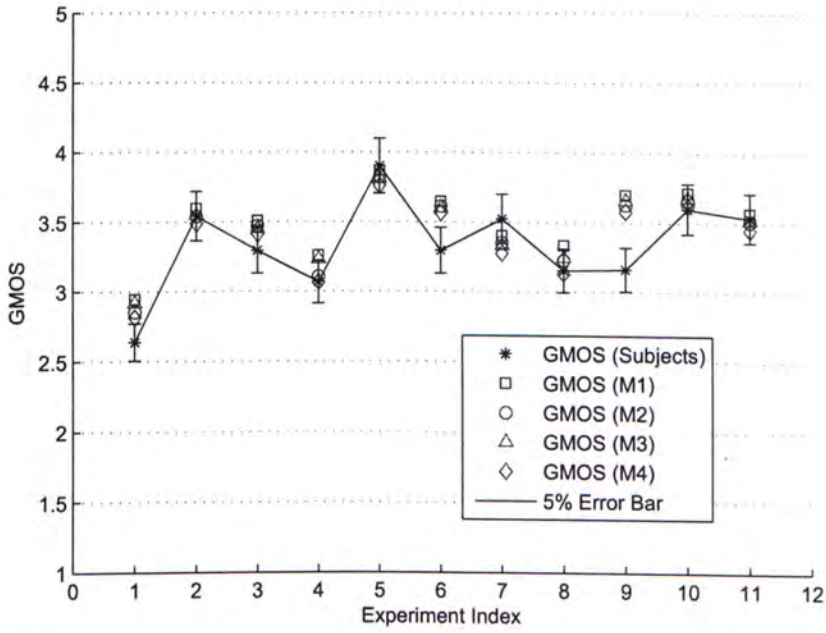


Figure 2.8: 4 types of Two-State Mapping Method with GMOS (Subjective)

is very small. It means the objective method we proposed (TSMM) to estimate the subjective GMOS works quite well.

Table 2.4: Illustration of average(Δy)

	M1	M2	M3	M4
Average(Δy)	0.1948	0.1486	0.1712	0.1417

Next, we check the correctness of the *leader selection strategy*(LSS) proposed in Section 2.2. Table 2.5 illustrates the GMOS given by subjects and the estimated GMOS derived from these four types of *two-step mapping method*(TSMM). These results are arranged into experiment groups. In order to check whether the selection of the leader by the *leader selection strategy*(LSS) is efficient or not, we perform an N-person conference experiment through SKYPE for N times so that each participants has the chance to be a leader for one time. This N times experiment with the same participants are called the ex-

Table 2.5: Illustration of Leader selection results of GMOS (Subjective) and Estimated GMOS by 4 types of the *two-step mapping method* (TSMM)

#	N	Persons and Host	GMOS (Sub)	GMOS (M1)	GMOS (M2)	GMOS (M3)	GMOS (M4)
1	3	'A(H)', 'B', 'C'	2.637	2.945	2.853	2.917	2.815
2	3	'A', 'B(H)', 'C'	3.541*	3.597*	3.541*	3.548*	3.486*
3	3	'A', 'B', 'C(H)'	3.293	3.507	3.471	3.450	3.411
4	4	'A(H)','B','C','D'	3.065	3.258	3.106	3.231	3.064
5	4	'A','B(H)','C','D'	3.900*	3.866*	3.824*	3.806*	3.760*
6	4	'A','B','C(H)','D'	3.295	3.647	3.614	3.600	3.565
7	4	'A','B','C','D(H)'	3.523	3.403	3.351	3.331	3.275
8	4	'J(H)','K','L','M'	3.152	3.333	3.227	3.246	3.129
9	4	'J','K(H)','L','M'	3.162	3.699	3.628	3.657	3.581
10	4	'J','K','L(H)','M'	3.600*	3.714*	3.665*	3.674*	3.622*
11	4	'J','K','L','M(H)'	3.536	3.570	3.509	3.517	3.451
		Correct / Total	–	3/3	3/3	3/3	3/3

periment group. We consider the participant who receives the largest GMOS_{L_i} given by all subjects in condition that he is the leader, is the correct server to be the leader. If any of these four types selects the same person as the leader, this shows a correct selection, otherwise an incorrect selection was made.

In summary, we have carried out three groups of conference experiments, two of which are 4-person conference and one is a 3-person conference. The GMOS_L value in Table 2.5 with a star "$*$" is the largest value of GMOS_L in each group. The person with the largest GMOS_L should be the leader according to the *leader selection strategy*(LSS). The leader selected by the "GMOS (Sub)" column in which the GMOS_L is given by subjects is considered as the correct one because it comes from subjective tests. "GMOS_L" in the last four columns comes from

the four types of applying the *two-step mapping method*(TSMM) to estimate the GMOS_L and selects the leader by the *leader selection strategy*(LSS). The results show that all the four types of applying the two-step mapping method have selected the *correct leader* in each group of conference experiments.

2.5 Applications of Proposals to Voice Conference

As we have discussed in the previous sections, our GMOS metric can be implemented to evaluate the overall quality of voice conference so that we can know how good the providing service is, and whether it can be improved from the provider view as well. The GMOS we propose can be used to evaluate many voice quality applications, e.g., USI[32].

The *leader selection strategy*(LSS) we propose can be applied in several ways:

a) Before N persons start a voice conference, the software can measure traffic between any two participants so as to estimate the network parameters. Then it can utilize the *two-step mapping method*(TSMM) to estimate $GMOS_{L_i}$ and then apply the *leader selection strategy*(LSS) to select the proper leader.

b) During a voice conference, the $GMOS_{L_i}$ could be an indicator to reveal the overall quality of the whole conference. And the software can maintain a light-weight testing traffic to select a leader candidate by the *leader selection strategy*(LSS) among the $N-1$ non-leader participants so that when the current conference gets disconnected due to some unforeseen network condition, it can restart with a new and proper conference leader.

□ **End of chapter.**

Chapter 3

P2P Application Identification

Summary

In this chapter, we describe the second part of the work, the method of identifying P2P applications based on behavioral-signature. we first describe a few generic types of periodic group communication behaviors commonly observable in P2P applications, and explain why such behavior exists in Section 3.1; we then explain the frequency-domain analysis that helps the identification in Section 3.2. In Section 3.3, we show the behavioral signatures we developed for a number of popular P2P applications, by individually observing their behavior in isolation. Then we evaluate how useful these signatures are in identifying P2P application (hosts or flows) from two type of traffic traces: (a) a campus traces, and (b) an ISP trace in Section 3.4. Finally we discuss the findings and future directions in Section 3.5.

3.1 Periodic Group Communication Patterns

P2P applications, whether it is to share files, streaming content, provide Voice-over-IP service or others, all need to form an overlay for peers to reach other peers. In order to form and maintain this overlay, and often in the use of this overlay, peers inevitably exhibit periodic group communication.

Naturally, we distinguish between two classes of periodic group communication patterns: (a) *control plane* - that used to form and maintain the overlay; (b) *data plane* - that used to multicast content. This terminology comes from the description of router activities. Routers also exhibit the same type of periodic group communication patterns. For example, when running RIP protocol, routers exchange routing tables with neighboring routers with a default timer of 30 seconds; when running OSPF, routers flood link state packets to neighbors every 60 seconds by default. Furthermore, in both routing protocols, neighbors exchange HELLO messages periodically. When routers implement multicast, group communication would be observed in the data plane, although any periodicity would depend on the application that is doing the multicasting.

In P2P systems, especially those P2P systems doing application layer multicasting, there are basically two kinds of overlays formed:

- Structured overlays: This includes overlays with mesh-based topology, such as ESM[41], and tree-based topology such as NICE[25], Yoid[39] and SCRIIBE[31]. In ESM, group members periodically generate refresh message and exchange their knowledge of group membership with their neighbors in the mesh. Similarly for tree-based topologies, peers also periodically refresh the overlay links so as to maintain the *soft state* information.

- Data-driven overlays: The classic example is Bittorrent[34].

In this case, the topology is more dynamic, driven by which neighbors have the right content needed by a peer. Such dynamic P2P systems are normally bootstrapped by a server known as the *tracker*. All peers may need to periodically update its information to the tracker for system health monitoring.

In both kinds of overlays, other active measurements may be used to optimize the efficiency of the overlay. For example, neighboring peers may periodically measure the distance (in terms of round trip time) between each other. In summary, these activities generate periodic group communication patterns.

3.1.1 Terminology for Behavioral Patterns

In the following, we describe three specific periodic group communication patterns that are common for many P2P applications, and are going to be used by our proposed behavioral signatures. Note that these three patterns of periodic group communication behaviors are just examples to illustrate our methodology. The particular values of periodicity of different behaviors are application dependent. A given P2P application may exhibit one or more of such behaviors.

In doing so, we first need to define some terminology in our framework.

First, time is divided into discrete time intervals. The length of the time interval is quite critical in the ability to identify the periodic behavior, and needs to be carefully chosen. Unless we state otherwise, the length of the time interval is set as 1 second for all our experiments.

For each host, it communicates with a number of neighbors. Such communications are organized into a sequence of flows, similar to the flows definition in Netflow, although the inactivity interval that starts and ends a flow is application behavior

dependent. The start and end of a flow is indicated with a Start Event (SE) and an End Event (EE), each event has an associated time. All flows are unidirectional. At any moment in time, each host has a number of flows. Some are considered *in-flows*, which are flows with destined to the considered host; others are *out-flows*, which are flows source-addressed at the considered host. These in and out flows can be thought as active links between pairs of overlay nodes.

3.1.2 Pattern 1: Gossip of Buffer Maps

A popular type of P2P applications is P2P streaming based on a data-driven overlay. This includes live streaming applications such as PPLive[4], PPStream[5], CoolStreaming[65] and P2P Video-on-Demand (VoD) streaming systems[5, 13, 14]. Because of the data-driven approach for forming and maintaining the overlay, all these applications rely on gossip of buffer maps to maintain active links between peers. Usually, the size of the buffer map is not large and can be represented using one or two packets.

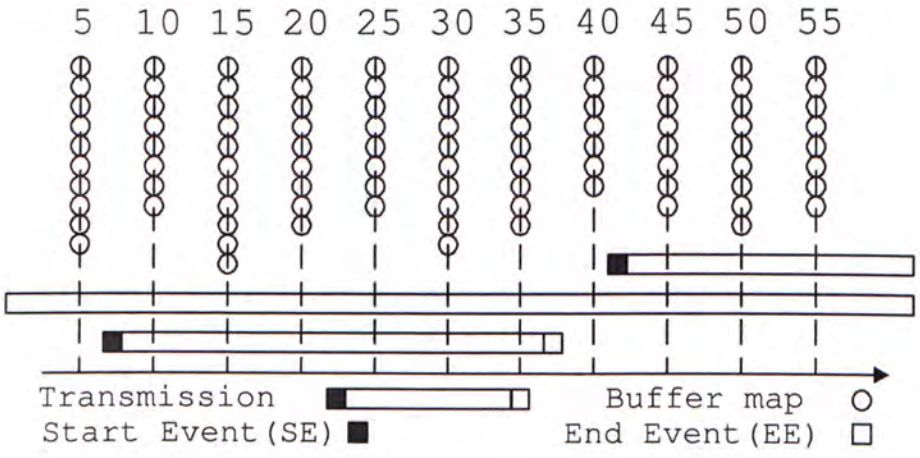


Figure 3.1: An example of traffic pattern of P2P Live Streaming

For P2P Live Streaming systems, because peers only store

the streaming content in their RAM (much smaller size compared with VoD) and remove the content as soon as it is played back, the buffer map information changes rather quickly. Each peer needs to periodically exchange its buffer map with neighbors to optimize the scheduling of content exchange to ensure good playback performance. The buffer map exchange period is as short as 5 seconds for some cases. Note, each peer must exchange this information with all its neighbors although it only exchanges content with a subset of these neighbors. For P2P VoD systems, peers also exchange buffer maps with their neighbors periodically, although the period may be longer. Figure 3.1 is an illustration of the traffic pattern for P2P Live Streaming systems. In this figure, we have three flows, e.g., there is a flow which starts right after $t = 5$ and ends between $[35, 40]$. The figure also shows the periodicity property. For example, every 5 seconds, this node sends out some packets to its neighbors (say at $t = 5$, this node sends out 10 packets which describe its buffer maps to its neighbors).

Since this traffic pattern does not involve content, we consider it to be information exchange in the control plane.

3.1.3 Pattern 2: Content flow control

The second pattern occurs in the data plane. For streamed video content, it can often reach that peers are faster than the playback rate, behaving much like file downloading. Although this is good for the peers, the content provider actually prefers the peers download at the pace of playback, to ensure all the peers stay around to help the server distribute content, rather than watch the content off-line. Content providers thus implement various mechanisms to make peers continue to contribute. One way is to make peers periodically send keep-alive messages to a tracker when they are watching the video, even after the whole

video has finished downloading (e.g. in the VoD case[42]). Another way is to perform the Pre-Downloading Control (PDC), which is a form of content flow control to make the download rate match with the playback rate. Such flow control often results in alternating bursts of download activities and sleep periods, as illustrated in Figure 3.2.

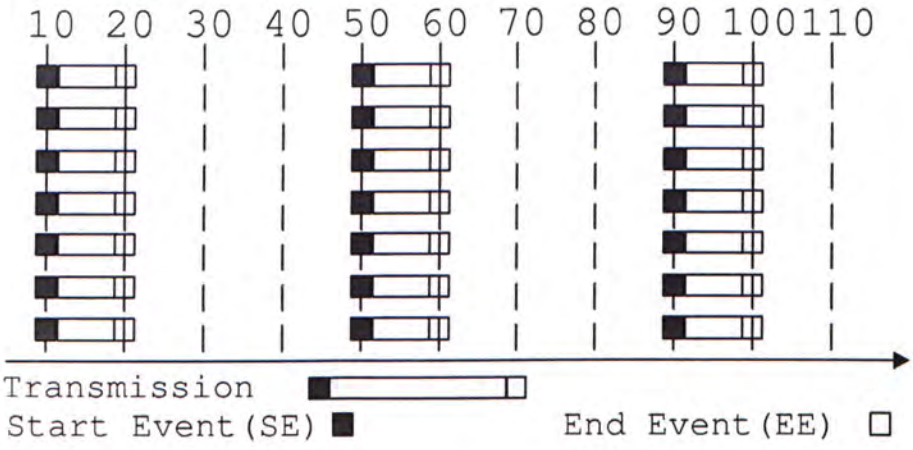


Figure 3.2: An example of traffic pattern of P2P VoD system

For example, PDC is deployed by PPStreamVoD[5] to keep peers contributing to the system. The actual implementation may decide to exert flow control every 40 seconds. At the beginning of the control interval, content is downloaded at full-speed. Suppose there is sufficient bandwidth to finish downloading 40-second worth of content in 10 seconds. As a result, we will see the download activity proceed for 10 seconds followed by 30 seconds of sleep, as shown in the figure.

3.1.4 Pattern 3: Synchronized Link Activation and Deactivation

It is well-known that BitTorrent implements the *tit-for-tat* mechanism to provide incentives for peers to serve each other. The third pattern of periodic group communication behavior is a

direct consequence of how BitTorrent-like protocols might implement the tit-for-tat mechanism. As described in [34, 47], each peer uses two timers (10 seconds and 30 seconds) to decide whether to choke and optimistically unchoke neighboring peers, respectively. This results in the synchronization of StartEvents and EndEvents at the beginning of the time intervals, as illustrated in Figure 3.3.

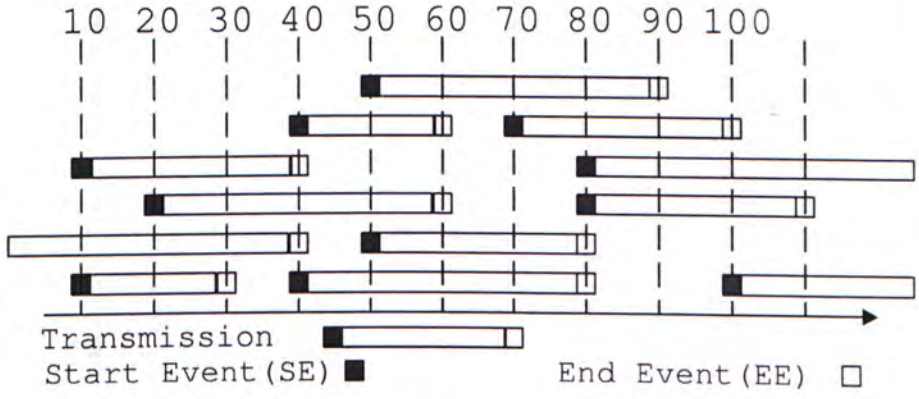


Figure 3.3: An example of transmission pattern of BitTorrent

As shown, the choking (every 10 seconds) and optimistic unchoking (every 30 seconds) actions always occur at the starting point of time intervals.

3.2 Identification Based on behavioral signatures

In this section, we describe our behavioral-signature based P2P application identification algorithm. The goal of our algorithm is to detect specific P2P applications (e.g., BitTorrent[34], PPLive[4] and PPStream[5]) from a traffic trace containing a mixture of traffic.

In Section 3.1, we describe several behavioral traffic patterns observable from P2P applications. Furthermore, we gave the specific P2P application designs that lead to these behavioral

patterns. In order to use these patterns for P2P application identification, we make the following two assumptions:

1. Most P2P applications have periodic behaviors while *other types* of applications (e.g., Web, FTP and Email) do not.
2. The types of periodic behaviors and the particular value of the periodicity are application specific.

In the next sections, we will apply our algorithm to real life traffic traces, to see if these assumptions hold.

3.2.1 Algorithm Overview

Generally speaking, the algorithm involves the following steps:

1. Filtering - to extract related traffic sequence
2. Sequencing - to convert related traffic sequence to a discrete time sequence
3. Transforming - to transform the time domain sequence to a frequency domain sequence
4. Analyzing - to determine whether the original traffic sequence match a behavioral signature

This whole process is illustrated in the block diagram found in Figure 3.4.

Note, the process is shown as an iterative process. If the algorithm is applied to detect hosts that are running certain P2P applications, then the filtering step tries to extract all the traffic related to a particular host; and the outcome of the analysis tells whether the target host is running the suspect P2P application or not. The iterative process shows that the same steps can be applied to many target host sequentially.

The three traffic patterns described in the last section are all patterns found with respect to a specific host. Our methodology

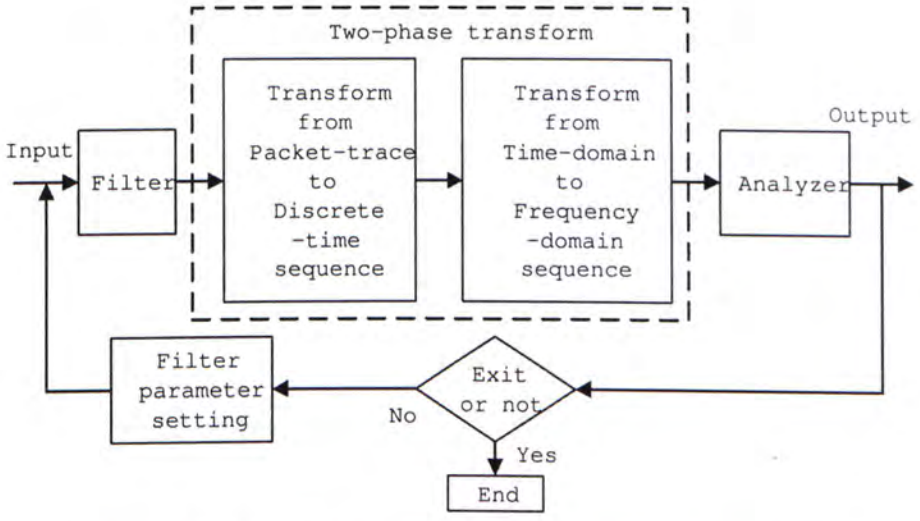


Figure 3.4: The block diagram of the identification process.

can be applied to objects of other granularity, e.g. a flow, or a group of hosts. But for the rest of this chapter, we are only concerned with detecting behavior for a target host.

As shown in the block diagram, we group step two and step three together as a *two phase transformation*. The first phase transforms the related traffic into a time series of values that can exhibit periodicity. This step is non-trivial and is where the secret of each behavioral signature is. The three behavioral patterns we described in the last section will yield three different time series. The second phase then transforms each time sequence into its frequency domain representation for ease of analysis. This is a common technique in detecting periodic behavior in analyzing signals in communications systems. For all three behavioral signatures, this step is the same.

Next, we explain how each of the three traffic patterns are turned into specific time series for signature matching.

3.2.2 Sequence Generation (SG1): Time Series for the Gossip Pattern

Recall that in our basic model, time is divided into intervals, of size T say. The filtering step has already yielded us a sequence of packets all related to a particular target host, say H. The time series we are generating can be denoted by $X_{in}[i]$ and $X_{out}[i]$, where i stands for the i^{th} time interval; and $X_{in}[i]$ stands for the number of in-flows (or in links) the target host has during the i^{th} interval; and $X_{out}[i]$ is correspondingly the number of out-flows (or out links) this target node has.

In other words, the target host is suspected of gossiping with a number of other hosts. The output $X_{in}[i]$ and $X_{out}[i]$ give us the number neighbors the target host is gossiping with over time intervals i. If we lump the uni-directional gossip into bi-directional gossip, then the time series can be merged into one series $X[i] = X_{in}[i] + X_{out}[i]$. The pseudo-code of this algorithm is given in Figure 3.5.

As an example, Figure 3.6 shows the value of $X_{in}[i]$ and $X_{out}[i]$ for a PPLive streaming session.

3.2.3 Transform Time-domain Sequence to Frequency-domain Sequence

Before describing how to generate the time domain series for the other two traffic patterns, let us describe how to transform the time-domain sequence into a frequency domain sequence. Again, this step is the same for all three behavioral patterns.

The input to this phase is the time-domain sequence, e.g., $X_{in}[i]$ and $X_{out}[i]$ as shown in Figure 3.6. In this phase, we first apply the Autocorrelation Function (ACF) and then apply the Discrete Fourier Transform (DFT).

The purpose of the Autocorrelation function is for finding periodic patterns in a signal, where the periodic pattern might

```
Input: Q //a queue of the output packet trace from Filter step in order.
       T //time interval of the sequence.
       targetHost //All the packets in Q is related to the targetHost
Output: Xin[i], Xout[i] .
1. int Start_ts = Head(Q).Timestamp/ T;
2. int End_ts = Tail(Q).Timestamp/ T;
3. int i = 1, ts = Start_ts;
4. Queue In, Out;
5. While (ts <= End_ts ) begin
6.   In.Clear(); Out.Clear();
7.   While( !Empty(Q) and Head(Q).Timestamp<= ts ) begin
8.        p = Dequeue(Q);
9.        if ( p.sourceHost== targetHost and !Out.Contain(p) ) begin
10.            Out.Add(p);
11.       end
12.       if ( p.destinationHost== targetHost and !In.Contain(p) ) begin
13.            In.Add(p) ;
14.       end
15. end
16. Xin[i] = Length(In);
17. Xout[i] = Length(Out);
18. if ( !Empty(Q) ) begin
19.       i++ ; ts = ts + T;
20. else
21.       break;
22. end
23.end
```

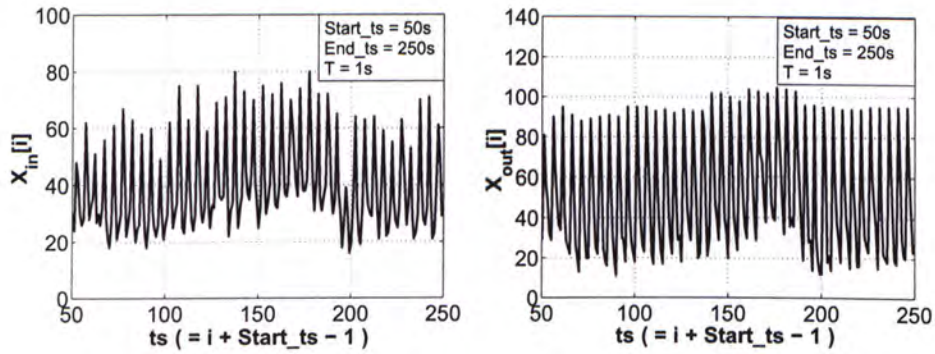
Figure 3.5: The algorithm to generate the sequence of number of concurrent flows of the target host

be buried under noise [27]. The ACF computation is based on Equation (3.1), where n is the lag of autocorrelation range in $[0, N-1]$, and N is the length of the sequence $X[i]$[1]. The result of the ACF function is still a sequence, denoted $r(n)$.

$$r(n) = \frac{1}{N-n} \sum_{i=1}^{N-n} X(i)X(i+n). \tag{3.1}$$

Finally, we apply the Discrete Fourier Transform, as shown in Equation (3.2) on $r(n)$ and get the frequency-domain sequence

[1]Here, $X[i]$ is any time series. It could be X_{in}, or X_{out} or the sum of the two.



(a) $X_{in}(i)$ (b) $X_{out}(i)$

Figure 3.6: Sequence $X_{in}[i]$ and $X_{out}[i]$ generated from the measurement of a host running PPLive Live Streaming Application.

$\mathbf{R}(k)$.

$$\mathbf{R}(k) = \sum_{n=0}^{N-1} r(n) e^{-\frac{2\pi i}{N} kn} \quad \text{where } k \in [0, N-1]. \tag{3.2}$$

The sequence, $\mathbf{R}(0), \mathbf{R}(1), \ldots, \mathbf{R}(N-1)$ is generally a sequence of N complex numbers (see [56]). For matching against the behavioral signatures, it is enough for us to take modulo of $\mathbf{R}(k)$ to get the magnitude of each frequency component:

$$\widehat{\mathbf{R}(k)} = |\mathbf{R}(k)|. \tag{3.3}$$

Let us illustrate this process by an example. We first apply ACF on both $X_{in}(i)$ and $X_{out}(i)$ shown in Figure 3.6(a) and 3.6(b). This gives us the two resultant sequence $r_{in}(n)$ and $r_{out}(n)$ as plotted in Figure 3.7(a) and 3.7(b). The length of both $r_{in}(n)$ and $r_{out}(n)$ are $N = 200$. Since the sequence repeats itself, we just show part of the sequence in each case.

The result of DFT, $\widehat{\mathbf{R}_{\mathbf{out}}(\frac{\mathbf{k}}{\mathbf{N}})}$ and $\widehat{\mathbf{R}_{\mathbf{in}}(\frac{\mathbf{k}}{\mathbf{N}})}$, are plotted in Figure 3.7(c) and 3.7(d). Here we need to explain some properties of DFT[56]:

a. If the input sequence to DFT is a real sequence, e.g., $r(n)$ is a real sequence, the output complex sequence $\mathbf{R}(k)$ will

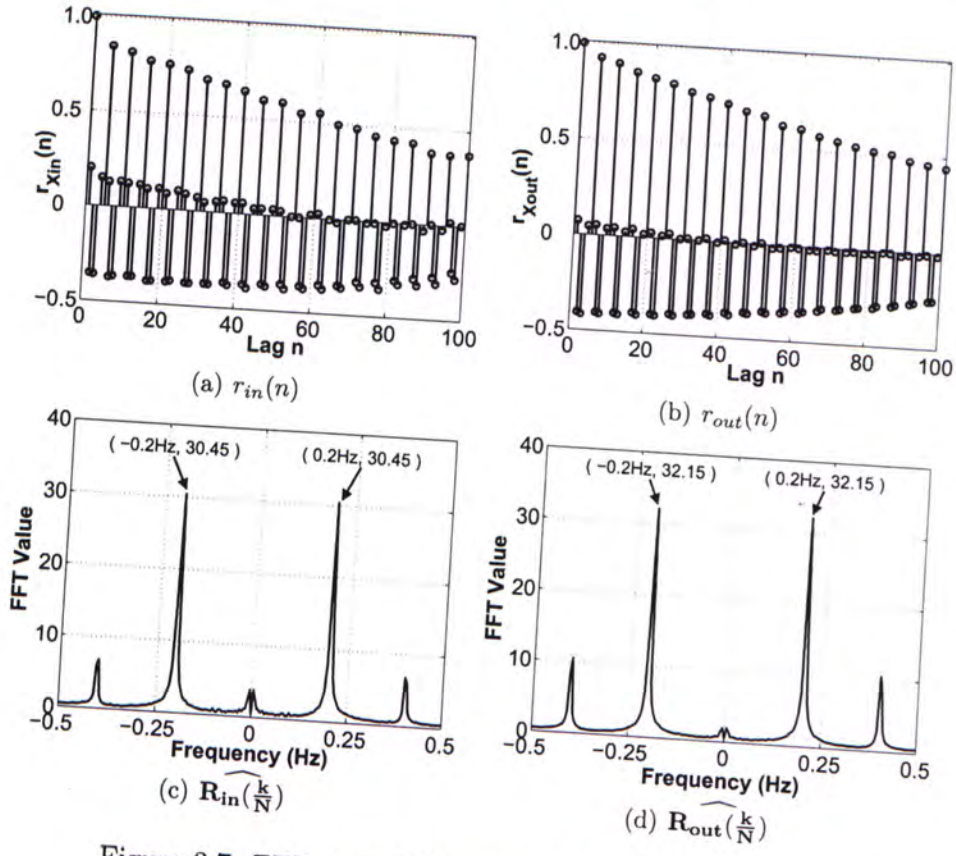


Figure 3.7: FFT and ACF results of $X_{in}[i]$ and $X_{out}[i]$.

satisfy:

$$|\mathbf{R}(-k)|_{mod\ N} = |\mathbf{R}(k)|. \tag{3.4}$$

In fact, the periods N is the common characteristic of all DFT result sequence. Eq. (3.4) says that the module of a real sequence after DFT is also symmetric within a period N. This explains why $|\mathbf{R}_{in}(\frac{k}{N})|$ and $|\mathbf{R}_{out}(\frac{k}{N})|$ are symmetric around the line $k/N = 0$ between $[-N/2, N/2]$ in Figure 3.7(c) and 3.7(d).

b. If the input sequence to DFT, $r(n)$, is a finite sequence, its DFT, $\mathbf{R}(k)$, will satisfy [56]:

$$\mathbf{R}(k) = \mathbf{R}(\Omega)|_{\Omega = 2\pi\frac{k}{N}} = \mathbf{R}(\frac{2\pi k}{N}),$$

$$\Omega = 2\pi f = 2\pi \frac{k}{N} \quad \Rightarrow \quad f = \frac{k}{N} \tag{3.5}$$

Eq. (3.5) gives the relationship between the Discrete Fourier Transform and Continuous Fourier Transform when the original sequence is finite. We apply this result and plot the X-axis in continuous frequency value in Figure 3.7(c) and 3.7(d).

3.2.4 Sequence Generation (SG2): Time Series for Content Flow Control Pattern

So far we have only shown how to transform the filtered traffic patterns into a time series for the first behavioral pattern (Gossip), and then turn the time series into a frequency domain sequence. We now continue with the other two behavioral patterns.

Recall the content flow control traffic pattern is about the rate a target node is sending or receiving content from all its neighbors. We represent these as two time series, $Y_{in}[i]$ and $Y_{out}[i]$.

The algorithm of this sequence generator is shown in Figure 3.8. The algorithm of this sequence generator is very similar to the one described in 3.2.2. Y_{in} and Y_{out} are used to accumulate data rate, rather than flow count in and out of the target node. After the time series for this behavioral pattern is generated, again ACF and DFT are applied to the time series. The corresponding results are shown in Figure 3.9.

Here,

$$\widehat{\mathbf{R}_{Y_{in}}(\frac{k}{N})} = |\mathbf{R}_{Y_{in}}(\frac{k}{N})|.$$

$$\widehat{\mathbf{R}_{Y_{out}}(\frac{k}{N})} = |\mathbf{R}_{Y_{out}}(\frac{k}{N})|.$$

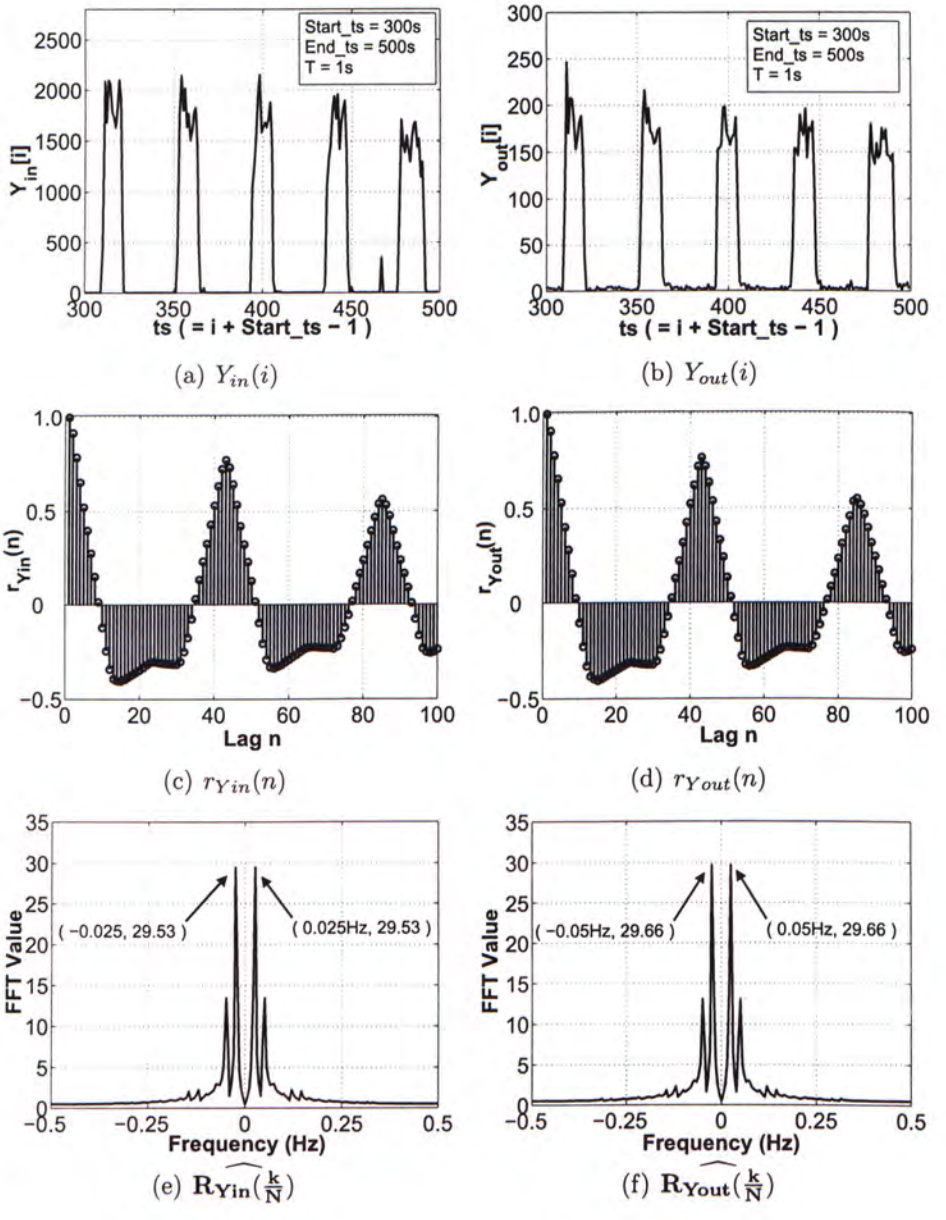
```
Input: Q //a queue of the output packet trace from Filter step in order.
       T //time interval of the sequence.
       targetHost //All the packets in Q is related to the targetHost
Output: Yin[i], Yout[i];
1. int Start_ts = Head(Q).Timestamp/ T;
2. int End_ts = Tail(Q).Timestamp/ T;
3. int i = 1, ts = Start_ts;
4. long InPacketSize, OutPacketSize;
5. While (ts <= End_ts) begin
6.   InPacketSize = OutPacketSize = 0;
7.   While( !Empty(Q) and Head(Q).Timestamp<= ts ) begin
8.        p = Dequeue(Q);
9.        if ( p.sourceHost== targetHost) begin
10.            OutPacketSize += p.packetSize;
11.       end
12.       if ( p.destinationHost== targetHost) begin
13.            InPacketSize += p.packetSize;
14.       end
15.  end
16.  Yin[i] = InPacketSize * 8;
17.  Yout[i] = OutPacketSize * 8;
18.  if ( !Empty(Q) ) begin
19.       i++ ; ts = ts + T;
20.  else
21.       break;
22.  end
23.end
```

Figure 3.8: The algorithm to generate the sequence of data transmission rate per unit time interval

3.2.5 Sequence Generation (SG3): Time Series for Synchronized Start and Finish of Flows

In BitTorrent-like applications, due to the periodic choking and optimistic unchoking mechanism, the occurrences of the flow StartEvent (SE) and EndEvent(EE) will also have the periodicity (Refer to Figure 3.3). This time, the results will be accumulated in time series $Z_{in}[i]$ and $Z_{out}[i]$. This algorithm is slightly more complicated than the first two algorithms, and is represented by pseudo code in Figure 3.10.

There are three parts to this algorithm. From line 1 to line



(a) $Y_{in}(i)$ (b) $Y_{out}(i)$

(c) $r_{Yin}(n)$ (d) $r_{Yout}(n)$

(e) $\widehat{\mathbf{R_{Yin}}(\frac{\mathbf{k}}{\mathbf{N}})}$ (f) $\widehat{\mathbf{R_{Yout}}(\frac{\mathbf{k}}{\mathbf{N}})}$

Figure 3.9: $Y_{in}[i]$, $Y_{out}[i]$, ACF and FFT results.

15 (Figure 3.10) is the first part. It reorganizes the input packet trace Q, related to the *targetHost* and sorted in ascending order of *Timestamp*. Then all the neighbors of the target host are put into the HostList data structure: if the neighbor is the

```
Input: Q //a queue of the output packet trace from Filter step in order.
       T //time interval of the sequence.
       targetHost//All the packets in Q is related to the targetHost
       effective_threshold//Minimum number of packet a host should own.
       interval_threshold//Packet interval between two session.
Output: Zin[i], Zout[i];
//Pakct trace reorganized into flow-based
HostList hostIn, hostOut ;
1. While(!Empty (Q)) begin
2.     p = Dequeue(Q);
3.     if ( p.sourceHost == targetHost) begin
4.           if ( !hostOut.Contain (p.destinationHost) ) begin
5.             hostOut.Add (p.destinationHost) ;
6.           end
7.           hostOut[Position(p.destinationHost)].packetQueue.Add(p) ;
8.     else
9.           if ( !hostIn.Contain (p.sourceHost) ) begin
10.            hostIn.Add (p.sourceHost) ;
11.          end
12.          hostIn[Position(p.sourceHost)].packetQueue.Add(p) ;
13.    end
14. end
15. EventlistQueue eventIn, eventOut//Establish events
16. for each host in hostIn begin
17.    if (Length(host.packetQueue) >= effective_threshold ) begin
18.    lastPacket = Head(host.packetQueue) ;
19.    //Start Event will be trigerred when first packet is checked.
20.    eventIn.Add ( SE, lastPacket.Timestamp ) ;
21.    while( !Empty(host.packetQueue) ) begin
22.          p = Dequeue(host.packetQueue) ;
23.          if ( (p.Timestamp - lastPacket.Timestamp) > interval_threshold ) begin
24.            eventIn.Add ( EE, lastPacket.Timestamp ) ;
25.            eventIn.Add ( SE, p.Timestamp ) ;
26.          end
27.          lastPacket = p ;
28.    end
29.    //End event will be trigerred when last packet is checked.
30.    eventIn.Add ( EE, lastPacket.Timestamp ) ;
31.    end
32. end
33. Repeat the procedure Line 16. to Line 32. on hostOut and eventOut.
34. Sort ( eventIn, "Timestamp" ) ;
35. Sort ( eventOut, "TimeStamp" ) ;
36. //Generate sequence
37. int Start_ts = Head(eventIn).Timestamp / T ;
38. int End_ts = Tail(eventIn).Timestamp / T ;
39. int i = 1, ts = Start_ts ;
40. int eventCounter;
41. While (ts <= End_ts ) begin
42.    eventCounter = 0;
43.    While( !Empty(eventIn) and Head(eventIn).Timestamp <= ts ) begin
44.          ev = Dequeue(eventIn);
45.          eventCounter ++ ;
46.    end
47.    Zin[i] = eventCounter ;
48.    if ( !Empty(eventIn)) begin
49.          i++ ; ts = ts + T;
50.    else
51.          break;
52.    end
53. end
54. Repeat the procedure Line 37. to Line 53. on eventOut and derive Zout[i].
```

Figure 3.10: The algorithm to generate the sequence of transmission session Start and End Event of the target host

destination, then it is put into *hostOut*; or if the neighbor is the source, then it is put into $hostIn$. The target host is considered to have a long term flow with each of its neighbors in $hostIn$

and *hostOut*.

In the second part, Line 15 to Line 35 in Figure 3.10, all the flows are divided into subflows one by one, each subflow with its distinctive StartEvent and EndEvent. Each subflow should correspond to content exchange between a pair of neighbors. Before this decomposition, the ineffective flows are dropped. An ineffective flow is one that contains too few packets (defined by a parameter *effective threshold*), hence is ruled out as content exchange. The rule for marking the beginning and end of subflows is that the time interval of any two consecutive packets of the same flow should not be larger than the given parameter interval_threshold. This is like the *inactivity timer* in Netflow. When a subflow ends, a flow EndEvent (EE) is triggered and the event has an associated *Timestamp* that is the time of the last packet of this subflow. When a new subflow starts, a flow StartEvent (SE) is triggered and this event will share the same *Timestamp* as that of the first packet of the new subflow. Figure 3.11 gives a simple example. In the end, all the triggered events are sorted into ascending order of *Timestamp*. Line 34 to Line 54 in Figure

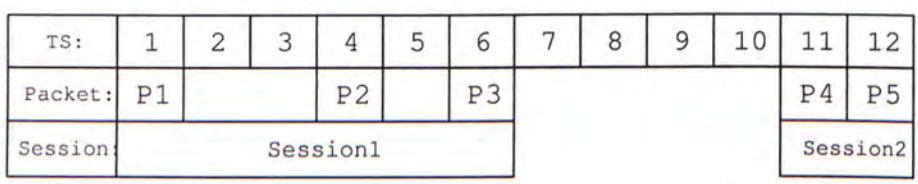
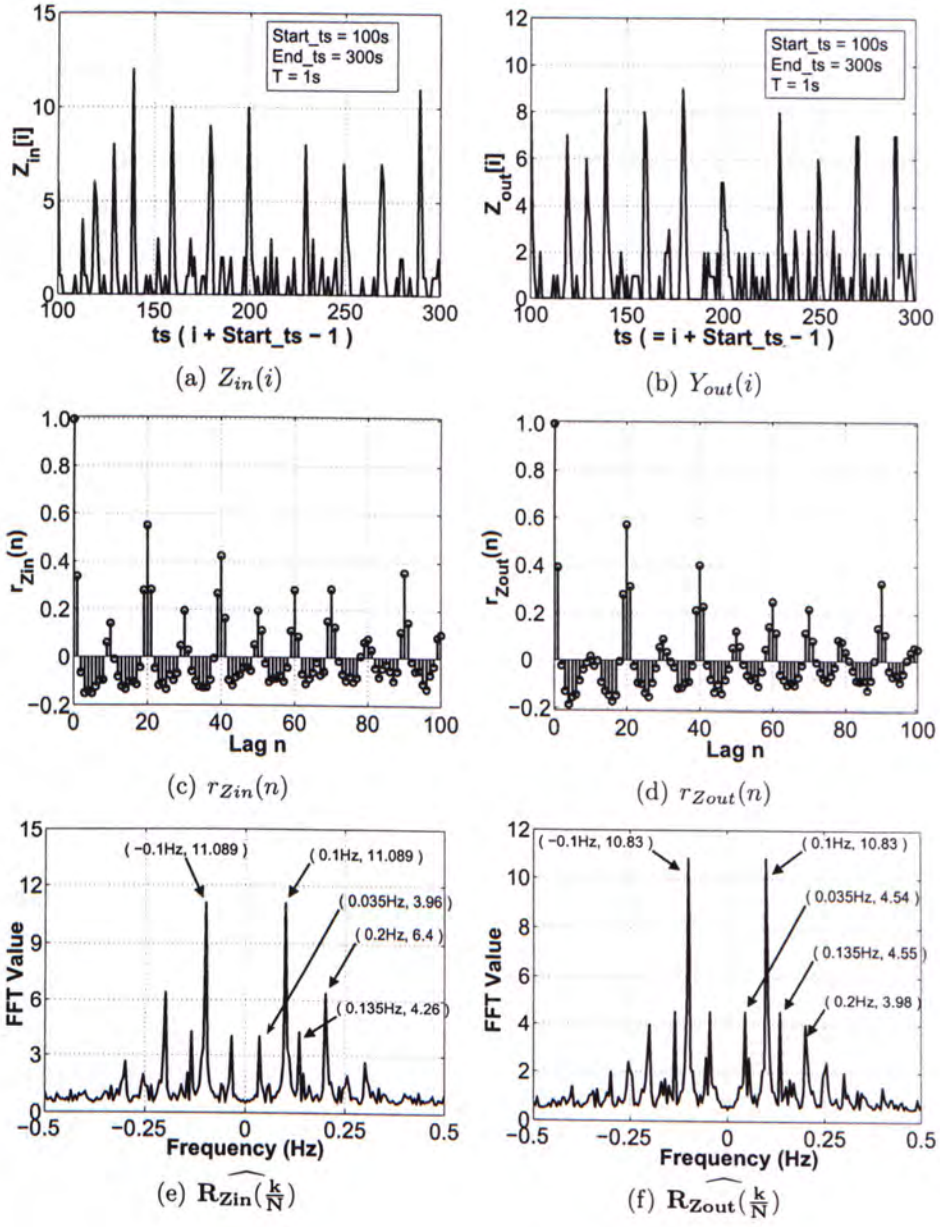


Figure 3.11: A simple example of how flows are separated into transmission sessions. The interval threshold is set to be 3

3.10 form the third part of the algorithm. This part is very similar to the algorithm in 3.2.4. This time, the algorithm uses the sequence $Z_{in}(i)$ and $Z_{out}(i)$ to record the number of StartEvents and EndEvents is triggered in each time slot. In fact, if we make some small change on the algorithm, we could derive more detailed sequences, e.g., $Z_{inSE}(i)$, $Z_{inEE}(i)$, $Z_{outSE}(i)$ and $Z_{outEE}(i)$ by treating the SE and EE separately. We show the results out-

put from our algorithm, $Z_{in}(i)$ and $Z_{out}(i)$, based on an example trace of BitTorrent in Figure 3.12. Here,



(a) $Z_{in}(i)$ (b) $Y_{out}(i)$

(c) $r_{Zin}(n)$ (d) $r_{Zout}(n)$

(e) $\widehat{\mathbf{R}_{\mathbf{Zin}}(\frac{\mathbf{k}}{\mathbf{N}})}$ (f) $\widehat{\mathbf{R}_{\mathbf{Zout}}(\frac{\mathbf{k}}{\mathbf{N}})}$

Figure 3.12: $Z_{in}[i]$, $Z_{out}[i]$, ACF and FFT results.

$$\widehat{\mathbf{R}_{Zin}(\frac{k}{N})} = |\mathbf{R}_{Zin}(\frac{k}{N})|.$$

$$\widehat{\mathbf{R}_{Zout}(\frac{k}{N})} = |\mathbf{R}_{Zout}(\frac{k}{N})|.$$

In this example, we set the two parameters, *effective threshold* and *interval threshold* to be 10 and 4 respectively. The setting of *effective threshold* is used to filter out some meaningless flow (that do not correspond to content exchange between two neighbors), because if a peer unchokes one of the neighbor and starts to upload data, it will not only send few number of packets. Theoretically, the value of *interval threshold*, which determines the start and end of flows, is likely to affect the frequency characteristics of the sequence. The larger the value of *interval threshold* is, the fewer number of events will be triggered and these is a higher probability that the real frequency will be buried by noise. So, *interval threshold* should take a relatively small value.

In Figure 3.12(e) and 3.12(f), we observe that there are four frequencies with large FFT values. In fact, the frequency points $f_1 = 0.035Hz$ and $f_2 = 0.1Hz$ are the frequency characteristics caused by choking (every 10 second, $f = 1/10Hz = 0.1Hz$) and optimistic unchoking (every 30 second, $f = 1/30Hz = 0.033Hz$). The remaining two frequencies, ($f_3 = 0.135Hz$ and $f_4 = 0.2Hz$) are the harmonic frequencies, which are the linear combination of the basic frequencies $0.1Hz$ and $0.035Hz$. Here, there is a small difference between $f_1 = 0.035Hz$ and $f = 1/30 = 0.033Hz$. This is due to the length of the original sequence $Z_{in}(i)$ and $Z_{out}(i)$. In Figure 3.12, the length of the original sequence is $N = (End\_ts - Start\_ts)/T = 200$. From Eq. (3.5), we know that $f = k/N, k = 0, 1, \ldots, N$ could be only finite values. Since $N = 200$, the granularity of $f = 1/N = 0.005Hz$ and this is why $f_1 = 0.035Hz$ but not $0.033Hz$. In later experiments, we use a larger N to make sure that the granularity will not bias the result. But N is limited by the duration of measurement.

3.2.6 Analyzer step

After the previous steps, the frequency characteristics of the packet trace related to the target host could be derived. We could set a *FFT value threshold* to pick out the frequencies with peak FFT values.

In order to conduct the analysis, we need to first develop the behavioral signatures for different P2P applications. This is done as follows. We first run the P2P application we are interested in detecting in a controlled environment. This means, we isolate a single host connected to a switch and catch all the traffic in and out of that host. In the mean time, only the P2P application of interest is run. We capture a traffic trace for this host, and apply our algorithm to determine the frequency domain characteristics.

In the previous subsections 3.2.2, 3.2.4 and 3.2.5, we built three separate behavioral signatures for PPLive live streaming, PPStream VoD and BitTorrent. In practise, a particular P2P applications may have multiple behavioral signatures, and we may develop all three sequences by using the three different generators. Using a combination of behavioral signatures for a give P2P application is likely to improve the accuracy for detecting that application.

3.3 Behavioral signatures of popular P2P applications

In this section, we present the behavioral signatures of several popular P2P applications. These signatures are derived by performing traffic measurement in a controlled environment. The captured traffic traces are then processed using the methodology we discussed in Section 3.1 and 3.2.

We access the Internet through our campus network and use

Wireshark[15] to capture all the packets. When we measure the traffic of one particular P2P application, we *turn off* all other network applications. The duration of each traffic measurement is about 30 minutes. When we capture these packets, we classify and separate them into TCP trace or UDP trace. Although many P2P applications use both TCP and UDP to realize the service, very often that one of these two types is used predominantly for data transfer. For example, PPLive live streaming[4] uses the UDP protocol to transfer most of the data while the data rate of TCP traffic is much less. On the other hand, PPStream live streaming[5] predominantly uses TCP to transfer its packets while the data rate of UDP traffic is much lower. As a result, one may find behavioral signature in TCP trace only, or UDP trace only, or both TCP and UDP traces. For other applications, both the TCP and the UDP data rates are comparable and have periodic features. In Table 3.1, we list the frequency characteristics of 16 popular P2P applications derived by our measurements and behavioral signature extraction. We also selectively plot the FFTs of these applications in Figure 3.13.

It is important for us to note that for each application, we repeat the measurement for several times to check whether we could find the its frequency characteristics every time. In addition, for some of the applications (e.g., PPlive, PPStream), we also carry out the measurement under another controlled environment in which the computer accessed to the Internet was through ADSL instead of the Campus network. Results showed that we could still find the frequency characteristics for these applications but the FFT values at those frequency points were a bit smaller.

When we examine Table 3.1, we could find that except for the PPlive Live Streaming and the TVAnt Live Streaming applications which share the same frequency characteristics, all other

P2P applications have unique fundamental frequency. Note that it is reasonable that the frequency characteristics may not be unique for some P2P applications. In this case, when we determine the behavioral signatures, one can combine those applications with the same frequency characteristics as a *group* of P2P applications. For example, we could take PPLive and TVAnt Live Streaming as a P2P Live Streaming application which has the fundamental frequency being 0.2Hz.

3.4 Experiment Results

To determine the effectiveness of our methodology, we apply our identification approach to two types of traffic traces: (a) campus traffic traces and, (b) ISP traffic traces.

The campus traffic traces were collected in the gateway of our department. We captured the packet header and the first 42 bytes of payload of the traffic going through the gateway. For privacy consideration, the IP addresses were anonymised and the payloads of some well-known applications such as HTTP, FTP were removed. We captured the traffic from September 12, 2007 to September 30, 2007. The size of each trace file is about 1G bytes in tcpdump format. The time duration of each trace file varies from one to two hours (in the afternoon) to seven to eight hours (at night).

Since the traffic is only from one department, the number of hosts running P2P applications is not large. Because the traffic pattern is similar between different days, we only present two of them (4 hours for each day and 2 hours in the afternoon and 2 hours at night) with the most traffic volume to analyze. After applying the behavioral signature based approach, we find that signatures of four different P2P application appeared and they are: 0.046Hz (PPStream Live Streaming), {0.2Hz, 0.4Hz} (PPLive-like Live Streaming), {0.048Hz, 0.192Hz} (Emule) and

{0.034Hz, 0.1Hz} (BitTorrent).

We use a combined method including payload signature checking and manual analysis for validation. Table 3.2 illustrates the payload signatures we use to identify P2P applications. Except for the payload signature of BitTorrent which is well studied and the signature has been published in the previous literatures, we propose the two payload signatures for PPStream and PPLive. We also use manual inspection to validate all P2P applications including Emule and TVAnt. The validation results show that the proposed behavioral-signature approach could identify all the hosts having running these applications accurately.

The ISP traffic traces were collected from an ISP core router from 17:40 p.m. to 17:50 p.m. on April 15, 2008. We captured the packet header and first 42 bytes of payloads. The data volume of the 10-minute traces is about 10GB. Since the trace is from an ISP router, there are a lot more hosts in this trace than the campus traffic trace. To this end, we only check for those *effective* hosts. A host is called effective when its average number of connected hosts is larger than a predefined *effective threshold*. The average number of connected hosts is calculated based on the following:

$$A = \frac{\sum_{i=1}^{M}(X_{in}[i] + X_{out}[i])}{|\{i|X_{in}[i] + X_{out}[i] \neq 0, i \in [1, M]\}|}. \tag{3.6}$$

In Eq.(3.6), M is the length the sequence $X_{in}[i]$ and $X_{out}[i]$. The numerator is to calculate the total number of connections in all M intervals (both *in* and *out*). The denominator is the number of the intervals during which there exits at least one connection, either *in* or *out*. We set the *effective threshold* equals to 4 which means that, on average, if a host is running a P2P application and it simultaneously communicates with at least two different peers, this host is considered as an *effective host*. After filtering out all *ineffective* hosts, we apply our behavioral-signature identification. From the analysis, we discover that there are 83

hosts which have the frequency characteristics that are within the known signatures (in Table 3.1) while there are 95 hosts which have frequency characteristics but they are not within the generated P2P behavioral-signatures set. For the UDP trace, there are 14 hosts which have the same frequency characteristics in our behavioral-signature set while there are 47 hosts which have frequency characteristics but are not within the generated P2P behavioral-signatures set. Among those hosts which have known frequencies, we could not find any running BitTorrent or PPStream Live Streaming application. On the other hand, we found that there were seven hosts running PPLive-like Live Streaming using our behavioral signature based approach. The manual validation method has found that four TCP hosts running BitTorrent and no hosts running PPStream Live Streaming. It also found three hosts running PPLive Live Streaming but no host running Emule. After checking the three hosts found by the manual validation, we found that all these three hosts are indeed included in the seven hosts found by our behavioral-signature approach. Table 3.3 lists the results. In the table, the "unknown" for validation means that we cannot accurately identify the application even we know the traffic patterns contain some periodicity.

Table 3.3 illustrates the result of our validation. One could find that although the behavioral-signature based approach was 100% accurate for the campus traces, the accuracy for the ISP traces is not as high. In particular, the behavioral-signature based approach failed to identify the four hosts running the BitTorrent applications. At the same time, only 3 out of the 7 hosts were running the PPLive applications while other 4 hosts were running different applications which have the same frequency characteristics as PPLive. We believe there are some possible explanations.

a) When the traffic capture and measurement are far away

from the host, the periodic characteristics will not be obvious because it is possible that only a small portion of the total flows belonging to one application are captured while most of the flows are not. This is because these flows may take a different route which does not pass the traffic capture point.

b) The periodicity of the traffic may be influence by queueing delay, transmission delay or packet loss between the host and the traffic capture point.

c) There may be different kinds of BitTorrent client software which use their own choking or unchoking methods, or they may set different time intervals for performing unchoking and optimistic unchoking. Therefore, these new mechanisms may not have the same periodicity as compare with the official BitTorrent protocol.

In Table 3.3, many hosts are found to have behavioral signatures which are similar to those in in Table 3.1. but currently these results could not be checked due to the limited method on validation and the lack of knowledge on the protocols that various softwares may use. To circumvent this problem, one may need to measure other different applications under controlled environment to enlarge the behavioral-signature set.

3.5 Discussion

In this section, we discuss some advantages and disadvantages of our behavioral signature based approach

There are number of advantages for using the behavioral signature based approach to identify application:

- There is no need to access any payload information of the packet trace. In fact,only packet header information is needed.

- The process of building a behavioral signature of an application is quite simple and fast.
- According to the behavioral signatures, our approach could identify from the traffic trace which particular P2P applications (such as PPLive, PPstream, BitTorrnet,...etc).
- Even when there is no known behavioral signatures, our approach could still classify the traffic based on their frequency-domain characteristics and find some unusual behaviors[64].

The proposed approach still has some limitations and these are our current effort. Some of the limitations are:

- The performance of the proposed method will be affected by the position of traffic monitoring points. The closer the monitoring point is to the host, the proposed approach can have a higher accuracy in identifying the application. In addition, because the proposed approach focuses on the time stamps of packets, traffic sampling will largely affected the accuracy of the approach.
- The identification results of our approach are host-level information and not flow-level information. Therefore, we do not know how many flows of the trace belong to this identified application. If different applications share the same frequency characteristics, our approach could not distinguish these applications from each other.
- The behavioral signatures of a large portion of existing applications are still unknown, one needs to build a large set of behavioral signatures database. In addition, if a P2P application changes its signature in different version (e.g., PPLive Ver 1.3.4.6 has different signatures with other versions), we need to rebuild the behavioral signatures. In other words, the behavioral signatures need to be updated regularly.

□ **End of chapter.**

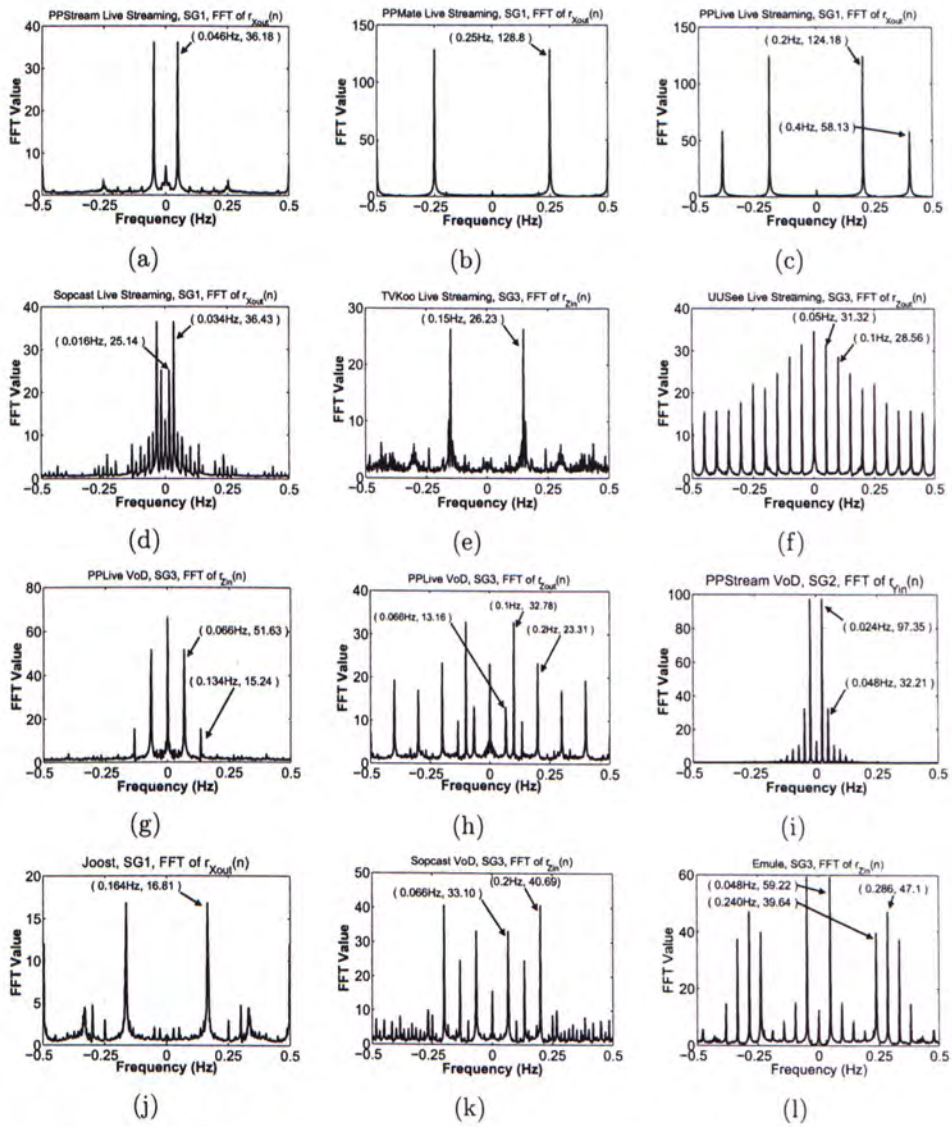


Figure 3.13: FFT of the selected measured P2P applications

P2P Application Name	Traffic (TCP UDP)	Using *In* or *Out*	Fundamental Frequency (Hz)	Harmonic Frequency (Hz)	Effective SGs
PPStream Live Streaming	TCP	Both	0.046		SG1, SG2 SG3
PPMate Live Streaming	TCP	Both	0.25		SG1
		In	0.2		
		Out	0.25		SG2
PPLive Live Streaming (Ver 1.3.4.6)	Both	Both	0.334		SG1 SG2
PPlive Live Streaming (other)	Both	Both	0.2	0.4	SG1 SG2
TVAnt Live Streaming	Both	Both	0.2	0.4	SG1 SG2
Sopcast Live Streaming	UDP	*Out*	0.016, 0.034	0.066	SG1
		Both	0.016, 0.034	0.066, 0.1	SG3
TVKoo Live Streaming	UDP	Both	0.15		SG1, SG3
		Both	0.034		SG2
UUSee Live Streaming	UDP	Both	0.05	0.1, 0.15, 0.2, 0.25	SG3
	TCP	Both	0.1	0.2, 0.3, 0.4	SG3
TVU Live Streaming	UDP	*In*	0.034, 0.066	0.1	SG1 SG3
PPStream VoD	UDP	Both	0.024	0.048, 0.072	SG1, SG2 SG3
PPLive VoD	UDP	*In*	0.066	0.134	SG3
		Out	0.066, 0.1	0.134,0.2	
		Out	0.1	0.2	SG1
Joost	UDP	Both	0.164	0.328	SG1, SG2
UUSee VoD	UDP	Both	0.05	0.1, 0.15	SG1, SG2 SG3
Sopcast VoD	UDP	Both	0.066	0.134, 0.2	SG3
Emule	UDP	Both	0.048	0.192, 0.24	SG3
Bittorrent	TCP	*Out*	0.034, 0.1	0.134	SG1, SG3

Table 3.1: The frequency characteristics of different P2P applications derived through our measurement.

Application	Payload Signature
BitTorrent	^get /announce\? | bittorrent protocol | d1:ad2:id20: | d1:rd2:id20: | ^azver
PPStream	^PsProtocol
PPLive	UDP ^\xe9\x03

Table 3.2: Payload signatures

Type	Approach	BitTorrent	PPStream	known Freq.	unknown Freq.
TCP	Behavioral Signature	Not found	Not found	83	105
	Validation	4	Not found	/	/
Type	Approach	PPLive	Emule	known Freq.	unknown Freq.
UDP	Behavioral Signature	7	Not found	14	47
	Validation	3	Not found	/	/

Table 3.3: The result of ISP traces

Chapter 4

Related Work

Summary

In this chapter, we discuss the related work of both two parts of the thesis work.

The related work of the first part of the thesis work is as follows. E-Model (ITU-T Rec. G.107 [23]) is designed to be a non-intrusive parametric model to estimate the subjective MOS (ITU-T P.800 [17]). Number of works have focused on the implementation and extension of E-Model. COLE et al. [35] propose to simplify the E-Model base on only two network related impairments, e.g., I_e, effects of packet loss and I_d, end-to-end delay. Alexander [58] proposes the $I_{e,eff}$, the effective equipment impairment factor quantifying the impairment of a codec under both random loss and bursty loss. There are more detailed description and discussions in[59], which covers all the related topics on assessment and prediction on speech quality of VoIP. The results in[59] estimating the $I_{e,eff}$ impairment with iLBC [10, 24] Internet low bit rate codec under random loss are important and related to our results. Samir et al. [51] propose other non-intrusive parametric model to estimate the subjective MOS. The proposal is a random neural networks-based (RNN)

approach, which could map to the subjective MOS very well, but the model needs a large sample space to train the coefficients of RNN. Kuan-Ta Chen et al. [32] introduce an innovative way to quantify the user's satisfaction on voice quality. In their work, they define the *user satisfaction index*(USI) and provide the method of deriving USI from network parameters via SKYPE [1] measurements, e.g. bit rate, jitter and RTT. However, the authors did not find the relationship between USI and subjective MOS, which so far has been considered as the standard way of measuring the quality of speech.

There are only a few articles focusing on quality of service for voice conference. Jonathan and Henning [48] demonstrate some existing voice conference topologies and also propose a protocol for decentralized conference. SKYPE [1] is becoming the most popular software due to its voice quality, robustness and free distribution. Experiments validating our models and proposals are using SKYPE because it supports concurrent voice conference very well. Early measurements on SKYPE, [26, 63, 38] reveal the basic properties of the software. Especially, in [26], the authors observe that the codecs used by SKYPE are iLBC [10, 24] and iSAC [11] and the network topology of SKYPE conference is the end system mixing topology [48].

The related work for the second part of the thesis work, behavioral-signature based P2P application identification is in the following.

The field of traffic classification has received continuous interest in recent years. Some of the traffic classification approaches develop discriminating criteria based on statistical observations of various flow properties in the packet traces. Based on these statistical observations, these studies employ classification, clustering and other machine learning techniques to assign flows to classes. Roughan et al. in [61] classify traffic flows into four classes suitable for quality of service applications. The authors

demonstrate the performance of the Nearest Neighbor and the Linear Discriminant Analysis algorithms. Moore et al. in [53] apply Bayesian analysis techniques to categorize traffic by application. The authors listed more than 200 different discriminators for traffic analysis[54]. The authors also apply FFT to build discriminators, but our approach are quite different with theirs. Their method is to be build flow-based discriminators and apply FFT on the interarrival times of packets belonging to a single flow. Our approach, on the other hand, focuses on the host-level behaviors and we inspect the periodicity of all flows related to the same host.

Another promising traffic classification approach is shown in [45]. Instead of classifying individual flows, Karagiannis et al. propose to associate Internet hosts with applications, and then classifying their flows accordingly. The authors attempt to capture the inherent behavior of a host at three levels of increasing detail: the social, the functional and the application level. Although both BLINC and our approach are host-level methods, but there is a significant difference between these two approaches. BLINC focuses on the behavior of host's connection patterns (spatial behaviors) while we focus on the periodic behaviors of a given host (temporal behaviors).

In [28], Bernaille et al. evaluate the feasibility of application identification at the beginning of a TCP connection. This approach distinguishes the behavior of an application by observing the size and the direction of the first few packets of the TCP connection. Crotti et al. in [37] propose a statistical approach to identify network application by building a set of protocol fingerprints that summaries its main IP-level statistical properties. In [50], Ma et al. analyze three mechanisms relying on flow content to automatically identify traffic that uses the same application-layer protocol.

In recent years, the tremendous growth of P2P traffic has

drawn a lot of attention from researchers. Several studies emphasize on identification of P2P traffic, such as the signature-based payload methodology in [62] and the identification method by transport layer characteristics in [44]. In [36], Collins et al. propose a set of tests for identifying masqueraded peer-to-peer file-sharing applications. Bonfiglio et al. in [29] propose a framework to reveal Skype traffic in real time by exploiting the randomness introduced at the bit level by the encryption process.

□ **End of chapter.**

Chapter 5

Conclusion

Summary

In this chapter, we carry out the conclusion our the thesis work

In the first part of the thesis, we propose *GMOS*, which is a a group-based MOS metric to evaluate the overall quality of voice conference. Note that this performance measure is important, not only because it lacks in this area but with this measure, it provides designers a systematic means to design group-based communication services as well. To leverage an existing work on MOS, we let GMOS be composed of MOS values plus a calibration parameter α. The parameter α can be calibrated in two ways. One is to indicate the user's perception on the conference quality, the second way is to consider it as an application dependent parameter.

Further, we propose the *two-step mapping method*(TSMM) to estimate the leader's GMOS_L from the network parameters between the leader and the non-leader participants. We also propose the *leader selection strategy*(LSS) to improve the overall quality of voice conference by properly selecting the conference leader.

To validate our proposals, we have invited 25 subjects to listen to and give the scores to the records of our conference experiments by SKYPE. These subjects are asked to provide the MOS to each speakers in the records, and also an overall score GMOS (a subjective test) to the whole conference record as well. The results of our experiments indicate that both of the *two-step mapping method*(TSMM) and the *leader selection strategy*(LSS) perform very well.

The conclusion of the second part of this thesis is that we propose a behavioral-signature based approach to identify P2P applications. Our approach only needs packet header information. We not only can identify *which* host is running the P2P application, but we can also identify a particular P2P application that is running by the host. We first extract all those packets which are related to a target host from the original traffic trace. Then we apply three types of time series generators to transform the traffic patterns to the time series. After that, these generated time series are served as inputs into ACF and FFT functions and be transformed into frequency-domain series. In the end, the frequency characteristics are analyzed and the identification results are obtained. In order to identify different P2P applications, we need to build the behavioral signatures database of different P2P applications. We have applied our approach to different traffic traces and use the combined method to validate the result. The result shows that our approach is quite promising as it performs quite well when the packet trace are collected near the host.

□ **End of chapter.**

Bibliography

[1] Skype, http://www.skype.com/download/.

[2] "BitComet", http://www.bitcomet.com/.

[3] "Emule", http://www.emule.com/.

[4] "PPLive", http://www.pplive.com/.

[5] "PPStream", http://www.ppstream.com/.

[6] "Sopcast", http://www.sopcast.com/.

[7] "NetFlow", http://www.cisco.com/en/US/ products/ps6601/products_ios_protocol_group_home.html.

[8] Ethereal, http://www.ethereal.com/download.html.

[9] Audition, http://www.adobe.com/downloads/.

[10] iLBC codec. http://www.gipscorp.com/files/english/datasheets/iLBC.pdf.

[11] iSAC codec. http://www.gipscorp.com/files/english/datasheets/iSAC.pdf.

[12] Global IP Solutions. http://www.gipscorp.com/default/customers.html.

[13] "PFSVOD", http://www.pplive.com/subject/20070808pfsvod/.

[14] "Joost", http://www.joost.com/.

[15] "Wireshark", http://www.wireshark.org/.

[16] Assessing voip call quality using the e-model. "White paper of IXIA", http://www.ixiacom.com/library/white_papers/.

[17] Methods for subjective determination of transmission quality. *ITU-T Recommendation P.800*, 1996.

[18] Objective quality measurement of telephone-band (300-3400hz) speech codecs. *ITU-T Recommendation P.861*, 1998.

[19] Definition of catergories of speech transmission quality. *ITU-T Recommendation G.109*, 1999.

[20] Method for objective measurements of perceived audio quality. *ITU-R BS. 1387*, 1999.

[21] Perceptual evaluation of speech quality (pesq), an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *ITU-T Recommendation P.862*, 2001.

[22] Single-ended method for objective speech quality assessment in narrow-band telephony applications. *ITU-T Recommendation P.563*, 2004.

[23] The e-model, a computational model for use in transmission planing. *ITU-T Recommendation G.107*, 2005.

[24] S. V. Andersen, W. B. Kleijn, R. Hagen, J. Linden, M. N. Murthi, and J. Skoglund. ilbc - a linear predictive coder with robustness to packet losses. In *Proceedings of IEEE Workshop of Speech Coding*, 2002.

[25] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *In Proc. ACM Sigcomm'02*, Aug 2002.

[26] S. A. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *Proceedings of IEEE INFOCOM'06*, April 2006.

[27] J. S. Bendat and A. G. Piersol. *Random Data - Analysis and Measurement Procedures.* John Wiley & Sons, 1986.

[28] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proc. CONEXT '06*, 2006.

[29] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing skype traffic: When randomness plays with you. In *Proc. SIGCOMM '07*, 2007.

[30] T. Bu, Y. Liu, and D. Towsley. On the tcp-friendliness of voip traffic. In *Proceedings of IEEE Conference on Computer and Communications(INFOCOM)*, 2006.

[31] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications*, 20(8), Oct. 2002.

[32] K. T. Chen, C. Y. Huang, P. Huang, and C. L. Lei. Quantifying skype user satisfaction. In *Proceedings of ACM SIGCOMM'06*, Sep 2006.

[33] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM'01*, Aug 2001.

[34] B. Cohen. Incentives build robustness in bittorrent. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf, May 2003.

[35] R. G. Cole and J. Rosenbluth. Voice over ip performance monitoring. *Computer Communication Review*, 31(2):9–24, April 2001.

[36] M. P. Collins and M. K. Reiter. Finding peer-to-peer file-sharing using coarse network behaviors. In *Proc. ESORICS '06*, 2006.

[37] M. Crotti, F. Gringoli, P. Pelosato, and L. Salgarelli. A statistical approach to ip-level classification of network traffic. In *Proc. ICC '06*, 2006.

[38] S. Ehlert and S. Petgang. Analysis and signature of skype voip session traffic. Fraunhofer FOKUS Technical Report, NGNI-SKYPE-06b.

[39] P. Francis. Yoid: Extending the multicast internet architecture. White paper, http://www.aciri.org/yoid/, 1999.

[40] T. A. Hall. Objective speech quality measures for internet telephony. In *Proc. SPIE*, volume 4522, pages 128–136. Voice over IP (VoIP) Technology, 2001.

[41] Y. hua Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proc. of ACM Sigmetrics*, 2000.

[42] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. *to appear ACM SIGCOMM'08*, 2008.

[43] J.Liang and R. Kubichek. Output-based objective speech quality. In *Proceedings of IEEE Vehicular Technology Conference*, pages 1719–1723, Stockholm, Sweden, 1994.

[44] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport layer identification of p2p traffic. In *Proc. IMC '04*, 2004.

[45] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In *Proc. SIGCOMM '05*, 2005.

[46] D.-S. Kim. Anique: an auditory model for single-ended speech quality estimation. *IEEE Trans. Speech Audio Process.*, 13(5):821–831, Sep 2005.

[47] A. Legout, N. Liogkas, and E. Kohler. Clustering and sharing incentives in bittorrent systems. In *Proc. of ACM Sigmetrics'07*, June 2007.

[48] J. Lennox and H. Schulzrinne. A protocol for reliable decentralized conferecing. In *Proceedings of ACM International Workshop on Network and Operating Systems Support for Diginal Audio and Video (NOSSDAV'03)*, June 2003.

[49] J. Linden. Achieving the highest voice quality for voip solutions. In *GSPx, The International Embedded Solutions Event*, Santa Clara, 2004.

[50] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of protocol inference. In *Proc. IMC '06*, 2006.

[51] S. Mohamed, G. Rubino, and M. Varela. Performance evaluation of real-time speech through a packet network: a random neural networks-based approach. *Performance Evaluation*, 57:141–161, 2004.

[52] S. Moller, A. Raake, N. Kitawaki, A. Takahashi, and M. Waltermann. Impairment factor framework for wideband speech codecs. *IEEE Trans. Audio, Speech Lang. Process.*, 14(6):1969–1976, Nov 2006.

[53] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proc. Sigmetrics '05*, 2005.

[54] A. W. Moore, D. Zuev, and M. Crogan. Discriminators for use in flow-based classification. Technical report, Intel Research, Cambridge, 2005.

[55] J. Okamoto, T. Hayashi, A. Takahashi, and T. Kurita. Verification of objective quality assessment method for arbitrary video sequence. IMQA, Sept. 2005.

[56] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time signal processing, 2nd Edition.* Prentice Hall, 1999.

[57] S. R. Quackenbush, T. P. Barnwell, and M. A. Clements. *Objective Measures of Speech Quality.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[58] A. Raake. Short- and long-term packet loss behavior: towards speech quality prediction for arbitrary loss distributions. *IEEE Trans. Audio, Speech Lang. Process.*, 14(6):1957–1968, Nov 2006.

[59] A. Raake. *Speech Quality of VoIP - Assessment and Prediction.* Chichester, U.K.: Wiley, 2006.

[60] A. W. Rix, J. G. Beerends, D. S. Kim, P. Kroon, and O. Ghitza. Objective assessment of speech and audio quality - technology and applications. *IEEE Trans. Audio, Speech Lang. Process.*, 14(6):1890–1901, Nov 2006.

[61] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for qos: A statistical signature-based approach to ip traffic classification. In *Proc. IMC '04*, 2004.

[62] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic. In *Proc. WWW'04*, 2004.

[63] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting relayed traffic: A case study using skype. In *Proceedings of IEEE INFOCOM'06*, April 2006.

[64] H. Sun, J. C. S. Lui, and D. K. Y. Yau. Defending aagainst low-rate tcp attack: Dynamic detection and protection. In *Proceedings ICNP'04*, 2004.

[65] X. Zhang, J. Liu, B. Li, and T. S. P. Yum. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. In *Proceedings of INFOCOM'05*, 2005.