Peer-to-peer multimedia streaming monitoring system

Wang, Z.

Award date:
2014

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Peer-to-Peer multimedia streaming monitoring system

Master Thesis

Zijian Wang

Supervisor:
dr. Dmitri Jarnikov

Eindhoven, August 2014

Abstract

Nowadays, many companies provide on-demand or live multimedia content over the Internet. The success of the content provider is measured by the number of subscribers. There are, however, pirates that re-distribute obtained content illegally, thus cutting into the legitimate subscribers base. The content providers would like to know how serious the problem is in order to address it. The goal of our project is to develop a monitoring system to help the content provider to learn: what is the popularity of the pirated content, who is using the illegal service, and where are the illegal users?

Our system focuses on services that are based on P2P streaming technology because its cost-effectiveness and scalability advantages are appealing to pirates. We aim to develop a system that can monitor an arbitrary P2P streaming system. To achieve this goal we will address the following questions in our project: How to perform analysis to understand the audience? What measurement data can we collect? How to measure a high-dynamic P2P network? How to gather data from an arbitrary (and proprietary) P2P streaming application?

In the end, we present a distributed monitoring system based on passive measurement. We conducted a set of evaluation experiments on three popular and representative P2P streaming systems: Acestream, Sopcast and Peerflix. The evaluation results proved the effectiveness of our measurement approach and strategy.

Preface

Six months ago, I started my master project. During the last half year I contributed all myself to solving the problem, using the best of my knowledge. It was a enjoyable period of time and this thesis is the statement of my achievements. I hope you enjoy reading. I wish to thank my parents and my girlfriend. Without their support and love I cannot finish this work. I also wish to thank my supervisor, dr. Jarnikov, who gave me innumerable guidance and help during my project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

In the market of Internet multimedia, there are content providers offering on-demand or live content to their audience. For example, Sky Sports aggregates games from sport leagues all over the world and provides live broadcasting to its audience; Netflix offers its subscribers movies and TV series online, which is its own production (e.g. House of Cards) or bought from other content providers. However, there are illegal services (i.e. pirates) pirating these content. These pirates receive content from a content provider as normal viewers, and re-distribute the content to other viewers illegally. The content providers' earnings are harmed, they are interested to know how serious the piracy problem is in order to address it. They ask questions such as:

- What content is being distributed by the illegal services?
- What is the audience of these services (e.g. location, scale etc.)?

The first problem is related to crawling techniques. Moshchuk et al. proposed a crawler that can be used to search pirated contents on Web [52]. Work of Indyk et al. is more specific [34], because it aims to crawl pirated video content. Since the problem of searching pirated content can be solved by crawling techniques, it is not in the scope of our project.

The focus of our project is placed on the second problem-illegal service audience measurement and analysis. We want to develop a monitoring system that can help content providers understand the extent of the piracy of their content. To achieve this goal, we need first answer the following questions by ourselves:

- What Internet multimedia delivery systems exist? And which are popular?
- What system do we want to investigate?
- How to collect and analyse data from the selected system?

We discuss these questions in the following sections. In Section 2, we introduce the Internet multimedia delivery from both business and technical perspectives. In Section 3 we propose our approaches of measurement and analysis.

1.2 Internet multimedia delivery

The Internet is a powerful platform for content delivery services. Multimedia content is the most prevalent content on the Internet [36]. Its popularity is not a new phenomenon, and the roll-out of the broadband Internet access has accelerated that. The large demand for multimedia content brings great opportunities to companies. In order to understand this market, we present a business

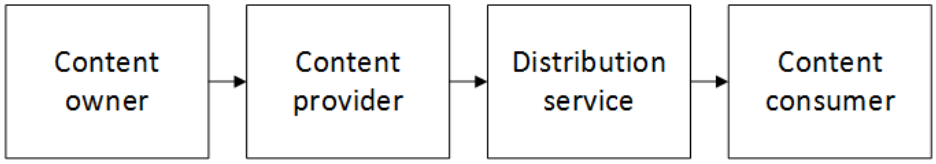

Figure 1.1: Value chain of Internet multimedia market

value chain in the Internet multimedia environment in figure 1.1. The value chain we present is a simplified view of the chain given in [56].

The role of each actor is explained underneath:

- Content owner: Content owner produces the content. The content owner may provide content not only for Internet multimedia delivery, but also for other distribution means (e.g. TV broadcasting).
- Content provider: Content provider buys copyright of content from the content owner. Content provider acts as an aggregator of multiple content sources. For example, Sky Sport provides live football games from leagues through out the world.
- Delivery service provider: Delivery service provider provides paid delivery service to the content provider to deliver content over the Internet in a scalable and reliable manner. They utilize delivery network architectures such as Content Delivery Network (CDN). An example is Akamai, who provides CDN service to Youtube to deliver video content.
- Content consumer: Content consumer consumes the content via their devices (e.g. PC, smart phone, and tablet).

Typically, pirates receive the content from content provider via a distribution service as normal content consumers. But afterwards they use content distribution techniques to re-distribute the content to other viewers. A large amount of viewers are attracted by these illegal services because its satisfying user experience, abundant amount of content and lower cost [44]. Because the income of content provider largely depends on the number of viewers, piracy results in heavy financial loss in Internet multimedia industry [44].

1.3 Approach

In this section, we introduce our measurement and analysis approaches of the monitoring system.

1.3.1 Measurement

The two categories of network measurement approaches are passive measurement and active measurement [51, 45, 62, 50]. The active approach relies on the capability to inject artificial packets into the network or send packets to servers and applications, then analyse the response (e.g Web crawler) [51, 52]. The volume and other parameters of the introduced traffic is fully adjustable and small traffic volumes are enough to obtain meaningful measurements [51]. In order to inject artificial packet, one needs to get access to the control flow of the Internet multimedia delivery system. However, most of these systems are proprietary because of their commercial nature [64], and, moreover, the encryption is often used to prevent access to the control flow [61, 51, 35, 23]. As suggested in [62], reverse engineering can allow us to extract information of proprietary protocol, but it is a non-trivial task even for a single system. Thus, we conclude that active measuring is not suitable for our system.

Passive measurement refers to the process of measuring a network, without creating or modifying any traffic on the network [51]. Passive measurement can provide detailed information about the monitored network entity (an application's process, a host or a local area network etc.) [51].

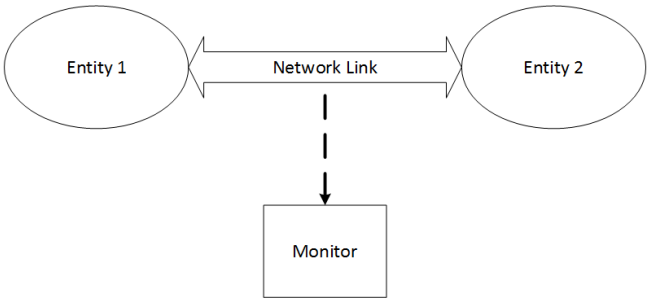

Figure 1.2: Passive measurement principle

The basic principle of passive measurement is shown in figure 1.2. The monitor observes any packets travelling through the link between networked entities. This is in contrast to the active measurement approach, in which specific packets are introduced into the network. Thus, passive measurement does not require specific knowledge of the Internet multimedia system, which means it can be applied to an arbitrary Internet multimedia network. In our system, we select the passive measurement approach.

1.3.2 Analysis

To acquire knowledge about the audience of illegal services, we need to analyse the collected measurement data. The first problem we need to solve is to identify the role of a node in a Internet multimedia delivery system. There are various roles of nodes in an Internet multimedia delivery system.

- Service node: A node owned by the service provider that offers service (content sourcing, channel information etc.) to other nodes.
- Viewer: A node that freely joins the multimedia delivery network, and exchanges multimedia data with other nodes.
- Non-viewer: A node that freely joins the multimedia delivery network, and exchanges not multimedia data with other nodes.

Our node role identification method is based on traffic classification: we identify viewers by detecting multimedia data traffic. Traffic classification is an automated process which categorizes computer network traffic according to various parameters (e.g., based on port number or protocol) into a number of traffic classes [68, 48]. There are three common methods for traffic classification: port analysis, payload analysis and traffic statistics analysis [48]. Port analysis is becoming invalid because many applications are using unregistered or dynamic port numbers [48]. Payload analysis requires inspection into the payload content of Internet multimedia application's packets to find application signatures [59]. It is accurate but needs manual payload inspection [59].

We find that only traffic statistics can fulfil our purpose. [30] proposed a heuristic method to filter video data traffic, based on their measurement studies of P2P streaming systems. The author of [30] states that this approach can be applied to many P2P streaming systems. Additionally, as shown in the works of [23, 50], clustering algorithm can help to distinguish different types of traffic.

We adopt both methods to detect video data traffic. For clustering, our work differs from previous traffic clustering works such as [23] and [50], because our aim is more fine-grained: we want to classify traffic by its function (data traffic or control traffic).

Another information we want to gather is the node's geolocation. Node's geolocation is relatively straight-forward to analyse because we directly infer it from IP address [66, 1]. We can employ IP mapping through public database to get the geolocation of node.

1.4 Thesis outline

The structure of the thesis is as follows:

- Chapter 2 introduces the Internet multimedia delivery technology.
- Chapter 3 proposes the solution and implementation details.
- Chapter 4 shows the evaluation result and analysis.
- Chapter 5 presents the future work and conclusions.

Chapter 2

Domain analysis

The aim of this chapter is to introduce the technical background of our project. We want to introduce how multimedia content is distributed (by pirates) over the Internet and explain why our focus is on P2P streaming systems.

2.1 Internet multimedia delivery technologies

To understand how pirates are distributing the multimedia content, we discuss characteristics of Internet multimedia delivery technologies. Each characteristic of 2.1 is explained in Appendix A.

The multimedia services can be classified into two groups: live streaming and video-on-demand(VoD) [47, 41]. Straight download is also considered for historical reasons. To serve the video content, the raw video must be compressed before transmission because of its huge size [67]. Once the raw video is compressed, it needs to be packaged into a container. Different systems vary in their support of compression format and container format. Delivery method deals with how the multimedia file is delivered to the end-user. While streaming is the mainstream delivery method, other options exist as well. Protocols are designed and standardized for communication between clients and streaming servers [67]. In the characteristics we discuss both standard protocols and proprietary ones. Network architecture deals with how the content is delivered over the network. And, the client software part describes how content data is received and played out to the user.

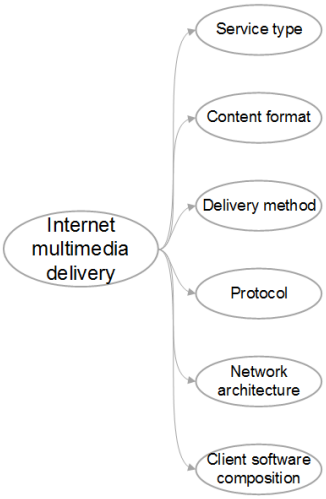

Figure 2.1: Characteristics of Internet multimedia delivery technology

2.2 Selected system

Based on our study, we find that P2P streaming technology is the most appealing one to the illegal service providers because of its two advantages: cost-effectiveness and scalability. P2P networks do not rely on a dedicated delivery infrastructure and therefore offer the possibility of rapid deployment at low cost. The upload capacity of hosts in a P2P network can be utilized for video transmission so as to reduce the server load dramatically, which offers great scalability. Studies of [44, 27, 31] also confirm that P2P streaming systems appear to be the most used mechanism for Internet multimedia delivery.

In the remainder of the thesis, we focus on developing a monitoring solution for P2P streaming systems.

2.2.1 P2P streaming architecture overview

P2P streaming, as its name suggests, is a technology utilizing P2P network for multimedia streaming service [70, 43]. In P2P streaming system, the video file is segmented into small pieces (called chunks) and sent to hosts (called peers) in the P2P streaming network. Peers actively contribute their resources by forwarding available content to other peers. By this approach, the available amount of content grows with the size of network [43].

Although P2P streaming netowrk has advantages for serving massive number of viewers, however, it is also challenging to design a P2P streaming system because of the stringent time constrains and hosts' diverse capacity (e.g. upload bandwidth) [18, 28]. To address these challenges, many systems have been designed [73, 49, 31, 6, 7].

By logical overlay topology classification, P2P streaming systems are roughly classified into two types; tree-based and mesh-based [32, 31, 70, 42]. Detailed study of mesh-based and tree-based architecture is in Appendix B. In tree-based architectures, nodes make tree shaped connections and contents are transmitted from root to leaf by overlay multicast. Tree-based architectures have well-organized overlay structure, which makes it has lower delays between a distributor and nodes. One major drawback of tree-based is that they are vulnerable to peers' departures. On the other hand, in mesh-based architectures, such as Coolstreaming [73], peers construct an overlay network without making clear hierarchy. In a mesh-based P2P streaming system, peers are not confined to a static topology, instead, peering relationships are established/terminated based on the content availability and bandwidth availability on peers [49]. A peer dynamically connects to a subset of peers in a swarm and pulls chunks from each other. Its major advantage is its simple design principle and inherent robustness. However, the delay performance in mesh-based cannot be guaranteed [43]. As stated in [31], mesh-based P2P streaming systems have been successfully deployed in commercial area. In contrast, tree-based systems largely have been running at the research stage. Study from [49] shows that mesh-based architecture exhibits a superior performance across a wide range of of scenarios. In our work, we focus on mesh-based architecture.

2.2.2 Generic architecture

We can separate the actions of P2P streaming system into two tasks: overlay construction and peer communication, which is similar with the discussion in [31, 49]. Overlay construction deals with how to construct an efficient and resilient architecture of a P2P streaming network; peer communication deals with how to efficiently distribute video chunks through the established overlay architecture. In addition, based on the peer's communication behaviour, we generalized four roles of peers in a P2P streaming architecture. We present a generic architecture of P2P streaming system as shown in figure 2.2. The elements on the architecture are discussed in the following subsections.

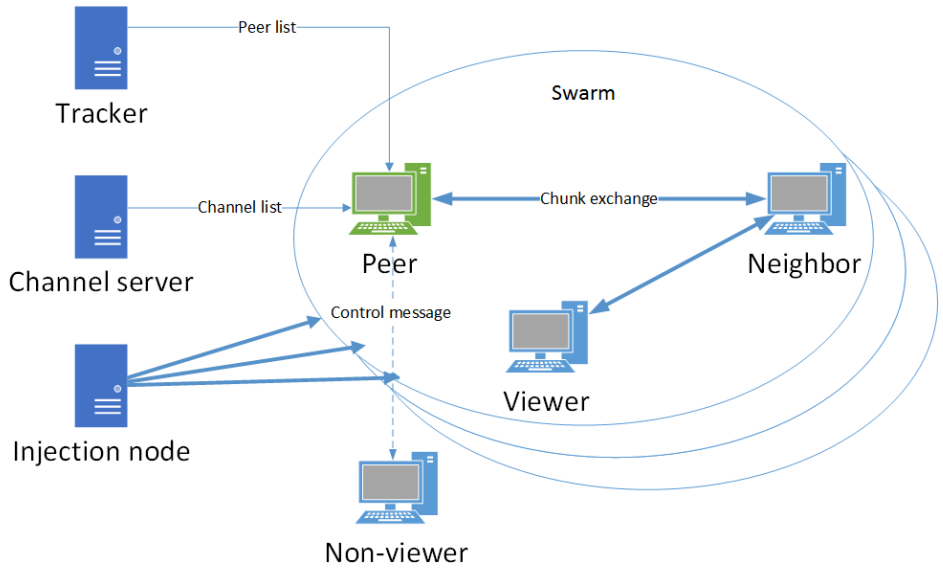

Figure 2.2: P2P streaming system generic architecture

Overlay construction

Overlay construction's purpose is to help a peer construct its neighbour relationship with other peers [31]. The efficiency of a P2P streaming network relies a lot on its overlay structure. In a P2P streaming architecture, an overlay swarm is formed by peers viewing the same content [49, 70]. In both mesh-based an tree-based systems, the media stream is transmitted from a source peer to all participating peers. However as shown in figures B.1 and B.2, in tree-based system there exists only one path from source peer to any other peer; in mesh-based topology there are multiple paths. We can break overlay construction into two sub issues: peer discovery and peer selection [28].

The purpose of peer discovery is to find information about other peers in the P2P streaming network [37]. There are two general peer-discovery approaches: centralized and decentralized. In centralized approach, a central node maintains the global peer information (e.g. a tracker node) [70]. It is the most commonly used method for locating peers, which has, unfortunately, the problem of single-point-failure [70]. In decentralized approach, each peer maintains a subset of the overlay information. If a peer wants to discover other peers, it has two methods: flooding method and gossip method [28]. In flooding method, a peer sends a probe message first to a starting peer, and then to its neighbours and their neighbours until finally finding a suitable point to join the network. In networks using the gossip-based method, a peer maintains information about a small list of randomly selected peers, initially obtained from a starting peer, and information is updated periodically by exchanging messages with other peers in the list [28].

Peer selection deals with how a peer construct its neighbour relationship. As stated in [57], a good peer selection algorithm should achieve the following goals: 1) minimizing packet delay; 2) achieving a minimum of total streaming rate at a peer; 3) being more resilient to peer and network dynamics. A key issue for peer selection is to measure peer's network connectivity [57]. Currently, the two common methods are: by measuring packet delay and by estimating bandwidth [28].

In studies of [49, 43, 70], we find that for overlay construction the network connectivity is an essential influencing factor. In tree-based architecture the bandwidth of a peer determines its place in the tree structure. In mesh-based, the upload capacity plays a major role in peer selection [43, 60], as well. A higher priority is assigned (dynamically) to peers that upload more data. This observation on network connectivity is important for our lately proposed approach.

Another observation is that, a peer tries to construct its neighbour relationship with other peers to exchange data in the most efficient way, which implicates that a peer only contact a

limited number of peers .

Peer communication

Tree-based architecture and mesh-based architecture differ in their chunk delivery behaviour. In mesh-based based, the video chunks are actively requested from other neighbour peers, and for a peer its neighbour relationship is dynamically changing. The amount of control traffic overhead for requesting and responding chunk is not negligible. In comparison, in tree-based architecture, video chunks are passively received from parent peers in a tree overlay, which means there is very small control overhead for chunk delivery.

Although two architectures exhibit different peer communication behaviours, we can generalize their commons. The peer's communication flow can be classified into two classes: control flow and data flow [18, 22] (a flow is a sequence of packets between two hosts [22]). Control flow is used to exchange control information (e.g. peer discovery, peer selection, chunk request). Data flow is composed of packets carrying data chunks and control information overhead. Based on the classification of peer's traffic characteristics, we define generic roles of peers in a P2P streaming network (from a local peer's perspective):

- Tracker: A node owned by service provider that provides overlay information to peers from multiple events' overlays. The tracker only has control flow. In a system, tracker is an optional node.
- Injection node: A peer owned by service provider that provides data as a source. Injection node only have data flow. Moreover, it has large amount of upload traffic and almost no download traffic. The injection node may provide uploads to multiple events (i.e. participate in distribution of different data simultaneously).
- Viewer: A peer that freely joins the P2P network, and exchanges data chunks with other peers in its overlay swarm. It has control and data flow. Its ratio of upload traffic and download traffic is not definable.
  - Neighbour: Same as viewer, but a neighbour peer has direct data exchange with the local peer.
- Non-viewer: A peer that freely joins the P2P network, and does not exchange data chunks with other peers. A non-viewer does not participate in the local peer's overlay.

The requirement on our P2P streaming monitoring system is to support measurement and analysis of the audience of a P2P streaming system. To be specific, we need to detect viewer peers in a P2P streaming network. From the peer role category above, we find the data traffic is an important clue in distinguishing viewers and non-viewers, which is the reason we adopt traffic statistic analysis.

2.3 Related work

There have been many measurement studies on P2P multimedia streaming systems [61, 20, 18, 26, 62, 30]. All of them focus on certain systems(e.g. PPlive, Sopcast, PPstream etc.) and certain characteristics, but none of them focus on the monitoring system's architecture for an arbitrary P2P streaming system. A very similar work with ours is NG-MON [29], which is also a distributed network monitoring system that adopts passive measurement approach. But NG-MON is not specific for P2P streaming system. Comparably, we proposed a system architecture that supports dynamic resource allocation, and we utilize the properties of P2P network to improve the measurement performance.

For traffic analysis, we use the heuristic of data flow proposed by [30]. In addition, we also use a machine learning method. [23, 50, 72] laid foundation of utilizing machine learning for traffic

analysis. However, all of them use clustering algorithm to classify traffic by type (e.g. P2P, Web, File transfering) or by application type. In comparison, our goal of analysis is more fine-grained: we want to classify traffic by its function (data traffic or control traffic). The work from Park et al. [54] and ours both work on fine-grained P2P traffic classification [54]. However, they use payload signature retrieval technique, whereas we use clustering algorithm to analyse traffic statistics.

Chapter 3

Solution and implementation

This chapter presents the solution of our P2P streaming monitoring system.

3.1 System description

The requirement on the monitoring system is that the system can monitor an arbitrary P2P streaming system, and can identify viewers in the system. The use case diagram of the system can be found in Appendix C.

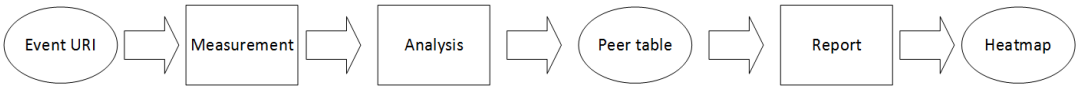

Figure 3.1: Work process of monitoring system

The monitoring task is divided into 3 phases: measurement, analysis and reporting, as shown in figure 3.1. The measurement phase deals with collecting measurement data from a given P2P streaming event, which can be a sport game, TV show, concert, and so on.

The analysis phase is responsible for distinguishing the roles of peers by performing flow statistics analysis. This phase is needed because we collect measurement data of all observed peers in the previous phase, but not all peers in a P2P streaming network are viewers [19]. The emphasis of our analysis task is mainly on the detection of viewers. In this phase, a peer table that contains each peer's IP address and peer's detected role is generated.

The report phase deals with finding each peer's geolocation based on the peer table and produces human-readable report.

3.2 Methodology

3.2.1 Measurement

As introduced in Chapter 1, we adopt passive measurement method. The monitor in our system collects packets associated with the target applications, which are used to construct flow-level traffic information [38, 68]. A flow can be a TCP connection or a UDP stream, which represents a continuous sequence of packets from a source to a destination. Each flow is identified by a 5-tuple: (source IP, source port, destination IP, destination port, type of transport protocol). We use flow-level information because studies from [72, 23, 29] show that it is the most used one in traffic analysis [29, 68].

3.2.2 Analysis

We propose a viewer detection method by analysing video data traffic. Currently, there are three types of traffic classification methods: port analysis, payload analysis, and flow statistics analysis [48, 68].

- Port analysis: this method depends on mapping of well-known port numbers to protocols. For example, the packet with destination port 80 is classified as HTTP traffic. This method is highly efficient if we can map port to known applications or protocols. However, this method is becoming less effective due to the fact that some applications allocate dynamic ports, which is often the case for P2P applications [48].

- Payload analysis: this method relies on the knowledge about the payload formats for Internet applications. Study of [59] shows that the payload-based analysis is accurate. However, encryption makes this method less practical [68]. And most of P2P streaming systems use proprietary protocols, which are often encrypted. This method is not applicable for our system.

- Flow statistics analysis: Flow statistics analysis maps instances of network traffic flow into different classes based on flow's statistics. A flow is described by a set of statistical features and associated feature values. A feature is a descriptive statistic (e.g average packet size, duration) that can be calculated from one or more packets. Some measurement studies use a heuristic filtering method to classify flows [30, 61]. Machine-learning is also a widely used approach for flow statistics analysis [23, 50].

From above, wed find that only flow statistics analysis can fulfil our goal. We adopted both heuristic filtering and machine learning approaches.

Heuristic filtering

The authors of [30] use this method to isolate video data traffic. The heuristic is based on their measurement studies on many P2P streaming systems. They found that if in a flow more than 10 large packets ($>$ 1000 bytes) are found, this flow is a video data flow [30, 61]. The authors of [30] state that heuristic method can also be applied to other P2P streaming systems. We use this approach to detect data flow.

K-means clustering

As shown in [23, 50], machine learning can help to distinguish different types of traffic. Machine learning falls into two categories: supervised and unsupervised [23, 50].

- Supervised machine learning: It is also known as classification. The goal of classification is to build a classification model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the new instances where the values of the predictor features are known, but the value of the class label is unknown.

- Unsupervised machine learning: It is also known as clustering. Clustering is the partitioning of objects into disjoint groups, referred as clusters, such that objects within a group are similar according to chosen criteria [23, 50]. There are many popular algorithms: K-Means, AutoClass, DBSCAN , which vary in speed and accuracy.

Supervised learning produces a model that fits the training data, where the training data is a priori available [23]. The training data is a collection of instances with known label values (instance's class). In contrast, unsupervised learning uses unlabelled training data to find similarities or patterns among objects in the data set. As our target is to monitor proprietary P2P streaming

systems, there is no ground truth of the traffic we measured [64, 50]. Because of the lack of labelled data, unsupervised machine learning is the only applicable machine learning approach for our system.

We use K-means algorithm since it is one of the quickest and simplest clustering algorithm [50]. K-means clustering has been shown having good performance in traffic classification in [23]. K-means clustering aims to partition instances into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. In our system, we select the average size of packets as the flow feature for clustering.

3.2.3 Report

The report module in our system maps IP address to its geolocation. The mapping of IP address is done through a public IP database. Based on the IP's geolocation information, the report module can generate a geographical map (an example is shown in figure 3.2).

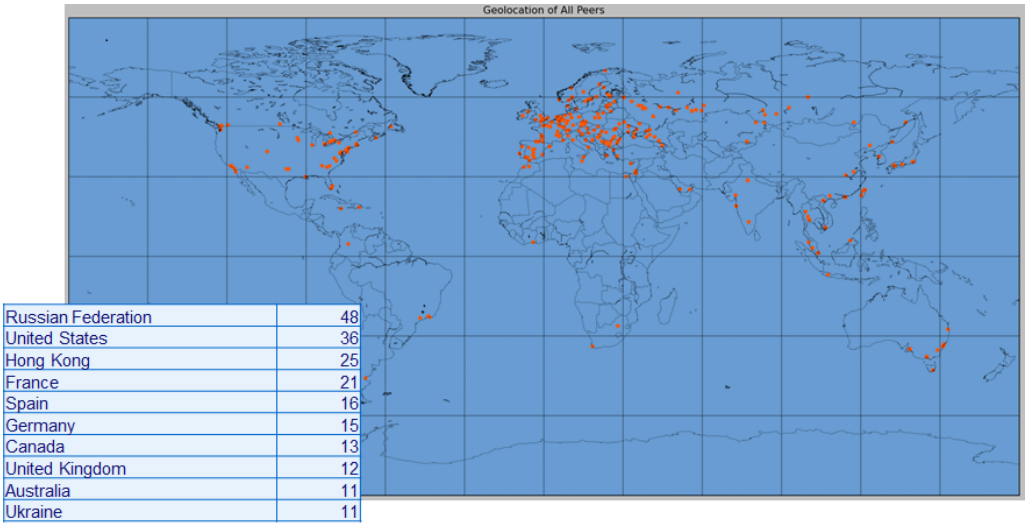

Figure 3.2: Report tool output example

3.3 System overview

The overall architecture of our system is shown in figure 3.3. There are four types of nodes in our system: manager, master, monitor and analyser. The manager, master and monitor are responsible for the measurement phase. The analyser is in charge of the analysis and report phase.

In the measurement phase, each event can be monitored by multiple monitors. A monitor implements passive measurement to observe the traffic associated with the P2P client installed on the same computer. Also, we use "choking" mechanism that improves the passive measurement performance. The idea of "choking" mechanism is to limit the P2P client's network connectivity to its current neighbours and force the client to contact new peers. We assume if we break our monitored client's connections to its neighbours, we can observe new peers.

A task group is a collection of monitor nodes dedicated to monitoring a single event. In a task group, there are a set of monitor nodes and a master node that coordinates monitors. By having multiple task groups, we can monitor multiple events simultaneously.

The manager node is responsible for allocating monitor nodes to each task group and balance the load of the monitoring system. As mentioned in Section 2.2, P2P streaming networks are diverse in scale, which means events differ in popularity. The manager can allocate different number of monitors to different tasks based on their popularity.

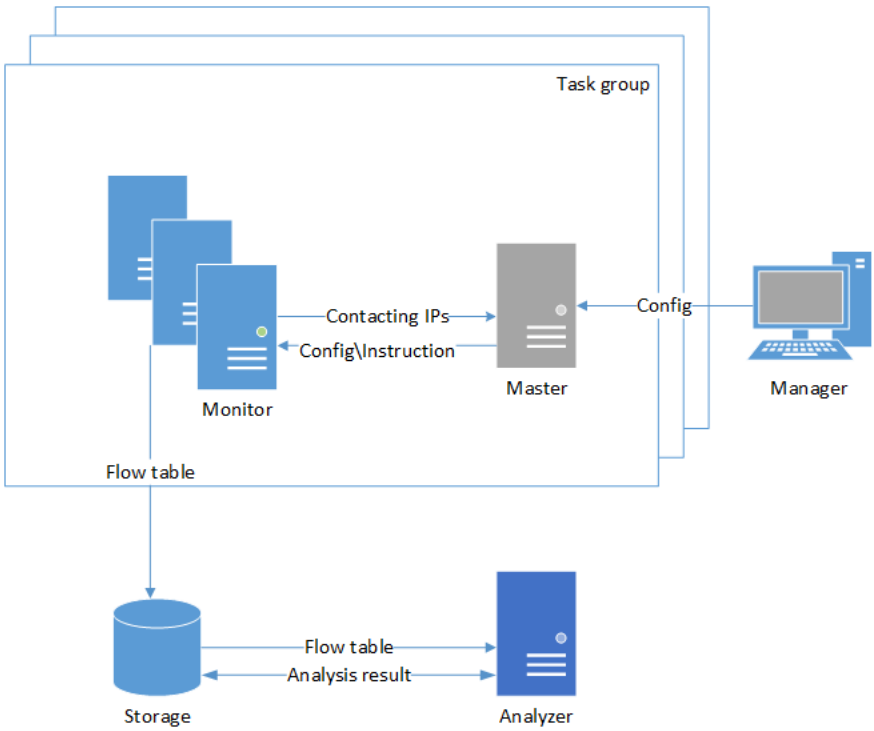

Figure 3.3: P2P streaming monitoring system overall architecture

In the next section, we introduce each of these components in detail.

3.4 System design

3.4.1 Monitor

We can launch a monitor to monitor an event. Monitor can start the installed P2P clients on the same computer and listen to the traffic associated with the client. During monitoring, the monitor stores captured traffic data to the storage and executes instructions from the master. Figure 3.4 shows the message diagram of the monitor.

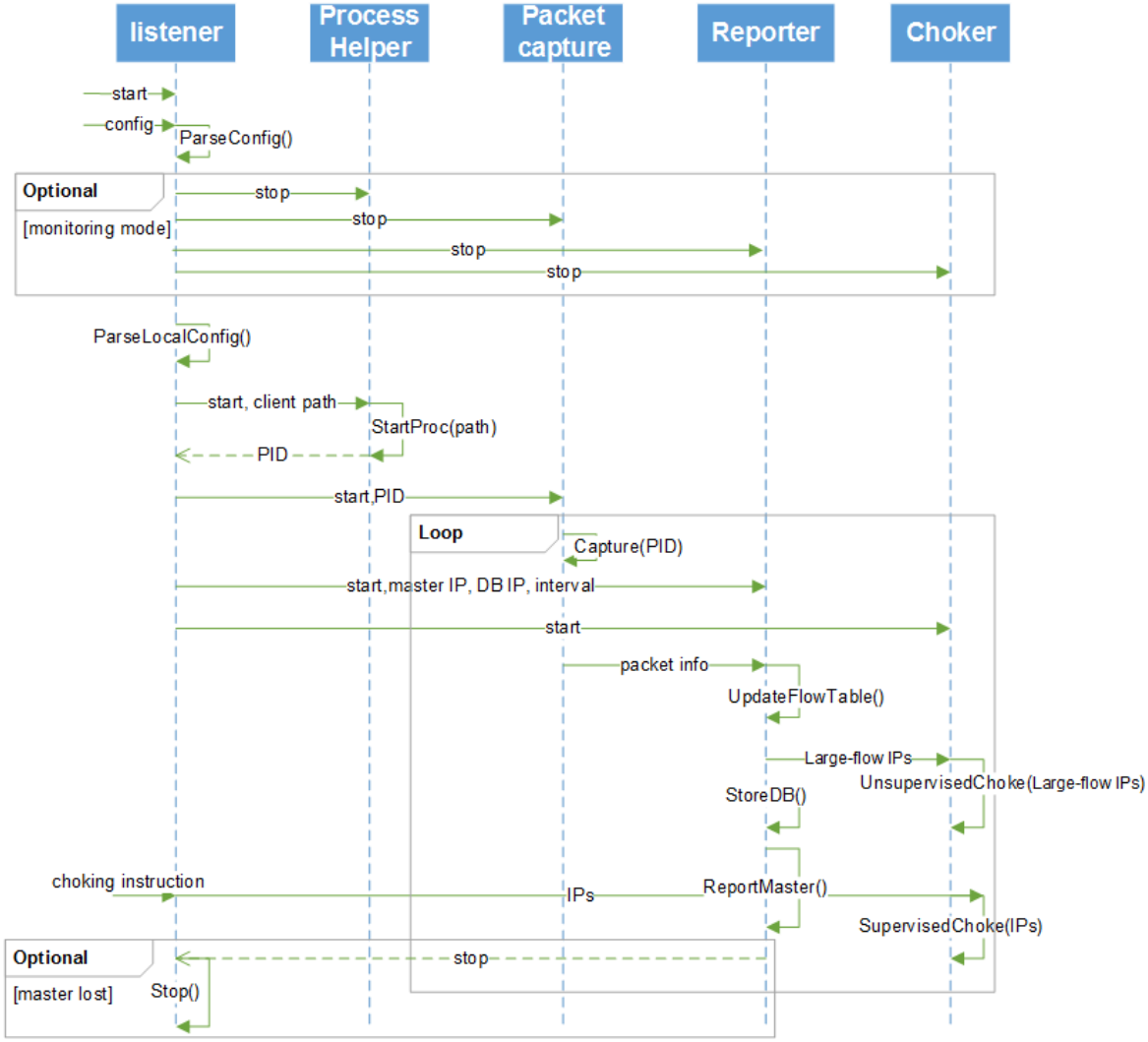

Figure 3.4: Message diagram of monitor

Listener

First, we present the work process of the listener. The listener starts and waits for configuration message from the master. After it has received a configuration (illustrated in Appendix D), the listener extracts information by parsing the configuration. The extracted information includes: the name of the P2P client, the IP address and port number of the master node and database, reporting interval and setting for algorithms. Then the listener checks whether other components are in working mode.

If the components are in working mode, the listener sends a stop message to each of them. Then the process helper shuts down the running P2P client, the packet capture clears the buffered packets and stops capturing, the reporter closes current connections to the master and database, and the choker empt the blocking IP list.

If they are not in working mode, the listener parses a local configuration (illustrated in Appendix D) to find the path of the P2P client. After locating the client, the listener sends a start command, with the client path, to the process helper. The process helper starts the client and returns the Process Identifier (PID) back to the listener. To capture the traffic of the client, the listener sends start message to the packet capture. The packet capture intercepts the traffic from/to the client and extracts packet information. The reporter receives a start message that contains the IP addresses and port numbers of the master node and database, and connects to them. Periodically, the reporter stores captured information (from the packet capture) to the database and sends observed IP addresses to the master. The listener also sends a start message to the choker, which creates an empty IP list. In the next steps, the IP list contains IP addresses that should be blocked bidirectionally.

The listener also accepts choking instruction from master. The choking instruction is a list of IP addresses that should be blocked by the monitor.

Packet capture

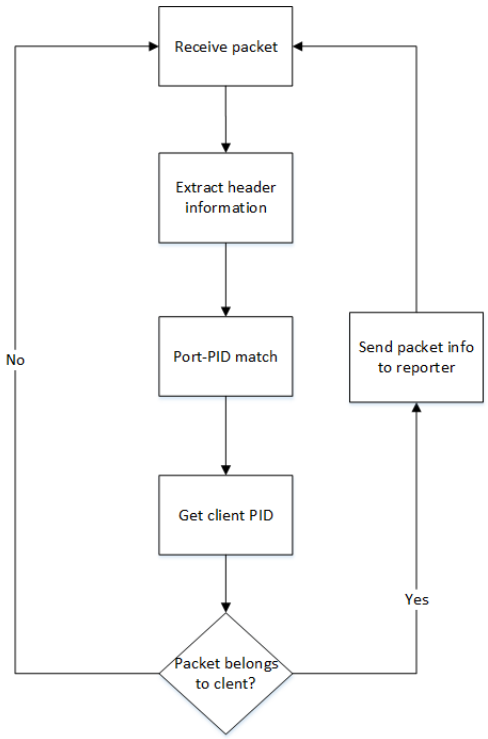

Figure 3.5: Work process of the packet capture

The work process of the packet capture is shown in figure 3.5. When the packet capture receives a packet, it extracts the following information from the packet's headers:

- source IP address
- source port number
- destination IP address

- destination port number
- protocol (TCP or UDP)
- size of packet (in bytes)
- captured time (in milliseconds)

The packet is captured through SharpPcap library, which utilizes Network Driver Interface Specification(NDIS). NDIS is an interface between the Network Interface Card (NIC) and other software. In OSI model, it is a part of layer 2. From NDIS, we can intercept the traffic from NIC to upper layer's software. Thus, through the SharpPcap library, we can get all the packets from the NIC.

Next, the packet capture gets a PID-port table through Windows IP-helper API. The PID-port table contains information of currently opening ports and ports' corresponding PIDs. The packet capture compares the PID of the P2P client with the PIDs we get from the packet's port numbers (source and destination). If the the source port's corresponding PID is equal to the PID of the P2P client, we regard the packet as a packet going from the client. If the destination port's corresponding PID is equal to the PID of the P2P client, we regard the packet as a packet going to the client. Otherwise, the packet is ignored.

When the packet capture gets a packet belonging to the P2P client, it sends the extracted packet information to the reporter.

Reporter

The reporter maintains a connection to the master node and a connection to the database. It also maintains a flow table data structure. The flow table contains the following fields:

- source IP address
- destination port number
- source IP address
- destination port number
- type of protocol (TCP or UDP)
- total number of packets
- total length of packets (in bytes)
- duration (the captured time of the last packet minus the captured time of the first packet)
- number of large packets ($>$ 1000 bytes)

In Section 3.2.2, we introduced the heuristic filtering method. This method is mainly implemented in the reporter. As shown above, the flow table has a "number of large packets" filed. When the reporter receives packet information from the packet capture, it updates the flow table, as shown in figure 3.6. From the table, the analyser is able to determine the traffic type of the flow by counting large packets. The reporter itself also detects large-flow IPs and sends to the choker.

Periodically, the reporter sends observed IPs to the master, stores captured traffic's flow table to the database and sends detected large-flow IPs to the choker. Since we adopted a distributed design, we did not provide a "stop" option to the monitor. Instead, if the reporter looses its connection with the master, the monitor stops working and waits for configuration in the idle mode.

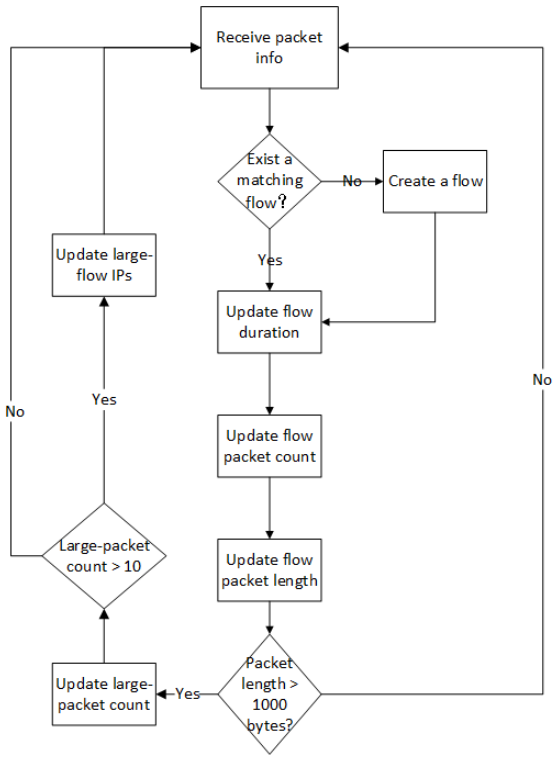

Figure 3.6: Flow updating process of reporter

Choker

We propose a choking mechanism to improve the measurement ability of the monitor. We assume that breaking the P2P client's connections with its neighbours can force the client to contact new peers. The assumption is made based on P2P networks' collaboration essence, which states that each peer shares its resource and tries to get resources from other peers [20].

The choking mechanism is implemented using Windows Firewall API. When the choker starts, it creates a firewall rule through the Windows Firewall API. Windows Firewall is an software that can block packets that match some give conditions. The firewall rule is initialized as a bidirectional blocking rule with an empty domain. If we add one IP to the rule's domain, the bidirectional traffic of this IP is blocked by the firewall. When the choker receives IP addresses from the listener, it populates the firewall rule with the received IPs.

The choking mechanism needs to be conducted by an algorithm. In the monitor, we use an unsupervised choking algorithm to improve the monitor's measurement ability. The principle of this algorithm is that we assume that peers which have large amount of traffic with the P2P client are neighbour peers, and we choke them. The reason we call this algorithm unsupervised is that the monitor decides what IPs to choke by itself, without external instructions. As shown in figure 3.4, periodically the reporter sends detected large-flow IPs to the choker, and the choker performs unsupervised choking.

We have a setting for choking start-up-delay. We only enable choking on the client after the client has been running for a period of time because the client has high peer discovery ability at its initial phase (shown in our experiments in Chapter 4), and we do not want to interfere with it.

3.4.2 Master

The message diagram of the master is shown in figure 3.7. Master starts and listens to configuration from the manager. After receiving a configuration, the master parses it and finds the IP addresses

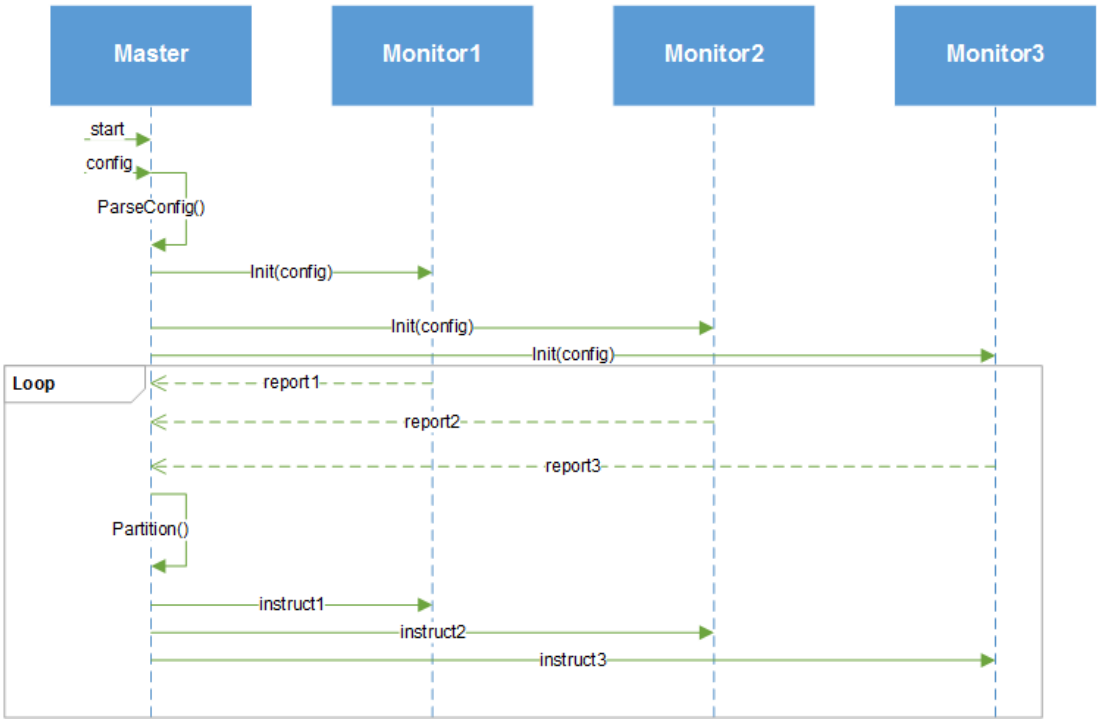

Figure 3.7: Message diagram of master with 3 monitors

of the monitor nodes in its task group. Then, the master distributes the configuration to all the monitors under its task group. As described in Section 3.4.1, the monitor that has received the configuration message will start monitoring the client. Periodically, each monitor reports back its observed peers' IP addresses to the master.

Having multiple monitors brings a challenge: peers can be monitored redundantly. To solve this problem, we use a supervised choking algorithm which aims to keep each peer monitored by a single monitor. The algorithm works as follows: each time the master receives all its monitors' reports, it looks for IP address that observed by more than one monitor. For each found IP address, the master identifies one monitor that shall allow its client to keep the connection with that IP. Other monitors will receive a choking instruction for that IP. When making the decision, the master will take duration of clients' connection with that IP in to account.

This decision strategy is based on the study of [61], which shows that the duration of data flow is much longer than control flow. Since the data flow is relatively stable, we consider monitor that has longer connection time with a remote peer is more likely to be in data flow communication, and we do not choke this monitor's connection.

3.4.3 Manager

The manager provides the operator with the access to the monitoring system through a configuration file (illustrated in Appendix D). Through the configuration file, the operator is able to configure the events of each task group and allocate monitors to the task group. When the configuration file is modified, the manager re-distributes the configuration to the masters. By updating the configuration file, the operator is able to dynamically allocate running monitors to different task groups.

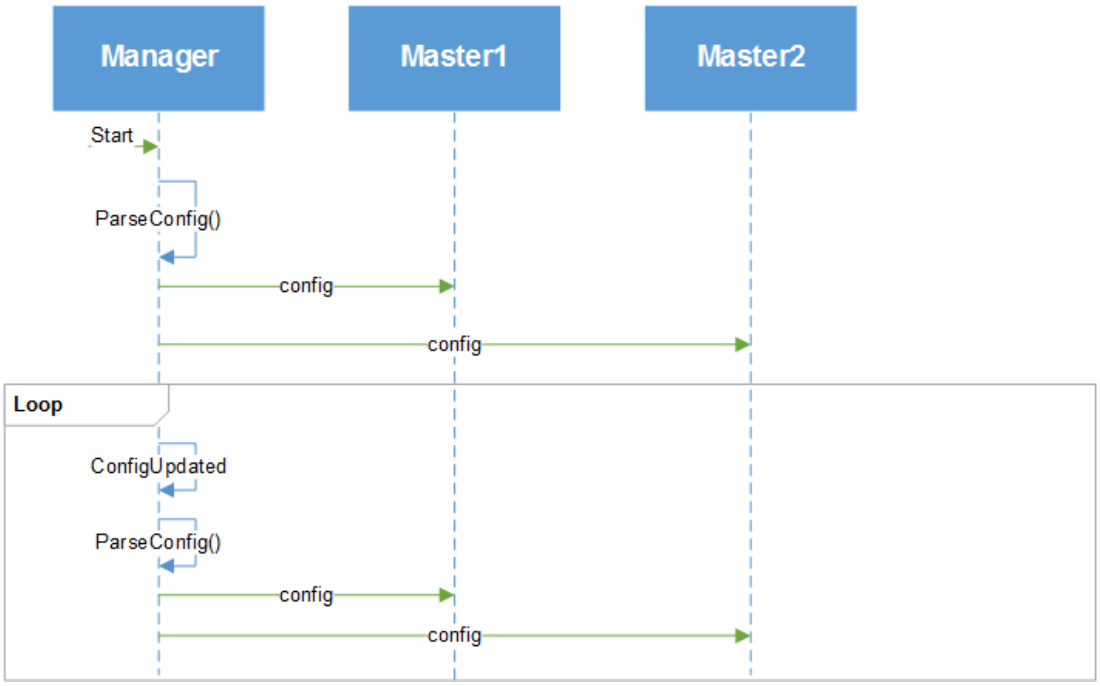

Figure 3.8: Message diagram of manager with 2 masters

3.4.4 Analyser

The analyser is responsible for detecting viewers and getting peer's geolocation information. We implemented both heuristic filtering (partial) and K-means methods in the analyser.

For heuristic filtering, the analyser reads the flow table of an event and finds all flows that has more than 10 large packets (> 1000 bytes). Then the analyser directly marks these flows as data flows.

The K-means algorithm has two parameters: number of clusters (K) and features. We set K to 2 since we are clustering data traffic and other traffic. For the clustering feature, we select average packet size of a flow because in this project we focus on the relation between packet size and traffic type. The clustering analysis process works as follows: when the analyser starts, it reads the flow table of an event, then it clusters the flow data into two groups. The flows in the group with larger average packet size are regarded as data flows.

Both methods produce the same result: detected data flows. The Analyser extracts IPs from the data flows and produces a peer table, which contains information of a peer's IP and its detected role (viewer or not). By using a public IP database [3], the analyser can find the geolocation of each IP in the peer table.

3.4.5 Storage

We use Redis database [8] as storage solution. In its outer layer, the Redis data model is a dictionary which maps keys to values. One of the main differences between Redis and other structured storage systems is that Redis supports not only strings, but also abstract data types:

- Lists of strings
- Sets of strings (collections of non-repeating unsorted elements)
- Sorted sets of strings (collections of non-repeating elements ordered by a floating-point number called score)

- Hashes where keys and values are strings

Table 3.1 shows the design of our redis database. For each key, the key name is split into name regions by ":" symbol. The name region rounded with * means a literal value and the other means variable. For example, $uri : *flow\_table* : round : flow\_key$ can be a value as $cctv1 : *flow\_table* : 2 : 107.178.208.111, 80 - 131.155.226.244, 52040 - TCP$, which represents the event is cctv1, the value name is $flow\_table$, the monitoring round is the 2, the key of the flow is $107.178.208.111, 80 - 131.155.226.244, 52040 - TCP$.

Key	Value Type	Explanation
uri:*max_round*	Int	Maximum number of round of an event.
uri:*flow_table*:round:flow-key	Hash	Flow table of an event.
uri:*peer_table*:round:ip	Hash	Peer table of an event.
uri:*IPs*:round	Set	All contacted peers' IPs of an event.
uri:*max_sessionno*:round	Int	Number of monitors.
uri:*partitions*:round:session-no	List	Each monitor's contacted peers' IPs.

Table 3.1: Storage design

- $uri : *max\_round*$: Largest number of runs of an event's monitoring. When the master receives a task, the master increase the value of $uri : *max\_round*$ by 1. From this value the monitors and analyser can know their round of monitoring.
- $uri : *flow\_table* : round : flow - key$: The item's value is a hash type, which has the same structure with the flow's we introduced in Section 3.4.1. The reporter stores the flow table in this item.
- $uri : *peer\_table* : round : ip$: The item's value is a hash type, which contains two fields: role and geolocation. The analyser stores the peer table in this item.
- $uri : *IPs* : round$: Each round's observed IPs of an event.
- $uri : *max\_sessionno* : round$: Each round's number of monitors.
- $uri : *partitions* : round : session - no$: Each monitor's contacted IPs in a round of an event's monitoring.

Chapter 4

Evaluation

In this section we present five sets of experiments on our monitoring system, first four evaluate the measurement performance of our system, the last one evaluates the analysis performance.

First we identify the target P2P streaming systems. Previous studies from [30] and [61] introduced some popular P2P streaming systems such as Sopcast, PPLive, PPStream and TVAnts. From these applications, we only selected Sopcast as one of our target systems. The reason is that PPstream and PPLive are used to distribute content legally [6, 7] and TVAnt is getting outdated.

Additionally, we selected another two emerging popular systems–Acestream and Peerflix. AceStream uses a P2P distribution based on BitTorrent protocol to share live video streams between users [16]. The majority of streamers are now using AceStream instead of SopCast as it offers higher quality [16, 14]. Peerflix is a P2P streaming engine based on the BitTorrent network, and it is the core engine of Popcorn time [24]. Popcorn time is a multi-platform, free and open source P2P streaming application. As stated in [17], the program could stream unauthorized copies of films directly from YTS (a pirated content download website) and other torrent trackers. The program quickly received unexpectedly positive media attention, with some comparing it to Netflix due to its ease of use [17]. We focus on Peerflix instead of Popcorn time because of two reasons: they both use the same P2P network; Peerflix program can accept URI as parameter, which makes monitoring easier.

	Acestream	*Sopcast*	*Peerflix*
Service type	Live, VoD	Live, VoD	VoD
Service discovery	Web	Web, client-embedded	Web
Client composition	Separated engine and player	Integrated engine and player	Separated engine
Protocol	TCP, Proprietary, BT	UDP, TCP, Proprietary protocol	TCP, BT
Codecs	h.264	wmv, h.264	No specification
Container	mpeg-ts	asf, wmv, mpeg-ts, mpeg-ps,	No specification

Table 4.1: Comparison of target P2P streaming systems

We select Sopcast, Acestream and Peerflix as our target P2P streaming systems not only because of their popularity, but also because they are representative of the overall population of P2P-based solutions. In table 4.1 we present our target systems and compare them based on technical features selected from the characteristics in Appendix A. We can see that these three systems have diverse technical features and we believe that the verification of our monitoring system on these three systems is able to show that our system can be applied to an arbitrary P2P streaming systems.

4.1 Measurement

4.1.1 Distributed experiment

The aim of this experiment is to verify that our distributed architecture can improve the overall measurement performance. We deploy two monitors (A and B) in the same task group, both with only unsupervised choking enabled. During experiment, we record Ips that they observed. At the end, we compare the average number of observed peers by each monitor with the union of their observations. We expect the task group's number of observed peers is larger than its individual monitor's.

During evaluation, we find that it is hard to guarantee all nodes have similar ability to observe peers. The number of observed peers of different nodes may vary a lot. Moreover, we only have limited number of nodes for experiments. Thus we run the experiment several times and create an average observation. Thus in our experiment, we run these two monitors on the same channel for 5 times. We assume the popularity of this channel is stable during our experiment time.

Figure 4.1 and figure 4.2 show the experiment results on Acestream and Peerflix respectively. Y-axis represents the average number of observe peers. X-axis represents the number of reporting rounds with one round taking up 10 seconds.

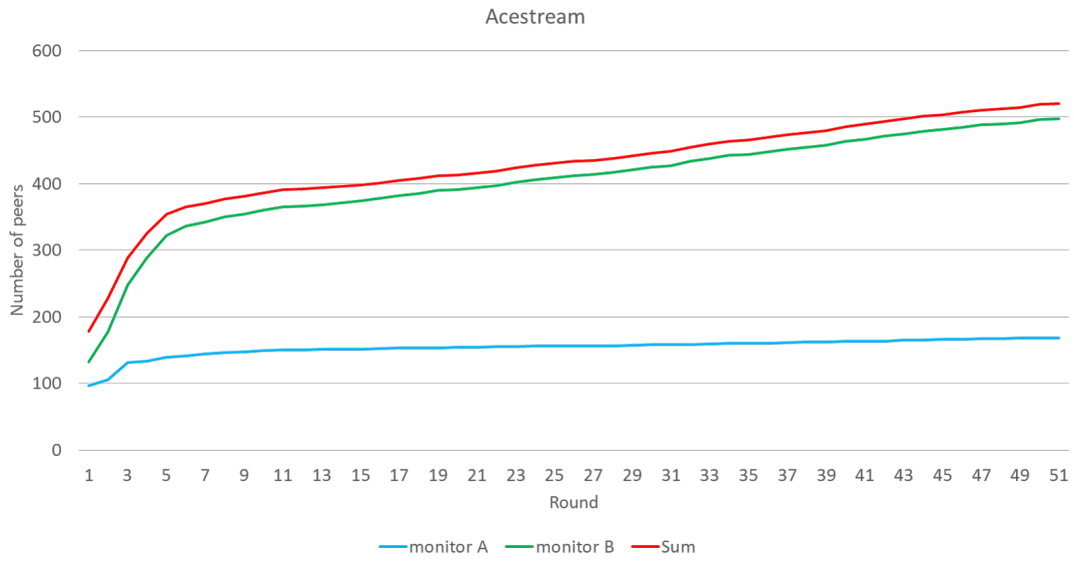

Figure 4.1: Distributed experiment on Acestream

From the result of Acestream we can observe that the number of observed peers by the task group is larger than any of its monitor's. We also find that monitor B contributes most of the observed peers, which means the peer discovery ability of nodes in Acestream are diverse. Another observation is there exists a large overlap between the each monitor's observation of the swarm, which shows the problem of inefficiency.

In figure 4.2, the experiment result of Peerflix also confirms that distributed architecture improves the overall measurement performance. However, there is no significant difference in the peer discovery ability of two monitors. We consider the difference in Acestream and Peerflix results is due to two systems' different peer discovery strategy implementation.

4.1.2 Unsupervised choking experiment

The aim of this experiment is to verify that unsupervised choking can improve a single monitor's measurement performance. The experiment is conducted in a continuous period of time. We

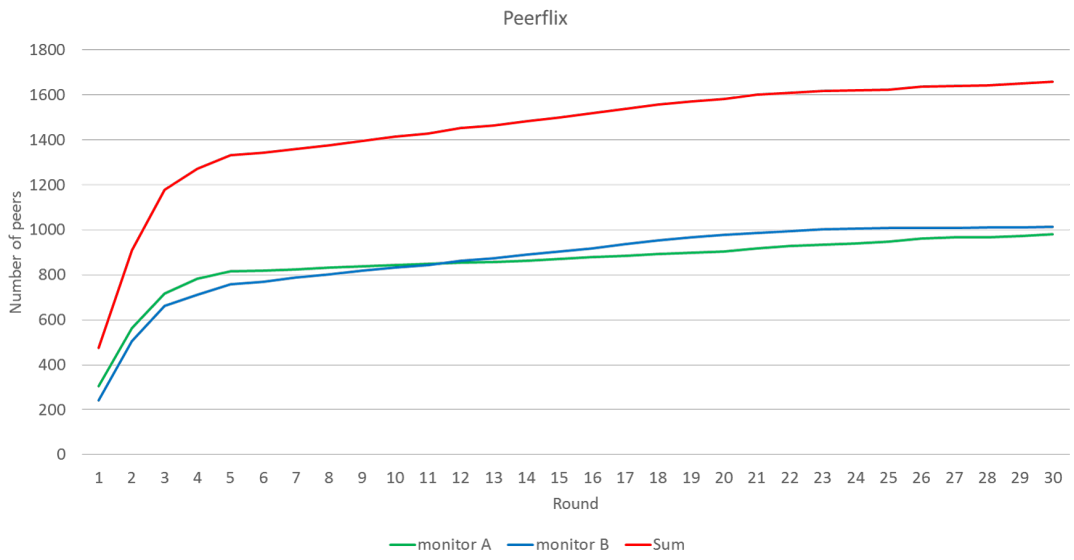

Figure 4.2: Distributed experiment on Peerflix

assume the popularity of selected channels are stable during evaluation period. We let a single monitor monitor a channel 5 times with unsupervised choking enabled and disabled by turns. The start-up-delay is set to 30 seconds for Peerflix, and 50 seconds for Acsetream and Sopcast based on our experience.

In the end, we compare the number of the observed peers with and without unsupervised choking enabled. We expect that monitor with unsupervised choking enabled can observe more peers.

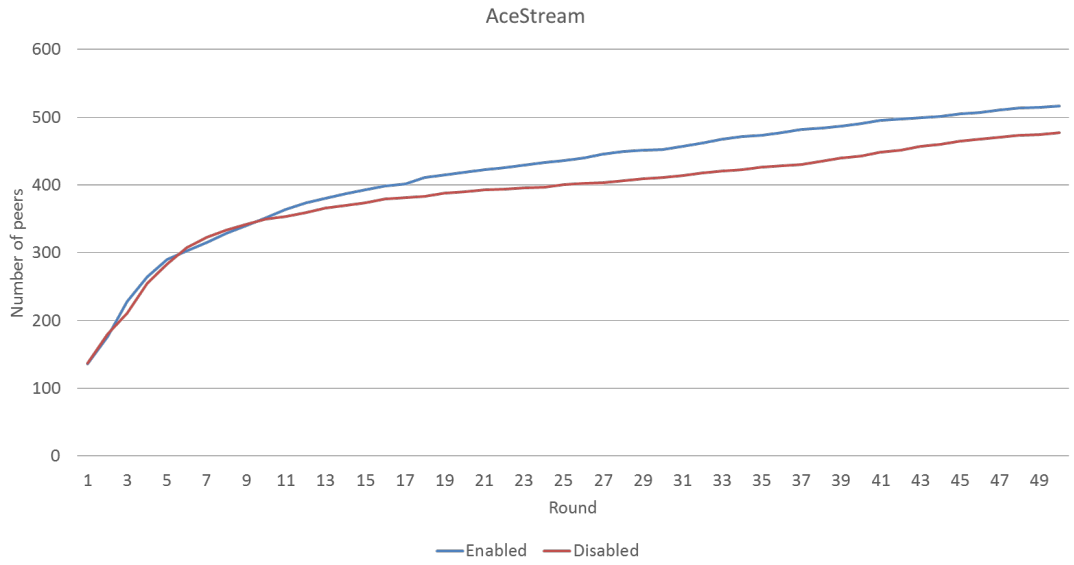

Figure 4.3: Unsupervised choking experiment on Acestream

Figures 4.3, 4.4, 4.5 show the experiment result on Acestream, Peerflix and Sopcast. Y-axis represents the average number of observed peers. X-axis represents the number of rounds (round is 10 seconds). From figure 4.3 and 4.4 we can see, the unsupervised choking enabled monitor has

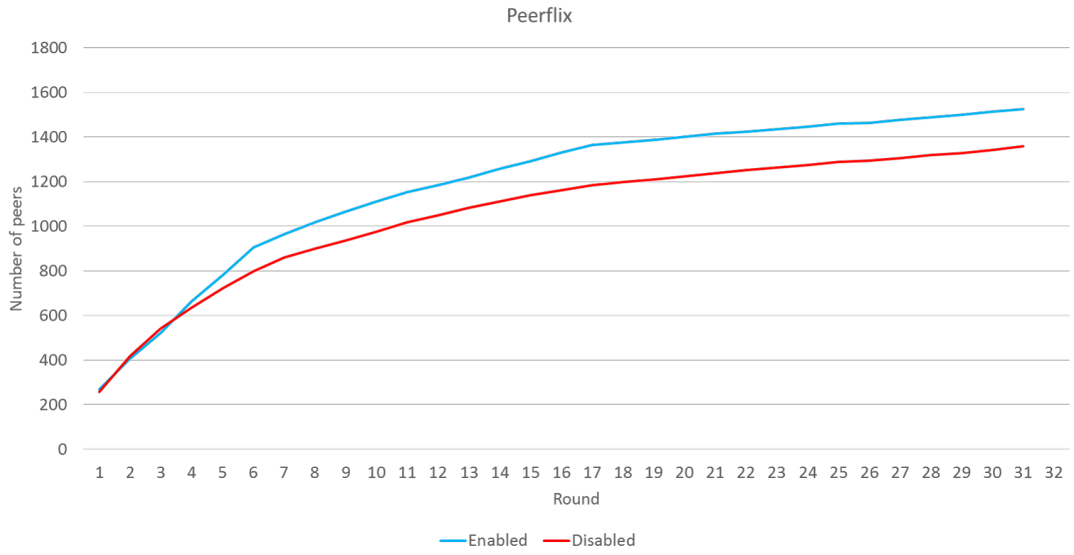

Figure 4.4: Unsupervised choking experiment on Peerflix

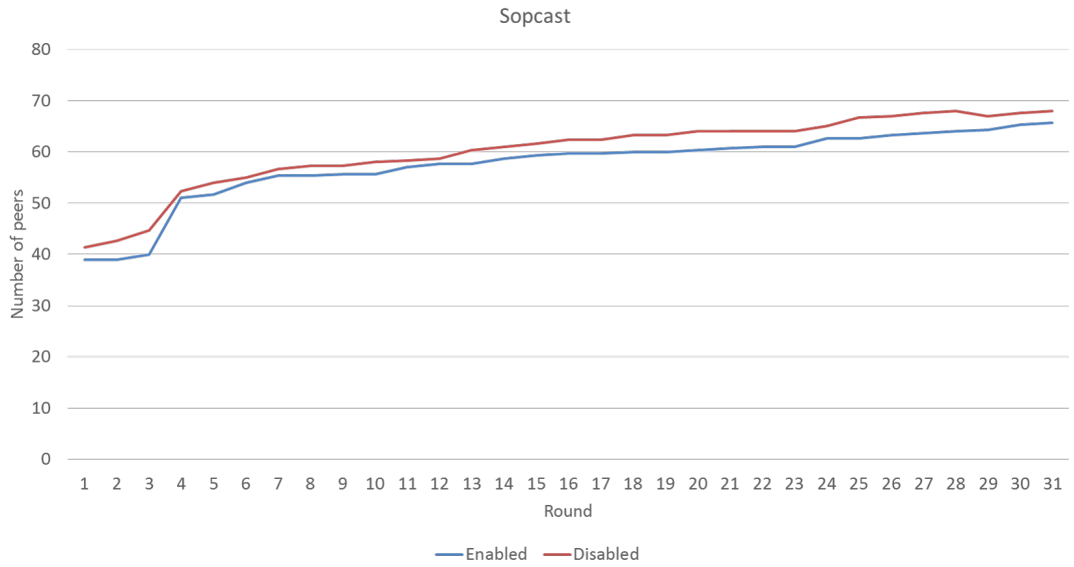

Figure 4.5: Unsupervised choking experiment on Sopcast

a larger average number of observed peers.

Figure 4.5 shows the experiment results of Sopcast. We find there is almost no difference between two cases. After inspection, we find that the firewall-traversing technique of Sopcast makes firewall-based choking mechanism invalid. It is our future work to adopt a new choking method instead of our currently-used firewall-based. Because of this finding, following experiments are only conducted on Acestream and Peerflix.

To sum up, unsupervised choking strategy can improve the monitor's measurement performance on Acestream and Peerflix.

4.1.3 Supervised choking experiment

The aim of this experiment is to verify that supervised choking can reduce the ratio of redundantly monitored peers among monitors, and improve the overall measurement performance.

Similar with the previous experiment, we run two monitor nodes and a master node for 5 times, with supervised choking enabled and disabled by turns. We also enabled unsupervised choking strategy in both cases, to have a better measurement performance of a single monitor. For each run of experiment, we recorded the number of observed peers and overlapping ratio ($\frac{no.redundantly\_monitored\_peers}{no.observed\_peers}$). We expect to see that supervised choking enabled case has larger average number of observed peers, and has lower overlapping ratio.

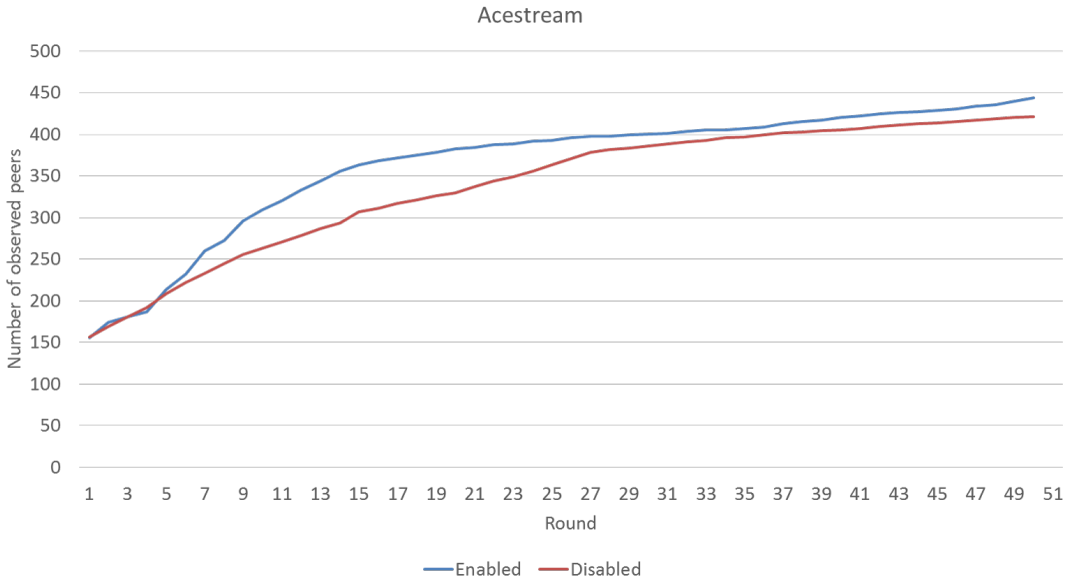

Figure 4.6: Supervised choking experiment on Acestream, number of observed peers

From figure 4.6, we can see that for Acestream, supervised choking enabled case has larger number of observed peers. From figure 4.7, we can see that supervised choking can obviously reduce the ratio of overlapping between two monitors. From figure 4.8, and figure 4.9, we can see similar results for Peerflix.

4.1.4 Dynamic assignment experiment

The aim of this experiment is to verify that our dynamic resource allocation design can improve the overall measurement performance of the system.

In this experiment, we set-up two task groups, named task 1 and task 2. Each group consists of 2 monitors, with both supervised and unsupervised choking enabled. We assign to task 1 a channel with high popularity, and assign to task 2 a channel with very low popularity. After two

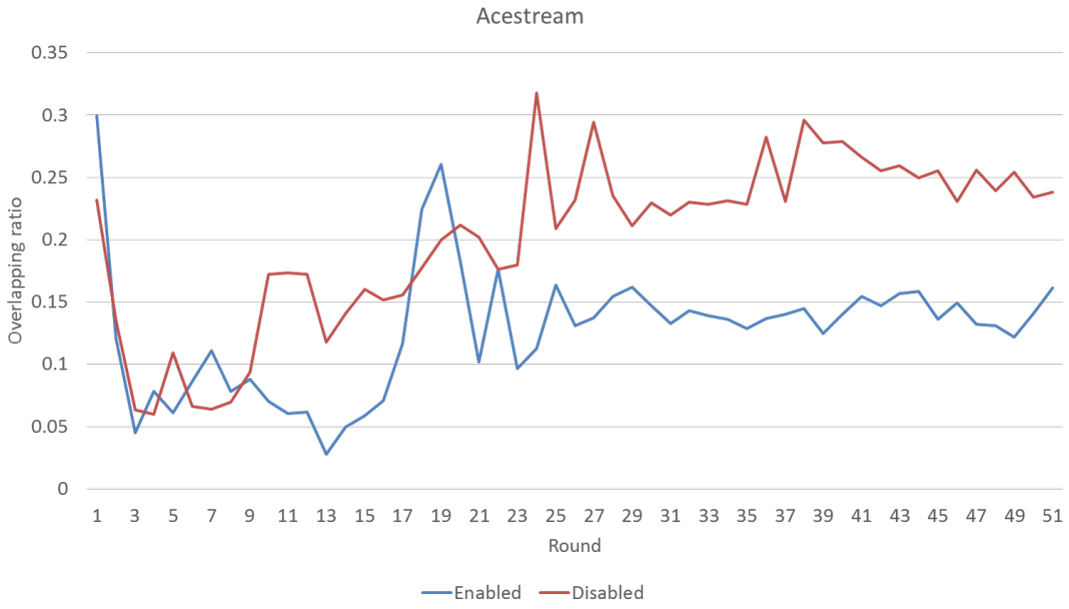

Figure 4.7: Supervised choking experiment on Acestream, overlapping ratio

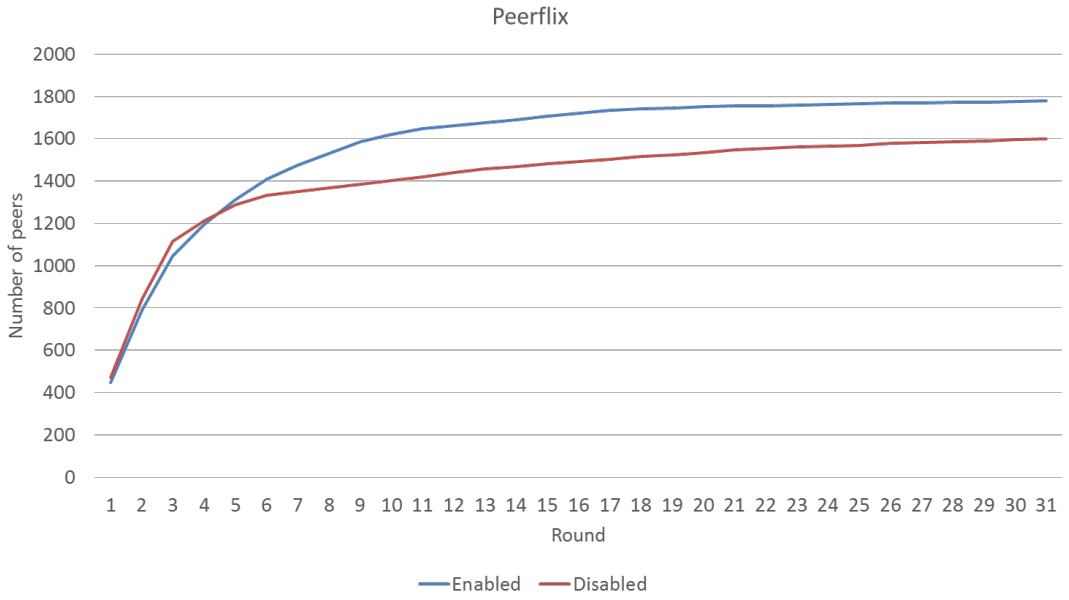

Figure 4.8: Supervised choking experiment on Peerflix, number of observed peers

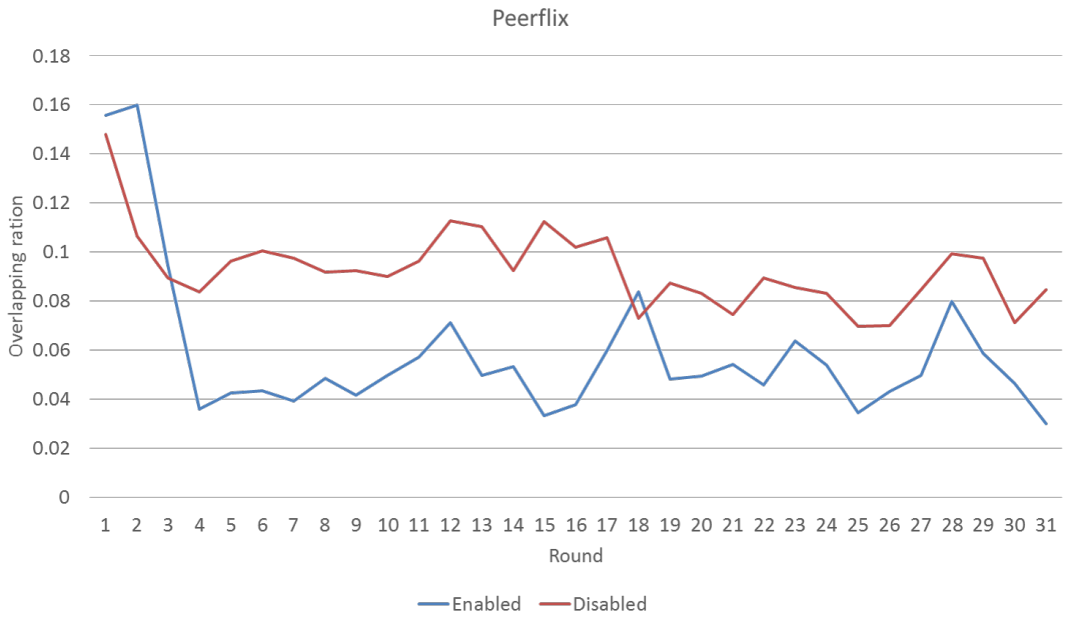

Figure 4.9: Supervised choking experiment on Peerflix, overlapping ratio

tasks have been running for a while (we choose 20 rounds on Acestream, 11 rounds on Peerflix), we re-assign one node from task 2 to task 1. We use this experiment to simulate the scenario that during monitoring we re-allocate under-utilized monitors to another task in order to balance the load of the whole system. We expect that overall observed number of peers can increase after reallocation.

There is a difference in Acestream experiment that we cannot use the number of observed peers as a metric of channel popularity. This is because we find Acestream client contacts a large number of non-viewers regardless of the popularity of channels. Instead, we use another metric: number of viewers detected by the heuristic filtering method.

From figures 4.10 4.11, we can see that around round 11 for Peerflix and round 21 for Acestream (which corresponds to re-allocation time), the number of observed peers increases.

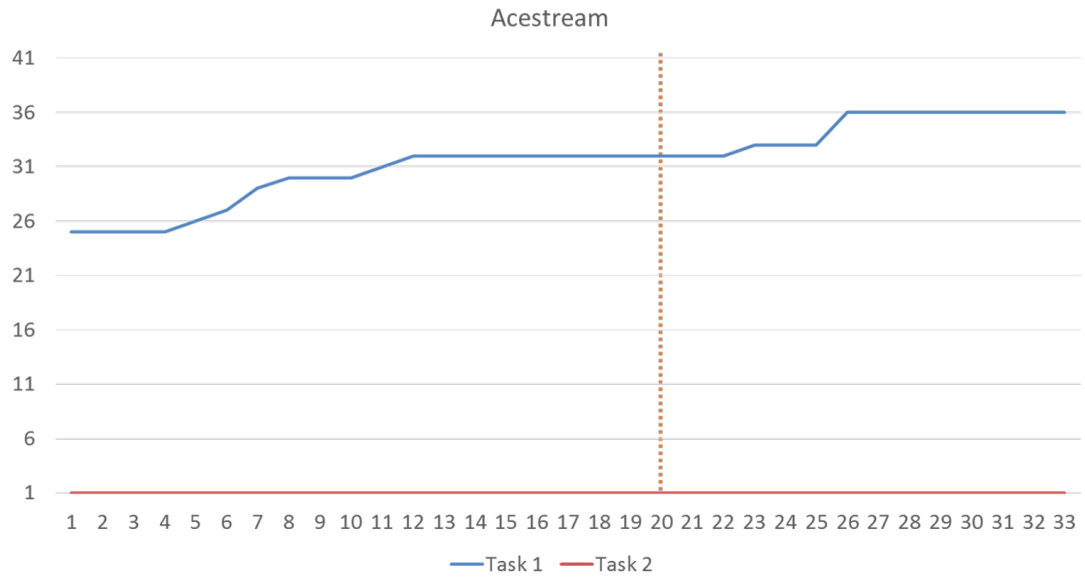

Figure 4.10: Dynamic monitor assignment experiment on Acestream

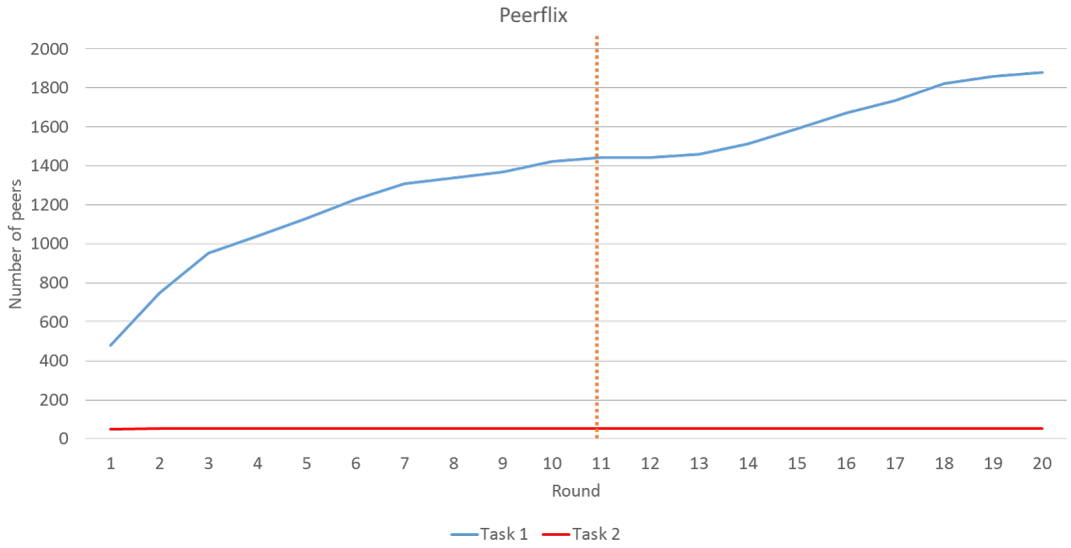

Figure 4.11: Dynamic monitor assignment experiment on Peerflix

4.2 Viewer detection

We evaluated the performance of our two traffic classification methods on Sopcast which has been widely studied [19, 61]. In this experiment, we compare the ratio of detected viewers among all observed peers in an monitored event by using two proposed methods, there is no ground truth in encrypted P2P communication [51].

In [19], the authors show that in Sopcast, the ratio of contacted viewers among contacted peers is around 37.29%. Similarly, in [33], the authors show this ratio is around 35%. We selected 4 Sopcast events to monitor. For each event, we monitored 3 times, each time for 20 minutes. We expect that our both methods provide a ratio that is comparable the other two studies' results.

The comparison result is shown in the table below.

Classification method	**Ratio of viewers among all observed peers**
Ciullo et al	37.29%
Horvath et al	about 35%
Heuristic filtering	32.72%
K-means	24.68%

Table 4.2: Viewer detection experiment

We find that the result of heuristic filtering is close with the previous studies'. We assume the reason for such similarity is that those two studies also use the size of packet as a heuristic. K-means detects less viewers, which means it performance is not as good as heuristic filtering on Sopcast. In future we need further study on K-means to improve its performance. We consider the input data and selection on features as two important factors for the future study.

Chapter 5

Conclusions

In this project, we proposed a solution for measurement and analysis of an arbitrary P2P streaming system. In order to achieve this goal, we studied the architectures of P2P streaming systems, traffic measurement methodologies and traffic analysis methodologies. In the end, we presented a monitoring system based on passive measurement and flow-level traffic statistic analysis.

The focus of our project is placed on the measurement part. In order to improve the measurement performance, we solved several challenges. The first challenge is the scalability of our measurement system: the monitor's measurement ability cannot scale with the size of the P2P network. To deal with this challenge, we adopted a distributed architecture, in which multiple monitors are coordinated by a central master node. The second challenge we solved is the visibility limitation of a single monitor. To address it, we proposed an unsupervised choking method that can force the monitor to observe more peers by breaking the client's connections with current neighbour peers. The third challenge we solved is the redundant observation of peers among multiple monitors. To solve this challenge, we proposed a supervised choking method. For each redundantly monitored peer, we keep one monitor's connection with that peer and choke the rest. The effectiveness of these approaches is verified in our evaluation experiments. In addition, our system architecture supports dynamic resource allocation, which can help us to balance the load among multiple monitoring tasks.

On the analysis side, we proposed a viewer detection method by traffic classification. We used a heuristic filtering approach to detect data flows and identify viewers. Additionally, we studied clustering algorithm to distinguish data traffic and control traffic. We found heuristic filtering has a good performance and clustering does not. However, the heuristic filtering method's accuracy largely depends on the setting of packet threshold, which may differ for different systems, whereas, clustering is universal for different systems. One limitation in our viewer analysis is that we only considered the role of viewer but not consider the role of uploader and downloader.

Currently, in our system the load balancing is a manual process as introduced above. In future, an automated load balancing module can be useful. For example, the module can automatically assign more monitors to the the task group with lower overlapping ratio ($\frac{no.redundantly\_monitored\_peers}{no.observed\_peers}$), since the lower ratio implies that a new monitor may contribute its resources to the task group.

One of the limitations of our monitor is that our firewall-based choking mechanism does not work with Sopcast, which has a firewall traverse feature. In the future we need to replace our firewall-based choking module with a new one. The new choking module should have the ability to catch and drop packets that matches a specified condition. Such module shall be based on the NDIS implementation (e.g. Netlimiter [4]).

Another limitation of our monitor is that each monitor can only run one monitoring task at a time, which causes a waste of computation resource. To break the limitation, we need to find a solution for two cases in figure 5.1. The first case is that each P2P player is connected with a corresponding engine. In this case, we only need to filter the packet to each engine, which is similar with our current implementation. The second case is more complex since only one engine receives packets from multiple players simultaneously, and we cannot associate packets with a particular

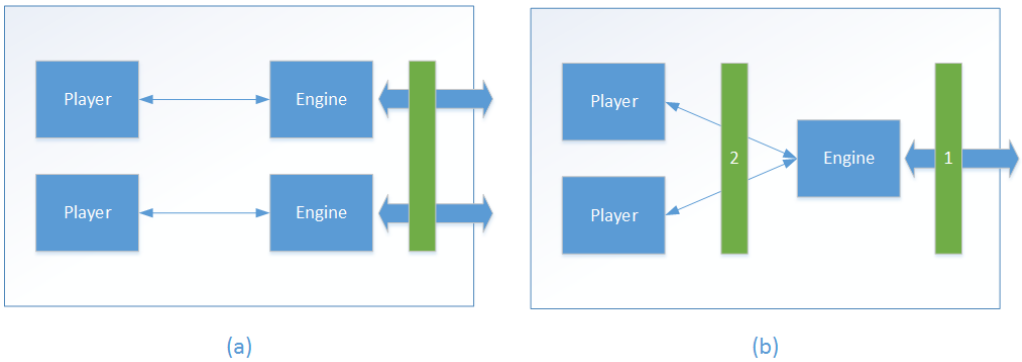

Figure 5.1: Principle of multi-client support

player. As the consequence, we cannot associate the observed IPs with the running events. For the cases shown in figure 5.1b. We need to work in two steps. First we filter the packets transited between remote hosts and the streaming engine. Second we filter the packets transmitted between the streaming engine and each player. For the packets captured in the second step, we cannot find its origin source IP address since we capture them locally. We need to match it with the original packet that captured in the first step. In this way, we can find the source IP address and port number of the packet belonging to each player's running event.

Bibliography

[1] Geolocation software. http://en.wikipedia.org/wiki/Geolocation_software. 3

[2] H.264 wiki page. http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC. 39, 40

[3] Maxmind GeoIP2 City. https://www.maxmind.com/en/city. 20

[4] Netlimiter homepage. http://www.netlimiter.com/. 31

[5] Olfeo protocol list. http://www.olfeo.com/sites/olfeo/files/english_site/pdf/protocol-list-olfeo.pdf. 43

[6] PPLive official website. http://www.pplive.com. 6, 22

[7] PPStream official website. http://www.pps.tvss. 6, 22

[8] Redis database. http://redis.io/. 20

[9] RSVP, Network Working Group. http://ftp.isi.edu/in-notes/rfc2205.txt. 43

[10] RTP, Network Working Group. https://www.ietf.org/rfc/rfc3550.txt. 43

[11] RTSP, Network Working Group. http://www.ietf.org/rfc/rfc2326.txt. 43

[12] Sopcast homepage. http://www.sopcast.com/. 46

[13] VC-1 wiki page. http://en.wikipedia.org/wiki/VC-1. 40

[14] Wizwig. http://www.wiziwig.tv/softwareitem.php?softwareid=28&part=software. 22

[15] WMV wiki page. http://en.wikipedia.org/wiki/Windows_Media_Video. 40

[16] Acestream guide. http://acestreamguide.com/, 2014. 22

[17] Popcorn time wiki page. http://en.wikipedia.org/wiki/Popcorn_Time, 2014. 22

[18] Shahzad Ali, Anket Mathur, and Hui Zhang. Measurement of commercial peer-to-peer live video streaming. Citeseer. 6, 8

[19] Delia Ciullo, Maria Antonieta Garcia, Akos Horvath, Emilio Leonardi, Marco Mellia, Dario Rossi, Miklos Telek, and Paolo Veglia. Network awareness of p2p live streaming applications: a measurement study. *Multimedia, IEEE Transactions on*, 12(1):54–63, 2010. 11, 30

[20] Delia Ciullo, Marco Mellia, Michela Meo, and Emilio Leonardi. Understanding p2p-tv systems through real measurements. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–6. IEEE, 2008. 8, 18

[21] Gregory J Conklin, Gary S Greenbaum, Karl Olav Lillevold, Alan F Lippman, and Yuriy A Reznik. Video coding for streaming media delivery on the internet. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(3):269–281, 2001. 40, 41

[22] Hrishikesh Deshpande, Mayank Bawa, and Hector Garcia-Molina. Streaming live media over a peer-to-peer network. *Technical Report*, 2001. 8

[23] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286. ACM, 2006. 2, 3, 8, 11, 12, 13

[24] Sebastian etc. Popcorn time github page. `https://github.com/popcorn-time/popcorn-app/`, 2014. 22

[25] Kevin R Fall and W Richard Stevens. *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011. 42

[26] Benedetto Fallica. Measurement study of the P2PTV application SopCast. Master's thesis, Delft University of Technology, 2007. 8

[27] A. Ganjam and Hui Zhang. Internet multicast video delivery. *Proceedings of the IEEE*, 93(1):159–170, Jan 2005. 6

[28] Jagannath Ghoshal, Lisong Xu, Byrav Ramamurthy, and Miao Wang. Network architectures for live peer-to-peer media streaming. 2007. 6, 7

[29] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju, and James Won-Ki Hong. The architecture of ng-mon: A passive network monitoring system for high-speed ip networks1. In *Management Technologies for E-Commerce and E-Business Applications*, pages 16–27. Springer, 2002. 8, 11

[30] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W Ross. A measurement study of a large-scale p2p iptv system. *Multimedia, IEEE Transactions on*, 9(8):1672–1687, 2007. 3, 8, 12, 22

[31] Xiaojun Hei, Yong Liu, and Keith W Ross. Iptv over p2p streaming networks: the mesh-pull approach. *Communications Magazine, IEEE*, 46(2):86–92, 2008. 6, 7, 44, 47, 48, 50

[32] Takayuki Hisada, Shusuke Yamazaki, Yusuke Hirota, Hideki Tode, and Koso Murakami. P2p live streaming system suitable for private contents distribution. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1–5. IEEE, 2010. 6

[33] Akos Horvath, Miklos Telek, Dario Rossi, Paolo Veglia, Delia Ciullo, Maria Antonieta Garcia, Emilio Leonardi, and Marco Mellia. Dissecting pplive, sopcast, tvants. *submitted to ACM Conext*, 2008. 30

[34] Piotr Indyk, Giridharan Iyengar, and Narayanan Shivakumar. Finding pirated video sequences on the internet. Technical report, Technical report, Computer science department, Stanford university, 1999. 1

[35] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, et al. Transport layer identification of p2p traffic. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134. ACM, 2004. 2

[36] Michael Karl, Tatiana Polishchuk, Thorsten Herfet, and Andrei Gurtov. Mediating multimedia traffic with strict delivery constraints. *2013 IEEE International Symposium on Multimedia*, 0:241–248, 2012. 1

[37] Mandar Kelaskar, Vincent Matossian, Preeti Mehra, Dennis Paul, and Manish Parashar. A study of discovery mechanisms for peer-to-peer applications. In *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, pages 444–444. IEEE, 2002. 7

[38] Myung-Sup Kim, Hun-Jeong Kong, Seong-Cheol Hong, Seung-Hwa Chung, and James W Hong. A flow-based method for abnormal network traffic detection. In *Network operations and management symposium, 2004. NOMS 2004. IEEE/IFIP*, volume 1, pages 599–612. IEEE, 2004. 11

[39] Franc Kozamernik. Media streaming over the internet–an overview of delivery technologies. *EBU Technical Review*, 10, 2002. 39, 40, 41, 42, 43

[40] Bo Li and Hao Yin. Peer-to-peer live video streaming on the internet: issues, existing approaches, and challenges [peer-to-peer multimedia streaming]. *Communications Magazine, IEEE*, 45(6):94–99, 2007. 44, 45

[41] Chunlei Liu. Multimedia over ip: Rsvp, rtp, rtcp, rtsp. *Handbook of Communication Technologies: The Florida*, 2000. 5, 39, 43

[42] Jiangchuan Liu, Sanjay G Rao, Bo Li, and Hui Zhang. Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of the IEEE*, 96(1):11–24, 2008. 6, 48

[43] Yong Liu, Yang Guo, and Chao Liang. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications*, 1(1):18–28, 2008. 6, 7, 44, 50

[44] Xiaosong Lou and Kai Hwang. Collusive piracy prevention in p2p content delivery networks. *Computers, IEEE Transactions on*, 58(7):970–983, 2009. 2, 6

[45] Bruce B Lowekamp. Combining active and passive network measurements to build scalable monitoring systems on the grid. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):19–26, 2003. 2

[46] Zhihui Lu, Ye Wang, and Yang Richard Yang. An analysis and comparison of cdn-p2p-hybrid content delivery system and model. *journal of communications*, 7(3):232–245, 2012. 44, 45

[47] Kevin J Ma, Radim Bartoš, and Swapnil Bhatia. A survey of schemes for internet-based video delivery. *Journal of Network and Computer Applications*, 34(5):1572–1586, 2011. 5, 39, 40, 41, 44

[48] Alok Madhukar and Carey Williamson. A longitudinal study of p2p traffic classification. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, pages 179–188. IEEE, 2006. 3, 12

[49] Nazanin Magharei, Reza Rejaie, and Yang Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1424–1432. IEEE, 2007. 6, 7, 48

[50] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. Flow clustering using machine learning techniques. In *Passive and Active Network Measurement*, pages 205–214. Springer, 2004. 2, 3, 8, 12, 13

[51] Venkat Mohan, YR Janardhan Reddy, and K Kalpana. Active and passive network measurements: a survey. *Int J Comput Sci Inf Technol, ISSN*, pages 0975–9646, 2011. 2, 30, 44

[52] Alexander Moshchuk, Tanya Bragin, Steven D Gribble, and Henry M Levy. A crawler-based study of spyware in the web. In *NDSS*, 2006. 1, 2

[53] Panagiotis Papadimitriou. Multimedia streaming over the internet. 2004. 40, 41, 42, 45

[54] Byungchul Park, JW-K Hong, and Young J Won. Toward fine-grained traffic classification. *Communications Magazine, IEEE*, 49(7):104–111, 2011. 9

[55] Andrea Passarella. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1–32, 2012. 44, 45

[56] D. Ratkaj. Distribution of multimedia services over the internet. In *ELMAR, 2013 55th International Symposium*, pages 1–5, Sept 2013. 2

[57] Reza Rejaie and Shad Stafford. A framework for architecting peer-to-peer receiver-driven overlays. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, pages 42–47. ACM, 2004. 7

[58] Janko Roettgers. Dont touch that dial: How youtube is bringing adaptive streaming to mobile, tvs. `http://gigaom.com/2013/03/13/youtube-adaptive-streaming-mobile-tv/`, March 2013. 41

[59] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*, pages 512–521. ACM, 2004. 3, 12

[60] P. Shah and J.-F. Paris. Peer-to-peer multimedia streaming using bittorrent. In *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pages 340–347, April 2007. 7

[61] Thomas Silverston and Olivier Fourmaux. Measuring p2p iptv systems. In *Proceedings of NOSSDAV*, volume 7, 2007. 2, 8, 12, 19, 22, 30

[62] Salvatore Spoto, Rossano Gaeta, Marco Grangetto, and Matteo Sereno. Analysis of pplive through active and passive measurements. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–7. IEEE, 2009. 2, 8

[63] Thomas Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011. 41, 42, 43, 44

[64] Yun Tang, J Luo, Qian Zhang, Meng Zhang, and S Yang. Deploying p2p networks for large-scale live video-streaming service. *IEEE Communications Magazine*, 45(6):100, 2007. 2, 13

[65] Duc A Tran, Kien A Hua, and Tai Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1283–1292. IEEE. 48, 50

[66] Andrew Turner. Geolocation by ip address. *Linux Journal, Oct*, 25, 2004. 3

[67] Dapeng Wu, Y.T. Hou, Wenwu Zhu, Ya-Qin Zhang, and J.M. Peha. Streaming video over the internet: approaches and directions. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(3):282–300, Mar 2001. 5, 40, 41, 42

[68] Gang Xiong et al. Research progress and prospects of network traffic classification. *integration technology*, 1(1), 2012. 3, 11, 12, 44

[69] Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai. Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11(4):383–399, 2006. 44, 45

[70] W-PK Yiu, Xing Jin, and S-HG Chan. Challenges and approaches in large-scale p2p media streaming. *MultiMedia, IEEE*, 14(2):50–59, 2007. 6, 7, 48, 50

[71] Alex Zambelli. Iis smooth streaming technical overview. *Microsoft Corporation*, 3, 2009. 40, 41, 42

[72] Sebastian Zander, Thuy Nguyen, and Grenville Armitage. Automated traffic classification and application identification using machine learning. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 250–257. IEEE, 2005. 8, 11

[73] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 2102–2111. IEEE, 2005. 6

Appendix A

Internet multimedia delivery technology characteristics

A.1 Service type

The multimedia services can be classified into two groups: live streaming and video-on-demand(VoD) [47, 41].

- Live streaming: Live streaming refers to content delivered live over the Internet. The video data to be served should be acquired in real-time. An application program called a broadcaster is needed. The broadcaster takes input from live sources, such as a video camera, microphone or an ordinary audio CD, compress these sources on the fly for transmission at the desired data rate, and creates a real-time audio-video stream [39].
- VoD: Video on demand (VOD) services allow users to select and watch/listen to video or audio content when they choose to, rather than having to watch at a specific broadcast time. In VoD service, video file is typically pre-recorded and available via persistent storage [39]. Users have the flexibility to watch any video at any moment in time, meaning that they do not need to synchronize their playback times. Moreover, they are capable to perform operations such as forward or backward on the video file.
- Straight download: Straight download typically is considered, for historical reasons [47]. The user can play the downloaded file only after the whole file has been downloaded from a server to his/her computer. The full file transfer, in the download mode, can often suffer unacceptably long transfer times, which depend on the size of the media file and the bandwidth of the transport channel [39].

A.2 Content formats

Typical video compression encoding schemes are introduced in the following.

- H.262/ MPEG-2: H.262 is a standard for the generic coding of moving pictures and associated audio information. It describes a combination of lossy video compression and lossy audio data compression methods, which permit storage and transmission of movies using currently available storage media and transmission bandwidth. While MPEG-2 is not as efficient as newer standards such as H.264, backwards compatibility with existing hardware and software means it is still widely used, for example in the DVD-Video standard [2].
- H.264 / MPEG-4 AVC: H.264/MPEG-4 Part 10 or AVC (Advanced Video Coding) is a video compression format that is currently one of the most commonly used formats for the

recording, compression, and distribution of video content [2]. The final drafting work on the first version of the standard was completed in May 2003, and various extensions of its capabilities have been added in subsequent editions. H.264 is likely becoming the most common codec used: Adobe supports it in Flash, HTML5 canvas can use H.264, YouTube is steadily moving to H.264, and Apple fully supports it [2].

- VC-1: SMPTE 421M, informally known as VC-1, which was initially developed as a proprietary video format by Microsoft. It is today a supported standard found in Blu-ray Discs, Windows Media, Microsoft's Silverlight framework, Slingbox, and the now-discontinued HD DVD [13].
- Windows Media Video(WMV): Windows Media (WMV) is Microsoft's family of proprietary video codec designs including WMV 7, WMV 8, and WMV 9. [15].

Popular video container formats are introduced in the following.

- MPEG-2 Transport Streams (MPEG-2 TS/PS): In the MPEG-2 standard, there are two types of audio video synchronization, TS and PS. TS is Transport Stream which are aimed for communication and broadcasting applications. PS is Program Stream which can be used for storage applications such as DVD. Therefore, it is recommended to select each type according to application and playback specifications. Supported codecs includes: MPEG-2, h.264 and VC-1.
- Advanced Systems Format (ASF): ASF is a Microsoft-based container format. There various file extensions for ASF files, including .asf, .wma, and .wmv. Note that a file with a .wmv extension is probably compressed with Microsoft's WMV (Windows Media Video) codec, but the file itself is an ASF container file. ASF files can, in theory, contain video and audio files compressed with any codec. In practice, it's playback that can become a problem, particularly with video compressed with H.264 codecs.
- Audio Video Interleave (AVI): Audio Video Interleave is a multimedia container format introduced by Microsoft in November 1992 as part of its Video for Windows software.
- QuickTime: QuickTime is Apple's own container format. QuickTime sometimes gets criticized because codec support (both audio and video) is limited to whatever Apple supports. This is true, but QuickTime supports a large array of codecs for audio and video. Apple is a strong proponent of H.264, so QuickTime files can contain H.264-encoded video.

A.3 Delivery method

According to the studies of [67, 21, 47], there are two modes for transmission of stored video over the Internet, namely file-downloading and streaming.

- File-downloading: In the file-downloading mode, a user downloads the entire video file and then plays back the video file. However, full file transfer in the download mode usually suffers long and perhaps unacceptable transfer time [53, 71]. File downloading is not a satisfactory solution since it costs disk space and takes time [21, 47].
- Streaming: Streaming video is content sent in compressed form over the Internet and displayed by the viewer in real time. In steaming media service, the video content is not distributed as a simple file, but it is delivered as a continuous ordered flow of data. On the user's side, a client application, known as streaming player, can start playing back streaming media as soon as enough data has been received without having to wait for the entire file to have arrived [39]. As data is transferred, it is temporarily stored in a buffer until enough data has been accumulated to be properly assembled into the next sequence of the media stream [21]. Due to its real-time nature, video streaming typically has bandwidth, delay and

loss requirements [21]. According to [67], the main advantage of streaming is that content of any length, even live content of unlimited length, can be played by the end user; the main disadvantage of streaming is that the playback quality depends on the network bandwidth. Poor network conditions and bandwidth fluctuations easily result in annoying disruptions to the end user.

From [21, 47] we know that there are subcategories of streaming technologies, which are traditional streaming, HTTP progressive download, and adaptive streaming.

- Traditional streaming: In traditional streaming, a streaming server provides streaming services and runs along with a Web server [53]. The streaming server, running in the background, operates similarly to a Web server, whereas it handles and oversees the distribution and access of the multimedia content. The streaming server normally waits for content request from the viewers. When it gets a request, the server looks in the appropriate folder for a hinted media of the requested name. If the requested media is located, the server streams it to the viewer, e.g. using RTP streams.

  Streaming server is the most important single element for providing quality streaming services [39]. Traditional streaming generally uses a stateful protocol, e.g., the Real-Time Streaming Protocol (RTSP). Once a client connects to the streaming server, the server keeps track of the client's state until the client disconnects again. After a session between the client and the server has been established, the server sends the media as a continuous stream of packets. In contrast, Web server runs HTTP, which is a stateless protocol [63]. If an HTTP client requests some data, the server responds by sending the data and the transaction is terminated. Each HTTP request is handled as a completely standalone one-time transaction [63]. From above discussion we know the difference with streaming server and Web server, which implies that traditional streaming cannot utilize wide-spread Web resource [39].
- HTTP progressive download: HTTP progressive download or pseudo streaming, is another common form of media delivery on the Internet today [71, 39]. It has tremendous popularity in industry. For example, Youtube has wide utilized this technology [58]. This technology is nothing more than a simple file download from an HTTP Web server [39]. The term 'progressive' stems from the fact that most player clients allow the media file to be played back while the download is still in progress before the entire file has been fully written to disk (typically to the Web browser cache) [71]. Clients that support the HTTP 1.1 specification can also seek to positions in the media file that haven't been downloaded yet by performing byte range requests to the Web server.

  The major advantage of pseudo streaming is that it can utilize existing Web infrastructures: Web server and CDN [71], which means it is cost-effective. Also, it can cross the firewalls and NAT without redundant efforts. Media protocols often have difficulty getting around firewalls and routers because they are commonly based on UDP sockets over unusual port numbers [39]. HTTP-based media delivery has no such problems because firewalls and routers know to pass HTTP downloads through port 80.

  Disadvantages of progressive download are mostly that: bandwidth may be wasted if the user decides to stop watching the content after progressive download has started, e.g., switching to another content; it is not really bitrate adaptive; it does not support live media services [71]. The key difference between streaming media and progressive download is in how the digital media data is received and stored by the end user device that is accessing the digital media [39]. In pseudo streaming, video content is cached on user's hard drive, thus it's very easy to copy and redistribute. In comparison, streaming video can be cache-less, which makes it inherently more secure [39].
- HTTP-based Adaptive streaming: Adaptive streaming is a hybrid delivery method that acts like streaming but is based on HTTP progressive download [71]. It's an advanced concept that uses HTTP rather than a new protocol [71]. In adaptive streaming, the raw video is compressed into different version of bitrate, and each of them is segmented

into small chunks (e.g. 3 seconds). Each chunk can be decoded independently of other chunks. A metadata file describes the relation of the chunks and how they form a media presentation(e.g. a TV program). The client uses the meatadata file to fetch the video content from origin Web server and can choose between chunks of different bitrate versions depending on estimated bandwidth [71].

Adaptive streaming addresses the weaknesses of traditional streaming and progressive download [63]. Firstly, adaptive streaming supports both on-demand and live streaming service [71]. Secondly, it's cheaper to deploy because adaptive streaming can use generic HTTP caches/proxies and doesn't require specialized servers at each node.

The disadvantages of adaptive streaming include: it requires client-side support. e.g. Adobe Flash Player only supports HTTP Dynamic Streaming after 10.1) [63] and different implementations are not compatible completely; adaptive streaming has longer delay compared with traditional streaming technologies (e.g. RTP, RTMP); adaptive streaming quality adaption is not as fast as traditional streaming technologies as the quality switch usually only occurs at a time boundaries [71].

A.4 Network protocols

Protocols are designed and standardized for communication between clients and streaming servers [67]. In the following two subsections, we discuss transport-layer protocol and application-layer protocol [25] used by Internet multimedia delivery.

A.4.1 Transport protocols

The basis of Internet, TCP/IP and UDP/IP, provides a range of services that multimedia applications can use [53].

- TCP: As stated in [53], Transmission Control Protocol (TCP) is the dominant protocol for data transmission over the Internet. However, because it constantly alters the rate of the transmitted data, it is not preferable for multimedia streaming applications. As a result, many TCP-based protocol extensions have emerged to overcome the standard TCP limitations concerning efficient multimedia streaming.

- UDP: Alternatively, some streaming implementations are based on the User Datagram Protocol (UDP). UDP is a fast, lightweight protocol without any transmission or retransmission control. Retransmission of packets adds delays and uses up bandwidth in the data channel. If network transmission errors are high, the receive buffer in the media player is emptied and the stream is interrupted. Therefore, the strategy for receiving the streams is to ignore lost packets. The User Datagram Protocol (UDP) does precisely that. The loss of packets may cause impairments to the subjective quality of the received stream or even the loss of several video frames [39]. Nevertheless, media players are often designed to conceal these errors. Consequently, UDP appears to be more suitable for applications, such as multimedia implementations, that tolerate some packet losses. However, the lack of a congestion control mechanism is a significant shortfall for UDP [39].

A.4.2 Application protocols

- HTTP: Hypertext Transfer Protocol (HTTP) is an application-level file transfer protocols [39]. HTTP is known as a protocol that effectively carries HTML pages and allows the hyperlinks to transfer the user to another document or Web site. The server and the client computers have a two-way connection, meaning that there is feedback from the client (receiver) computer. Thus, lost or damaged packets can always be retransmitted, so that the received file can be fully restored. HTTP can also be used for media download [63], especially if the files are small and the number of concurrent users is limited. If the connection

speed is lower than the media data rate, the media still gets through but it may not play smoothly. The transfer time of file download depends on the size of the file and the speed of the connection.

Study in [63] gives reasons that HTTP has advantage as the delivery protocol for streaming services. The main reasons includes: HTTP-based delivery enables easy and effortless streaming services by avoiding NAT and firewall traversal issues; HTTP-based delivery provides reliability and deployment simplicity due as HTTP and the underlying TCP/IP protocol are widely implemented and deployed; HTTP-based delivery provides the ability to use standard HTTP servers and standard HTTP caches.

- RTP: Realtime transport protocol (RTP) is an UDP-based protocol providing support for the transport of real-time data such as video and audio streams [63, 10]. The services provided by RTP include time reconstruction, loss detection, security and content identification. RTP is designed to work in conjunction with the auxiliary control protocol RTCP to get feedback on quality of data transmission and information about participants in the on-going session. As discussed in the first section, Internet is a shared datagram network.

- RTSP: Real Time Streaming Protocol (RTSP) is a client-server multimedia presentation protocol to enable controlled delivery of streamed multimedia data over IP network [11]. It provides VCR-style remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning. Sources of data include both live data feeds and stored clips. RTSP is an application-level protocol designed to work with lower-level protocols like RTP, RSVP to provide a complete streaming service over Internet. It provides means for choosing delivery channels (such as UDP, multicast UDP and TCP), and delivery mechanisms based upon RTP. It works for large audience multicast as well as single-viewer unicast.

- RTCP: RTP Control Protocol (RTCP) is used in conjunction with RTP and uses TCP for bi-directional client-server connection. It provides feedback to the service provider on the network reception quality from each participant in an RTP session [39]. The messages include reports on the number of packets lost and the jitter statistics (early or late arrivals). This information can be used by higher-level applications to control the session and improve the transmission; for example, the bit-rate of a stream could be changed to combat network congestion.

- RSVP: Resource Reservation Protocol (RSVP) is the network control protocol that allows data receiver to request a special end-to-end quality of service for its data flows [9]. Real-time applications use RSVP to reserve necessary resources at routers along the transmission paths so that the requested bandwidth can be available when the transmission actually takes place. RSVP is a main component of the future Integrated Services Internet which can provide both best-effort and real-time service. RSVP is used to set up reservations for network resources. When an application in a host (the data stream receiver) requests a specific quality of service (QoS) for its data stream, it uses RSVP to deliver its request to routers along the data stream paths. RSVP is responsible for the negotiation of connection parameters with these routers. If the reservation is setup, RSVP is also responsible for maintaining router and host states to provide the requested service [41].

- RTMP: Real Time Messaging Protocol (RTMP) is developed by Macromedia, which supports for live audio and video streaming. RTMP supports a non-encrypted version over the TCP or an encrypted version over a secure SSL connection. RTMP can also be encapsulated within HTTP requests to traverse firewalls that only allow HTTP traffics.

- Proprietary protocol Despite those standard protocol discussed above, proprietary protocols are also widely used, as shown in [5]. A proprietary protocol is a communications protocol owned by a single organization or individual. A proprietary protocol does not comply to any open standards. The intent of a proprietary protocol is to limit communication only to

nodes that implement a specific application. Proprietary protocols are wide used in Internet video delivery systems. Typically, proprietary protocol is encrypted and no specification is available [68]. An attacker is most likely to need time and resources (usual exploiting tools will not work as an example) to understand how the proprietary protocol is designed and implemented before exploiting any weaknesses. Typically, reverse engineering is the process of retrieving proprietary protocol's details. Methods of reverse-engineering a protocol include packet sniffing and binary decompilation and disassembly [51].

A.5 Network architecture

The network architecture is the most important characteristic to identify a content delivery system [40], which deals with how media content is distributed from the source to destinations. Currently, CDN, P2P and P2P/CDN hybrid are popular architectures that used to distribute multimedia content [47]. CDN schemes distribute data to the network edge so that data does not need to cross the network core reducing congestion in the core. P2P schemes push network load away from centralized data centers toward localized communities at the network edge to not only limit congestion in the core, but to also limit requests serviced by the origin data center [47]. The benefits of each of the schemes described above are not mutually exclusive, P2P/CDN hybrid architecture can lower the cost of CDN capacity reservation without compromising the media quality delivered [69].

In following, we introduce CDN, IP multicast, P2P and P2P/CDN hybrid scheme in details.

- CDN: Content Distribution Networks (CDN) have successfully been used to serve Web pages, offloading origin servers and reducing download latency [63]. Such systems generally consist of a distributed set of caching Web proxies and a set of request redirectors located strategically across the wide-area Internet. In a CDN architecture, a media file is first pushed to multiple CDN servers, each of which serves clients in its designated domain(s). A CDN server has dedicated storage space and out-bound bandwidth for high-quality media streaming.

  Given the scale, coverage, and reliability of CDN systems, it is appealing to use them as base to launch streaming services that build on this existing infrastructure [63]. This can reduce capital and operational expenses, and reduces or eliminates decisions about resource provisioning on the nodes. CDNs are tightly integrated into the existing Web architecture, relying either on DNS interposition or on URL rewriting at origin servers to redirect HTTP requests to the nearest CDN replica [63].

  However, CDN servers are expensive to deploy and maintain [69, 46]. The server capacity (including processing power and out-bound bandwidth) that can be allocated to the distribution of one media file is limited, and it incurs a non-trivial cost to the provider and/or clients of this media file.

- IP multicast: IP multicast was demonstrated as a promising technique that can significantly reduce the duplication of data transmissions [40]. However, IP multicast encountered the following problems in the deployment: (i) Scalability in that there are potentially a large number of multi-cast groups that must be managed in a large network. (ii) A requirement for coordination of dynamic spanning tree(s) construction at routers across different autonomous subnets, which often makes it practically infeasible. (iii) Routers must maintain the state, which violates the stateless principles and creates difficulty in the design of high-level functions.

- Peer-to-Peer network: P2P netowrk is formed by hosts (peers) that equally share the burden of providing services to each other in a cooperative fashion [55]. Each peer has equivalent capabilities and responsibilities in a P2P network. All peers provide resources, such as bandwidth, storage spaces and computing power, to the whole network. Thus, with the increasing number of peers, the total capacity of the system increases [55, 43, 31].

An important property of P2P network is its overlay structure. P2P networks can be categorized by overlay structure into 3 classes: structured, unstructured and hybrid [55]. Structured overlays means a virtual address space if defined and each peer is assigned to a address. The virtual space is organized according to a given topology. In a structured overlay, the route between any two peers are determined. The most common usage of such overlay is to locate objects. Core functions of structured overlays are address assignment, join and leave operations, structure maintenance, routing and forwarding [53]. Unstructured overlay does not enforce any structure in the network. There is no neighbourhood relationship among peers and any two peers can communicate without determined routing [53].

In recent years, P2P technology has captured the interest of the research community as well as the industry [55]. By allowing peers to serve each other, P2P solutions overcome many limitations of traditional client-server architectures. They can handle flash crowds(that is, very large and sudden surges of demand) as well as achieve bandwidth scalability(that is, overcome the bandwidth limitations of the server). In addition, P2P solutions do not require any special support from the network. But for P2P technology, the legal aspects related to the distribution of copy-righted content is clearly an issue. P2P technologies are largely used to illegally exchange content due to its cost-effectiveness [40].

- P2P/CDN Hybrid: P2P and CDN are complementary solutions [55]. CDN can provide robust services with guaranteed performance, while the cost of CDN service is non-negligible for users. P2P solutions can seldom provide guaranteed services, but it can improve the scalability of the architecture by providing cheap and flexible delivery of content with statistical guarantees. Many works in both industrial and academic area have been done as stated in [46, 69]. The details in our of the scope of our project, we are only interested in their basic principle: CDN and P2P technologies are complementary, which makes their combination possible.

A.6 Client software composition

In this section, we discuss end-user client software composition. Based on our study of popular Internet video delivery service including Youtube, Netflix, PPlive, Sopcast, Acestream and Popcorn time, we generalize the function of a client software into three parts: service discovery, video content reception and video playout.

Service discovery deals with finding multimedia content. From our observation web portal and client-embedded are two common means to discover content. Web portal service discovery is not part of the client software. Users can browse the content on the web portal. The content typically are listed as an URI that is an acceptable parameter to the client software(e.g. magnet URI, Acestream URI), or a metafile contains resource information(e.g. torrent file). Examples includes BitTorrent clients, Sopcast, Acestream. Client-embedded service discovery means user can find content through client software, and it is not exclusive of web portal service discovery. E.g., PPlive client has embedded service discovery function, Sopcast supports both manners.

Content transmission deals with reception and buffering of the multimedia content, which is the responsibility of the reception engine. The received video content is cached in local storage, either in memory (e.g. traditional streaming) or in disk (file-downloading and progressive download). reception engine is the most important component of a end-user client because it is responsible for content locating and reception.

Video playout deals with viewing the received content, which is dealt by the media player. The player can be integrated with the engine, or be an external program that can be ported to the engine. An example of the later case is the browser plug-in of Acestream and Sopcast.

Based on discussed components above, we introduce different types of end-user client software composition.

- Integrated engine and player: In this case, the engine and media player are not separated programs. Typically implementation types include flash player used by many web video

service providers(e.g. Youtube), and desktop program(e.g. PPlive desktop client, Popcorn time client).

- Separated engine and player: In this case, the engine and player are separated programs. The received data of the engine is transmitted to the player through a dedicated interface(e.g Peerflix uses HTTP streaming interface). Typically, use has the flexibility to select an external player.

Some systems support both integrated and separated player. For example, Sopcast uses Window Media player as default and it also supports external player [12].

Appendix B

P2P streaming system architecture

B.1 Mesh-based architecture

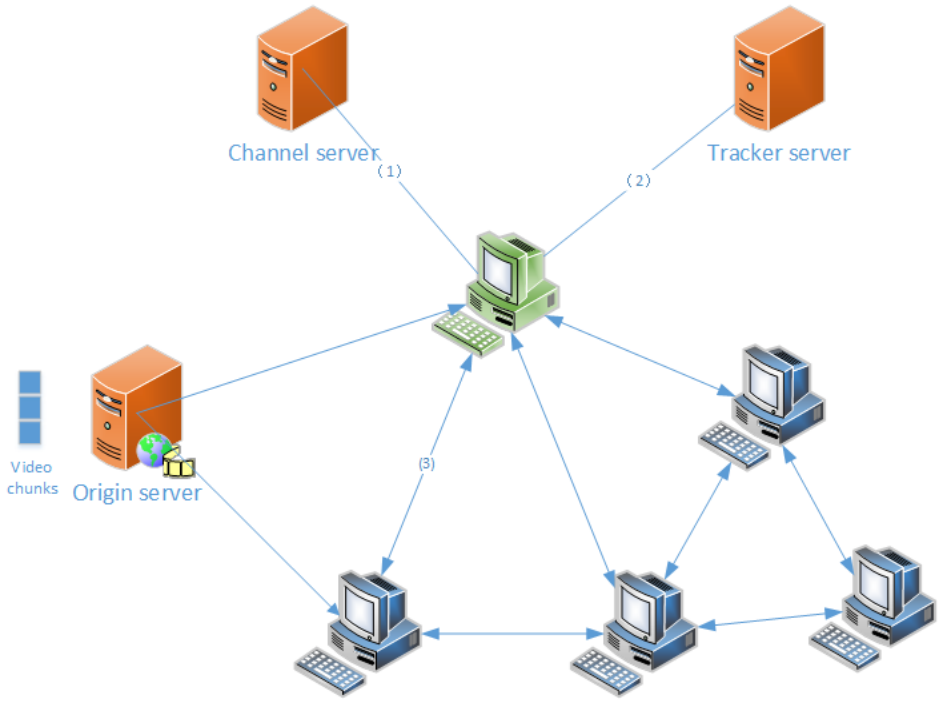

Figure B.1: Mesh-pull P2P streaming architecture

We give an overview of mesh-pull architecture in the following, which is referred from [31]. As shown in figure B.1, in a mesh-pull P2P architecture a video content is divided into small segments and is broadcast from origin peer to other peers. Information of channels(or events) is available from a channel server. Tracker server provides information of all peers downloading/uploading a particular content. After a peer selected interested channel(step (1)), it gets the list of peers who are also watching the same content(step (2)). The peer then establishes data transport connections with other peers in this list. Peers have established connections are called neighbours. Neighbour peers exchange messages to know each other's status of content owning. Then each peer requests

its interested chunks from its neighbours.

In step(2), the local peer's overlay is constructed. The mesh-based overlay construction approach is inspired by file swarming mechanisms (such as BitTorrent) where participating peers form a randomly connected mesh and employ a swarming content delivery mechanism over a recent time window [49]. Each peer periodically fetches peer list from the tracker and establishes relationship with subset of peers from the peer list. Note that for a peer in mesh-based overlay, it's neighbours are not static, a peer keeps looking for peers that holds the content and it selects peers based on its peer selection algorithm [31]. Different systems vary in their own implementation of peer selection algorithm. Mostly, peers with larger uploading capacity are preferred [49].

In step (3), neighbour peers exchange data chunks. If two peers have established neighbour relationship, they exchange buffer map message with each other. In this way, the content availability information can be exchanged. Based on the content availability information, a peer can request specific chunks from another neighbour peer or respond to a neighbour peer's chunk request. In this process, the packet scheduling algorithm is a main design issue. The packet scheduling algorithm determines the order of packet to download. As stated in [49], an efficient packet scheduling algorithm should achieve following goals:(i) effectively utilizing the available bandwidth from all parent peers, (ii) pulling a proper number of descriptions(i.e., desired quality) from all parent peers and (iii) ensuring in-time delivery of requested packets.

The major advantage of mesh-based architecture is fault-tolerance [31, 49]. In mesh-based architecture, each peer has a dynamic neighbour relationship with multiple peers. This gives advantage that a peer's media quality won't degrade a lot if one of its neighbour peer leaves. The service quality of mesh-pull P2P system is relying on the contributions of all participating peers' upload contribution, which means if a peer cannot find a neighbour peer with desired contend and large upload capacity, it is difficult to have good playback performance. However, the peers in a mesh-pull swarm are very heterogeneous in terms of upload bandwidth. Super peers or injection nodes may have upload capacity that is 10 times bigger than of a "normal" peer. These factors make the video distribution performance of mesh-based architecture unpredictable [31]. Another disadvantage of a mesh overlay is that the delay performance. The amount of control overhead is the main reason for delays in mesh-pull system [31, 42].

B.2 Tree-based architecture

The design of tree-based P2P streaming networks was initially motivated by IP multicast [31, 70]. For IP multicast, a multicast tree is built at the IP layer to deliver packets from a source node to all of the destination nodes. Similarly, tree-based P2P streaming networks construct an logical multicast tree on application layer instead of network layer, in which the media stream is pushed from the source to other peers. When a peer joins the tree-based P2P streaming network, it first establishes its overlay. The tree construction algorithm then determines its position in the overlay tree. In a tree overlay, each node is placed as an internal node or leaf node. Then the peer download video content from its parent node, and uploads to its children nodes. Similar with the last section, we focus on step (2) and (3) in figure B.2.

For tree-overlay construction, the end-to-end delay from the source peer to a receiver peer is a critic metric in media streaming system [31]. In tree-based architecture, the longest delay in a multicast tree depends on the height of the tree [65]. As stated in [65], the major design goal of a tree-based architecture is to shorten the end-to-end delay by reducing overall height of the multicast tree. Different tree-based architecture implements their own tree-construction algorithm. ZIGZAG [65], for example, organizes the peers into a hierarchy of clusters. Their algorithm guarantees that the height of the tree built is $O(\log_k N)$ where N is the total number of peers and k is a constant. According to [70], the essential influencing factor of tree-push architecture's overlay construction is the peer's upload bandwidth. If a peer has low upload bandwidth, it is more likely to be put at the bottom of the tree overlay; in contrast, mostly internal node has higher upload capacity that can serve multiple children nodes. After the overlay tree is constructed, the peer receives media stream from its parent, until its parent departures. If an internal node

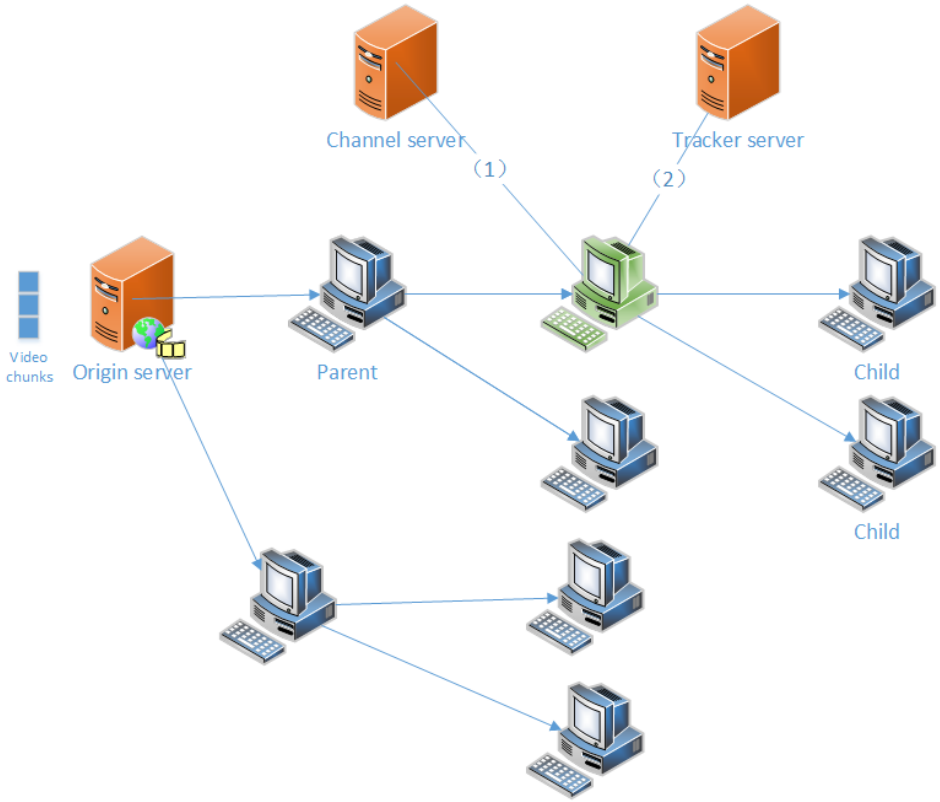

Figure B.2: Tree-push P2P streaming architecture

departures, the tree overlay has to be reconstructed because some nodes lost their data source [70].

The peer communication in tree-based architecture is relatively simple. After tree structure is formed, chunks are forwarded from the root of the tree to different levels of the tree hierarchy. The tree construction process is costive [31], but for peers in a constructed tree overlay, the communication overheads are very low [43]. Thus, the delay performance of tree-based system is, in general, good.

Tree-based P2P streaming system has a major drawback: it is vulnerable to peer churn. Peer churn is a common phenomenon in P2P network, which means peers randomly joins or leaves the P2P network [43]. As we discussed, when an internal node leaves, the tree overlay needs to be reconstructed, which has a heavy cost [65]. Another disadvantage is inefficiency. As we can see from figure B.2, the leaf node in a tree does not contribute their upload capacities.

Appendix C

Use case diagram of monitoring system

The use case digram of our system is shown in figure C.1.

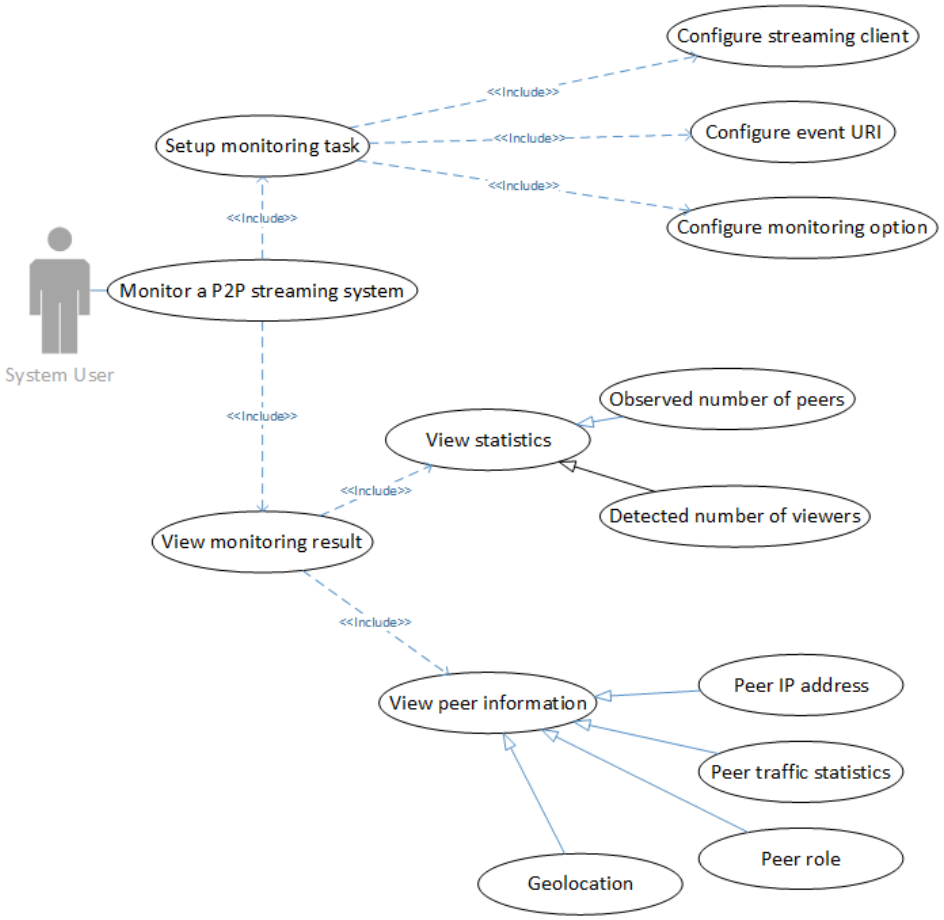

Figure C.1: System use case diagram

- Monitor a P2P streaming system: User can monitor an event of an P2P streaming system.
- Setup: The user sets up configuration of the monitoring task, and monitoring system.

  - Configure streaming client: User configures the streaming client by indicating the name of the target system in the configuration file. In addition, on each monitor host the user needs to fill in a configuration file containing the paths to the installed clients.
  - Configure event URI: User indicates the event URI. This parameter is only useful when the P2P client can accept the URI as a parameter to start.
  - Configure monitoring option: User indicates the monitoring interval, logging option and storage option.

- View monitoring results: The user views the result of the monitoring task.
  - View statistics: The user views statistics including:
    * Observed number of peers
    * Detected number of viewers
    * Country table (each entry has a number of peers and viewers)

    From these statistics, the user can know the popularity of the monitored event.
  - View peer information: The user views monitored peer's information, including:
    * IP address
    * Role (viewer or a service node)
    * Geographical location
    * Traffic statistics

To conclude, in our monitoring system, the user can monitor a channel of an arbitrary P2P streaming service. From the monitoring result, the user can have information of geographical distribution of peers and peer's role.

Appendix D

Configuration specification

We use two examples to illustrate the configuration of our system.

The example below is a configuration for the manager, which is used by the operator to configure monitoring tasks.

```
{
  "task": [// a list of tasks, all task are managed by the manager.
    {
    "monitor": ["10.56.62.161,31001","10.56.62.162,31001"],
    //This task has two monitors, "10.56.62.161" is the IP of a monitor node;
    //"31001"is the port number, you cannot specify other port number.
    "redisIP": "10.56.62.161",// IP address of the host running redis-server.
    "redisPort": 6379,// Default port number of redis-server, if you run
       // redis-server with specified port number, please also
       // change the port number here.
    "masterIP": "10.56.62.161",// IP address of the host running master
    "masterPort": 31003,// Port number of the master process, cannot be changed.
    "managerIP": "10.56.62.161",// IP address of the host running master
    "managerPort": 31004,// Port number of the master process, cannot be changed.
    "record": false,//if true, flow data will be dumped to the storage host, you
         //can perform analysis
    "monitorInterval":10,//interval of seconds for monitor to report to master,
    "packetLogging":false,//if true, monitor record the header of each packet.
    "firewall": false,//if true, monitor use firewall
    "filtering": false,//if true, choking is enabled
    "partitioning": false,//if true, partitioning is enabled
    "uri" : "6183e6bc46fd44fc5c692c73ccde965175b4838a",
    // target channel's acestream content ID,
    // it should be an acceptable parameter to
    // the target P2P streaming application.
    "client": "ace_player",
    //name of client, should be consistent with the name in exe_config.json
    "engine": "ace_engine"
    // name of engine, should be consistent with the name in exe_config.json

    //for application with separated engine(e.g. acestream and peerflix),
    //you need to have both keys
    //for application without seperated engine(e.g. sopcast), you don't need
    //"engine" key
    //example of peerflix
```

```
    //      "client": "peerflix",
    //      "engine": "node"
    //example of sopcast
    //      "client": "sopcast",
    }
  ]
}
```

The local configuration of each monitor node contains information of installed client's path. The key name of each entry should be consistent with the name in the last configuration. The value of each key is the path to install application's component. If the target application has separated client component and engine component, please input the path of two parts separately. For example, *peerflix* is a client running on *node* engine, thus we have two keys of both. In another case, if the target application does not have a separated engine, input the path of the only application just as *SopCast*.

```
{
    "exeTable": {
    "ace_player": "C:\\ACEStream\\player\\ace_player.exe",
    "ace_engine": "C:\\ACEStream\\engine\\ace_engine.exe",
    "SopCast": "D:\\Program Files (x86)\\SopCast\\SopCast.exe",
    "peerflix": "C:\\Users\\Zijian\\AppData\\Roaming\\npm\\peerflix.cmd",
    "node": "C:\\Program Files (x86)\\nodejs\\node.exe"
  }
}
```