Using BitTorrent for Measuring End-To-End Internet Path Characteristics

TOMAS ISDAL

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2006

Using BitTorrent for Measuring End-To-End Internet Path Characteristics

T O M A S I S D A L

Master's Thesis in Computer Science (20 credits)
at the School of Computer Science and Engineering
Royal Institute of Technology year 2006
Supervisor at CSC was Olof Hagsand
Examiner was Stefan Arnborg

TRITA-CSC-E 2006:148
ISRN-KTH/CSC/E--06/148--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Using BitTorrent for measuring end-to-end Internet path characteristics

Abstract

This report presents *BitProbes*, a system utilizing participants in peer-to-peer systems as targets for measuring the link characteristics of Internet end hosts. By connecting measurement nodes to popular swarms of the widely used peer-to-peer system *BitTorrent*, it is showed that properties of BitTorrent make users of it well suited as targets for Internet measurements. The design and implementation of a system for large scale end host measurement is presented. The system is capable of measuring the upload capacity and link latency of end hosts as well as recording the path to the host. The report also includes the design of a probing tool that can measure the download capacity of end hosts in what looks like a BitTorrent application level handshake. This tool exploits TCP-packet reordering to disable delayed acknowledgements on remote hosts. An early version of BitProbes is then evaluated on *PlanetLab* to verify both potential coverage and usability of results. A later version is currently running at 8 dedicated machines in a server cluster at the University of Washington, covering roughly half a million end-hosts per week. The measurements are stored in a database accessible by the *iPlane*, a system developed at the University of Washington that provides predictions about Internet path performance. With the addition of end-host measurements, the iPlane is able to predict the properties of the link between two arbitrary end hosts.

Mäta egenskaper hos Internet-anslutningar genom att utnyttja BitTorrent

Sammanfattning

I denna rapport presenterar jag *BitProbes*, ett system som begagnar användare av det populära peer-to-peer programmet *BitTorrent* som mål för mätning av egenskaper hos slutanvändares Internetanslutning. Genom att ansluta speciella mätnoder till populära BitTorrent svärmar visar jag att egenskaper hos just BitTorrent gör användare av det väl lämpade att använda som mål för mätningar. Jag presenterar design och implementation av ett system byggt för storskalig mätning av slutanvändares anslutning. Systemet är kapabelt att mäta bandbreddskapaciteten uppströms och länkens latens hos slutanvändaren så väl som att notera vilken Internet väg som använts. Jag presenterar också designen av ett verktyg för att mäta bandbreddskapaciteten nedströms i vad som för slutanvändaren ser ut som en vanlig handsskakning i BitTorrent protokollet. Detta verktyg skickar TCP-paket i ändrad ordning för att inaktivera TCPs ”delayed acknowledgement”. En tidig version av systemet utvärderades på *PlanetLab* för att verifiera om det är möjligt att få god täckning så väl som användbara data. En senare version körs nu på 8 dedikerade maskiner i ett serverkluster vid University of Washington, och ansluter till ungefär en halv miljon slutanvändare per vecka. Mätresultaten sparas i en databas som är tillgänglig för iPlane, ett system utvecklat på University of Washington för att förutspå egenskapen hos Internet vägen mellan två godtyckliga noder på Internet.

Table of contents

1 Introduction

Detailed estimates of Internet path properties allow applications to make better decisions. Content distribution networks (CDNs) such as Coral (Freedman, Freudenthal et al. 2004), CoDeeN (Wang, Park et al. 2004) and Akamai (Akamai 2006) have the ability to use Internet topology information to redirect clients to the node which provides the best performance. Peer to peer services such as Skype (Skype 2006) and BitTorrent (Cohen 2003) could use knowledge of both the core and the edge to be able optimize peer selection to improve end-user performance (Madhyastha, Isdal et al. 2006). Overlay services such as RON (Andersen, Balakrishnan et al. 2001) can optimize routes based on metrics such as loss rate, latency, or bandwidth capacity to allow for applications to select routes based on specific needs.

The *iPlane* is a scalable service that provides predictions about Internet path performance. To be able to make accurate predictions, the iPlane requires measurements of a large number of Internet routes. When measuring the Internet core, a concern is to make accurate measurements without using too much bandwidth. Techniques for this have been published, yielding tools such as Spruce (Strauss, Katabi et al. 2003) and PathLoad (Jain and Dovrolis 2003) for available bandwidth, CapProbe (Kapoor, Chen et al. 2004) and SProbe (Saroiu, Gummadi et al. 2002) for bandwidth capacity, and `ping` and `traceroute` for latency and topology. Even with these tools, researchers are limited when performing measurements to end-hosts (Spring, Peterson et al. 2006) even though this is where the greatest diversity of the network link properties exists.

Measurements to the edge have proved to be difficult due to uncooperative and even hostile hosts. End hosts are often firewalled, which makes them not respond to active probing and rules out many of the tools that are used for mapping the core. Hosts can also mistake a measurement probe for an intrusion attempt, causing alarms in intrusion detection systems (IDS). To avoid these problems, researchers have been forced to take alternative approach to end host measurements. Instead of active probing, systems such as PlanetSeer (Zhang, Zhang et al. 2004) have taken an opportunistic approach to end host measurements as described in (Chen, Bindel et al. 2004). These systems infer link properties by passively monitoring TCP streams of hosts using a service already in use by the measuring node. This allows measurements of end-hosts without triggering alarms in IDSs. These systems, while successful, have the drawback that they are limited by the popularity of the service offered. Getting providers of popular content to install the required measurement application on their servers has also proven difficult.

Instead, I propose a solution where the measuring nodes participate in popular peer-to-peer systems. These systems are widely deployed and provide millions

of hosts that can be measured. One of the most widely deployed systems, BitTorrent, provides just the level of free-riding support required to be able to perform a wide array of measurements. The information that can be collected includes bandwidth capacity, latency and topology. While this information is collected, the measuring node still looks like any other peer from the perspective of the measured host. This is crucial since it avoids triggering alarms in IDSs.

In this report, I present the design and implementation of BitProbes, a system for large scale end-host measurements. I show that the use of BitTorrent provides a large user-base that can be measured unobtrusively. The system is capable of measuring the upload capacity and link latency of end hosts as well as recording the path to the host. I also show that measurements to one end host generalize to other hosts close in IP address space. I then present the design of a probing tool that can measure the download capacity of end hosts in what looks like a BitTorrent application level handshake. This system is currently running on 8 dedicated machines in a server cluster at the University of Washington. The measurements are stored in a database accessible by the iPlane. These nodes provide roughly half a million connections to distinct end hosts per week, connections that are used to transparently measure the hosts.

These measurements allow the iPlane to not only provide predictions about Internet core routes, but also to provide predictions about the link performance between two arbitrarily selected end hosts. A description of the iPlane is available in (Madhyastha, Isdal et al. 2006). In that paper we implement a content distribution service, a peer-to-peer swarming file-sharing application and Voice-over-IP application that each use predictions made by the iPlane. In all of these applications we show that prior knowledge about the characteristics of Internet paths, such as capacity, latency and loss-rate leads to significant improvement in end-user performance.

The rest of this report is organized as follows. Section 2 provides background information about the technologies used in this system. Section 3 gives the reader an overview of the system design followed by a detailed description of the implementation of the system in Section 4. Section 5 presents an evaluation of the methods used as well as statistics about the collected measurements. Section 6 gives an overview of previous systems that have used similar techniques to collect measurements. Section 7 concludes the report and discusses the success of the method.

2 Background

This chapter provides background information for readers that are unfamiliar with the iPlane, the BitTorrent protocol or Internet measurements.

The iPlane section provides a short introduction to the iPlane. The iPlane provides a query interface for applications that benefit from prior knowledge of Internet path characteristics.

The BitTorrent section introduces the BitTorrent protocol and the different components of a BitTorrent system. It then goes into more technical detail about the parts of the protocol that make BitTorrent suitable to Internet measurements.

The measurement section contains an extensive overview of previous Internet measurements research. The focus is on bandwidth capacity measurements since these are the measurements performed by this system that are the most difficult to perform accurately.

2.1 The iPlane

iPlane: an Information Plane for Distributed Services (Madhyastha, Isdal et al. 2006) is a system developed at the University of Washington. The iPlane, when queried by an application, is able to predict properties of arbitary Internet paths, allowing the application to make better decisions about for example peer or server selection.

The iPlane has been evaluated with three applications: a CDN service, a Voice-over-IP, or Skype-like, application and BitTorrent. Results from the CDN and the BitTorrent experiment are shown below.

In the BitTorrent experiments, we used a modified BitTorrent tracker. The standard BitTorrent tracker will, when queried, return a random subset of the peers associated with the particular swarm. We modified the standard BitTorrent tracker to instead take predictions made by the iPlane into account. Instead of randomly returning peers, the modified tracker returns only 50% of the peers selected randomly. The other 50% of the peers returned are peers that are predicted to be well connected with the querying peer. To check if our technique is better than proposed optimizations to the BitTorrent protocol, we also compared against an extension to BitTorrent that utilizes Vivaldi (Dabek,

Cox et al. 2004), a system for Internet coordinates. A cumulative distribution function (CDF) plot of comparative results is given in Figure 1.

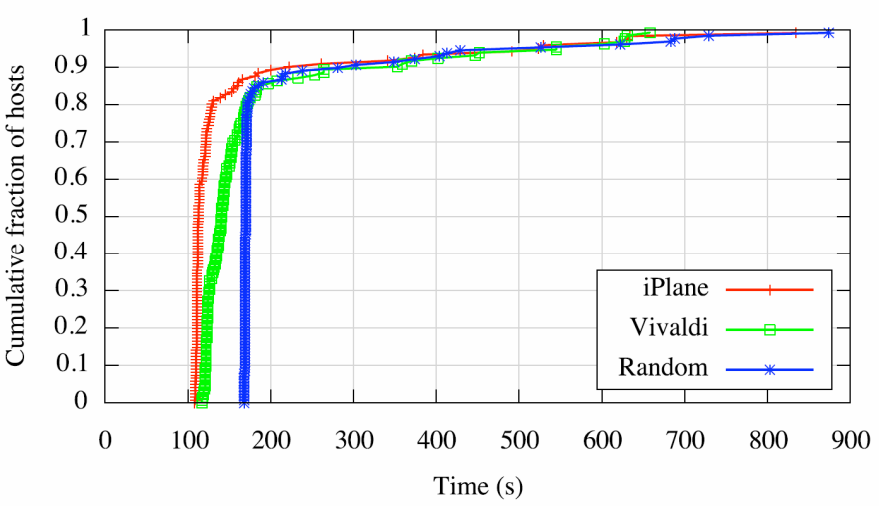

Figure 1: A comparison of BitTorrent performance with and without path-prediction on the tracker.

As can be seen in the figure, 80% of the hosts using the iPlane tracker have completed their download within the first 130 seconds. At this time only 35% of the Vivaldi hosts and none of the hosts using the normal BitTorrent tracker had completed the download. It can also be seen that the slowest 10% see only a very limited improvement with the iPlane's tracker, suggesting that this optimization primarily is important for hosts that do not utilize their potential capacity with regular BitTorrent. The conclusion of this experiment is that a tracker that is querying the iPlane will improve the performance of BitTorrent file transfers.

We also verified that the iPlane can help content providers such as Akamai and CoDeeN that direct hosts to the server which will provide the highest TCP throughput as suggested by the PFTK model (Jitendra, Victor et al. 1998) and the iPlanes predictions link characteristics. Current CDNs instruct their name servers to direct clients to the server that is closest in terms of latency. To see if the predictions made by the iPlane could perform better we compared download times of clients connecting to the closest server, and clients that connected to the server that the iPlane predicted to have best performance. It should be noted that the choice of the closest server was done by an "oracle" having complete knowledge of the network; something that would requires extensive probing, and therefore is infeasible in the real world.

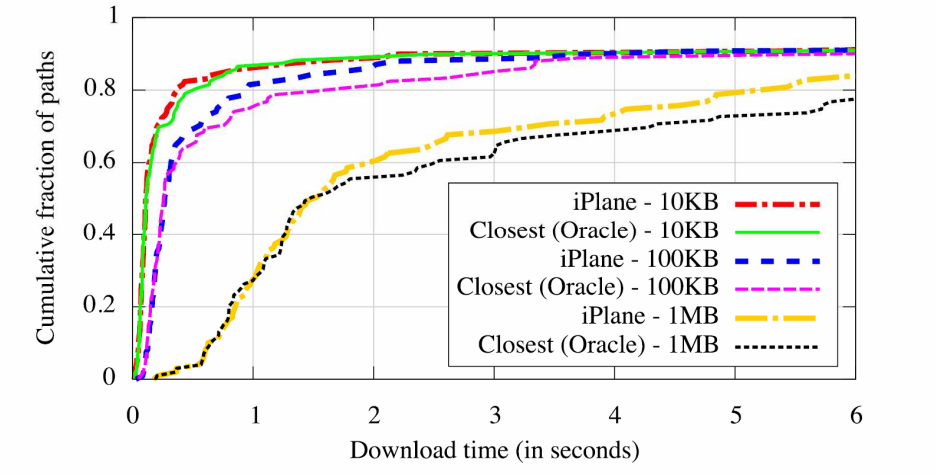

Figure 2: Download times from CDN using iPlane's performance predictions or the closest server in terms on latency.

As seen in Figure 2, the iPlane and the oracle have very similar performance for small files. For larger files, the iPlane performs marginally better than the oracle. This is very encouraging results since the iPlane only uses predictions, while the oracle relies on active probes to create a complete map of the network.

2.2 The BitTorrent protocol

The BitTorrent protocol is a peer-to-peer file transfer protocol that has become increasingly popular the last couple of years. Sources state that as much as one third of all Internet backbone traffic originate from BitTorrent transfers (Parker 2004).

2.2.1 General idea of BitTorrent

The BitTorrent protocol was developed to allow people with limited resources to make their content available to a large group of people. The traditional client-server model of the Internet, illustrated in Figure 3, requires the server to send one copy to each client causing the load on the server to increase linearly with the number of clients.

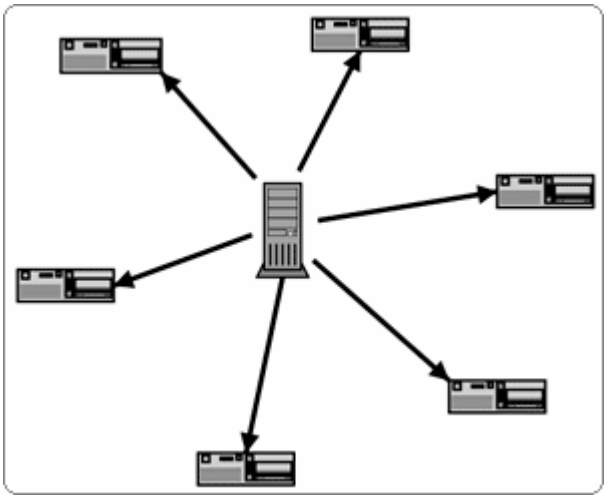

Figure 3: Centralized file distribution.
Figure source: (bittorrent.org 2006)

The purpose with BitTorrent is to utilize the upload resources available on the participating clients, resulting in increasing system resources as clients connect. An illustration can be seen in Figure 4. The clients, or peers, are grouped into swarms, where each swarm contains the clients interested in the same data. As soon as a peer has received a piece of the data, it announces to the other peers that it has that data. The peers are therefore able to request data from any participant in the swarm which has the requested data, not only the original source. Utilizing the resources of the clients allows for file distribution that scales very well.

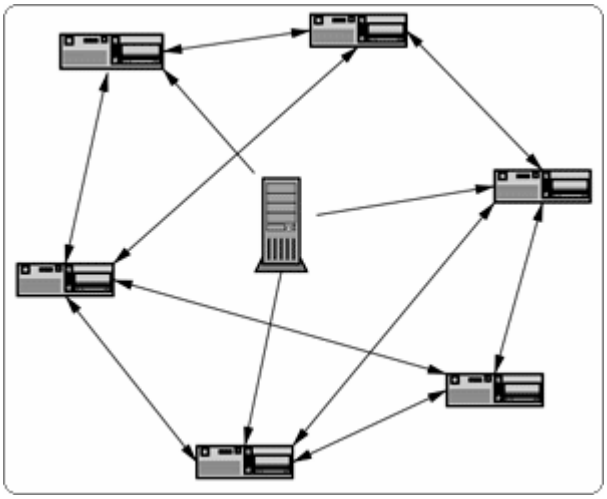

Figure 4: File distribution with BitTorrent
Figure source: (bittorrent.org 2006)

BitTorrent is able to distribute both single files and multiple files. All the information about the structure of the files transmitted is contained in a .torrent file. This .torrent file includes checksums of the contents of the files transferred to make it possible for the peers to verify that the information received in fact is the information sent by the initial source. To make the transfer more efficient the content is divided into pieces, each piece being around 64-256 Kbytes, although larger pieces sometimes are used. The .torrent file contains the checksum of all the pieces which makes it possible for clients to recover from transmission errors without any significant penalty. This also makes it difficult to cheat the system since any malfunctioning or hostile peer sending bogus information immediately is detected (bittorrent.org 2006).

Using pieces makes it is possible for peers to upload data to other peers, despite that they only have received a small fraction of the total content. To optimize availability of all content the peers request those pieces that they see are in most demand among the peers to which they are connected. This policy is called Local Rarest First (LRF) and has shown to be a simple yet efficient way of improving block availability in most circumstances (Bharambe, Herley et al. 2006).

2.2.2 Components of a BitTorrent system

A BitTorrent file distribution system requires the following components:

- An ordinary web server
- A static .torrent file containing meta-information about the torrent
- A BitTorrent tracker
- An initial seed possessing the whole file
- The end user web browsers
- The end user downloaders

Serving files using BitTorrent requires one additional centralized component, the tracker, compared to serving files using http. The BitTorrent tracker serves as the hub of the swarm, allowing the clients of the system, the peers, to find other peers interested in the same data. During the time which a peer is downloading it is called a *leecher*, when the peer is done downloading but still serving the file, the peer is called a *seeder*. A BitTorrent download is initiated when the user downloads a .torrent file and supplies it to a BitTorrent client. The .torrent file can be downloaded using any file-transfer method; the most common way is to download the .torrent file from an ordinary http-server.

The .torrent file contains administrative information about the torrent; including the address of the tracker. The tracker keeps track of the peers currently associated with each of the torrents currently served by the tracker. Each new peer queries the tracker to get information about other peers

interested in the same torrent. The tracker returns the addresses of randomly selected peers as response to the query. The tracker will also remember the address information about the querying peer, to be able to return that address when queried by other peers interested in the same torrent. The BitTorrent protocol allows the tracker to specify how often a peer is allowed to query for new peers. Trackers usually limit the peers to send one query every 300 to 600 seconds. This query rate is too low to be able to perform measurements of end-hosts at a high rate, and the BitTorrent client used by BitProbes takes a unique approach to increase the rate at which new peers are discovered without violating the BitTorrent protocol.

2.2.3 Protocol information

All BitTorrent clients have to adhere to the BitTorrent protocol. The protocol specifies how clients communicate with each other, and is well documented. This has allowed a large number of BitTorrent clients to be developed by users that do not like or wish to improve the main-line client. For the purposes of Internet measurements two parts of the protocol specification are important. The specification for when to send data to other clients and the instructions for how to notify other clients that a piece has been received. All information in this section is based on version 1.0 of the BitTorrent specification (Cohen 2002).

2.2.3.1 Choking, unchoking and optimistic unchokes

For peer-to-peer file sharing applications to be efficient there must be incentives for the participants to contribute recourses to the system. In BitTorrent this is achieved via a Tit-For-Tat (TFT) approach. The core of this approach basically mandates; if you send data to me, I will send data to you. This has the effect that the more a peer contributes to the swarm, the faster the download rate of that peer will be. When a peer decides to start sending data to another peer, the sending peer is *unchoking* the receiving peer. If that peer later notices that it does not benefit from the data exchange the sending peer stops transmitting data, the receiving peer is said to be *choked*.

The problem with a pure TFT approach is that it provides no reason for a peer to unchoke a newly joined peer as that peer does not have any pieces to upload and will not be able to reciprocate. To cope with this, BitTorrent uses *optimistic unchokes* that are designed to aid in the bootstrapping of new peers. Each time period, usually 30 seconds, each peer looks over its relations with other peers and chokes the peer that has sent it the smallest amount of data. It then randomly unchokes a new peer in its *peer-set*. The peer-set is the peers to which the peer has active TCP connections to. This serves two purposes: firstly it helps to bootstrap new peers so that they can contribute their resources. Secondly, it allows each peer to search for other peers with which it can have a

profitable relationship. The peer getting unchoked will hopefully reciprocate allowing a data transfer to occur in both directions.

For the purpose of end-host measurements, the optimistic unchokes are crucial. BitProbes relies entirely on optimistic unchokes, and therefore the behavior of the BitTorrent client, running on the measurement nodes, is optimized to maximize the possibility of being optimistically unchoked. The key observation is that keeping the measurement node in the peer-set of as many peers in the swarm as possible maximizes the probability of getting unchokes.

2.2.3.2 Bitfields and "have" messages

BitTorrent is a request driven protocol. Each peer requests pieces from the other peers in the swarm. Each peer is responsible for notifying the other peers in its peer-set at to which pieces it has. These are the pieces the other peers can then request. This information is distributed in two ways.

During the handshake between two newly connected peers an optional BitField message can be sent. This message is as many bits long as the total number of pieces in the torrent. The BitField contains a 1 at the indices where the peer has the corresponding piece and zeros at all other positions. This allows newly connected peers to efficiently update each other's views of which pieces that are available for request.

The other message type is the *"have"* message. Every time a peer successfully receives a piece, it sends a "have" message to all the peers in its peer-set. This tells the other peer that the corresponding piece now is available for download. It is in the interest of each peer to advertise their pieces as quickly as possible to the peers in their peer-group since it makes the peer more interesting and therefore increase the chances of being unchoked.

By monitoring the rate at which a peer sends out "have" messages it is possible for the other peers in the swarm to infer the download rate of that peer. This is done by multiplying the rate at which "have" messages are sent by the size of each piece. This information is useful for end-host measurements since it allows a very conservative estimate of the download capacity of the host. This information can then be used to eliminate evidently erroneous measurements done with other techniques.

Unfortunately, from the standpoint of end-host measurements, the "have" messages are sent in the same TCP-stream as all other information sent between peers. This includes the actual data stream which can result in blocks of 16 Kbytes to be queued in front of "have" messages. Because of this, the time which "have" messages are received by the other peers might not correspond to the exact time at which the peer received the piece.

2.2.4 Favorable properties of BitTorrent for Internet measurements

End-hosts running BitTorrent have many properties that make them amenable to Internet measurements. The most important one are:

- Because the way the tracker is implemented it is possible to connect to a significant portion of the peers in a swarm. The tracker has knowledge of all peers in the swarm and will provide peers with lists of other peers upon request. This list is a random subset of all peers in the swarm, which makes it possible to extract a large portion of the swarm's population by repeatedly querying the tracker for new peers.
- When connecting to a BitTorrent swarm, the other peers in the swarm will send data to the newly connected peer as a part of the optimistic unchoke bootstrapping mechanism. This allows measurement nodes to receive data without providing any data in return. During the time of which the TCP-connection is active, it is possible to perform measurements on the end-host.
- BitTorrent is immensely popular. BitTorrent has a large user group that frequently uses it to download files. Recently several large companies have started to use BitTorrent as a way of distributing content (Crawford 2005). This is a way for them to improve user experience by providing faster downloads as well as to externalize bandwidth costs. BitTorrent shifts the cost of content distribution from the content providers to the Internet Service Providers. This makes it probable that the use of BitTorrent will increase in the future, which is good for opportunistic measurements.
- BitTorrent provides the ability to track other users download rate by monitoring their "have" messages. This information is useful as it provides a conservative estimate of the download capacity of other users.

2.3 Measurement theory

To be able to perform accurate measurements it is important to have and understanding about current research in the area. In this section I will provide an overview over the previous tools and methods that can be used in BitProbes.

2.3.1 Topology

The topology of the Internet can be discovered using two techniques, the traceroute based technique and the IP Record Route based technique.

2.3.1.1 Measurements with traceroute

The goal of the *traceroute* tool is to attempt to follow the path an IP packet takes through the network. It sends packets with increasing IP time to live (TTL) values. The IP TTL field specifies how many hops through the network a packet is allowed to propagate. Each time the packet passes through a router the TTL field is decremented by one. The router that decrements the TTL field to zero sends an ICMP "time exceeded" message to the source of the message and discards the packet.

To figure out the address of all routers a packet passes through on the path to the destination, the program starts by sending a packet to the destination with a TTL value of one. The first router on the path will decrement the TTL to zero and therefore send back a time exceeded message. This message is received by the traceroute application and the source of the message is displayed to the user. traceroute then continues with a TTL of 2 and so on. The original traceroute sends UDP packets to a random port on the remote host. When the target host receives the UDP packet it will responds with an ICMP "port unreachable" packet. This is the sign that the traceroute has reached the target. For the traceroute to work the port on the target host must be unused, not firewalled and the target must allow outgoing ICMP port unreachable packets (Jacobson 1988).

2.3.1.2 Measurements with IP record route

The IP Record Route (RR) option is described in RFC 791 (ISI-USC 1981). It allows the source of a packet to request all intermediate routers to store their address in the header of the packet. When the packet reaches the destination it contains the path of which the packet traversed the network. The main drawback with IP RR option is that it only stores the first 9 hops of the path. Many paths on the Internet are longer than this and can therefore not be entirely mapped with IP RR.

The advantage with IP RR is that is records the address of the outgoing interface of the router, the interface from which the router sends the packet to the next hop. This differs from a traceroute which records the address of the incoming interfaces of the router. By using both traceroute and IP RR it is therefore possible to get gather more complete information that yields a higher quality mapping of Internet paths (Sherwood and Spring 2006).

2.3.2 Available bandwidth and loss

For many applications knowing the available bandwidth and loss-rate of an end-host can provide valuable information about the expected performance of that host. One of the main disadvantages of employing BitTorrent users as targets for the measurements is that there is a risk that the fact that the end-hosts actively participate in a BitTorrent swarm affects the measurement

results. For measuring the loss this is certainly the case. Since TCP uses loss events as markers that the link is congested, it is difficult to know if the loss is caused by congestion or link quality. Because of this is it difficult to derive an accurate loss-rate measurement from end-hosts using the technique described in this paper.

For measurements of available bandwidth the measurements become affected by the fact that the end-host is participating in a BitTorrent swarm. The peers in a BitTorrent swarm are encouraged to saturate their connection to minimize their download time. Measurements of available bandwidth during this time will results in a low available bandwidth, which is true at the time but it is also difficult to draw any conclusions about the amount of available bandwidth once the hosts leaves the BitTorrent swarm. It is believed that most end-hosts have a low utilization of their Internet connection most of the time, which makes the bandwidth capacity of a link a better measure of expected performance than available bandwidth.

2.3.3 Latency

The latency of a path is the time required for a packet to traverse it. The term latency is in literature often used interchangeably with terms such as Round Trip Time (RTT) and ping. In this report I will use the term latency to describe the RTT of a path.

There is a couple of ways to find the latency of a path. The classic way is to send ICMP Echo Request packets to the destination and wait for the echo reply. This operation is described in RFC 791 (ISI-USC 1981), which also states that “every host MUST” implement this functionality. The `ping` application sends echo requests and records the time taken before the reply is received; the time is then displayed to the user. To use pings to measure latency is the first choice, and also the way latencies are measured in the Internet core. Sending pings to end-hosts is unfortunately less straightforward. Many end-hosts are behind firewalls which filter ICMP traffic, making the ping application less useful. To make matters worse, many firewalls also notify the user about the incoming ICMP packet which might lead to abuse letters being sent to the researchers responsible. Policies disallowing pings to end-hosts are therefore typical on measurements platforms such as PlanetLab.

Another way to measure latency is to use application level latency. This means sending a packet and recording the time for an application level response to be received. This can for example be the time taken for the TCP handshake. When a TCP connection is initiated the client sends a TCP SYN packet to the destination. The destination then acknowledges the packet with a TCP SYN ACK packet. The time required for this is usually slightly longer that the time required for an ICMP ECHO Request / Response. In spite of this, it is a useful way to record the latency of a path without sending any ICMP packets.

2.3.4 Bandwidth capacity

Measurements of topology and latency are uncomplicated compared to measurements of bandwidth capacity. Measuring the bandwidth capacity in packet switched networks is an active research area with new methods being published frequently. I will in this report give an overview of the current status in the subject. So far research in the area of bandwidth capacity can be grouped into two different techniques: one packet and packet pair technique.

2.3.4.1 The one packet technique

The one packet technique was first proposed in (Bellovin 1992) and can be summarized as follows: the propagation delay of a packet is a sum of the following delays: signal speed, queuing delay and transmission delay.

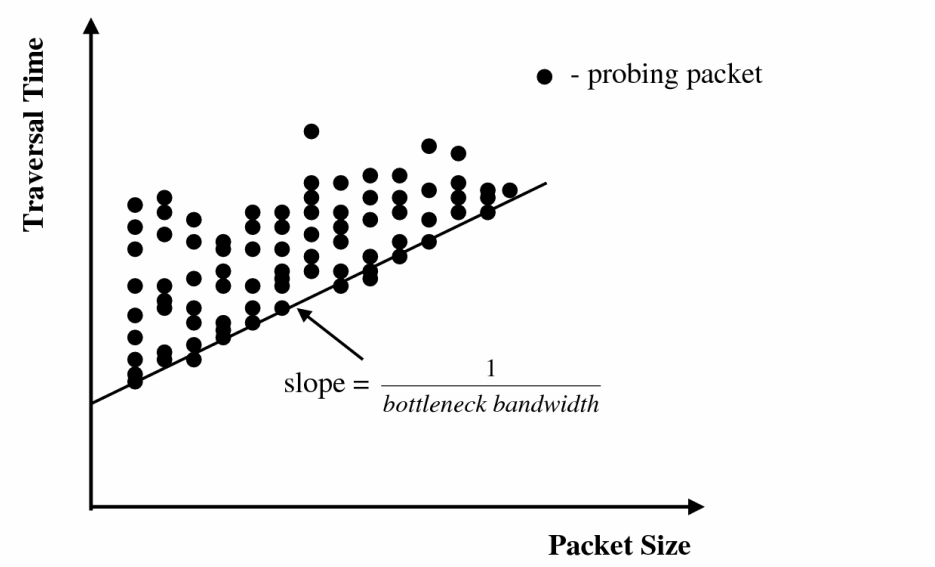

Figure 5: The one packet technique. Ideally one packet of each probe size traverses the network without experiencing cross-traffic. By looking at minimum delay packets, the bottleneck bandwidth can be inferred.
Figure source: (Saroiu, Gummadi et al. 2002)

- Signal delay; the time required for the signal to propagate in the physical medium. In fiber the speed is roughly two thirds of the speed of light. This delay is constant as long as the path between the hosts is the same.
- Queuing delay can be assumed to be constant if the link has no cross traffic. In the presence of cross traffic the queuing delay can be coped with by taking a large number of measurements and using the ones

with the lowest delay. This measurement is assumed to have propagated through the network without any queuing delay.

- Transmission delay is the time it takes to transmit the data over the network. This delay varies with the bandwidth capacity of the link. In an ideal network this delay should increase linearly with packet size.

By varying the packet size and measuring how quickly the packet traverses the network, the bandwidth capacity can be inferred. The probe which has the smallest Round Trip Time (RTT) is assumed to have suffered least queuing from cross-traffic. An illustration of the one packet technique can be seen in Figure 5.

2.3.4.2 The packet pair technique

The packet pair technique was introduced by van Jacobsen in (Jacobson 1995). The idea is that packets that are sent back to back by the source often will traverse the bottleneck link back to back. When the packets then exit the bottleneck link and continue into a link with higher capacity they will be spread out. As illustrated in Figure 6 the delay between these packets when they arrive to the destination will then be proportional to the bottleneck link bandwidth.

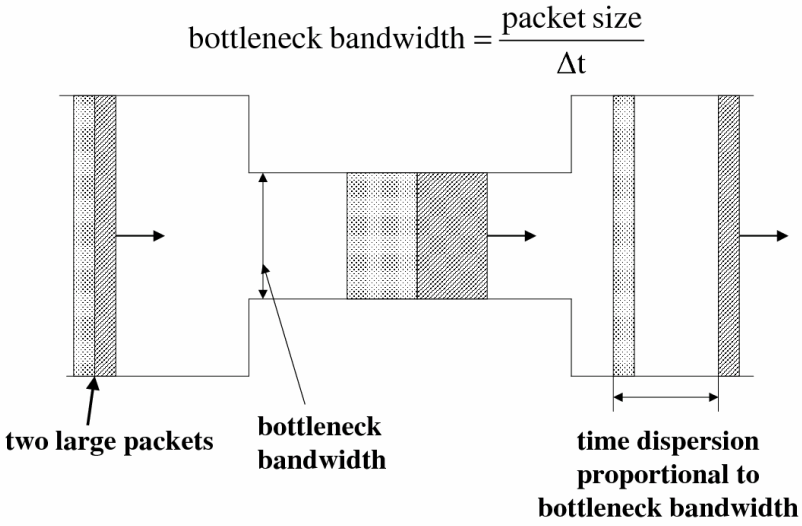

Figure 6: The packet pair technique. The bottleneck introduces a distance between packets that is proportional to bottleneck bandwidth capacity.
Figure source: (Saroiu, Gummadi et al. 2002)

The packet pair technique is also sensitive to cross traffic. If the packets are queued on intermediate routers on their way from the source, the measurements can both indicate a higher or lower bandwidth capacity than the

true capacity. If the second packet is queued, the result will be an underestimate. If the first packet is queued after the bottleneck link it will result in an overestimate. To cope with the problems of cross traffic a Probability Density Function (PDF) is computed. By searching for peaks in the PDF it is possible to find the inter-arrival times of packets that traversed the network without any queuing. It is assumed that cross traffic will have a more random distribution while the measurements without cross traffic will have a more compact distribution (Lai and Baker 2001). An example of the PDF of a flow experiencing cross traffic can be seen in Figure 7.

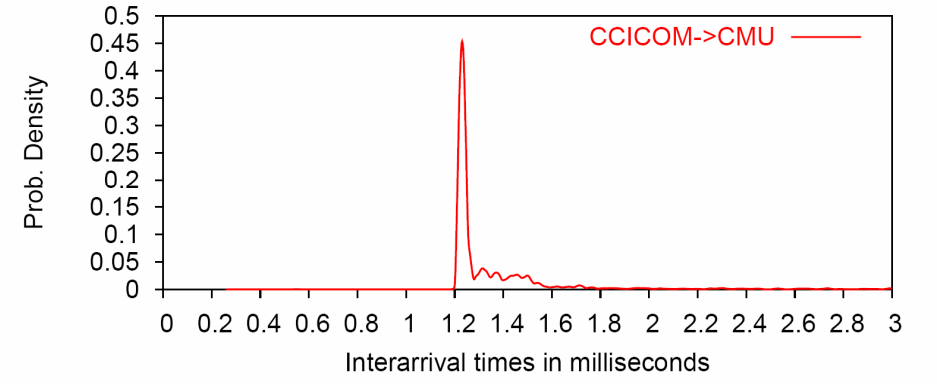

Figure 7: Probability distribution of a flow from a high capacity host to a host with lower capacity. Limited queuing is experienced after the bottleneck link. Figure source: (Katti, Katabi et al. 2004)

To get even more information about the bandwidth capacities along the end-to-end path, recent tools also take mode gaps into account. A mode gap is the distance between common packet inter-arrival times. By looking for reoccurring distances it is possible not only to find the bandwidth capacity of the bottleneck link, but also to find the capacity of up to 3 bottlenecks on the path. The way this works is that the packets traversing the path often are 1500 bytes large, that is the Maximum Transfer Unit (MTU) of standard Ethernet. Cross-traffic will therefore cause queuing in discrete 1500, 3000, 4500 Kbytes lengths and so on, corresponding to a queue of 1, 2, 3 … packets. Because of this the packet inter-arrival times will have a probability distribution similar to the one in Figure 8. The figure shows two different queuing delays, the dominant one at 1.2 ms, indicating a bottleneck bandwidth of 10 Mbit/s. Then a distinct mode gap of 0.12 ms indicating that packets often get queued at an additional link where the time needed for a 1500 byte packet to traverse is 0.12 ms. This indicates that the second bottleneck link has a capacity of 100 Mbit/s. (Katti, Katabi et al. 2004)

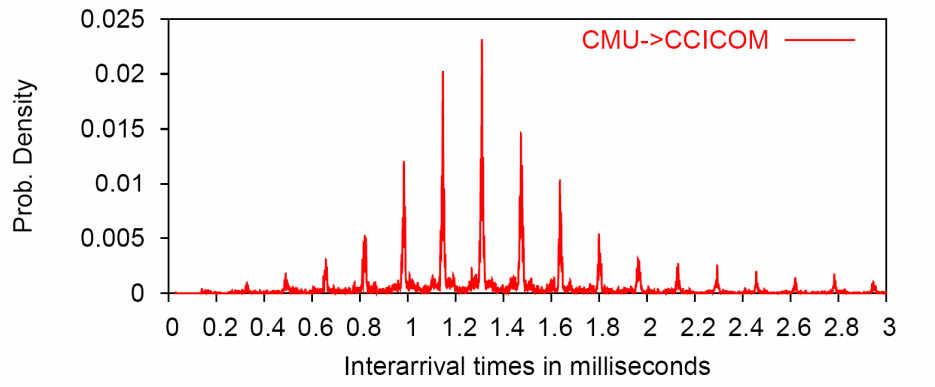

Figure 8: Probability distribution with mode gaps. Packets are queued because of cross-traffic after the bottle-neck link.
Figure source: (Katti, Katabi et al. 2004)

The packet pair technique has the requirement that the intermediate routers use First In First Out (FIFO) queuing. If the bottleneck router uses another queuing policy the measurements will be inaccurate.

2.3.5 Overview of existing tools for bandwidth capacity estimation

There are currently three tools that all perform capacity measurements with high accuracy. Two are passive, MultiQ and NetTimer, and one is active, PathRate. All these tools, when running in their most accurate modes, are capable of measuring the path capacity within 10% of the true capacity 85% of the time.

2.3.5.1 Passive tools

MultiQ

MultiQ, (Katti, Katabi et al. 2004), is a tool developed at Massachusetts Institute of Technology designed to find the bottleneck bandwidth capacity from TCP traces. Specifically, it is designed to obtain accurate measurements while only requiring traces from the receiver side of the TCP connection. MultiQ use the packet pair technique to find the bottleneck bandwidth capacity. By introducing the Equally-spaced Mode Gaps (EMG) algorithm in addition MultiQ also has the ability to find additional bottleneck amount the path, finding non minimum capacity bottlenecks on 64% of paths.

MultiQ has an accuracy that is as good as Pathrate, the most accurate active tool. The accuracy of MultiQ is also comparable to the accuracy of NetTimer when NetTimer operates in RBPP mode, that is when NetTimer has access to the packet traces from both then sender and the receiver. Because of the accuracy for receiver side traces this is the tool used for measurements of

upload capacity of end hosts in BitProbes. To allow future change of the algorithm used to measure bandwidth capacity, all timestamps of incoming large packets are save and stored.

MultiQ has the disadvantage that measurement of the downlink capacity of the end-host is fairly inaccurate. The paper states that 70% of the measurements are within 20% of their true value. This big difference in accuracy is because the only information accessible to the measurement node is the inter-arrival times of incoming TCP ACK packets. This introduces a new source of error since the ACK might have been queued on their way back to the measuring node. Many TCP implementations also use TCP delayed ACK, as specified in (Allman, Paxson et al. 1999). This causes the receiver to send an ACK only every other packet which decreases the number of data points by a factor of 2. The fact that the data uploaded by our system is very limited also makes an ACK based approach infeasible. To be effective the amount of data uploaded must be significant enough to generate many pairs of back-to-back maximum size packets. This is not the case in the current system.

For the purposes of this system, the MultiQ technique works very well for measurement of end-host uplink bandwidth capacity. For measurement of downlink capacity it is less suitable and therefore another method to measure the download bandwidth of end-hosts had to be developed. The design of this tool is presented later in the report.

NetTimer

NetTimer, (Lai and Baker 2001), was created at Stanford University and is designed to enable content providers to vary the size and quality of content vary depending on the bandwidth of the path. There is no reason to try to stream high quality video to a receiver that only has a dial up internet connection. NetTimer use kernel density functions (Scott 1992) to eliminate measurements where cross traffic might have affected the result. The NetTimer measurements can be performed in different modes. Receiver Based Packet Pair (RBPP), first described in (Paxson 1997) is the most accurate mode, but requires packet traces from both the sender and the receiver. Because of this limitation this mode cannot be used in this system.

For the purpose of this paper the most interesting modes are the Receiver Only Packet Pair (ROPP), first described in (Lai and Baker 1999), and Sender Based Packet Pair (SBPP). ROPP look at the inter-arrival times of incoming data packets. SBPP is using transport or link layer acknowledgements to infer the arrival times of the data packets. These modes have the same properties and limitations as the corresponding modes in MultiQ with the difference that they are slightly less accurate.

2.3.5.2 Active tools

The active tools can be classified into two: tools that require the person measuring to have control of both end hosts of the path measured, and those that only require control of one of the hosts. The tools that require both end points have the advantage that they generally are more accurate than the ones that only require control of one end point. The disadvantage is obviously that control over both end points is required.

Pathrate

Pathrate, (Dovrolis, Ramanathan et al. 2004), is an active tool that utilizes the packet-pair technique. It requires control of both hosts, so the method can not be used by BitProbes. Pathrate is very accurate, so it is despite this limitation interesting to study it. When measuring the end-to-end bandwidth capacity between two hosts, Pathrate sends a large number of packet-pair probes. The inter-arrival times of these packets in each probe is then recorded. To handle queuing from cross-traffic Pathrate use a technique much similar to the technique used by MultiQ based on the distribution of inter-arrival times. Since Pathrate has control over both end-hosts it is possible to send a large number of probes, resulting in accurate measurements. The capacity measured by Pathrate is comparable to the accuracy of MultiQ and NetTimer making it the most accurate active tool studied in this report.

SProbe

SProbe, (Saroiu, Gummadi et al. 2002), is a tool developed at the University of Washington used to measure the bandwidth capacity in uncooperative environments, and therefore only requires control of one host. SProbe sends a probe of two TCP SYN packets with a large payload (1460 bytes). The probe is sent to a closed port on the remote host. It then uses the fact that the remote host will respond with a RST packet when it receives a packet on a closed port. Unfortunately many hosts are behind firewalls, which will make them unresponsive to this.

Since the packets sent are large they will be queued up at the bottleneck link. The RST packets sent by the remote host are small leading to the assumption that they will not be queued at any intermediate router. Therefore the time between the RST packets received at the probing host should be the same as the time between the SYN packets received by the remote host. To improve handling resilience to cross traffic SProbe sends the two large packets within a train of small packets. If the small packets are reordered the measurements is discarded because of cross traffic. This way SProbe can filter probes that have experienced cross-traffic.

SProbe has an accuracy of 80% within a factor of two and 60% within 10% of the bandwidth as estimated by NetTimer.

The main disadvantage with SProbe is that the probes easily can trigger Intrusion detection alarms causing trouble for researchers using the tool extensively. If a service generates too many complaints it might be suspended from PlanetLab. The goal is that the system described in this paper eventually will run on PlanetLab, which makes any obtrusiveness unacceptable. The fairly inaccurate results are also a problem. Because of these two problems SProbe is not used by BitProbes.

CapProbe

CapProbe, (Kapoor, Chen et al. 2004) developed at University of California at Los Angeles, combines the packet-pair technique with the one packet technique. It only requires control of one side of the path which makes it eligible for measurements of end-hosts. The authors observe that a packet pair that produces an over- or under-estimate must have been queued by cross traffic. To cope with this, CapProbe only marks probes as valid if they experience minimal delay during the transit over the network. This makes it possible to achieve good accuracy with a limited number of probes. The number of probe pairs sent varies between 40 and 100.

The probes used in the paper are ICMP echo requests, but the authors have also been able to incorporate probes in regular TCP stream. The disadvantage with ICMP echo probes is that the receiver will respond with an ICMP echo reply which will contain the same payload as the echo request. This means that the packets going to the target will have the same size as packets going from the target. Because of this the current implementation of CapProbe only measures the link of the host with the smallest capacity.

The CapProbe paper also states that CapProbe functions poorly when the amount of cross traffic exceeds 50% of the link capacity. This is because the possibility to get a sample where no queuing occurred gets smaller as the congestion of a link increases. This makes CapProbe unsuitable for the measurement of hosts participating in BitTorrent swarms, our target environment, since these hosts tend to saturate their upload bandwidth. Because many end hosts have asymmetric connections to the Internet, these hosts will saturate their upload capacity while still having available download capacity. In an environment which is favorable for CapProbe it is as accurate as Pathrate.

A possible modification to CapProbe that would allow it to measure the downlink capacity while not suffering from low capacity uplink is to send TCP SYN packets to unused ports, much the same way as SProbe does. The target host would then send TCP RST packets back and the inter-arrival time between these would indicate the inter-arrival time of the original probe packets. The probe packets can be as large as 1500 bytes, while the RST responses will be only 40 bytes large. Unfortunately, a large number of TCP SYN packets to

unused ports have a tendency to trigger IDS alarms, which makes this solution unsuitable for BitProbes.

3 System design

This chapter gives the reader a short introduction to the design of BitProbes. The goals of the system are presented, as well as a description the target environment in which the system will operate. The chapter also contains short descriptions of the operation of the system to give the reader an understanding of how the system works.

The purpose of BitProbes system is to provide measurements of end-hosts to the iPlane. An overview of the system is presented in Figure 9. I start by discussing the requirements of BitProbes.

- *Accuracy*: It is important that the measurements made have sufficient quality to make them useful for the clients of the iPlane.
- *Wide coverage*: For measurements of end-hosts to have high value, a significant number of measurements must be made. Hosts that are close in IP-space have a high probability to have similar connection properties. The more measurements made, the higher is the probability to find a measurement close to any given IP address.
- *Resource efficiency*: The parts of the system that are running on remote nodes must consume a small amount of resources.
- *Unobtrusiveness*: The end-hosts measured must not be probed with anything that can be mistaken for an intrusion attempt.
- *Scalable components*: The components of the system that are centralized must be able to support a large number of remote nodes. When running on PlanetLab, up to 500 remote nodes could be used.

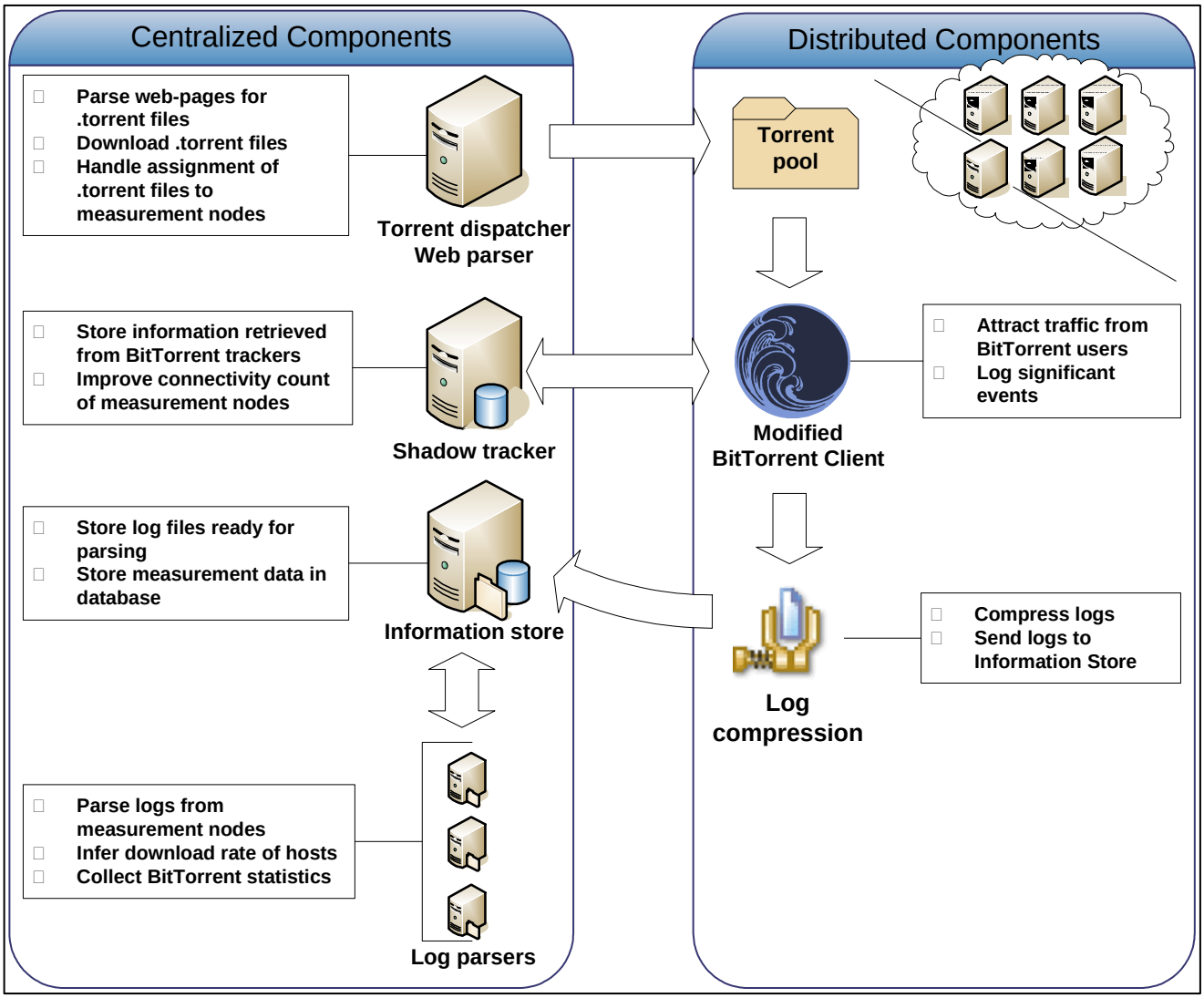

Figure 9: Overview of the BitProbes system

3.1 Overview

Measurements of end-hosts are more difficult than measurements of the internet core. As mentioned earlier, end-hosts are often behind firewalls which make them unresponsive to the measurements probes sent. Some end-hosts also run software that trigger alarms when any unusual network activity is happening. Probes used for active measurements of Internet characteristics look very different from normal Internet traffic, and might trigger alarms on the end-hosts. This causes the owners of the hosts to wonder what caused the alarms, to then look at the firewall logs, and to often misinterpret the probe as an intrusion attempt. This results in e-mails being sent to abuse departments at the ISP from which the probe came (Saroiu, Gummadi et al. 2002). Because of

this, there are often policies against active probing of end-hosts on distributed testbeds, and since this system is designed to run on *PlanetLab (PlanetLab 2006)* on which probes to end-hosts are prohibited (Spring, Peterson et al. 2006) another approach must be taken.

Instead of active probing, the system relies on opportunistic measurements of hosts that it is connected to. The number of measurements possible is therefore closely tied to the number of hosts to which the system has connections. One way to achieve a high connection count is for example to co-locate the measurements software on servers that already have a large number of clients. This has the disadvantage that it is necessary to persuade the owners of these servers that it is in their interest to agree to run the software. Instead I have taken a more active approach.

Each day thousand of Internet users use the popular peer-to-peer file distribution application BitTorrent to download files on the Internet. These files include for example Linux distributions, videos and music. BitTorrent allows a person with a low-end Internet connection to serve content to a large number of people. This is achieved by utilizing the upload capacity of everyone downloading the file. This feature has made BitTorrent very popular and some sources state that over one third of all Internet traffic in 2004 was BitTorrent transfers (Parker 2004). By joining BitTorrent swarms, it is possible for measurements nodes to connect to the hosts associated with that swarm thereby allowing measurements of those hosts.

3.1.1 Finding end-users to utilize as targets

The key to achieving good coverage of end-hosts is to make sure that the BitTorrent swarms connected to have a large number of members. Fortunately, many websites that provide files available for download over BitTorrent also specify how many users that currently is associated with that specific torrent. By connecting to the swarms with the highest number of users it is possible to direct the measurement nodes towards the swarms that have capability to provide a large number of measurements. To be able to connect to BitTorrent swarms a .torrent file is required. A .torrent file is a file that contains all information necessary for a BitTorrent client start downloading a file. To get the .torrent files, popular homepages that provide links to these files are crawled. These files are then distributed to the measurement nodes so that they can start to join swarms and attract traffic.

3.1.2 Attracting traffic

Many of the available techniques to transparently measure properties of an Internet path require a data transfer to already exist between the target and the measuring node. The way BitProbes attract traffic is to participate in BitTorrent swarms by having the measurement host run a modified BitTorrent

client. The client is modified in such way that it does not store or serve any of the content provided in the actual BitTorrent swarm. This allows the system to connect to a larger number of swarms since the risk of serving copy-righted content is eliminated.

A second modification I made to a regular BitTorrent client is to increase the rate at which it connects to new end-hosts. Many BitTorrent swarms limit the rate at which new peers are received, which decreases the number of possible hosts a measurement node can be connected to. To increase the number of connected hosts, all measurement nodes monitoring the same swarm inform each other about the users of the swarm. This allows them to connect not only to the clients they discovered themselves, but also to the clients discovered by other measurement nodes.

The reason why it is important to have a high connection count is that BitTorrent clients prefer to exchange content with other clients that previously have provided them with content. Since the measurement nodes do not have any content to provide, they have to rely on the other ways to get the clients to send data to them. Fortunately, the way the BitTorrent protocol works, it is required by the clients to send data to at least one randomly chosen host to which it is connected, this is called an *optimistic unchoke*. By being connected to a large number of hosts, the possibility of being unchoked accumulates causing each measurement node to receive a large amount of traffic.

3.1.3 Analyzing incoming TCP-streams

Attracting traffic is not enough to perform Internet measurements. By closely monitoring the rate at which packets sent by each end-host are received at the measuring node, it is possible to collect information about the Internet path between them. It is also possible to inject extra packets into an existing stream. These packets will look like legitimate TCP packets both to the end-host being probed and to any intermediate firewalls. These extra packets can be sent in such a way that they for example record the path between the end-host and the measuring node.

3.1.4 Analysis of log data

The BitTorrent protocol also requires that clients send information about their download progress to other clients to which they are connected. These messages contain information that when analyzed provide the rate of which each client in the swarm is receiving content. This is useful information since it provides a conservative estimate of the download capacity of the client. To collect this information, each of the measurement nodes logs all BitTorrent protocol events. These logs are then analyzed at a central location, the reason for this being that a global view of the activity in the swarm makes it possible to infer each client’s download rate with higher accuracy.

4 Implementation

This section describes the different components of the system. The section is divided into three subsections. The section titled centralized components describes components that are designed to run at University of Washington to manage the system. The distributed components run on measurements nodes and are designed to be able to run on PlanetLab. The section titled measurements describes the methods that were chosen to provide measurements, the methods that would provide the most accurate measurements given the environment in which BitProbes operates.

4.1 Centralized components

The centralized components are all running at University of Washington. These applications are designed to orchestrate the measurements by assigning jobs to measurements nodes. All centralized components are designed to be able to support up to 500 measurements nodes. The centralized tasks that require more computation power, such as the analysis of logs, can be split up on several local machines to make it possible to support a large number of measurement nodes.

4.1.1 Web parser

The web parser consists of a number of scripts that crawl popular BitTorrent websites. The crawler then adds all links to .torrent files it can find to a queue for later download. It also records how many seeders and leechers each torrent has. The accuracy of the numbers stated on tracker websites has been studied (Pouwelse, Garbacki et al. 2005) and is reliable. The number of seeders and leechers is useful when assigning torrents to measurement nodes since it allows both to assign more nodes to popular torrents as well as a chance to avoid unpopular torrents.

The torrents that pass the requirements of popularity are then downloaded. To avoid overwhelming the BitTorrent website, the downloader sleeps 5 seconds between downloads. This allows a download rate of roughly 6 torrents per minutes which is fast enough. The web parser is designed to make a new pass for torrents every 6 hours to make sure that recently added torrents get measured, as well as to remove old inactive torrents.

4.1.2 Torrent dispatcher

The torrent dispatcher has two tasks; making sure to reassign new torrents to the measurement nodes, done every 2 hours, and ensuring that the logs from the previous run is sent back for analysis.

Every two hours the torrent dispatcher runs a script on all measurement nodes that shuts down all currently running torrents, starts new torrents, and moves the old logs to the log store. To decrease log file size the logs are stored in a binary format as well as getting compressed with bzip2 before being transferred over the network.

4.1.3 Log analyzer

The log analyzer is a simple application that read the bzip2 compressed log files, analyses the information and inputs it into the measurement database. Information collected includes all BitTorrent protocol messages received by the measurement nodes. An anonymized version of the logs will be made public to the research community.

The log analyzer has information from all measurement nodes participating in a single torrent. It can get more accurate information about for example a single peers download rate by monitoring the times of which that peer sends BitTorrent have messages. Another reason to use logs as the way to transfer information about the BitTorrent swarms participated in is that the information collected later can be used in other studies. These logs can provide more information than what is usable by the iPlane, and I invite other researchers to study these logs to look at how BitTorrent functions. Hopefully statistics derived from the logs will provide researches with information about how to design more efficient file distribution protocols than what is available today.

The drawback from using logs is that the information collected is delayed up to one hour before it is entered in the database. This only applies to connection information and not the actual measurements, but one could imagine a scenario where information about a large number of disconnects could indicate a problem in the network. Timely information could better help service providers to locate problems in the network rapidly.

4.1.4 Shadow tracker

Since the system relies on optimistic unchokes to obtain measurements it is advantageous to be connected to as many peers as possible. To possibility to get unchoked by each individual peer is constant, but by being connected to a large number of peers the number of unchokes per hour increases. A challenge is that trackers often specify how often a peer can contact it to receive a list of new peers. This time is usually 10 minutes. To increase the number of peers each measurements node is connected to, I developed a shadow tracker

infrastructure. Each time a measurement node receives information about a peer from the tracker; this information is reported to the shadow tracker. The shadow tracker is then queried every minute by the measurement nodes for new peers. This allows the measurement nodes to share the peers received from the tracker and thus be connected to a higher number of peers, providing more measurements from more vantage points.

4.1.5 Database

The database, running MySQL 5 (MySQL 2006), provides the interface between the end-host measuring system and the rest of the iPlane. All measurements of end-hosts are inserted in the database, together with timestamp information to allow the iPlane to filter out old data.

To decrease the load on the database, the updates are sent in batches every minute. This might seem unnecessary but has proven important in practice. An early version of the client sent each insert as an individual SQL INSERT and that caused the database to become overloaded. Despite upgrading the database machine to a Dual Xeon 2.2 GHz server with 4 GB of ram, update latencies of 30 seconds were not uncommon. Now the inserts are made with SQL prepared statements, followed by ADD BATCH followed by EXECUTE. This allows the database to only recompute indices once per batch, as compared to once per insert, and has decreased the load on the database to almost zero.

4.2 Distributed components

The distributed components are designed to allow the system to take advantage of multiple computers spread across the Internet. These run on measurement nodes that are often under high load. Because of this the distributed components are designed to require only a small amount of resources.

4.2.1 Modified BitTorrent client

The goal of the BitTorrent client is to make Internet hosts send data to the measurement node. To do this it must look like any other BitTorrent client to the peers in the swarm. Therefore the base code of the modified client comes from the popular BitTorrent client Transmission (Transmission 2006). This client was chosen because it is written in pure C and has an extremely small footprint. One instance of the client only consumes 40KB of memory and almost no CPU (0.1-0.3% of a 2 GHz x86 CPU). Because of the small footprint many instances of the client can run on each measurement node simultaneously. During experiments each node has been running 40 instances of the BitTorrent client without difficulty. The large number of clients running on each measurements node has the positive effect that random assignments of

torrents to measurement nodes work well. Some of the torrents might not be very active, but then the client is consuming almost no resources.

The BitTorrent client is modified in several ways to make it more suitable to use for measurements.

- First, all disk I/O code is removed, since I only want to download data, not upload anything. All incoming data packets are discarded instantly.
- Second, the client disconnects peers after it has received 1 MB of data. This number was chosen since the tools used to measure download capacity, MultiQ, only see slight improvements in accuracy when the TCP traces are longer than a couple of hundred packets.
- Third, additional code to enable communication with the shadow tracker is added, this is made to allow measurement nodes connected to the same swarm to share peer information. As soon as a peer is received from the normal tracker, the IP address and BitTorrent port of that peer is added to the shadow tracker. This allows peers to quickly build up a large active peer set, which increases the rate of which the measurement node gets optimistically unchoked by the other peers in the swarm.

4.2.2 Packet analyzer

The purpose of the packet analyzer is to monitor the incoming packets and take appropriate action for certain events. The packet analyzer uses libpcap to capture packets direct from the network stack, allowing the application to get kernel-level timestamps from when the packets arrived. The measurements are very sensitive to the accuracy of the timestamps of incoming packets, and in the highly loaded environment on which the measurements are running user-level timestamps would provide less accurate results. More about the importance of accurately recorded times of incoming packets is mentioned in the evaluation section. The downside of kernel-level timestamps is that they require root access on the host machine. This might be a problem in some environments, but in the target environment for this system, PlanetLab, it is not a problem.

To allow measurement of the upload capacity of remote hosts the timestamp of all incoming packets on each path is recorded. To save space only the packets of the largest packet size seen so far is stored. That is, suppose a host sends a train of packets with sizes (42, 42, 1500 and 1500) only the arrival-times of the two largest packets are stored. The reason for this is that only the large packets are of interest for the MultiQ algorithm. As soon as the packet analyzer sees a TCP RST (reset) packet for a certain source IP:port pair, it assumes that that host disconnected and starts the MultiQ algorithm on the packet stream for that specific IP:port pair. When it gets the result from the algorithm the information

is added to the queue of database updates that will be sent to the central database.

The packet analyzer runs an individual thread that is responsible for managing connections with the database. Since the load on the database can be high with more than 300 clients doing frequent inserts, all communication with the central database runs in a separate thread to avoid the entire application to have to wait for database traffic. This is also important since the central database is located at the University of Washington and the round trip time form UW to for example the PlanetLab nodes in Sunet is around 200 ms.

4.3 Measurements

4.3.1 Measurement of upload capacity with MultiQ

In peer-to-peer systems such as Skype and especially BitTorrent the uplink capacity is the most important link characteristic of end-hosts. For the Skype case there is no reason to try to forward a call through an intermediate host if that host does not have the bandwidth capacity to support the data stream. For normal calls the capacity required is so limited that most hosts can support it, but for video calls capacity information is essential. In the BitTorrent case it is often advantageous for a client to connect to high capacity peers since they will have a higher bandwidth / peer ratio than clients with lower capacity connections.

To obtain the upload capacity of the end hosts the method proposed in MultiQ (Katti, Katabi et al. 2004) is used. The algorithm is implemented as a part of the Click modular router project (Kohler, Morris et al. 2000) and was extracted from the source tree of that project. To fit the purposes of BitProbes the code was converted into a stand-alone application. The stand-alone MultiQ application takes a file of inter-arrival times and outputs the discovered bottlenecks.

Since the iPlane is interested in the bottleneck capacity of the path, I only store the minimum of the discovered bottlenecks in the database. All the measurement nodes are well connected, and it is assumed that the measured capacity is the last-hop capacity of the end-host measured. It would be possible to use the other discovered bottlenecks and try to map them to another link in the path, but there are several reasons why we avoid this. First there is the problem of asymmetric paths, wherein the path to the end-host, the forward path, differs from the path from the end host, the reserve path. Unfortunately, the reverse path cannot be measured with any traceroute style tool, and the bottlenecks measured are the bottlenecks on the reverse path. Second the accuracy of the measurements of the additional bottlenecks, the tight links, is much lower than the measurements of the least capacity bottleneck. The

MultiQ paper states that it finds 64% of the tight links, misses 21% and mislabels 15%.

4.3.2 Measurement of download capacity

4.3.2.1 Passive listening to BitTorrent have messages

A conservative estimate of a BitTorrent peer's download capacity is its download rate. Because of the way the BitTorrent protocol works it is possible for any peer in a swarm to calculate the download rate of any peer to which it is connected. Each time a BitTorrent peer receives a piece, it will notify the other peers to which it is connected, in order to advertise that it has the piece so other peers can request it. By monitoring the rate of which a peer sends "have" messages it is possible to know that peer's download rate.

Common BitTorrent piece sizes are 64KB, 128KB and 256KB but pieces as large as 4MB are used. The size of the blocks limits the accuracy of this technique. It is not possible to get a higher accuracy of the measurements than the piece size divided by the time windows of which the piece is assumed to have been downloaded.

4.3.2.2 TorrentProbe

TorrentProbe is a probing tool that encapsulates CapProbe like measurements in the BitTorrent handshake. The implementation is not completed, and it will therefore not be evaluated in this report.

The goal is to be able to measure the download capacity of end-hosts. This can be done by using a probing technique, for example CapProbe (Kapoor, Chen et al. 2004). As mentioned in section 2.3.5.2, CapProbe has a couple of drawbacks when used on end-hosts:

- CapProbe relies on ICMP packets, packets that often are filtered by end-hosts as well as might cause intrusion alarms.
- CapProbe has difficulties to measure asymmetric links (Chen, Sun et al. 2005), links on which the bandwidth capacity in the uplink direction differs from the capacity in the downlink direction.

Both these drawbacks can be solved by instead performing the probing with TCP SYN packets, much as in SProbe (Saroiu, Gummadi et al. 2002). Sending large amounts of TCP SYN packets might still trigger intrusion alarms though, which makes this solution unfeasible.

Instead I propose to encapsulate the probes in a BitTorrent application level handshake. The regular BitTorrent handshake is 68 bytes long, which is not long enough to be able to perform any probing. During the handshake a client can send an optional BitField message, the size of this messages is as many bits as there are pieces in the torrent + a padding to make it a real number of bytes.

The number of pieces in a torrent varies, but is often between 500 and 1,000. This corresponds to a length between 60 and 120 bytes. Fortunately it is perfectly normal for a client to send a BitTorrent “have” message for each piece in the torrent. Each have-message is 5 bytes long, which brings up the total size to between 2.56 Kbytes and 5 Kbytes. This makes it possible to send between 8 and 14 probe packets in a single handshake.

The relationship between the probing packet size and a TCP ACK packet size limit the degree of asymmetry in bandwidth capacity that this technique can handle. A TCP ACK packet is 40 bytes, so a probing packet size of 320 bytes makes it possible to correctly measure a download link that has up to 8 times higher capacity than the upload link. For most purposes this limitation is acceptable.

Because the way TCP works the measured hosts do not have to send an acknowledgement per incoming packet, it is enough to acknowledge every other packet as specified in the Delayed Acknowledgement section of RFC 2581. When sending probes it is preferable to get an acknowledgement per packet sent, especially in this case since the number of probes that can be sent is limited. Fortunately, the same RFC also requires TCP to send an ACK each time an out-of-order packet is received.

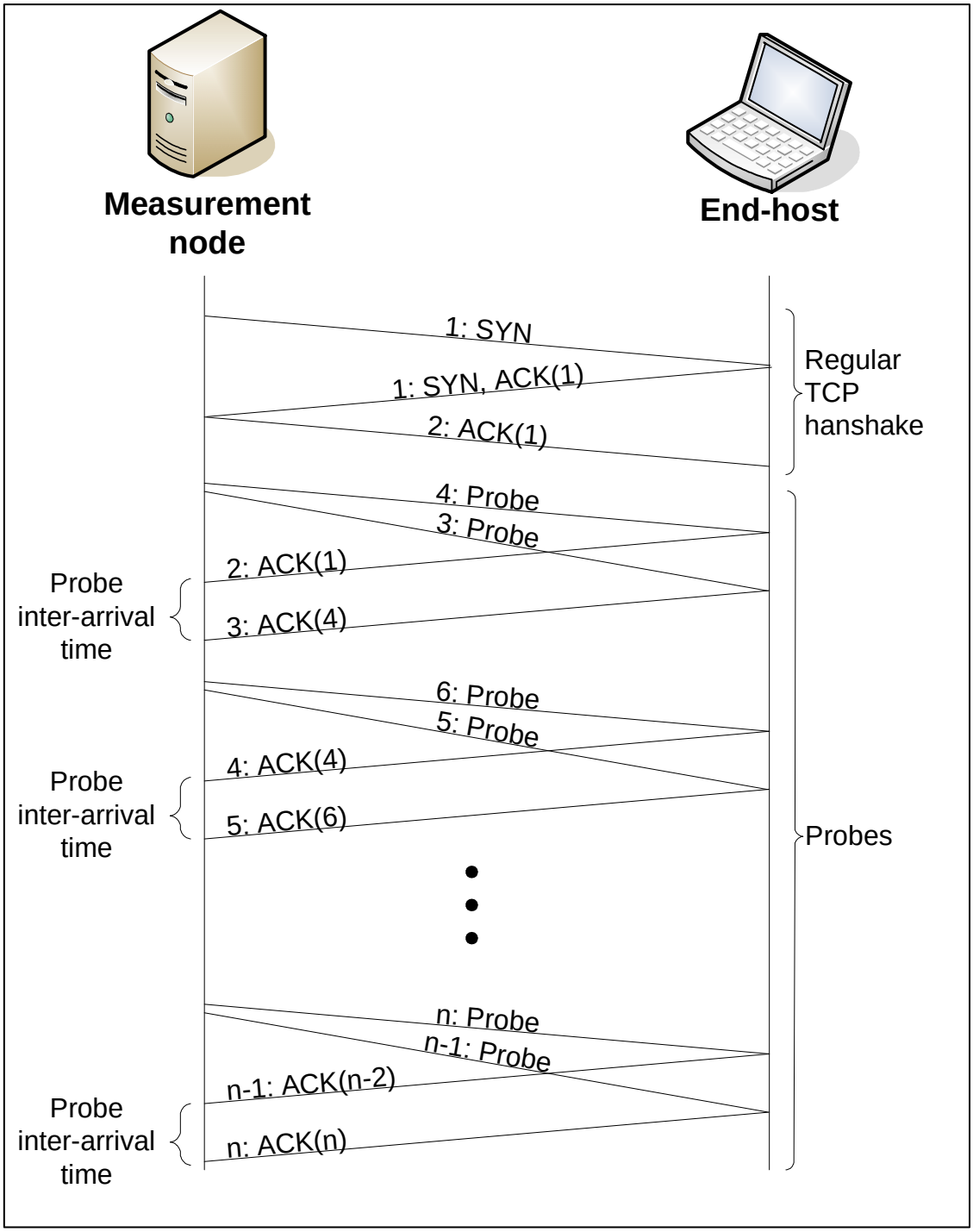

Figure 10: Operation of TorrentProbe. Probe packets are reordered to disable TCP Delayed Acknowledgements on the remote hosts. Probe pairs are sent pack to back and the inter-arrival time of acknowledgements indicate remote host download capacity.

To force the end-host to send an ACK after each probe the packets are reordered. A detail illustration of the packet order can be seen in Figure 10. As illustrated in the figure, the probing host first performs a regular TCP hand-shake. After that it delays the sending of packet three, but instead sends the fourth packet, causing the remote hosts to resend the acknowledgment of the second packet. The probing host sends packet three immediately after it sends packet four, in the hope that both packets will traverse the network back-to-back. When the remote host receives packet three, it will send an acknowledgement to the probing host that it now successfully has received all packet up to four. This way the remote host will send an acknowledgment after each packet, despite that TCP Delayed Acknowledgments is enabled. It is important that the actual payload of each packet remains the same, the only change is in which order the packets are sent.

After the probes have been sent the data is post-processed. This is done the same way as in (Kapoor, Chen et al. 2004) using minimum delay packets. If a packet has been queued, the queuing can introduce error in the measurements. To avoid this, only the packets which have traversed the network in the least time is used. The accuracy of TorrentProbe should be comparable to the accuracy of CapProbe.

4.3.3 Topology mapping

4.3.3.1 Topology mapping with tracreroute

The standard implementation of traceroute sends UDP packets with increasing TTL. This works well in most circumstances. However in the case of large scale measurements of end hosts, this unfortunately have some negative properties.

- Most firewalls are configured to not send the ICMP port unreachable reply when they receive a UDP packet to an unused port. This has the effect that it is difficult to know when the traceroute has reached the destination.
- Traceroutes from several vantage points to the same end host can trigger alarms in firewalls. This might result in angry emails being sent to the researchers responsible. During the initial run on PlanetLab this happened and traceroutes were disabled for the rest of that run.
- To make sure that a host only is sent one traceroute requires that information is shared among all measurement nodes. For example a central server database could be queried before each traceroute is initiated. This would solve the previously mentioned problem, but with the downside that only one route to every given destination is discovered.

To cope with all problems described above the following solution is used. A modified version of tcptraceroute (Toren 2006) is used to send the traceroute in the already existing TCP stream to a given host. The tcptraceroute application was modified in such a way that it allows the TCP sequence and TCP acknowledgement numbers to be specified from the command line. The traceroute is sent as ACK packets with the same source IP, source port, destination IP, destination port, and proper sequence numbers as a previously sent packet. These trace packets are interpreted as duplicated ACKs and are therefore ignored at the target host. The result is a traceroute that is completely transparent to the target. This technique is the proposed method for mapping the Internet with IP record route in (Sherwood and Spring 2006) and during their experiments they performed traces to over 22,000 end-hosts without receiving a single complaint.

4.3.3.2 Topology mapping with IP record route

Recent work in (Sherwood and Spring 2006) demonstrated that combining normal traceroutes with probes that use the IP Record Route (RR) option provides an even better understanding of the network topology. The IP RR option has rarely been used in network research since it was assumed that it was disabled by most routers. The RR option also has the limitation that it only allows the first 9 hops to be discovered. Sherwood and Spring show that the first assumption is wrong, IP RR is enabled in most routers on the Internet. They also show that when performing the probes from PlanetLab 89% of all hosts the probed can be reached in 9 hops or less.

The IP RR has the advantage that it captures the IP of the outgoing interface on the intermediate routers. Traceroute usually captures the incoming interface. The combination of the two giving a more complete topology map can be seen in Figure 11. This makes it easier to map discovered IPs to physical routers which is useful for services such as the iPlane.

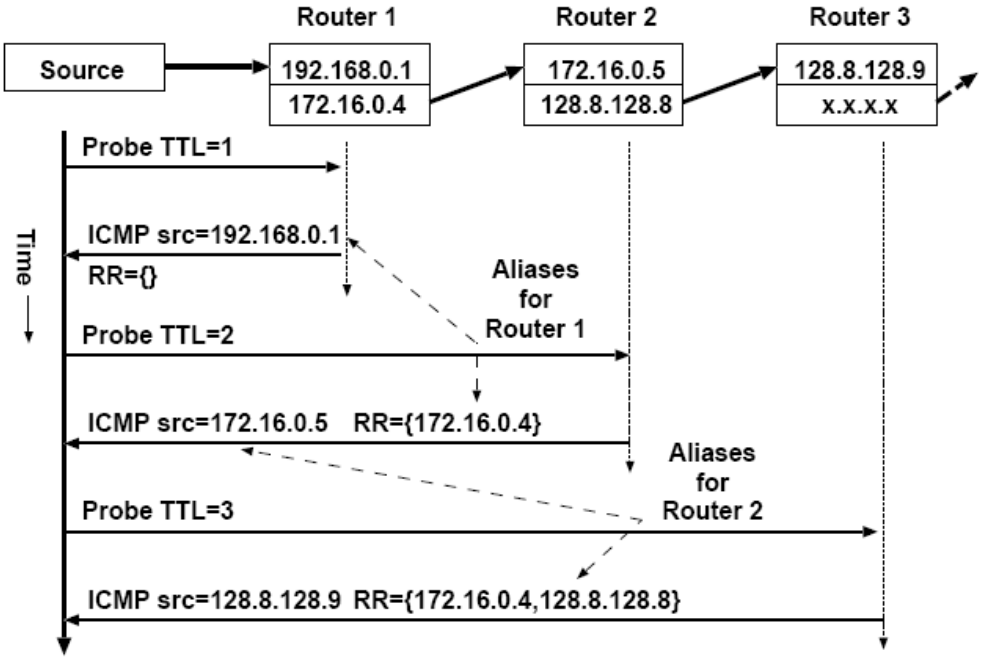

Figure 11: Internet mapping with both traceroute and IP RR. Both incoming and outgoing IP addresses of intermediate routers are discovered.
Figure source: (Sherwood and Spring 2006)

5 Evaluation

This section first describes how the system was evaluated for both potential coverage and for the quality of the results. Then the results that were gathered are presented, results that quantify the potential of the method. Some of the statistics about the data collected is also presented.

5.1 Initial evaluation on PlanetLab

The system was initially evaluated to verify the feasibility of the method. In terms of coverage and quality of results an initial evaluation were done on PlanetLab. A Java version of the client was deployed on 367 PlanetLab nodes. In this setup a crawler was run every hour that parsed well-known public websites for .torrent files. From these the 120 most popular swarms were chosen. The reason for why only 120 swarms were chosen, despite that over 360 measuring nodes were available, was to provide multiple measurement vantage points in each swarm. The number of measurement nodes designated to a swarm was proportional to the number of peers participating in it. Each measurement node was running only one instance of the modified BitTorrent client.

The main difference between the first version and the second version of the client is that the first one was written in Java. While this had the benefit that it was easy to develop, the main disadvantage was that it consumed much memory. Memory is a very scarce resource on PlanetLab and therefore it was only possible to run one instance on the client per PlanetLab node. The second difference was that this client kept a small cache of recently received pieces in memory. Data in this cache was offered for upload so that the client would not have to depend only on peers optimistically unchoking it to initiate data transfer. This difference had the effect that each instance of the old client was collecting roughly 5 times more measurements than the current client. On the other hand the new client is only using 0.1% of the amount of memory of the old client.

5.1.1 Coverage

To verify that it would be possible to cover a large number of Internet hosts, coverage data was examined for 48 hours. The rate at which measurements were gathered is summarized in Figure 12. During this 48 hour period, the measurement nodes connected to 301,595 distinct IP addresses. The number of unique IPs for which upload bandwidth capacity estimates was gathered were 70,428. Connections were initiated with IP addresses in 3,591 distinct ASs and

19, 639 distinct BGP prefixes. The run covered end-hosts in 160 different countries. These initial results showed that opportunistic measurements obtained from BitTorrent makes it possible to measure a large number of end-hosts and that the hosts measured come from many different parts of the Internet.

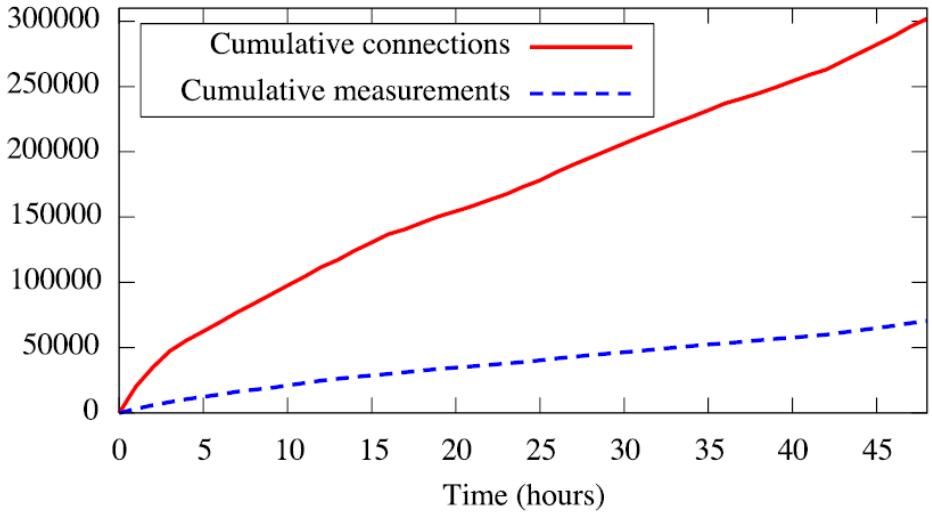

Figure 12: Rate of increasing connections and measurements during the 48 hour run on PlanetLab.

5.1.2 Clustering of results

Even if the coverage of end hosts is significant, it is far from complete. Instead, it is assumed that hosts that are close in IP space also will have similar link characteristics. By clustering hosts based on IP prefixes it is possible to generalize results for prefixes to which there is only a few measurements. The validity of this assumption is explored in Figure 13. For every /24 prefix in which measurements was performed to multiple end-hosts from the same vantage point, a ratio was computed of the maximum to the minimum measured bandwidth capacity. For 70% of /24 prefixes, the capacities measured differ by less than 25%. During the 48 hour measurement period, a total of 61, 294 /24 prefixes were covered, which are representative of measurements to over 15 million end-hosts.

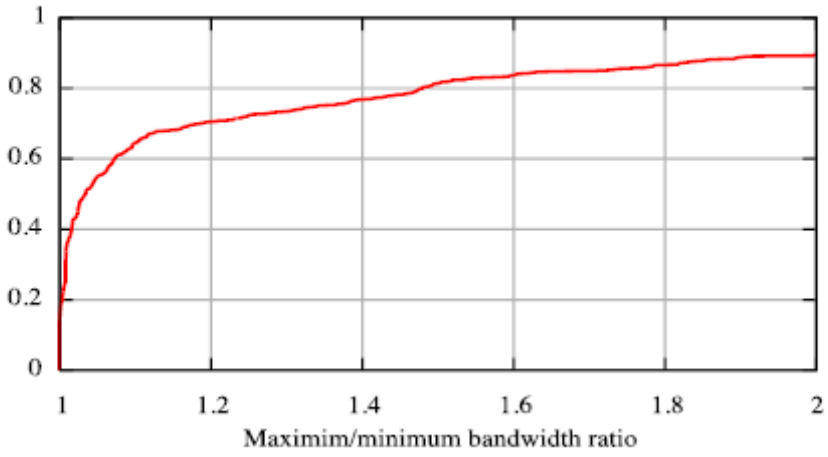

Figure 13: Maximum / minimum quota of discovered upload capacity of hosts located within the same /24 subnet.

5.2 Results from UW machines

The results presented in this section were collected by 8 machines running at the University of Washington. Each of these was running 40 instances of the modified BitTorrent client. The results presented were collected between September 2nd 2006 and September 9th 2006. Two popular web-sites[1] providing links to .torrent files was crawled every 12 hours, and the 100 most popular torrents discovered was assigned to the measurements nodes.

5.2.1 Rate of connections

The rate of which new hosts are discovered is an important measure of the success of the technique described in this report. Firstly it gives an indication of with what speed new hosts are discovered. Secondly it indicates whether the system begins to deplete the pool of clients using torrents hosted on the web-sites crawled.

The rate of new end-hosts in presented in Figure 14. Since the rate of new connections shows no sign of flattening out, I conclude that the pool of end-hosts using torrents from the two web-sites is significantly larger than the number that can be covered by this system in a week. It is therefore possible to increase the rate of newly discovered end-hosts by adding more measurement nodes to the BitProbes system. Since BitProbes is designed to eventually run

[1] The web-sites crawled were http://www.mininova.org and http://www.thepiratebay.org.

on PlanetLab, it will be interesting to see how an increase from 8 to over 350 measurements nodes will increase the rate at which end-hosts are discovered.

During the week analyzed in this report, the measurement nodes initiated connections to, or were connected from, a total of close to 500,000 unique IP addresses, each connection providing the opportunity to perform a wide range of measurements.

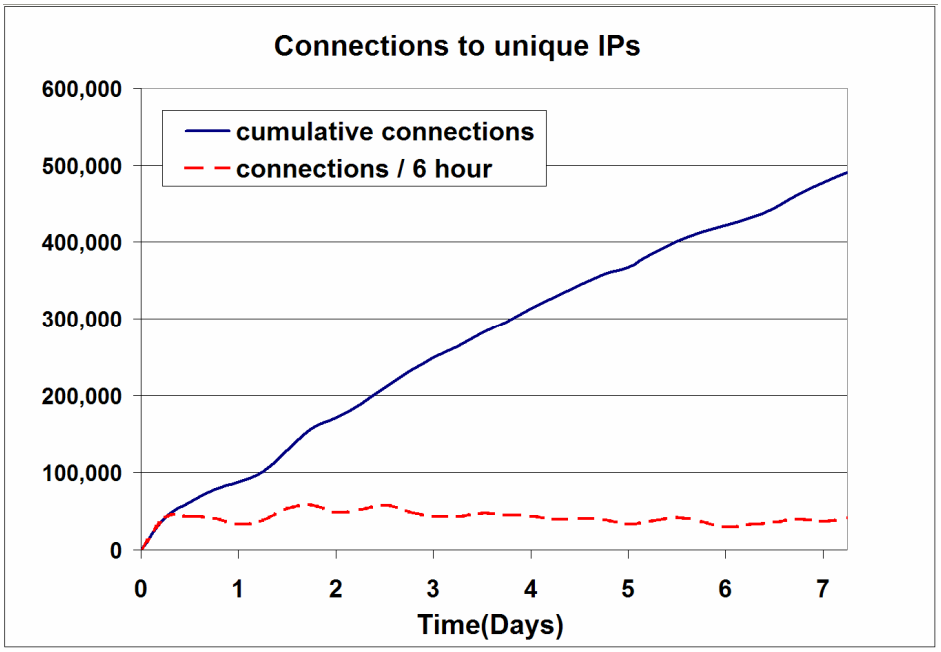

Figure 14: Rate of increasing connections to unique IP addresses

As data for longer time periods become available I suspect that the system eventually will see a decrease in the rate of which new end-hosts are discovered. When this happens it will be necessary to find new web-pages providing .torrent files. Currently both web-sites crawled are European, but as new web-sites from other continents are added, it will be possible to extend the period for which new hosts are discovered at a high rate. It should be noted that the current the web-sites were chosen because both of them are immensely popular and both serve content in several languages, causing them to have to global reach.

5.2.2 Rate of capacity measurements

Measurement of the upload bandwidth capacity of remote hosts is the most difficult measurement performed by this system. To be able to infer the upload capacity of end-hosts, a TCP flow with a significant number of packets has to be sent by the end-host to the measurement node. The flow is then analyzed by MultiQ resulting in a measurement of the bandwidth capacity. All flows with more than 100 full-size packets are analyzed, and during the week covered in

this report, significant flows from 176,487 unique IP addresses were received. In 96,080 of these (54%), MultiQ is able to tell the upload capacity. The rate which new end-hosts are successfully measured is presented in Figure 15.

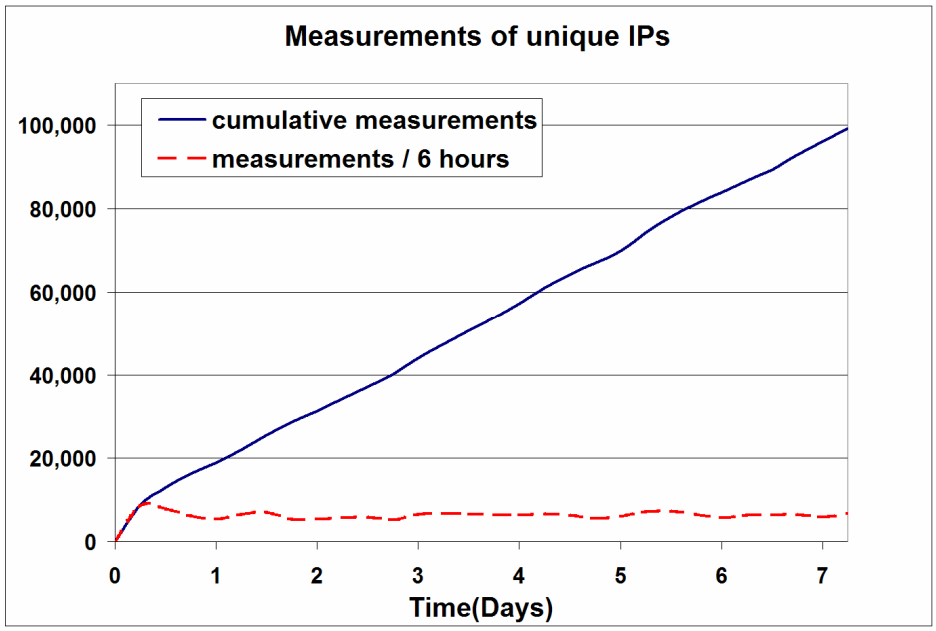

Figure 15: Rate of increasing measurements of unique IP addresses

As can be seen in Figure 15 the number of measurements is increasing steadily during the 7 days analyzed. The number of cumulative measurements is the number of unique IP addresses that has been measured. As seen the system manages to successfully measure the upload capacity of roughly 100,000 unique IPs in a week, which means that roughly 1 in 5 discovered end-hosts upload capacity is measured. As the system runs the number of measurement should get closer and closer to the number of discovered hosts, since the rate of which new hosts are discovered can be assumed to decrease over time.

5.2.3 Network coverage

In terms of network coverage, the results show a trend of depleting the source. During the 7 day period, hosts in 21,032 BGP unique prefixes were connected to. A BGP prefix corresponds to an entry in the global BGP routing table. As seen in Figure 16 the number of prefixes discovered flattens out significantly. During the 24 first hours of operation almost half of the total number of prefixes was discovered. This shows that even though the swarms joined have a large number of end-hosts associated with them, most of these hosts are located in a limited section of the Internet. The reason for this could be that the web-sites used to collect .torrent files focus on types of content that are of interest only in some parts of the world. Finding web-sites that either have a

more diverse clientele, or a clientele that is disjoint from the other websites crawled will hopefully mitigate this problem.

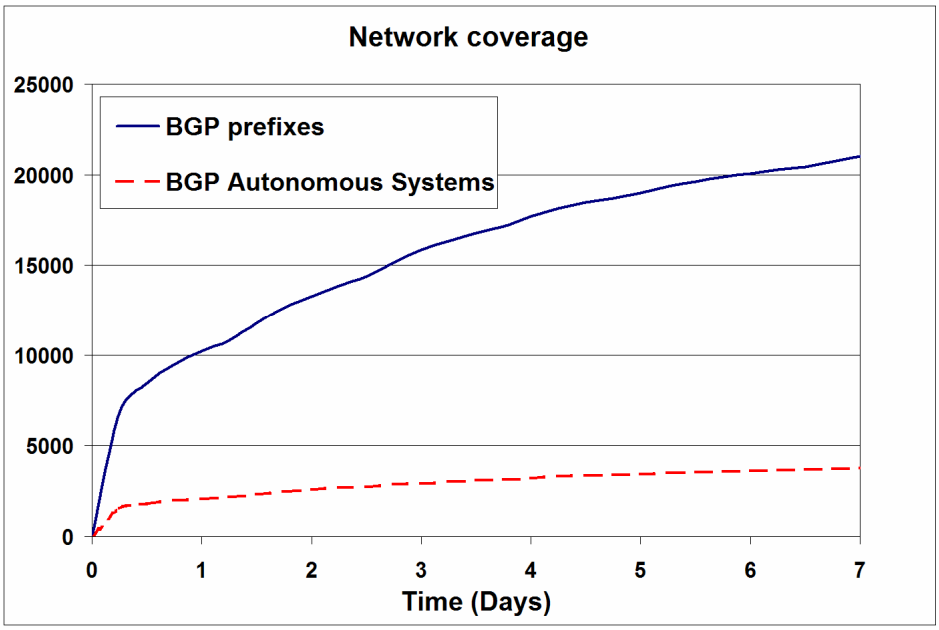

Figure 16: Network coverage

The coverage of BGP Autonomous Systems (AS) shows similar properties as the coverage of BGP prefixes. During 7 days a total of 3763 ASs were connected to, with close to half of them being discovered in the first 12 hours of operation. The method to increase the number of ASs covered is the same as with BGP prefixes. Since one AS often corresponds to one Internet Service Provider (ISP), the key in getting higher coverage is to find sources with material that either is interesting globally, or finding more sources that target different markets than the sources currently used. It should also be noted that the activity and size of different ASs varies, with the effect that some ASs will be difficult to reach with this method. On the other hand will the ASs measured be the biggest and most active ones, causing them to be the most valuable to measure.

5.2.4 Geographic coverage

To see what level of geographic coverage that is possible with BitTorrent measurements, I used a tool from CAIDA (CAIDA 2006) used to map IP addresses to countries. During the 7 days analyzed in this report, IP addresses from 165 different countries were connected to. A chart of the 22 countries that received the most connections can be seen in Figure 17.

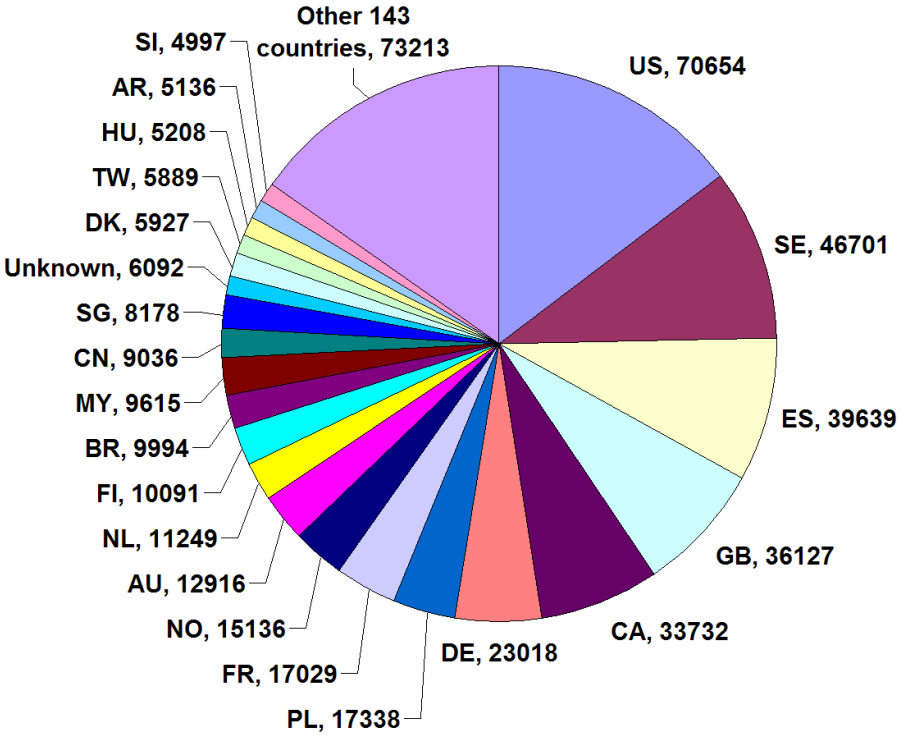

Figure 17: Number of connections to the top 22 countries

The figure shows that some countries are better covered than others, suggesting that the source web-pages are better know in some countries. Surprising is for example the high number of measurements of Swedish hosts, despite the small population of that country. I attribute this fact to two causes; the websites used have been mentioned in mainstream media in Sweden, making it know by the general population. The other cause is that the percentage of broadband users differs in different countries. Additionally it can be noted the geographic coverage is impressive, hosts from 165 countries spreading the entire globe has been connected to.

5.3 Capacity distribution

To know the distribution of upload capacities of users of peer-to-peer systems is important when designing new peer-to-peer systems. Since the upload capacities of a large number of end-hosts were measured, I take the opportunity to present some of the results in this report. Since the measurements are of a large number of hosts running BitTorrent, is can be assumed that this distribution is fairly representative for users of BitTorrent and for users of peer-to-peer swarming systems in general. The upload capacity distribution measured is presented in Figure 18.

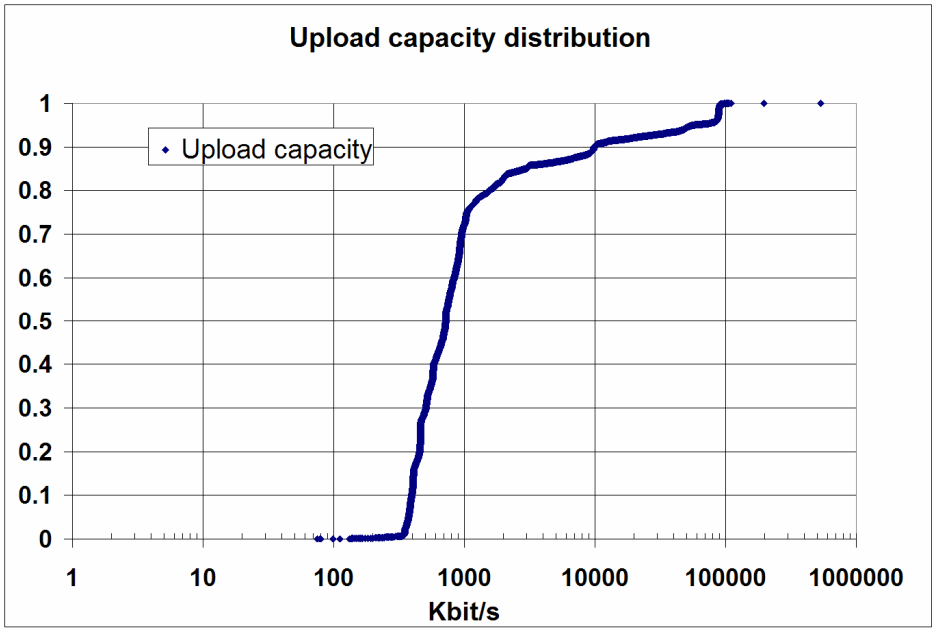

Figure 18: Upload capacity distribution of BitTorrent users

As seen a significant majority (70 %) of the hosts have an upload capacity between 350 Kbit/s and 1 Mbit/s. Only 10% of hosts have an upload capacity of 10 Mbit/s or more. What also can be noted is that the 5% of hosts with bandwidth capacities between 55 Mbit/s and 110 Mbits/s, contribute with 64 % of the available resources in the system, suggesting that successfully incorporating the resources of the high capacity clients is a important trait of efficient peer-to-peer systems. Further analysis of the data is needed before any conclusions about the efficiency of BitTorrent can be presented, and that is not within the scope of this report. The data will be made public, making it possible for other researchers to answer questions such as; what is the correlation between a BitTorrent peers upload capacity and download rate.

5.4 Validation of capacity measurements

The bandwidth capacity measurements rely on inter-arrival times observed between data packets in the connections I maintain with BitTorrent peers. As previously mentioned the MultiQ technique is used to infer end-to-end bottleneck bandwidth capacity from these inter-arrival times. Although the performance of MultiQ presented in previous studies is encouraging, with 85 % of measurements based on data packets within 10 % of the true bottleneck capacity, the properties of PlanetLab hosts might introduce additional problem. The system is designed to eventually run on PlanetLab and that environment can be extra challenging for Internet measurements. To see if the purposed technique would work in the PlanetLab environment an

additional experiment was set up. The results were then compared to results made by S³ (Lee, Sharma et al. 2005).

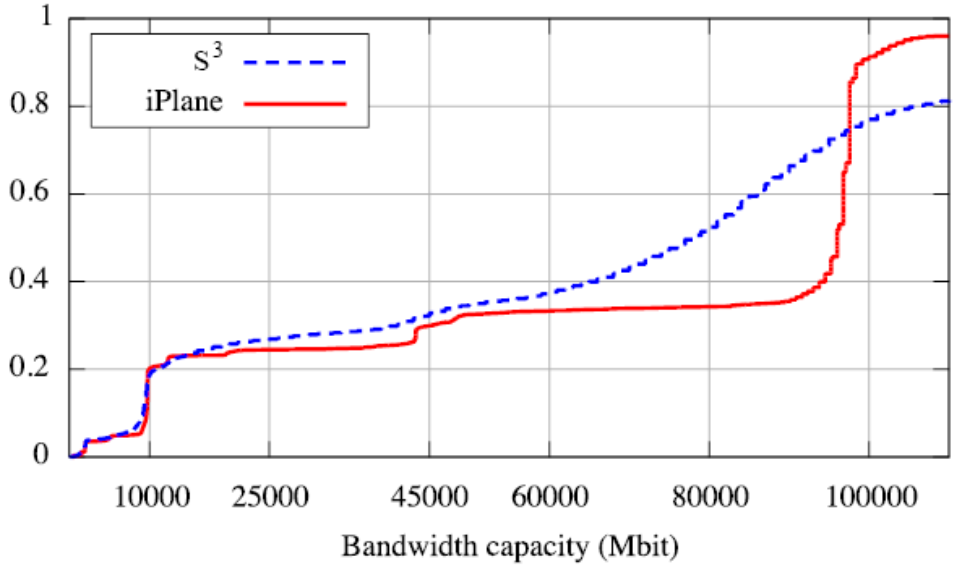

Figure 19: Bandwidth capacity distribution of 10,879 paths between PlanetLab nodes as measured by the iPlane and by S³.

A test torrent was set up and 357 PlanetLab nodes joined the torrent. The paths between the PlanetLab nodes were then opportunistically measured using the same technique that is used to measure end hosts. The experiment resulted in 10,879 paths in common with measurements made by S³ on that same day. Figure 19 compares the bandwidth capacities measured by the two methods. The measurements made by the iPlane closely match those of S³ for capacities less than 10 Mbps. At higher bandwidth capacities, they are only roughly correlated. This difference is probably from S³ using Pathrate that, as noted previously in the paper can be a very accurate tool. Unfortunately for Pathrate to work accurately, the CPU on the system should be idle, preferably Pathrate should be the only running application (Dovrolis 2006). The reason for this is that Pathrate uses user-level timestamps, which will be inaccurate in an environment under heavy load. On PlanetLab the load on the system is high, load averages of above 10 are common. The inter-arrival times recorded by BitProbes come from kernel-level timestamps, which are more accurate in environments with high load. When measuring high bandwidth path accurate time-stamping is crucial. A 1500 byte packet will have a transfer time of only 0.8 ms. On paths with capacity higher than 10 Mbit/s, the MultiQ technique records many paths as either around 45 Mbit/s (T3) or around 100 Mbit/s although with a small systematic under-estimation. Since both these are common capacities for large corporations and institutions these results are plausible for the environment measured.

5.5 Results summary

The results collected by BitProbes show that the use of BitTorrent to attract traffic from end-hosts is a successful method. Using 8 servers for one week has yielded connections from close to half a million end-hosts from 165 countries. The coverage in terms of Internet geography is also impressive, with hosts from over 20,000 different entries in the global BGP routing table and over 4,000 unique Autonomous Systems. All these connections provide the opportunity to unobtrusively measure the remote host. The most challenging measurement performed by BitProbes is measurements of the end-hosts upload capacity. This measurement requires the end-host to send a significant flow of maximum size TCP packet to the measuring node. BitProbes has during the week analyzed in the report successfully inferred the upload capacity of close to 100,000 different IP addresses.

The results also show that hosts that are close to each other in IP address space also, with high probability, have similar connection characteristics. By analyzing the bandwidth distribution of hosts within the same /24 IP subnet, it can be concluded that measurements of one host in a /24 subnet is a useful for prediction the bandwidth capacity of other hosts is the same subnet. In one week, BitProbes successfully measured the upload capacity of hosts in 69,026 different /24 subnets, corresponding to over 17 million hosts.

6 Related work

In this section I investigate previous systems that have taken an opportunistic approach to measurements, focusing on advantages and disadvantages of the methods used in the systems described. I note that, while the systems have been successful for their purposes, none of them is able to perform measurements to such a large number of uncooperative end-hosts as BitProbes.

6.1 PlanetSeer

PlanetSeer (Zhang, Zhang et al. 2004) monitor users of the CoDeeN (Wang, Park et al. 2004) CDN, to detect failures on the Internet. PlanetSeer monitors existing TCP connections and notes when an unexpected disconnect event occurs. When noticing a disconnect PlanetSeer tries to probe the remote node from different vantage points, and if a node is reachable from some vantage points but not others, it is noted as a route abnormality.

The coverage of PlanetSeer is on the order of 9 to 12 thousand clients peer day, with no information about what fraction of these that are clients that are new to the system. There is also no information in the paper whether all these clients come from unique IP addresses or if many of them originate from the same source, but (Sherwood and Spring 2006) note that when monitoring the CoDeeN network for a week, 22,428 unique IP addresses were seen. The system described in this report initiates around 500,000 connections to unique IP addresses per week. The measurement infrastructure described in this paper covers more than one magnitude more end-hosts than PlanetSeer.

6.2 TCP sidecar

TCP-sidecar (Sherwood and Spring 2006) performs traceroutes to end-hosts to be able to construct an accurate router-level topology of the Internet. Since traceroutes often cause IDS alarms, the traces are embedded into existing TCP connections. TCP sidecar use TCP connections from two sources:

- Passive monitoring of CoDeeN, resulting in measurements to 22,428 unique end-hosts per week.
- Downloading robots.txt from web-servers. This method yielded 166,745 unique IP addresses, although this method obviously can cover a significant higher number.

Downloading content of web-servers is a technique that is promising when the goal is to record topology. For measurements of bandwidth capacity there are some drawbacks with this method. The robots.txt file usually fits in one IP packet, making analysis of the TCP stream to infer bandwidth capacity impossible. Instead it would be necessary to download larger or multiple files from each web-server. This is possible, and putting the PacketAnalyser in front of a web crawler would definitely yield a large number of measurements, making it reasonable to investigate this more thoroughly. In this report however, the investigation is left for future work. An other drawback of this method is that web-servers generally are better connected than ordinary end-hosts, causing any statistics, for example bandwidth capacity distribution generated with this method to be biased.

6.3 Spurious Traffic

By looking at unconventional sources of network traffic, (Casado, Garfinkel et al. 2005) is able to achieve great coverage for their measurements. The sources used include SPAM traffic, traffic from worms and automated scans, yielding a potential coverage of several hundred thousand IPs. For example the authors state that CAIDA received probes from 359,000 infected servers during the first Code Red outbreak. The authors use a similar suit of tools to perform their measurements, for example is MultiQ used to discover the bandwidth capacity of the end-hosts. Although this technique is promising in terms of coverage, it has some problems associated with it.

- Traffic flow length. The amount of traffic sent to each host in for example an IP scan or a worm attack is very limited. This makes it difficult to use any of the existing passive tools to infer path properties.
- Biased results. The hosts infected by worms are often unrepresentative for the Internet as a whole. For example, one of the most severe worm attacks, the SQL slammer attack, only infected servers running Microsoft SQL server. This causes any statistics derived from the measurements to be difficult to use in other studies.

Even if measurements of BitTorrent users also show a biased view of the Internet, the bandwidth capacity of peer-to-peer users is more interesting for research than the bandwidth capacity of servers running Microsoft SQL.

6.4 Planet Scale Software Updates

In a paper from Microsoft Research (Gkantsidis, Karagiannis et al. 2006) geared towards the behavior of software updates, the authors had access to packet trace data from Windows Update servers. Although the paper did not focus on measuring Internet path properties they were able to present an

impressive coverage of Internet hosts. During the year during which the traces were collected, over 150 Million unique IP addresses were seen. Analyzing packet traces from update services certainly has potential to achieve great coverage, with the downside that the amount of traffic flowing from the end-hosts to the servers is small, making measurements of end-host upload capacity difficult. On the other hand Microsoft has control over the application running on the client machines, making it possible for them to alter the behavior of the end-hosts. The paper discusses how a peer-to-peer distribution strategy would decrease the load on the update servers, and if that strategy is implemented it would be possible to incorporate measurement code into the update client allowing for better peer selection. In the paper the authors discuss grouping clients into distribution groups depending on AS. Since bandwidth capacity of end-hosts is very divers, varying between 36 Kbit/s dialup to 100 or 1000 Mbit/s Ethernet it would improve download performance to not only group clients by AS but also to make sure that high capacity nodes are used more efficiently.

7 Conclusion

In this report I have presented BitProbes, a system performing measurements to Internet end-hosts. BitProbes attracts connections from close to half a million unique end-hosts per week, connections that are used to opportunistically infer link latency, Internet topology and upload bandwidth capacity of the hosts. All these measurements are performed unobtrusively, no probes that might trigger IDS alarms are sent. I also present the design of a tool that utilizes the BitTorrent handshake and TCP packet reordering to unobtrusively measure the download capacity of end-hosts. The number of connections covered by BitProbes is more than an order of magnitude higher than what previous systems relying on opportunistic measurements have attracted, suggesting that the method used by BitProbes is successful.

BitProbes attract traffic by connecting to swarms of the popular file-distribution application BitTorrent. Results collected by BitProbes, show that the pool of end-hosts that can be measured using this technique is significantly larger than the close to 500,000 connected to during the week analyzed in this report. It also suggests that an even higher rate of measurements can be achieved if the number of measurement nodes is increased.

The measurements collected by BitProbes are fed to the iPlane, allowing the iPlane to predict link performance not only for Internet core routes, but also for the link between two arbitrary end-hosts. By making it possible for applications to a priori know the properties of other hosts, the applications can choose which hosts to connect to based on metrics such as link latency or capacity. With this information it has been shown that the iPlane is able to improve performance of CDNs, voice-over-IP applications and BitTorrent.

BitProbes, as a system, has exceeded expectations both in terms of rate of measurements and in terms of discovered hosts. It can be concluded that utilizing users of BitTorrent as targets for end-host measurements is an excellent method to unobtrusively perform measurements of large numbers of end-hosts.

Glossary

AS **Autonomous System**

A collection of networks and routers under control by the same entity sharing the same routing policy

BGP **Border Gateway Protocol**

Protocol used by Internet Service Providers to communicate routing information

CDN **Content Distribution Network**

A system for large-scale content distribution over the Internet

FIFO **First In First Out**

Queuing policy used by many Internet routers

ICMP **Internet Control Message Protocol**

Protocol used by Internet hosts to send control messages

IDS **Intrusion Detection System**

A system monitoring network activity to detect intrusion attempts

IP **Internet Protocol**

The protocol handling addressing on the Internet

IP RR **IP Record Route**

Option in the IP header asking routers to append the interface which packets exit to the packet header

ISP **Internet Service Provider**

Company or organization providing Internet access to end-users

LRF **Local Rarest First**

Piece selection strategy used by BitTorrent to make sure that the rarest pieces are downloaded first.

MTU **Maximum Transmission Unit**

The largest packet size that can travel over a link without getting fragmented

RBPP **Receiver Based Packet Pair**

NetTimer mode used when both sender and receiver traces are available.

RFC **Request For Comment**

Document describing Internet standards and drafts

ROPP **Receiver Only Packet Pair**

NetTimer mode used when only receiver traces are available

RTT **Round Trip Time**

The time it takes for a packet to travel to a remote host plus the time it takes for the response to travel back to the source

SBPP **Sender Based Packet Pair**

NetTimer mode used when only sender traces are available

TCP **Transmission Control Protocol**

Reliable byte stream protocol used on the Internet

TCP-ACK **TCP Acknowledgement**

TCP packet type sent to acknowledge that data has been received successfully

TCP-RST **TCP-Reset**

TCP packet type used to terminate a TCP connection

TCP-SYN **TCP-Synchronize**

TCP packet type used to initiate a TCP connection

TFT **Tit-For-Tat**

Policy for trading resources in which the default state is trusted

TTL **Time To Live**

Field in the IP header specifying the number of hops a packet can travel before it is discarded

UDP **User Datagram Protocol**

Unreliable datagram based protocol used on the Internet

Table of figures

References

Akamai. (2006). "Akamai: The Trusted Choice for Online Buisniess." Retrieved Sept 8th, 2006, from http://www.akamai.com/.

Allman, M., V. Paxson and W. Stevens (1999). "RFC 2581: TCP Congestion Control."

Andersen, D., H. Balakrishnan, F. Kaashoek and R. Morris (2001). "Resilient overlay networks." *Proceedings of the eighteenth ACM symposium on Operating systems principles*, Banff, Alberta, Canada, ACM Press.

Bellovin, S. M. (1992). "A Best-Case Network Performance Model." Retrieved Sept 8th, 2006, from ftp://coombs.anu.edu.au/pub/net/papers/netmeas.ps.

Bharambe, A., C. Herley and V. Padmanabhan (2006). "Analyzing and Improving a BitTorrent Network's Performance Mechanisms." *Proceedings of InfoComm 2006*, Orlando, FL.

bittorrent.org. (2006). "What is BitTorrent?" Retrieved Sept 8th, 2006, from http://www.bittorrent.org/introduction.html.

CAIDA. (2006). "CAIDA : tools." Retrieved Sept 18th, 2006, from http://www.caida.org/tools/.

Casado, M., T. Garfinkel, W. Cui, V. Paxson and S. Savage (2005). "Opportunistic Measurement: Extracting Insight from Spurious Traffic." *Proceedings of the 4th ACM Workshop on Hot Topics in Networks*, College Park, MD, USA.

Chen, L.-J., T. Sun, G. Yang, M. Y. Sanadidi and M. Gerla (2005). "End-to-end asymmetric link capacity estimation." *In IFIP Networking 2005*.

Chen, Y., D. Bindel, H. Song and R. H. Katz (2004). "An algebraic approach to practical and scalable overlay network monitoring." *In SIGCOMM, 2004*.

Cohen, B. (2002). "BitTorrent Protocol Specifications v1.0." Retrieved Sept 8th, 2006, from http://www.bittorrent.org/protocol.html.

Cohen, B. (2003). "Incentives build robustness in BitTorrent." *In Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems 2003*.

Crawford, K. (2005). "Hollywood looks to BitTorrent as ally." Retrieved Sept 20th, 2006, from http://money.cnn.com/2005/04/29/technology/movie_piracy/.

Dabek, F., R. Cox, F. Kaashoek and R. Morris (2004). "Vivaldi: a decentralized network coordinate system." *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, Portland, Oregon, USA, ACM Press.

Dovrolis, C. (2006). "Pathrate tutorial." Retrieved Sept 19th, 2006, from http://www-static.cc.gatech.edu/fac/Constantinos.Dovrolis/pathrate_tutorial.html.

Dovrolis, C., P. Ramanathan and D. Moore (2004). "Packet-dispersion techniques and a capacity-estimation methodology." *IEEE/ACM Transactions on Networking* **12**(6): 963-977.

Freedman, M. J., E. Freudenthal and D. Mazières (2004). "Democratizing Content Publication with Coral." *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA.

Gkantsidis, C., T. Karagiannis, P. Rodriguez and M. Vojnovic (2006). "Planet scale software updates." *SIGCOMM Computer Communication Review* **36**(4): 423-434.

ISI-USC (1981). "RFC 791 - Internet Protocol."

Jacobson, V. (1988). "traceroute.c." Retrieved Sept. 18th, 2006, from http://rpm.rutgers.edu/repository/solaris/SOURCES/traceroute.tar.Z.

Jacobson, V. (1995). "Congestion avoidance and control." *SIGCOMM Comput. Commun. Rev. 0146-4833* **25**(1): 157-187.

Jain, M. and C. Dovrolis (2003). "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput." *IEEE/ACM Trans. Netw.* **11**(4): 537-549.

Jitendra, P., F. Victor, T. Don and K. Jim (1998). "Modeling TCP throughput: a simple model and its empirical validation." *ACM SIGCOMM 1998*, Vancouver, British Columbia, Canada, ACM Press.

Kapoor, R., L.-J. Chen, L. Lao, M. Gerla and M. Y. Sanadidi (2004). "CapProbe: A Simple and Accurate Capacity Estimation Technique." *ACM SIGCOMM 2004*.

Katti, S., D. Katabi, C. Blake, E. Kohler and J. Strauss (2004). "MultiQ: automated detection of multiple bottleneck capacities along a path." *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Sicily, Italy, ACM Press.

Kohler, E., R. Morris, B. Chen, J. Jannotti and M. F. Kaashoek (2000). "The Click modular router." *ACM Transactions on Computer Systems* **18**(3): 263-297.

Lai, K. and M. Baker (1999). "Measuring bandwidth." *In Proceedings of IEEE INFOCOM 1999*, New York, NY.

Lai, K. and M. Baker (2001). "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth." *Proceedings of the USENIX Symposium on Internet Technologies and Systems 2001*.

Lee, S.-J., P. Sharma, S. Banerjee, S. Basu and R. Fonseca (2005). "Measuring Bandwidth between PlanetLab Nodes." *Passive and Active Measurement Workshop '05*, Boston, MA, USA.

Madhyastha, H. V., T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy and A. Venkataramani (2006). "iPlane: An Information Plane for Distributed Services." *To appear in the 7th USENIX Symposium on Operating Systems Design and Implementation*.

MySQL. (2006). "MySQL: The World's Most Popular Open Source Database " Retrieved Sept 18:th, 2006, from http://www.mysql.org/.

Parker, A. (2004). "The True Picture of P2P Filesharing." Retrieved Sept 8th, 2006, from http://www.cachelogic.com/home/pages/studies/2004_01.php.

Paxson, V. E. (1997). "Measurements and Analysis of End-to-End Internet Dynamics." *University of California at Berkeley*. **PhD Dissertation**.

PlanetLab. (2006). "PlanetLab: Home." Retrieved Sept 8th, 2006, from https://www.planet-lab.org/.

Pouwelse, J., P. Garbacki, D. Epema and H. Sips (2005). "A measurement study of the bittorrent peer-to-peer file-sharing system." *The 4th International Workshop on Peer-To-Peer Systems*.

Saroiu, S., P. Gummadi and S. Gribble. (2002). "A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments." *Proceedings of IEEE INFOCOM 2002*.

Scott, D. (1992). "Multivariate Density Estimation: Theory, Practice and Visualization." New York, NY, *Addison Wesley*.

Sherwood, R. and N. Spring (2006). "Touring the Internet in a TCP Sidecar." *To appear in Internet Measurement Conference 2006*.

Skype. (2006). "Skype -The whole world can talk for free." Retrieved Sept 8th, 2006, from http://www.skype.com/.

Spring, N., L. Peterson, A. Bavier and V. Pai (2006). "Using PlanetLab for network research: myths, realities, and best practices." *SIGOPS Operating Systems Review* **40**(1): 17-24.

Strauss, J., D. Katabi and F. Kaashoek (2003). "A measurement study of available bandwidth estimation tools." *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, Miami Beach, FL, USA, ACM Press.

Toren, M. C. (2006). "tcptraceroute." Retrieved Sept 18th, 2006, from http://michael.toren.net/code/tcptraceroute/.

Transmission. (2006). "Transmission homepage." from http://transmission.m0k.org/.

Wang, L., K. Park, R. Pang, V. S. Pai and L. Peterson (2004). "Reliability and Security in the CoDeeN Content Distribution Network ". *Proceedings of the USENIX 2004 Annual Technical Conference*, Boston, MA.

Zhang, M., C. Zhang, V. Pai, L. Peterson and R. Wang (2004). "PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services." *Proceedings of the Sixth Symposium on Operating Systems Design and Implementation*.

TRITA-CSC-E 2006:148
ISRN-KTH/CSC/E--06/148--SE
ISSN-1653-5715

www.kth.se