UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

Dipartimento di Ingegneria dell'Informazione

Scuola di Dottorato di Ricerca in Ingegneria dell'Informazione

CICLO XXI

Network Access Capacity Estimation through Passive Traffic Measurement

Direttore della Scuola: Ch.mo Prof. Matteo Bertocco

Supervisore: Ch.mo Prof. Nicola Zingirian

Dottorando: Martino Fornasa

L'utente di una linea di accesso a Internet (ad es. l'utente di una ADSL) è sensibile alla qualità del servizio, che è determinata da vari aspetti, tra cui la banda disponibile. La stima per mezzo di misure passive della massima banda ottenibile a livello IP (*capacity*) su un link di accesso ad una rete TCP/IP è un problema interessante sia dal punto di vista scientifico che considerando le sue applicazioni industriali.

Dal punto di vista scientifico, il problema è interessante perché l'estrazione della capacity a partire da osservazioni passive sul TCP/IP richiede lo sviluppo di modelli e algoritmi appropriati.

Dal punto di vista industriale, la stima della banda di accesso è uno strumento fondamentale per la verifica delle condizioni minime di servizio che possono essere stipulate tra un internet service provider ed un utente finale o tra il fornitore dell'accesso (access service provider) e il fornitore della connettività Internet (network service provider).

La tesi propone dei modelli e delle tecniche aventi lo scopo di stimare dall'interno della rete e in maniera passiva la capacity di un link di accesso a Internet. Il metodo proposto estende i tradizionali approcci basati sulle tecniche *packet-pair* considerando sequenze di pacchetti TCP più lunghe di due elementi allo scopo di minimizzare l'impatto del rumore di misura. Lo scopo è quello di ottenere una stima affidabile senza il bisogno di raccogliere grandi moli di dati su cui applicare tecniche statistiche.

Abstract

This work proposes models, techniques and tools aimed at passively estimating the maximum achievable downlink network-layer bandwidth (*capacity*) of an access link to the Internet from inside a network. The Internet access capacity estimation by mean of passive measurements is an interesting issue from a scientific and from an industrial perspective.

From a scientific perspective the problem, still open, is of interest because of the packet based and best effort based nature of the TCP/IP, that makes the user perceived rate depend not only on the access rate but also on the backbone performance and on the endpoint server performance. Extracting the access rate from TCP/IP observations requires the development of appropriate models, algorithms and techniques.

From an industrial perspective the problem is relevant considering the Internet's evolution is at a point at where the TCP/IP suite protocols best effort nature needs to be paired with appropriate provisioning policies based on enforceable service level agreements (SLA) between service providers and service users and/or between different types of service providers. The availability of widely accepted techniques to measure the service levels is fundamental to such an evolution.

This thesis proposes a method that extends the well-known packet-pair approach to network capacity estimation by considering longer TCP packet sequences to minimize the impact of measurement noise and to obtain reliable estimation without the need of a large amount of data. In order to obtain such a result, the method augments the traditional packet timing analysis with a model driven data analysis, similar to what is done in pattern recognition to extract known items from large data sets (for example to recognize known objects in images). A two layer process is proposed, in which the first layer performs packet timing analysis, based on statistical techniques, to extract the main traffic features, while the second layer combines the features extracted through appropriate heuristics, to compute the access capacity. The second layer takes advantage of the knowledge of the application environment (i.e. the way TCP/IP networks behave) to analyze the features in a global way, so as to eliminate the false features, i.e. those deriving from noise or artifacts, and identify the true features, i.e. those deriving from the known and expected network behaviour.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Internet access rate measurement at service provision time is a very relevant issue both from a scientific perspective and from an industrial perspective.

From a scientific perspective the problem, still open, is of interest because of the packet based and best effort based nature of the TCP/IP, that makes the user perceived rate depend not only on the access rate but also on the backbone performance and on the endpoint server performance. Extracting the access rate from TCP/IP observations requires the development of appropriate models, algorithms and techniques.

From an industrial perspective the problem is relevant considering the Internet's evolution is at a point where the TCP/IP suite protocols best effort nature needs to be paired with appropriate provisioning policies based on enforceable service level agreements between service providers and service users and/or between different types of service providers. The availability of widely accepted techniques to measure the service levels is fundamental to such an evolution.

Network operation characterization for performance measurement purposes is based on a set of definitions. In a computer network environment the words *bandwidth* and *rate* can refer to different concepts, so there is a large consensus in literature on the need of more precise definitions. This work will focus on the maximum achievable network-layer bandwidth on a link, that is usually called *link capacity*. In particular, we will rely on the definitions summarized in Table 1.1.

Traffic load measurements on network links are commonly performed by network operators from information provided by network devices (e.g. routers, switches) gathered via SNMP (Simple Network Management Protocol) [17]. Tools like MRTG [41] and RRDTool [42] permit automation of collection and graphical representation of rate values obtained by router counters (see for example [7]). However, this *router-based* approach requires access to the routers, which is possible only for the network owner.

In this work we are interested in *end-to-end capacity estimation*, an approach

Term	Meaning
Node	Host, router, Ethernet switch
Link	Connection between two nodes
Path	Series of links
Capacity	Maximum IP-layer bandwidth that a link can deliver
Narrow link	Link having the smallest capacity in the path

Table 1.1: Definitions

requiring no access to a router interface, but only to one or both the connection ends. In particular, this work addresses the problem of measuring the narrow link capacity in a TCP/IP network, i.e. the link having the smallest capacity in a path. In most cases, the narrow link corresponds to the access link. We can classify the capacity estimation techniques in several ways. A capacity estimation technique can be:

Active If the method requires actively probing a network, injecting traffic.

Passive If the method requires only to take measurements (timestamp, traffic traces) without injecting traffic onto a network.

End-to-end methods can be classified based on where the measurement point is. A method can be:

Receiver based If the measurements are taken at a receiver[1].

Sender based If the measurements are taken at a sender.

Network based If the measurements are taken in one node in the network path between the receiver and the sender.

This work proposes models, techniques and tools aimed at passively estimating downlink capacity of an access link from inside a network.

The proposed method is based on a widely accepted model of a bottleneck link, that is considered as a point-to-point link with a constant data rate, a First In First Out, FIFO, scheduling and a tail-drop queue management. While in some cases such a model might not be the most accurate way to describe an access link (for example a cable modem access link, or a Wi-Fi campus-wide access, as stated in [26]), it fits on the majority of scenarios well. In particular, such a model describes well a DSL access link.

[1] *Receiver* and *sender* are relative concepts. For example, measuring the downlink capacity of an ADSL line, the user host will be the receiver. On the contrary, while measuring the uplink capacity, the user host will be the sender.

1.1 Contributions

The original contributions of the work can be summarized as follows.

A first contribution is related to network organization. The work proposes a technique that permits estimating the access downlink capacity of a large number of access links to the Internet in a passive way. Let's consider the case in which a Network Service Provider (i.e. a Corporate Network Service Provider or an Internet Service Provider) reaches its customers through an access network operated by another provider, namely an Access Service Provider. This is the case, for example, of the Internet Service Providers that reach their residential customers by means of ADSL lines owned by the national incumbent operator. The Access Service Provider establish a contractual relationship based on a Service Level Agreement (SLA), but in general it has no easy way to verify the level of service in a passive way, and needs to be given appropriate techniques and tools to monitor the compliance.

The second contribution is related to methodology. The thesis proposes a method that extends the well-known packet-pair approach to network capacity estimation by considering longer packet sequences. This method permits to minimize the impact of measurement noise and to obtain reliable estimation without the need of a large amount of data. In order to obtain such results, we augment the traditional packet timing analysis with a model driven data analysis, similar to what is done in pattern recognition to extract known items from large data sets (for example to recognize known objects in images). A two layer process is proposed, in which the first layer performs packet timing analysis, based on statistical techniques, to extract the main traffic features (see Chapter 4), while the second layer combines the features extracted through appropriate heuristics, to compute the access capacity (Chapters 5 and 6). In this second layer we take advantage of the knowledge of the application environment (i.e. the way TCP/IP networks behave) to analyze the features in a global way, so as to eliminate the false features, i.e. those deriving from noise or artifacts, and identify the true features, i.e. those deriving from the known and expected network behaviour.

1.2 Roadmap

The aim of this work is the development of technology and of a tool to estimate the downstream access link capacity in TCP/IP networks, and in particular in the Internet, in a passive way, exploiting the TCP connections originating from hosts placed downstream of an access link. The presentation is organized as follows.

Chapter 2 describes the measurements arrangement considered in this work, in which the measurement tool, running in a network node called measurement node (MN), grabs the packet time-stamps, i.e. the times at which the packets pass

through the measurement node and computes the access link capacity of any host or network located downstream (e.g. the ADSL downlink capacity). Next, a description of the time components that form the round trip time of a TCP network connection is provided. In particular it is shown that under some conditions it is possible to split the round trip time in a fixed component and a variable component, with the variable component depending only on the queue sizes on the access link.

Chapter 3 gives an overview of the possible approaches to end-to-end capacity estimation. In particular the chapter proposes a classification of such methods. Section 3.1 describes some methods based on the average data passed during a time interval, and proposes a novel passive sender-based capacity estimation method that in some cases can provide a good lower bound of the actual capacity. Section 3.2 describes the 'packet-pair' estimation methods rationale, and Section 3.3 presents a brief literature review about packet-pair based capacity estimation methods.

Chapter 4 starts describing a mathematical model of packet transmission. Then, the model behaviour is analyzed in a simple case in which i) the traffic directed to the client passes entirely through the measurement node, ii) acknowledgment packets are never queued on the uplink access queue. The analysis leads to the definition of the concept of Packet Burst (PB), i.e. a burst of packets that are so close in time to each other that they need to be buffered in the access downlink device located at the network side of the access link. It is only during these bursts that the parametric mathematical model can be fruitfully applied to estimate the access link capacity. The deliverable of the work described in Chapter 4, presented in Section 4.2.1, is an algorithm that processes a set of packet time-stamps taken at the measurement node and generates a set of estimated bursts directed to the same client, each of which is associated whit an access link capacity.

Chapter 5 analyzes the effects of interfering traffic, i.e. i) the traffic that arrives to the downlink access queue without passing through the measurement node and ii) the data traffic originated by the TCP connections in which the end host acts as a sender. Section 5.1 describes interfering traffic effects on the downlink access queue, which can cause a capacity underestimation. Section 5.2 analyzes the effects on the interfering traffic on the uplink access queue, that can cause the 'ACK compression' phenomenon, which can cause a wrong capacity estimation.

Chapter 6 provides a set of heuristics aimed at refining the outcome of the Packet Burst extraction algorithm presented in Chapter 4 to obtain the actual access link capacity. In particular, after having obtained a set of Packet Burst, the heuristics are used to locate the time intervals during which there is interfering traffic, in order to filter them out.

Chapter 7 analyzes the fact that the TCP acknowledgements packets are sent according to the delayed acknowledgement scheme and suggests a modification of the capacity estimation approach to adapt it to the delayed acknowledgement scenario.

Chapter 8 presents a capacity estimation architecture suitable for high speed links. In particular, the main issues of high-speed packet capture are discussed, and some packet analysis architectures are proposed.

Chapter 9 describes some experiments performed to validate the approach and presents the results of such experiments.

Chapter 10 presents some concluding remarks.

Chapter 2

Scenario

Network operation characterization for performance measurement purposes is based on a set of definitions. In a computer network environment, the words *bandwidth* and *rate* can refer to different concepts, so there is a large consensus in literature on the need of more precise definitions. This paper will focus on the maximum achievable bandwidth on an access link, that is usually called *link capacity*. In particular, we will rely on the definition of the link capacity provided by RFC 5136:

> We define the IP-layer link capacity, C(L,T,I), to be the maximum number of IP-layer bits that can be transmitted from the source S and correctly received by the destination D over the link L during the interval [T, T+I], divided by I. [6]

A *link* is defined as:

> We define nodes as hosts, routers, Ethernet switches, or any other device where the input and output links can have different characteristics. A link is a connection between two of these network devices or nodes. [6]

Whereas a *path* is defined as:

> We then define a path P of length n as a series of links (L1, L2, ..., Ln) connecting a sequence of nodes (N1, N2, ..., Nn+1). [6]

The link having the smallest capacity is called the *narrow link* of the path.

2.1 Measurement arrangement

This thesis aims to measure the downlink capacity of an access link from inside a network in a passive way.

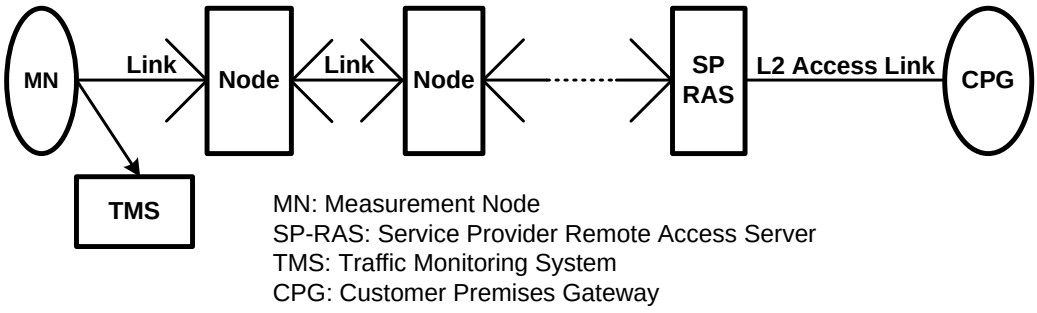

Figure 2.1: Description of the scenario

The capacity estimation method described in this work takes advantage of the TCP protocol. A Transmission Control Protocol, TCP (see [33]) is a transport-layer IP-based network protocol that employs a positive acknowledgement technique with retransmission in order to achieve a reliable data transmission. The receiver host periodically sends back to the sender host a packet[1] containing an acknowledgement. In particular, the TCP uses a cumulative acknowledgement scheme — when the receiver acknowledges a segment, it also means all the former segments have been correctly received.

In particular, we consider the scenario shown in Figure 2.1. We assume that a Traffic Monitoring System (TMS) is placed in a specific interface of a node, namely the Measurement Node (MN) (see Figure 2.1). Such a node is connected to a Customer Premises Gateway (CPG) by means of a chain of links bound together by a set of nodes. Such an MN may be figured out somewhere inside the network of an Internet Service Provider (ISP), at the border of such a network, for example in a Neutral Access Point (NAP), or at a network endpoint, for example in the Content Provider (CP) premises. We focus our attention on the access service between the CPG and the Service Provider Remote Access Server (SP-RAS). The access service is provided by the so called Access Provider (AP), that is usually the national incumbent operator.

As we are interested in estimating the access downlink capacity, we consider the TCP half connection in which the host attached behind the CPG (Customer Premises Gateway) acts as a receiver, and the host where the TMS (Traffic Monitoring System) is placed acts as a sender. This corresponds to the usual case in which an end-user acts as server client on the Internet. For example, when a residential user accesses a web page. In this case, we consider the data packets flowing towards the end-user host, and the acknowledgement packets coming from the end-user host.

[1]The TCP packets are called 'segments'. In the following, we will use the terms 'packet' or 'segment' to refer to a TCP segment.

The TMS captures the packets passing through the MN, marks them with appropriate time stamps, detects the TCP forward packets and the TCP acknowledgment packets and fills out an array of (Frame size, TCP Forward Frame Time Stamp, TCP Acknowledgment Frame Time Stamp) triples:

$$(w_0, t_0^s, t_0^a)$$

$$(w_1, t_1^s, t_1^a)$$

$$(w_2, t_2^s, t_2^a)$$

$$\dots$$

$$(w_{n-1}, t_{n-1}^s, t_{n-1}^a)$$

where w_i is the forward segment IP total size, t_i^s is the forward packet passing time, t_i^a is the acknowledgment packet passing time, and n is the number of TCP acknowledgment packet passed during a given observation.

The TCP does not always send an ACK for every data packet received. In fact, the *delayed acknowledgment* technique is often employed in order to send an ACK about every two data packets, thus making the number of acknowledgment packets lower than the number of data packets. This issue will be discussed in Section 7. For this reason, for defining the triples, we choose to not consider data packets having no acknowledgement.

2.2 Network delay components

The downstream delay between the MN and the CPG, i.e. the time needed for a TCP data packet to travel from the MN to the CPG, can be seen as the sum of the following components:

DD1 The sum of the delays due to signal propagation along the links, i.e. $DD1 = \sum_i \frac{l_i}{v_i}$, where l_i and v_i respectively denote the physical length and the propagation speed of the i-th link.

DD2 The sum of the delays due to packet transmission over the links, i.e. $DD2 = \sum_i \frac{w^d}{r_i^d}$, where w^d is the IP-level total size of the TCP data packet and r_i^d denotes the downstream capacity of the i-th link.

DD3 The sum of the delays due to processing in the nodes, i.e. $DD3 = \sum_i L_i$, where L_i denotes the processing delay of the i-th node.

DD4 The sum of the delays due to queuing in the node output interfaces, i.e. $DD4 = \sum qd_i^d$, where qd_i^d denotes the output queuing delay in the downstream i-th node.

Symbol	Meaning
w_i	IP datagram total size field of the i-th TCP forward segment (bit)
t_i^s	Passing timestamp of the i-th TCP forward segment at MN (seconds)
t_i^a	Passing timestamp of the i-th TCP acknowledgement segment at MN (seconds)
T	Network fixed delay (seconds)
ξ_i^d	Additive noise on the downlink at measurement i
ξ_i^u	Additive noise on the uplink at measurement i
r	Access link downstream capacity (bits per second)
q_i	Size of the access link downstream queue upon reception of the i-th TCP data segment (bit)

Table 2.1: Notation

Similarly, the upstream delay between the CPG and the MN, i.e. the time needed for a TCP acknowledgment packet to travel from the CPG to the MN, can be seen as the sum of the following components:

DU1 The sum of the delays due to signal propagation, i.e. $DU1 = \sum_i \frac{l_i}{v_i}$.

DU2 The sum of the delays due to packet transmission over the links, i.e. $DU2 = \sum_i \frac{w^u}{r_i^u}$, where w^u is the IP-level total size of the TCP acknowledgment packet and r_i^u denotes the upstream capacity of the i-th link.

DU3 The sum of the delays due to processing in the nodes, i.e. $DU3 = \sum_i L_i$, where L_i denotes the processing delay of the i-th node.

DU4 The sum of the delays due to queuing in the node output interfaces, i.e. $DU4 = \sum qd_i^u$, where qd_i^u denotes the output queuing delay in the upstream i-th node.

Let us examine the aforementioned components.

1. DD1 and DU1 are constant, as they depend only on constant parameters. So, we define $k_1 \equiv DD1 + DU1$.

2. DD2 and DU2 depend on the capacities (supposed fixed) of the links and on the frame sizes. If we consider that the TCP MSS (Maximum Segment Size) is usually around 1500 bytes for legacy reasons, the size of the vast majority of TCP forward packets is about 1500 bytes [38]. The size of the ACK packets (if we do not consider the piggyback packets) is 40 bytes. Under such hypothesis, the DD2 and DU2 components can be considered as fixed. So, we define $k_2 \equiv DD2 + DU2$.

3. DD3 and DU3 are constant if no special computing intensive packet scheduling technique is implemented. So, we define $k_3 \equiv DD3 + DU3$.

4. DD4 and DU4 depend on the network load. In particular, they depend on the amount of data present on the queues that compose the downstream path and the upstream path.

    - Regarding DD4, being k the downstream access link (i.e. the link between the SP-RAS and the CPG), if we assume that $r_k^d \ll r_i^d$, $\forall i \neq k$, i.e. that the downlink access capacity is small compared with the other downstream link capacities, the DD4 term can be approximated by the sum of a fixed term (k_4^d) and a noise (ξ^d) plus the delay induced by the downlink access queue in the SP-RAS (qd_k^d).

- The same is true for the DU4; if $r_k^u \ll r_i^u$, $\forall i \neq k$, i.e. the uplink access capacity is small compared with the other upstream link capacities, the DU4 term can be approximated by the sum of a fixed term (k_4^u) and a noise (ξ^u), plus the delay induced by the uplink access queue in the CPG (qd_k^u).

In summary:

$$DD4 = k_4^d + \xi^d + qd_k^d$$
$$DU4 = k_4^u + \xi^u + qd_k^u$$

Summarizing, we can say that the round trip time (RTT) of the network, i.e. the sum of downstream and upstream delays can be written as:

$$RTT = T + \xi^d + \xi^u + qd_k^d + qd_k^u \tag{2.1}$$

where the *fixed delay* T is defined as: $T \equiv k_1 + k_2 + k_3 + k_4^d + k_4^u$. Equation (2.1) provides an expression for the time interval between the passing time of a TCP data packet through the TMS and the return time of its TCP acknowledgment packet. In particular we showed that the RTT is the sum of a fixed delay (T), a noise component ($\xi^d + \xi^u$), a component that depends on the downstream queue size on the SP-RAS (qd_k^d), and a component that depends on the upstream queue size on the CPG (qd_k^u).

It can be noted that the access downlink and access uplink queue sizes depend on the timing properties of the traffic passing by the access link — both the measured traffic (i.e. the one that also passes through the TMS) and the interfering traffic (i.e. the one that not passes through the TMS).

Formula (2.1) will be exploited in Chapter 4 to devise a capacity estimation method in the absence of interfering traffic and in Chapter 5 to take into account the interfering traffic influence.

Chapter 3

End-to-end capacity estimation

In the Introduction we classified the end-to-end capacity estimation techniques using two classification criteria: active versus passive probing, and receiver-based versus network or sender-based probing. In the following we examine in detail the characteristics of such approaches, whereas Section 3.3 proposes a review of the method proposed in literature.

Active vs. Passive An active method requires actively probing a network, injecting traffic, whereas a passive method requires only to take traffic measurements. In general, passive measurements are harder then the active ones, because it has to rely only on the existing traffic on a link. The main problem here is the used bandwidth on a given link varying with time, making hard to measure the capacity, which is the maximum achievable transfer rate on that link.

Receiver-based vs. Network or Sender-based Receiver-based measurements are taken at a TCP receiver side, sender-based are taken at a sender side, whereas network based measurements are taken in one node in the network path between the receiver and the sender. In general, a receiver-based approach is easier than the other two, because network and sender-based approaches have to rely on the information carried by the ACK packets. The main problems here are the noise on the upstream path, the TCP 'delayed acknowledgment' scheme, and the interfering traffic on the upstream.

Section 3.1 describes some methods based on the average data passed during a time period, and a novel method is proposed. Section 3.2 describes the packet-pair approach, and Section 3.3 is a review of the end-to-end measurement method proposed in recent years.

In the following the notation of Table 2.1, page 22, will be used. In particular, the symbol r will be used for the access downlink capacity.

3.1 Average methods

Receiver side An active capacity measurement performed with the cooperation of the receiver is not hard to perform (see for example [9, 25, 26, 30]). It consists of running a number of long-lived bulk TCP transfer (for example FTP downloads from a server placed on a well-provisioned link) and the capacity is the number of received bits divided by the time interval between the first (namely i) and the last (namely j) received TCP data packets. If t_k^r is the arrival time of the k-th TCP packet to the receiver, we can write the following expression for the average rate during the time interval $[t_i^r, t_j^r],\ i < j$.

$$\bar{r}_{i,j} = \frac{\sum_{k=i}^{j} w_k}{t_j^r - t_i^r} = \frac{\sum_{k=i}^{j} w_k}{\Delta_{i,j}^r}$$

$\bar{r}_{i,j}$ is equal to the capacity of the link ($\bar{r}_{i,j} = r$) only if during the packet interval $[i, j]$ the traffic saturates the link, i.e. the downlink access queue is non-empty for the whole duration of the interval.

Exploiting the measurements taken at the receiver side, the formula takes into account the packets belonging to all the different TCP connections that involve the receiver. Figure 3.1 shows the time diagram of an interval composed of four data/ACK packet pairs.

Sender or network side Estimating the capacity on the sender or network side is harder than doing so on the receiver because the measurements have to rely on the TCP acknowledgment packets arrival times:

1. The uplink backbone delay path is noisy.
2. In general, not all traffic involving the receiver passes from the measurement node.
3. TCP acknowledgments are sent according the delayed acknowledgment scheme, i.e. the TCP protocol acknowledges more than one data packet a time.

Lying on the sender or network side, we only know the send time of the k-th TCP packet t_k^s and its corresponding ACK passing time t_k^a. So, we can write the following expression for the average rate during the $[i, j]$ interval:

$$\bar{r}_{i,j}^* = \frac{\sum_{k=i}^{j} w_k}{t_j^a - t_i^s} = \frac{\sum_{k=i}^{j} w_k}{\Delta_{i,j}^s} \simeq \frac{\sum_{k=i}^{j} w_k}{\Delta_{i,j}^r + RTT}$$

It can be noted that $\bar{r}_{i,j}^*$ is an underestimation of $\bar{r}_{i,j}$ as the difference between $\Delta_{i,j}^s$ and $\Delta_{i,j}^r$ is about a one round-trip time. So, the two quantities difference is lower

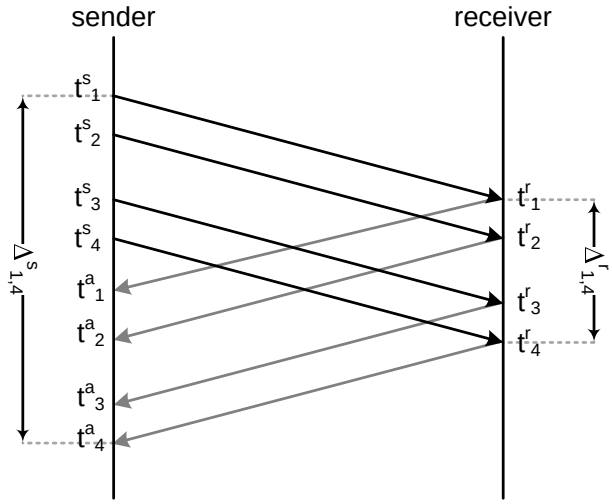

Figure 3.1: Time diagram of a sequence of four data/ACK packet pairs.

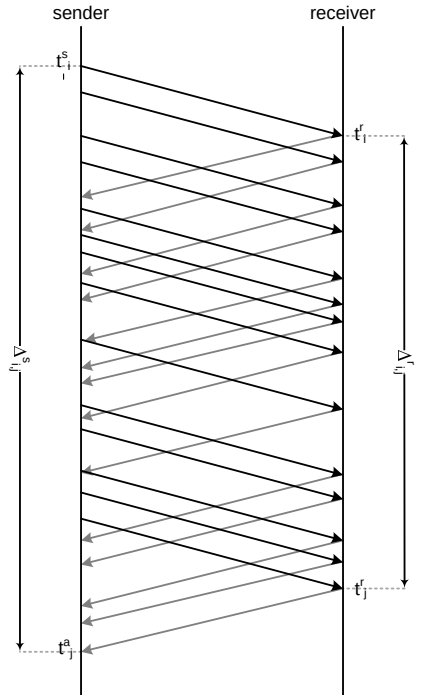

Figure 3.2: Time diagram of a long sequence of data/ACK packet pairs.

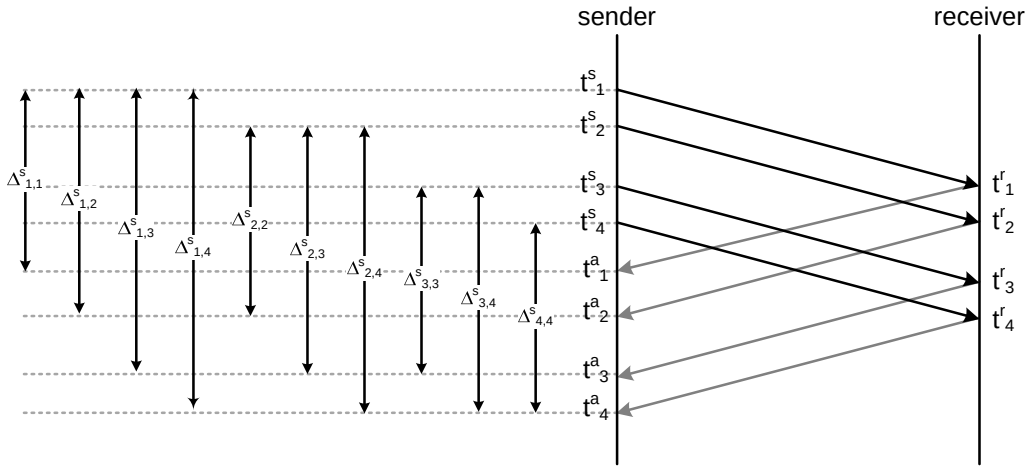

Figure 3.3: $\hat{r}$ calculation

when the considered interval is large. As $\bar{r}$, also $\bar{r}^*$, need the downlink access link saturation in order to approximate the capacity.

In summary, the main problem with the average methods is that the used bandwidth on a link is most of the time lower than the capacity. On the sender or network side, the fact that a fraction of the traffic could not pass the measurement node worsens the problem.

In order to improve such aspect it is possible to consider the quantity $\hat{r}$, defined as follows:

$$\hat{r} \equiv \max_{i\geq 1, j\leq n;\, i\leq j} \left(\frac{\sum_{k=i}^{j} w_k}{t_j^a - t_i^s} \right) = \max_{i\geq 1, j\leq n;\, i\leq j} \left(\frac{\sum_{k=i}^{j} w_k}{\Delta_{i,j}^s} \right) \tag{3.1}$$

$\hat{r}$ selects the maximum-rate interval $[i, j]$ from all the possible intervals during the observation period. This helps to find a suitable interval in which the link saturates. In general, we have that $\hat{r} \leq r$, i.e. $\hat{r}$ is a lower bound on r. Figure 3.3 shows the time interval considered by Equation (3.1) on a four-TCP packet trace.

3.2 Packet pair techniques

A number of end-to-end link capacity estimation techniques have been proposed in recent years, and a good review can be found in [34]; the authors classify the capacity estimation techniques in two main categories:

- Variable Packet Size Probing, aiming to estimate the capacity by using an active probing with packets of varying sizes.

- Packet Pair / Train Dispersion Probing, aiming to estimate the capacity measuring the time interval between back-to-back packets, because such a time is influenced by the constrained links on the path.

The variable packet size methods are inherently active, whereas in general the dispersion techniques can be passive.

The packet pair dispersion technique is based on the dispersion (i.e. the time difference between the last bit of the first packet and the last bit of the second packet) of a pair of back to back packets passing through a link can be modified. In general, the dispersion of a back to back pair after a link of capacity r will be:

$$d = w/r$$

where w is the size of the two packets. The above formula is valid assuming no interfering traffic on the link. In general, the dispersion of two back to back packets that traverse a path is the one induced by the narrow link (i.e. the link having the smallest capacity on the path). Using the aforementioned formula, it is possible to calculate the narrow link capacity as:

$$r = w/d$$

In a TCP/IP network the packet pair method can be applied:

1. On the receiver side, by measuring the dispersion of the data packets that arrive to the receiver.

2. On the sender side or network side, by measuring the dispersion of the TCP acknowledgement packets, exploiting the fact that to a first approximation the interarrival times of the ACK packets to the measurement node reflects the interarrival times of the data packets to the receiver.

On the receiver side, the main issues causing capacity estimation errors are (as described, for example, in [13]):

1. Interfering traffic on the narrow link can increase packet dispersion, leading to a rate underestimation.

2. Interfering traffic on the post-narrow links, i.e. the links placed after the narrow link on the path, can decrease the packet dispersion, leading to a rate overestimation.

In general, the main concern is due to the second issue. In fact, in the presence of post-narrow links, it is impossible to estimate the narrow link capacity using the minimum of dispersion measurements. However, in the downlink access capacity

estimation scenario considered in this work (described in Section 2.1), the narrow link is the last link on the downstream path, removing the overestimation problem.

Performing the capacity estimation on the network side or on the sender side poses some additional issues not present in the receiver side estimation. In particular such issues are:

1. The uplink backbone path delay is noisy.
2. The TCP acknowledgments are sent according to the delayed acknowledgment scheme, i.e. the TCP protocol acknowledges more than one data packet at a time.
3. Interfering traffic on the uplink queue can cause the queuing of the ACKs on the queue, causing a decrease of the ACK pair dispersion, namely the ACK compression phenomenon.

Regarding noise, we can note that any noise on the uplink path can modify the dispersion of the TCP ACK pairs. Being ξ_i the path delay noise affecting the i-th packet pair, we have:

$$d_i = \frac{w_i}{r} + \xi_i$$

Our tests show that in the Internet, ξ_i is usually comparable to $\frac{w_i}{r}$, leading to a considerable error on the dispersion measurement. This is mainly due to the low value of $\frac{w_i}{r}$, caused by the MSS (Maximum Segment Size) of the TCP being usually around 1500 bytes for legacy reasons, irrespective of the ever increasing link capacities (on a typical 7 Mbps ADSL link, $\frac{w_i}{r}$ is about 1.7 ms, with a typical noise value $\xi_i = \pm 0.7$ ms).

In order to overcome this issue, we propose measuring the dispersion of sequences composed by more than two back-to-back packets, often called packet trains, in order to decrease the impact of the noise on the capacity estimation. The dispersion of a packet train composed by n packets of the same size w_i can be written as:

$$d_i = \frac{(n-1)w_i}{r} + \xi_i$$

It can be noted that the impact of the noise on the dispersion measurement decreases with the length of the packet train.

As we stated before, our objective is to perform a passive capacity estimation, i.e. to avoid injecting traffic on the network. So, we can not generate the packet trains needed to perform the estimation. In order to passively measure the dispersion of a packet train, it is necessary to exploit the bursty nature of the TCP traffic by identifying what we call Packet Bursts, i.e. bursts of packets that are so close in time to each other, they need to be buffered in the access downlink queue. During such bursts, the time difference between each successive burst packet entering the

downlink queue is $d < \frac{w}{r}$. So, the dispersion of the ACK packets generated by these bursts can be exploited to perform a capacity estimate, even if the bursts were not strictly composed by back to back packets.

Dovrolis et al. [13] argued, in general, considering packet trains instead of packet pairs is ineffective, due to longer sequences being more prone to the interfering traffic on the downlink. We believe this is not an issue in the access capacity measurement scenario. In fact, the absence of post-narrow links ensures all the interfering traffic distortion will be rate underestimation. This allows determining capacity by using the minimum dispersion, thus filtering the rate underestimations.

3.3 Previous work on packet-pair techniques

The packet pair dispersion techniques aimed to estimate the capacity of a link originate from the work of Jacobson [18], Bolot [2] and Keshav [22]. Several extensions to the original concept have been presented, both active and passive.

3.3.1 Active techniques

The authors of [4] propose an active capacity estimation method based on packet-pair technique and ICMP echo replies, based on ACK traces. They recognize a set of problems with the simple approach:

- Queueing failure, i.e. the sender sends probe packets too slow to have such packets queued.
- Interfering traffic along the path.
- Packet drops.
- Congestion on return path.

The authors propose some statistical filtering methods and the use of variable size probes in order to attenuate the wrong modes on the interarrival times histogram.

In [29] the active packet-pair probing method is augmented by considering the effect of the probe size on the rate estimation, due to the different lower layer overhead that affects packets of varying sizes. This method considers the end-to-end delay variations and uses a peak detection technique.

In [13] (an extension of [12]) the authors note that:

- The congestion on the narrow link can cause an increased packet dispersion
- The congestion on a post-narrow link can cause a decreased packet dispersion

So, in general it is impossible to use the minimum of the measured dispersion in order to estimate the capacity of a narrow link. The authors used simulation and live experiments to demonstrate that in presence of heavy interfering traffic, the under-estimation modes prevails on correct capacity estimation. The paper also investigates the effect of the probing packet sizes. They point out, using probing pairs of different sizes can flatten the under-estimation modalities, making the capacity mode stronger. Finally, the authors analyze the dispersion measurement made on a sequence of more than two back-to-back packets, namely packet trains, asserting that a long packet train is not apt for capacity measurement, because they are more prone to the interfering traffic. Their paper proposes a capacity estimation method named Pathrate, a double-end active method.

In [20] the authors present an active capacity estimation technique called CapProbe. CapProbe combines the usual packet-pair dispersion measurement with the end-to-end delay measurement, and exploits the minimum delay packet pair to estimate the capacity. This behaviour is justified when one of the packets composing the packet pair is delayed, the total end-to-end delay increases. So, the method takes the minimum delay pair dispersion to perform the capacity estimation. In addition, the paper presents a statistical performance analysis supposing different types of interfering traffic, and gives some algorithm to detect the method convergence, i.e. to detect when the minimum delay of a packet pair is actually the delay in the presence of no queueing.

In [5] a CapProbe derivation called AsymProbe is presented. AsymProbe is an active, sender-side capacity estimation method that uses variable size probes in order to correctly estimate the uplink and downlink capacities of asymmetric links.

In [11] the authors present a large scale measurement study of the characteristics of the link performances of DSL and cable access in Europe and North America. The authors employed an active method consisting of large probes sent to a large number of hosts attached to a residential broadband access link. The aim of the work was to actively measure the access link characteristics such as the capacity in uplink and downlink, the queue dimensions, the RTT of the last hop, the packet loss rate, and the packet drop policies.

In [31], the authors present an extension of the CapProbe method ([20]) called TCP Probe. TCP Probe requires a modification of TCP on the sender-side in order to put a sufficient number of back-to-back data packet on the network and inverting the sequence number of the packet pairs to overcome the delayed ACK issue

3.3.2 Passive techniques

In [24] the authors propose a filtering method based on the assumption that the interfering traffic has random packet size and arrives randomly. However, the technique performs badly running on the sender side.

In [21] the authors argue that, due to the fact that the distribution of IP packet sizes is dominated by 40 byte and 1500 bytes packet, the distribution of interarrival times of packet pairs in the presence of interfering traffic is composed of several equally spaced modes. The spacing will be the interarrival time of a 1500 byte packet in some of the queues on the path. The analysis starts from a large number of interarrival samples, and computes the probability distribution of such gaps at increasing resolution, in order to identify multiple congested links traversed by the packets. The authors propose a passive (received-based or ACK-based) capacity estimation technique. The proposed method used the so-called *equally-spaced mode gaps*, exploiting the fact that the distribution of IP packet sizes is dominated by 40 byte and 1500 byte packets. The authors state that the distribution of interarrival times of packet pairs in the presence of interfering traffic is composed of several equally spaced modes. The spacing will be the interarrival time of a 1500 byte packet in some of the queues on the path. The analysis starts from a large number of interarrival samples, and computes the probability distribution of such gaps at increasing resolution, in order to identify multiple congested links traversed by the packets.

Lakshminarayanan et al. analyze in [26] the conditions in which the model based on a FIFO queue and point to point access link does not apply. In particular they focus on:

- Wireless based access links, that employ dynamic multi-rate regulation schemes, and non-FIFO scheduling.
- Cable modem access link, that usually comprise a token-bucket regulation scheme and non-FIFO scheduling.

Chapter 4

Model analysis: No interfering traffic

In this chapter we devise an access downlink capacity estimation method. In order to simplify the analysis, we make two hypotheses:

1. There is no interfering traffic on the downlink, i.e. all the traffic that passes through the access link passes also through the TMS.

2. Acknowledgment segments are never queued on the uplink access queue, i.e. the uplink access link gives a fixed contribution to the RTT of each forward/acknowledgment pair.

After having devised a capacity estimation method that works under such conditions, we will come back to a more general scenario in Chapter 5.

In Chapter 2 we presented a formula for the sum of the downstream and upstream delays on the path from the TMS to the CPG, which can be written as follows:

$$RTT = T + \xi^d + \xi^u + qd_k^d + qd_k^u$$

The aforementioned hypotheses have some consequences on the RTT formula. In particular:

1. Using the first hypothesis, the term qd_k^d can be written in a closed form, as it is possible estimating the downlink queue size, because such a size is determined only by the measured traffic pattern.

2. Using the second hypothesis, the term qd_k^u can be cancelled, as the ACK packets are never queued on the uplink of a well dimensioned access line in the absence of uplink data transfer. This is due to the fact that the size of a data packet can be up to 1500 bytes, whereas the size of an ACK packet

is around 40 bytes. So, symmetric access links never show queueing on the uplink, and also asymmetric access links are always correctly dimensioned in order to avoid such a phenomenon.

So, we can write: 1) an expression for the downlink access queue size after the arrival of each forward segment (q_i) and 2) an expression for the acknowledgment segments passing time (t_i^a), as follows:

$$\begin{cases} q_i = w_i + \max\left(q_{i-1} - (t_i^s + \xi_i^d - t_{i-1}^s - \xi_{i-1}^d)r, 0\right) & \text{(4.1a)} \\ t_i^a = t_i^s + T + \xi_i^d + \xi_i^u + \dfrac{q_i}{r} & \text{(4.1b)} \end{cases}$$

$$\text{for } i = 0, 1, \ldots, n-1$$

where n is the total number of forward segments during the observation period.

Equation (4.1a) describes the evolution of the downlink queue size. The queue size after the arrival of the i-th segment (q_i) is the sum of two components: the size of the i-th segment (w_i) and the queue size before the arrival of such a segment. For such a second component there are two possibilities, represented by the max function in the formula:

- If the queue size upon reception of the (i-1)-th segment (q_{i-1}) has already been processed at the time of the arrival of the i-th segment, the i-th segment reaches an empty queue. In this case the quantity $\left(q_{i-1} - (t_i^s + \xi_i^d - t_{i-1}^s - \xi_{i-1}^d)r\right)$ is negative or zero, and the max function takes zero.

- In the other case, the i-th segment reaches a non-empty queue, and the max function takes a positive value.

It can be noted that the behaviour of the first equation is nonlinear, due to the presence of a max function.

Equation (4.1b) gives an expression for the passing time of the i-th acknowledgment segment (t_i^a), which is the sum of the passing time of the i-th forward segment (t_i^s), plus the channel delay (T) and the associated noise in the downlink (ξ_i^d) and in the uplink (ξ_i^u), plus the time spent by the segment in the queue ($\frac{q_i}{r}$).

In the following, we first carry an analysis in a simplified case, when the measurement interval comprises only three data packet/ACK packet pairs (Section 4.1). Next, we will provide a general solution and we will describe a capacity estimation algorithm (Section 4.2).

4.1 Three triples

This section obtains a solution for the system described in Equation (4.1) for a sequence composed of three triples:

$$(w_1, t_1^s, t_1^a)$$
$$(w_2, t_2^s, t_2^a)$$
$$(w_3, t_3^s, t_3^a)$$

Let K be the queue size at the arrival time of the first segment to the queue.

4.1.1 Three triples: no noise

We first suppose that the noise components of Equation (4.1) are not present. Under such condition, the equation applied to three measured triples becomes:

$$\begin{cases} q_1 = w_1 + K \\ t_1^a = t_1^s + T + \frac{q_1}{r} = t_1^s + T + \frac{w_1 + K}{r} \end{cases}$$

$$\begin{cases} q_2 = w_2 + \max\left(q_1 - (t_2^s - t_1^s)r, 0\right) \\ t_2^a = t_2^s + T + \frac{q_2}{r} \end{cases}$$

$$\begin{cases} q_3 = w_3 + \max\left(q_2 - (t_3^s - t_2^s)r, 0\right) \\ t_3^a = t_3^s + T + \frac{q_3}{r} \end{cases}$$

Our purpose is to solve the linear system, i.e. to calculate the three unknown quantities r, T, K that satisfy the six equations. Having three unknown quantities, three equations (i.e. three triples) are needed. The equation system is non-linear for two reasons:

- Reason 1: the max function in q_2 and q_3 equations.
- Reason 2: the $\frac{K}{r}$ ratio that appears in t_1^a equation, and can also appear in the t_2^a and t_3^a equations after the substitution of the Q_2 and Q_3 terms.

Reason 1 (the max function) can be eliminated by considering the four possible cases for the max function argument. Having two max functions, there are four cases, namely A1, A2, B1, B2, summarized in Table 4.1.
The system that we obtain in the four cases is still non-linear for Reason 2. Next, we analyze the four cases in detail.

	A		B	
	A1	A2	B1	B2
$q_1 - (t_2^s - t_1^s)$	≤ 0		> 0	
$q_2 - (t_3^s - t_2^s)$	≤ 0	> 0	≤ 0	> 0

Table 4.1: Three triples analysis: the four possible cases

Cases A1 and A2 The quantity $q_1 - (t_2^s - t_1^s)$ is less than or equal to zero, so the max function in equation q_2 takes zero. In other words, segment 1 and segment 2 queue arrival times are spaced far enough, that segment 2 arrives to an empty queue.

$$\begin{cases} q_2 = w_2 \\ t_2^a = t_2^s + T + \frac{w_2}{r} \end{cases}$$

There are two sub-cases:

- **Case A1**: The quantity $q_2 - (t_3^s - t_2^s)$ is less than or equal to zero, so the max function in equation q_3 takes zero. This means that segment 2 and segment 3 are spaced far enough that segment 3 is not influenced by the previous situation (i.e. it arrives to an empty queue).

$$\begin{cases} t_1^a = t_1^s + T + \frac{w_1+K}{r} \\ t_2^a = t_2^s + T + \frac{w_2}{r} \\ t_3^a = t_3^s + T + \frac{w_3}{r} \end{cases} \implies \begin{cases} r = r_{A1} = \frac{w_2 - w_3}{RTT_2 - RTT_3} \\ K = K_{A1} = r(RTT_1 - RTT_2) - w_1 + w_2 \\ T = T_{A1} = RTT_3 - \frac{w_3}{r} \end{cases}$$

- **Case A2**: The quantity $q_2 - (t_3^s - t_2^s)$ is greater than zero. Segment 2 and segment 3 are spaced in a way that segment 3 arrives to a non-empty queue (Such a non-empty queue is set only by segment 2 properties, because we are in the case A).

$$\begin{cases} t_1^a = t_1^s + T + \frac{w_1+K}{r} \\ t_2^a = t_2^s + T + \frac{w_2}{r} \\ t_3^a = t_3^s + T + \frac{w_2+w_3}{r} - (t_3^s - t_2^s) \end{cases} \implies \begin{cases} r = r_{A2} = \frac{w_3}{t_3^a - t_2^a} \\ K = K_{A2} = r(RTT_1 - RTT_2) - w_1 + w_2 \\ T = T_{A2} = RTT_2 - \frac{w_2}{r} \end{cases}$$

Cases B1 and B2 The quantity $q_1 - (t_2^s - t_1^s)$ is greater than zero. In other words, segment 1 and segment 2 are spaced in a way that segment 2 arrives to a non-empty queue (Such a non-empty queue is a consequence of segment 1 and the K quantity).

$$\begin{cases} q_2 = w_2 + q_1 - (t_2^s - t_1^s) = w_2 + w_1 + K - (t_2^s - t_1^s) \\ t_2^a = t_2^s + T + \frac{w_1+w_2+K}{r} - (t_2^s - t_1^s) \end{cases}$$

There are two sub-cases:

- **Case B1**: The quantity $q_2 - (t_3^s - t_2^s)$ is less than or equal to zero, so the max function in equation q_3 takes zero. Segment 2 and segment 3 queue arrival times are spaced far enough that segment 3 is not influenced by the previous situation (i.e. it arrives to an empty queue).

$$\begin{cases} t_1^a = t_1^s + T + \frac{w_1+K}{r} \\ t_2^a = t_2^s + T + \frac{w_1+w_2+K}{r} - (t_2^s - t_1^s) \\ t_3^a = t_3^s + T + \frac{w_3}{r} \end{cases} \implies \begin{cases} r = r_{B1} = \frac{w_2}{t_2^a - t_1^a} \\ K = K_{B1} = r(RTT_1 - RTT_3) - w_1 + w_3 \\ T = T_{B1} = RTT3 - \frac{w_3}{r} \end{cases}$$

- **Case B2**: The quantity $q_2 - (t_3^s - t_2^s)$ is greater than zero. Segment 2 and segment 3 are spaced in a way that segment 3 arrives to a non-empty queue (Such a non-empty queue is set by segment 1 and segment 2 properties, and by the K quantity, because we are in case B).

$$\begin{cases} t_1^a = t_1^s + T + \frac{w_1+K}{r} \\ t_2^a = t_2^s + T + \frac{w_1+w_2+K}{r} - (t_2^s - t_1^s) \\ t_3^a = t_3^s + T + \frac{w_1+w_2+w_3+K}{r} - (t_3^s - t_1^s) \end{cases} \implies \begin{cases} r = r_1 = \frac{w_2}{t_2^a - t_1^a} \\ r = r_2 = \frac{w_3}{t_3^a - t_2^a} \\ r = r_3 = \frac{w_2+w_3}{t_3^a - t_1^a} \\ T = \dots \\ K = \dots \end{cases}$$

  In this case, due to the non-linearity (Reason 2) it is possible to obtain three different expressions for the capacity ($r = r_1 = r_2 = r_3$), but it is impossible to obtain a unique solution for K and T; in other words, K can be expressed only in terms of T, and vice versa.

Discussion It can be noted that given three successive segments, in general we do not know in which case the system evolved (A1, A2, B1, or B2), so it is impossible to choose the correct system solution.

However, if the system evolved in the B2 case, the quantities r_1, r_2 and r_3 are equal. Summarizing, there are two possible cases:

- If the three quantities r_1, r_2, r_3 are equal, the measured system evolved in the B2 case (i.e. the system satisfies the B2 conditions) and the downlink access capacity is $r = r_1 = r_2 = r_3$.
- If the three quantities r_1, r_2, r_3 are not equal, the measured system evolved either in A1, A2, or B1 case.

In the latter case, we can try to calculate the system solutions for the three cases ($\{r_{A1}, K_{A1}, T_{A1}\}$,$\{r_{A2}, K_{A2}, T_{A2}\}$,$\{r_{B1}, K_{B1}, T_{B1}\}$). If only one of the solutions appears to be valid, i.e. $r > 0, T \geq 0, K \geq 0$; such a solution identifies the correct

case (either A1, A2, or B1), and provides the capacity estimation. If more than one solution appears to be valid, there is no way to identify the correct case. As a consequence, it is impossible to obtain the capacity.

In summary, we showed that, given three triples, corresponding to three successive segments, we can:

- Verify if the three triples satisfy the B2 conditions
- If the triples are in B2 conditions, calculate the access downlink capacity.

Example The following set of triples is an example that does not satisfy the B2 conditions:

$$F1 : (10000, 0.000, 0.040)$$
$$F2 : (5000, 0.020, 0.050)$$
$$F2 : (6000, 0.027, 0, 058)$$

The following Table reports the system solution for the four possible cases obtained in the previous section, applied to the aforementioned triples:

A1	r_{A1}	$1.00e6$
	K_{A1}	5000
	T_{A1}	0.0250
A2	r_{A2}	$7.50e5$
	K_{A2}	2500
	T_{A2}	0.0233
B1	r_{B1}	$5.00e5$
	K_{B1}	500
	T_{B1}	0.0190
B2	r_1	$5.00e5$
	r_2	$7.50e5$
	r_3	$6.11e5$

It can be noted, the capacity values obtained for case B2 are not equal. So, we know that the system evolved either in an A1, A2, or B1 cases. However, the solutions for such three cases appears all valid. In conclusion, this is a case in which it is not possible to obtain the access downlink capacity by means of the three measured triples.

Multiple packets in B2 conditions The capacity expression valid in B2 conditions devised in the previous section can be extended to an arbitrary-long sequence of triples in B2 conditions as follows:

$$r = \frac{\sum_{j=2}^{i} w_j}{t_i^a - t_1^a} \tag{4.2}$$

4.1.2 Simple case: Dealing with the noise

We now revert to consider the noise components on Equation (4.1). The purpose of this section is to provide an expression for the error on the capacity estimation due to noise. We assume the three segments are in B2 conditions, as this is the case in which we can correctly estimate the capacity.

Three triples The capacity estimation based on three triples has been provided in the previous section. One of the equations that provides the capacity is the following:

$$r = \frac{w_2}{t_2^a - t_1^a} \tag{4.3}$$

Suppose now that the quantity $t_2^a - t_1^a$ is affected by an error Δ_T. In this case, the measured capacity will differ from the real capacity:

$$r_{meas} = \frac{w_2}{t_2^a - t_1^a + \Delta_T} \tag{4.4}$$

The error on the capacity can be obtained by making the difference between the measured capacity and the real capacity:

$$\begin{aligned}
\Delta_r = r_{meas} - r = \frac{w_2}{t_2^a - t_1^a + \Delta_T} - \frac{w_2}{t_2^a - t_1^a} = \\
= \frac{w_2(t_2^a - t_1^a) - w_2(t_2^a - t_1^a + \Delta_T)}{(t_2^a - t_1^a + \Delta_T)(t_2^a - t_1^a)} = \frac{-w_2}{(t_2^a - t_1^a)} \frac{\Delta_T}{(t_2^a - t_1^a + \Delta_T)} = \\
\overset{(4.3)}{=} -r \frac{\Delta_T}{t_2^a - t_1^a + \Delta_T} \overset{(4.3)}{=} -r \frac{\Delta_T}{\frac{w_2}{r} + \Delta_T} = -r \frac{\Delta_T}{\frac{w_2 + r\Delta_T}{r}} = \\
= -r^2 \frac{\Delta_T}{w_2 + r\Delta_T}
\end{aligned}$$

If we denote with Δ_T^{MAX} the maximum absolute value that can be assumed by Δ_T ($r_{meas} = \frac{w_2}{t_2^a - t_1^a \pm \Delta_T^{\mathrm{MAX}}}$), we can bound the maximum error on r:

- If $\Delta_T = -\Delta_T^{\mathrm{MAX}}$, then $\Delta_r^{(1)} = r^2 \frac{\Delta_T^{\mathrm{MAX}}}{w - r\Delta_T^{\mathrm{MAX}}}$

- If $\Delta_T = \Delta_T^{\mathrm{MAX}}$, then $\Delta_r^{(2)} = -r^2 \frac{\Delta_T^{\mathrm{MAX}}}{w + r\Delta_T^{\mathrm{MAX}}}$

Thus, the upper bound on the measured capacity is $r_{meas} = r + \Delta_r^{(1)}$ and the lower bound is $r_{meas} = r + \Delta_r^{(2)}$ (Note that $\Delta_r^{(2)}$ is negative or zero). Figure 4.1 shows a plot of $\Delta_r^{(1)}$ versus Δ_T^{MAX}. It can be noted that:

- If the error on $(t_2^a - t_1^a)$ is zero, the error on r will be zero.
- The error on r increases with Δ_T^{MAX}, asymptotically converging to the $(-r)$ value.

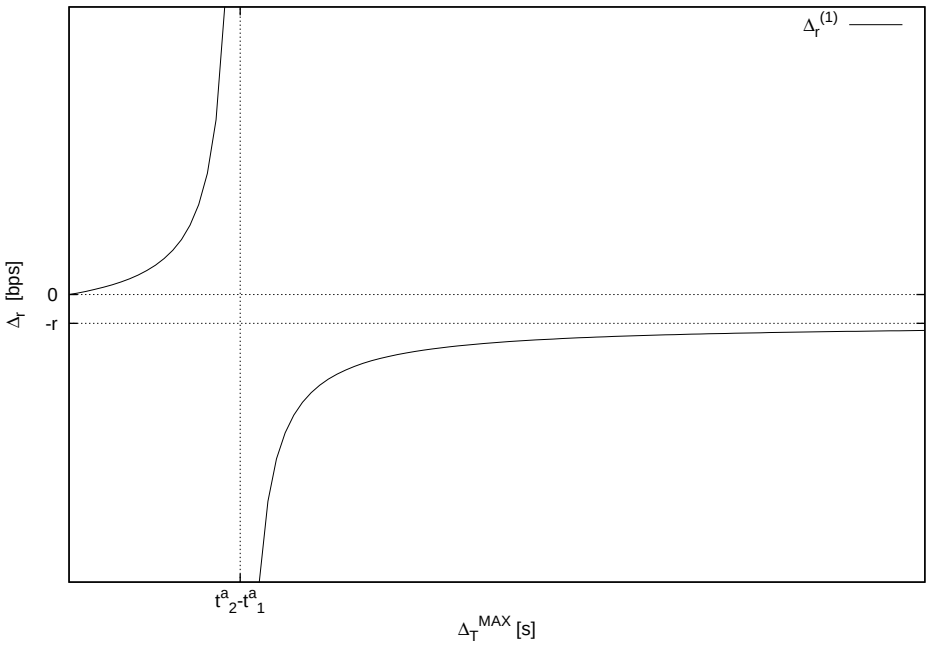

Figure 4.1: $\Delta_r^{(1)}$ versus Δ_T^{MAX}

Figure 4.2 shows a plot of $\Delta_r^{(2)}$ versus Δ_T^{MAX}. In can be noted that:

- If the error on $(t_2^a - t_1^a)$ is zero, the error on r will be zero.
- The absolute value of the error on r increases with Δ_T^{MAX}. The error asymptotically converges to the $(-r)$ value.

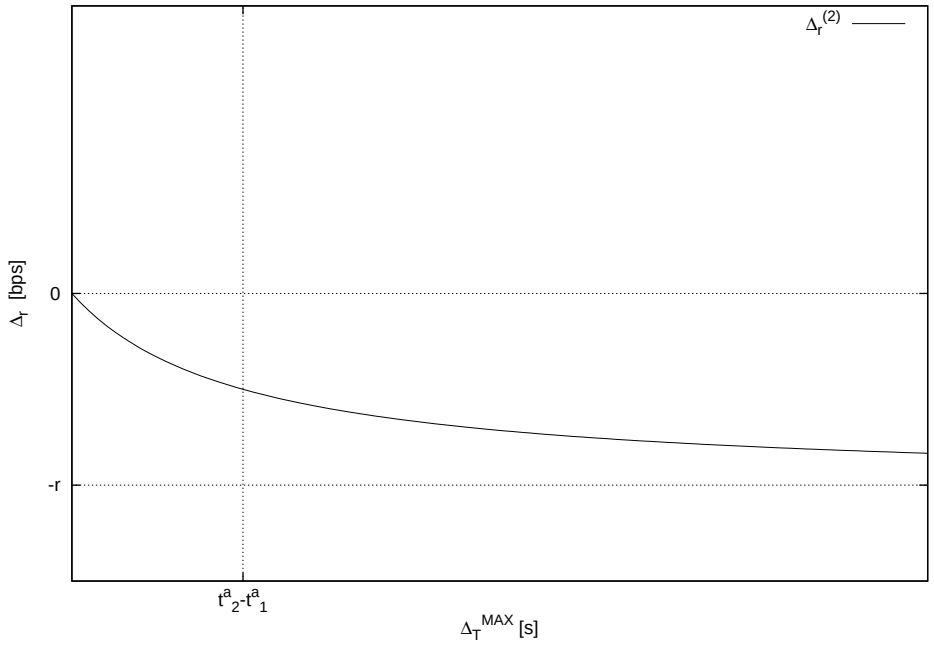

Figure 4.2: $\Delta_r^{(2)}$ versus Δ_T^{MAX}

Figure 4.3 shows a plot of $\Delta_r^{(1)}$ (continuous line) and $\Delta_r^{(2)}$ (dashed line) versus Δ_T^{MAX} for x-values between zero and $(t_2^a - t_1^a)$. For example, in order to limit the error on r in the range $\pm 25\%$, it is necessary to have a relative error on $(t_2^a - t_1^a)$ less than or equal to the 20%.

Multiple packets Suppose now to consider the capacity calculation method applied to a sequence of packets in B2 conditions. The capacity can be obtained using Equation (4.2).

As we have done in the previous section, we suppose the quantity $(t_i^a - t_1^a)$ is affected by an error Δ_T. The measured capacity will differ from the real one:

$$r_{meas} = \frac{\sum_{j=2}^{i} w_j}{t_i^a - t_1^a + \Delta_T}$$

In this case, r_{meas} will be affected by an error Δ_r (see previous section for the

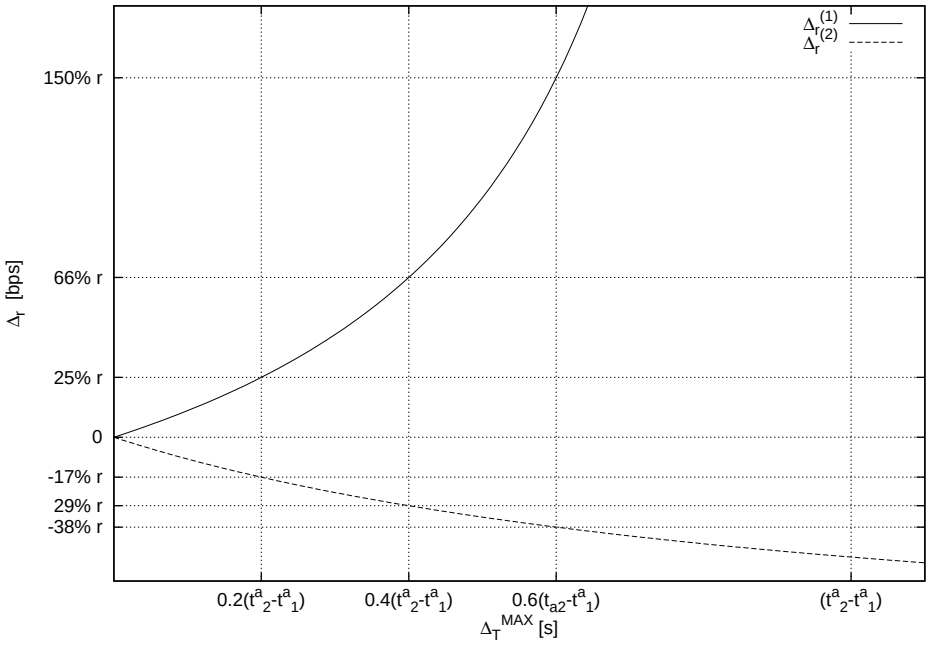

Figure 4.3: $\Delta_r^{(1)}$ and $\Delta_r^{(2)}$ versus Δ_T^{MAX}

calculation of Δ_r):

$$\Delta_r = r_{meas} - r = -r^2 \frac{\Delta_T}{\sum_{j=2}^{i} w_j + r\Delta_T}$$

If we denote with Δ_T^{MAX} the maximum absolute value that can be assumed by Δ_T ($r_{meas} = \frac{\sum_{j=2}^{i} w_j}{t_i^a - t_1^a \pm \Delta_T^{\text{MAX}}}$), we can bound the maximum error on r:

- If $\Delta_T = -\Delta_T^{\text{MAX}}$, then $\Delta_r^{(1)} = r^2 \frac{\Delta_T^{\text{MAX}}}{\sum_{j=2}^{i} w_j - r\Delta_T^{\text{MAX}}}$
- If $\Delta_T = \Delta_T^{\text{MAX}}$, then $\Delta_r^{(2)} = -r^2 \frac{\Delta_T^{\text{MAX}}}{\sum_{j=2}^{i} w_j + r\Delta_T^{\text{MAX}}}$

As before, the upper bound on the measured capacity is $r_{meas} = r + \Delta_r^{(1)}$ and the lower bound is $r_{meas} = r + \Delta_r^{(2)}$ (Note that $\Delta_r^{(2)}$ is negative or zero).

For simplicity, we assume now that all the packets size are equal (i.e. $w_2 = w_3 = \ldots = w$). We define the queue time of one packet as Δ_{t^a}.

$$\Delta_{t^a} = \frac{w}{r}$$

Under such assumptions, we have:

$$\Delta_r^{(1)} = r^2 \frac{\Delta_T^{\mathrm{MAX}}}{\sum_{j=2}^{i} w_j - r\Delta_T^{\mathrm{MAX}}} = r^2 \frac{\Delta_T^{\mathrm{MAX}}}{(i-1)w - r\Delta_T^{\mathrm{MAX}}} = r \frac{\Delta_T^{\mathrm{MAX}}}{(i-1)\Delta_{CT} - \Delta_T^{\mathrm{MAX}}}$$

$$\Delta_r^{(2)} = -r^2 \frac{\Delta_T^{\mathrm{MAX}}}{\sum_{j=2}^{i} w_j + r\Delta_T^{\mathrm{MAX}}} = r^2 \frac{\Delta_T^{\mathrm{MAX}}}{(i-1)w + \Delta_T^{\mathrm{MAX}}} = -r \frac{\Delta_T^{\mathrm{MAX}}}{(i-1)\Delta_{CT} - \Delta_T^{\mathrm{MAX}}}$$

Figure 4.4 shows the plot of the error on r versus Δ_T^{MAX} for different values of i, i.e. for different lengths of the packet sequence. The figure shows that the longest B2 sequence can assure a better capacity estimation.

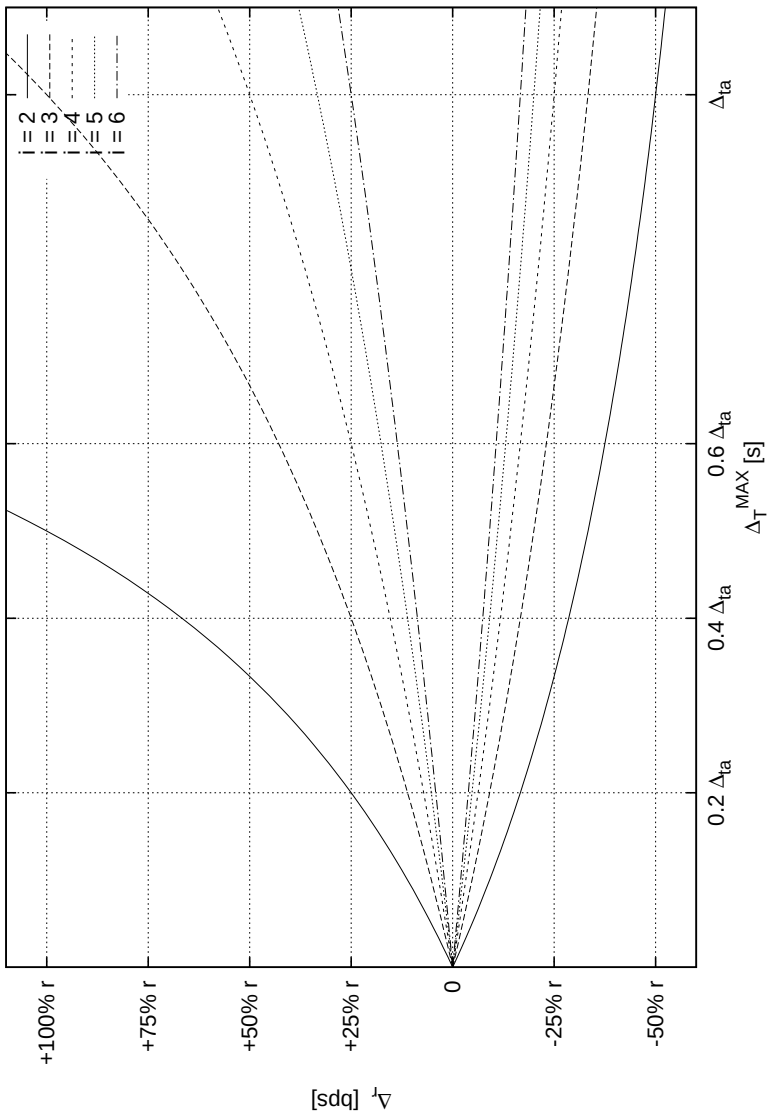

Figure 4.4: Multiple packets. $\Delta_r^{(1)}$ and $\Delta_r^{(2)}$ versus Δ_T^{MAX} for different values of i

4.2 General case

In the following, we devise the behaviour of the system described by Equations (4.1) in the general case, i.e. we start from a sequence of n triples to obtain the values of the unknown quantities T and r. In order to do that, it is necessary to linearize the system. We observe that it is possible to consider a sequence of data segments where all but the first segment arrive to a non-empty queue; we call such a sequence a 'Packet Burst' (PB). During a burst the max function first argument is always positive, making it possible to eliminate the max from the equation. So, we can take advantage of the burst in order to devise a solution for the system. In general, given a sequence of segments, we can identify a (possibly empty) set of bursts.

We suppose now that the segments in the sub-sequence $[i, i+1, i+2, \ldots, i+l-1]$ form a burst. To be more precise, we suppose that:

- Before the arrival of the forward segment i, the access downlink queue is empty ($q_i = w_i$);
- The queue does not empty up to the arrival of the segment $i + l - 1$;
- Before the arrival of the segment $i + l$ the queue is empty ($q_{i+l} = w_{i+l}$).

We divide the analysis in two phases. First we write an expression for the passing time of the acknowledgment segments of the pairs forming the burst. Second, we write an expression for the passing time of the l-th acknowledgment, the one that does not belong to the burst.

During the burst For the segments that form the burst, the system described in Equation (4.1) can be rewritten without the max function as:

$$\begin{cases} q_i = w_i \\ t_i^a = t_i^s + T + \xi_i^d + \xi_i^u + \frac{q_i}{r} = t_i^s + T + + \xi_i^d + \xi_i^u + \frac{w_i}{r} \end{cases}$$

$$\begin{cases} q_{i+1} = w_{i+1} + q_i - (t_{i+1}^s + \xi_{i+1}^d - t_i^s - \xi_i^d)r = \\ \qquad = w_{i+1} + w_i - (t_{i+1}^s + \xi_{i+1}^d - t_i^s - \xi_i^d)r \\ t_{i+1}^a = t_{i+1}^s + T + \xi_{i+1}^d + \xi_{i+1}^u + \frac{q_{i+1}}{r} = \\ \qquad = t_i^s + (T + \xi_i^d) + \xi_{i+1}^u + \frac{w_{i+1}+w_i}{r} \end{cases}$$

and so on. The generic term $i + m$ can be written as

$$\begin{cases} q_{i+m} = \sum_{j=i}^{i+m} w_j - (t_{i+m}^s + \xi_{i+m}^d - t_i^s - \xi_i^d)r \\ t_{i+m}^a = t_i^s + (T + \xi_i^d) + \xi_{i+m}^u + \frac{1}{r}\sum_{j=i}^{i+m} w_j \end{cases} \tag{4.5}$$

$$\forall m : 0 < m < l$$

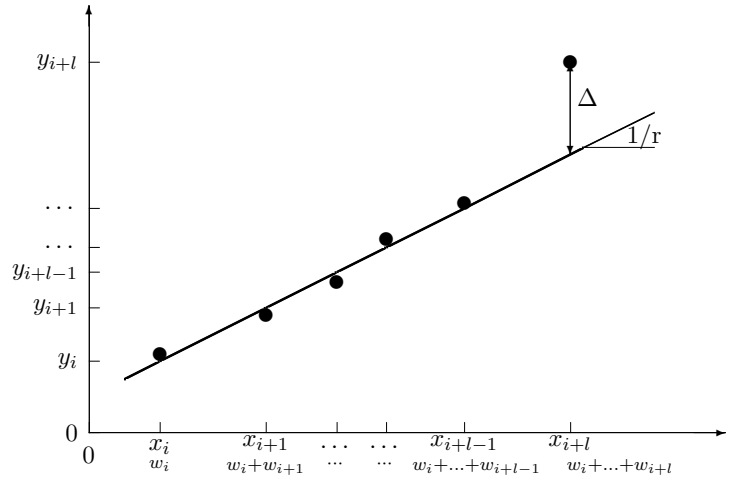

Figure 4.5: Linear relationship

If now we define $x_i \equiv \left(\sum_{j=i}^{i+m} w_j\right)$ and $y_i \equiv (t^a_{i+m} - t^s_i)$, we can write a linear relationship between x_i and y_i:

$$\underbrace{(t^a_{i+m} - t^s_i)}_{y_{i+m}} = (T + \xi^d_i) + \xi^u_{i+m} + \frac{1}{r}\underbrace{\sum_{j=i}^{i+m} w_j}_{x_{i+m}} \quad \forall m : 0 < m < l. \tag{4.6}$$

In short:

$$y_{i+m} = (T + \xi^d_i) + \xi^u_{i+m} + \frac{1}{r}x_{i+m} \tag{4.7}$$

Thus, as long as the queue does not empty, the points $\{x_i, y_i\}$ are approximately arranged on a line with slope $\frac{1}{r}$ and y-intercept $T+\xi^d_i$ (see Figure 4.5). The reciprocal of the slope of such a line represents the capacity of the downlink access queue (r). In order to devise the fitting line parameters, it is possible to apply the linear regression method[1] to equation (4.7).

Equation (4.7) shows that the noise on the downlink (ξ^d) affects all the points on the line in the same manner (in fact, ξ^d_i does not depend on m). This means the noise on the downlink affects only the estimation of the y-intercept T, but not the capacity estimation. Conversely, the noise on the uplink (ξ^u) affects all the points on the line. In other words, we can say that during a burst, the FIFO queue cancels the error on the downlink.

[1] See, for example, [14].

After the burst We now write the value of t^a_{i+l}, that is the y-value of the point that represents the $(i+l)$-th segment, the one that does not belong to the burst. For $m = l$ the segment arrives to an empty queue; we have that $q_{i+l} = w_{i+l}$ and the system becomes, for $m = l$:

$$\begin{cases} q_{i+l} = w_{i+l} \\ t^a_{i+l} = t^s_{i+l} + T\xi^d_{i+l} + \xi^u_{i+l} + \frac{w_{i+l}}{r} \end{cases}$$

obtaining:

$$\underbrace{(t^a_{i+l} - t^s_i)}_{y_{i+l}} = T + \xi^d_{i+l} + \xi^u_{i+l} + \frac{1}{r}\underbrace{\sum_{j=i}^{i+l} w_j}_{x_{i+l}} + \underbrace{\left[(t^s_{i+l} - t^s_i) - \frac{1}{r}\sum_{j=i}^{i+l-1} w_j\right]}_{\Delta}$$

One can easily verify that Δ is greater than zero (see Appendix A). Thus the point $\{x_{i+l}, y_{i+l}\}$ is not aligned with the previous ones, but it is shifted upward of a Δ quantity (see Figure 4.5).

In general, given a sequence of triples (w_i, t^a_i, t^s_i) it is possible to identify a set of bursts in which the linear relationship exists. As shown in Figure 4.6, such bursts form a set of fitting lines, at different y-intercepts, but with the same slope $(1/r)$.

4.2.1 Capacity estimation algorithm

In this section we propose an access downlink capacity estimation algorithm. The goal of the capacity estimation algorithm is to identify the maximum-sized subsequences of TCP segments in which the linear relationship described by Equation (4.6) is valid (i.e. the segments that form a burst) in order to estimate the access downlink capacity. The main idea is outlined in Figure 4.7. The algorithm takes as input n triples, that are the measurements performed during the observation period and identifies a number of maximum sized bursts (A and B in the figure). A quintuple (t^s, r, T, n, R^2) is associated with every discovered burst, where:

t^s is the passing time of the first forward segment in the burst.

r is the downlink access capacity obtained as a result of the linear regression performed on the burst (i.e. the reciprocal of the slope of the fitting line).

T is the fixed delay value (i.e. the y-intercept).

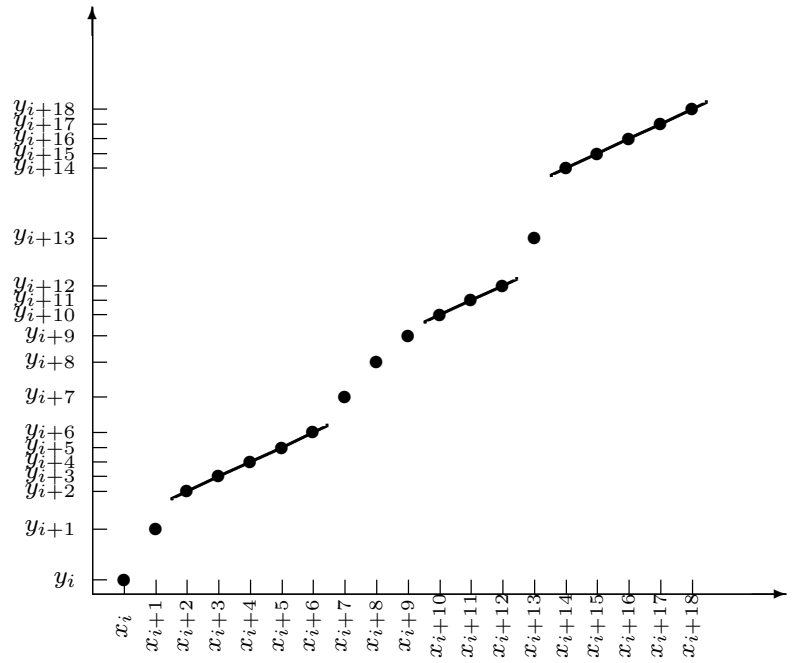

Figure 4.6: Linear relationship: multiple packet bursts

n is the number of forward segments that compose the burst.

R^2 is the coefficient of determination of the linear regression, that describes how well the linear models fit to the measured points.

The algorithm consists of successive linear regression tests over growing subsequences, to find the maximum-sized sub-sequence of segments that shows a 'good' fit to the linear model described in Equation (4.7).

The algorithm starts considering the sub-sequence composed of the first three forward segments ($[i, j], i = 0, j = 2$). At each iteration, the algorithm:

- Calculates the $\{x_k, y_k\}$ pairs over the considered interval, i.e. $\{x_k, y_k\}_{k=i \text{ to } j}$ according to x_i and y_i definition provided in chapter 4. The $\{x_k, y_k\}$ pairs are can be represented as points on a Cartesian graph.

- Performs a linear regression test on such points. Then:

  - if the linear regression is not good[2], the considered sub-sequence is shifted up by one ($i \leftarrow i + 1;\ j \leftarrow j + 1$) and the next iteration is started.

[2]The meaning of 'good' and 'not good' linear regression will be specified in Section 4.2.2.

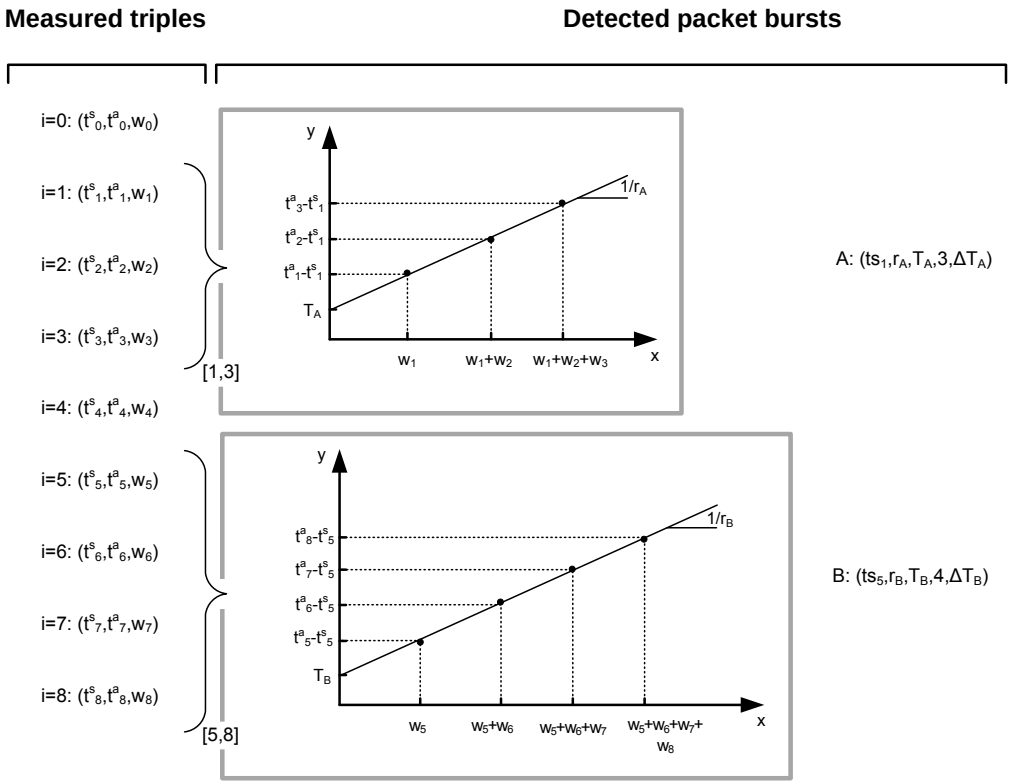

Figure 4.7: Capacity estimation algorithm outline

– if the linear regression is good, the quintuple associated with such a sub-sequence is saved, the sub-sequence is enlarged by one ($j \leftarrow j + 1$), and a new iteration is started; if the regression on the larger sub-sequence is good, the sub-sequence is enlarged another time, and so on. However, if the regression on the larger sub-sequence is not good, the last valid quintuple (the one found in a previous iteration) is retained, and the next three element sub-sequence is selected.

At the end of the iterations, the algorithm has identified several bursts, each one characterized by a quintuple.

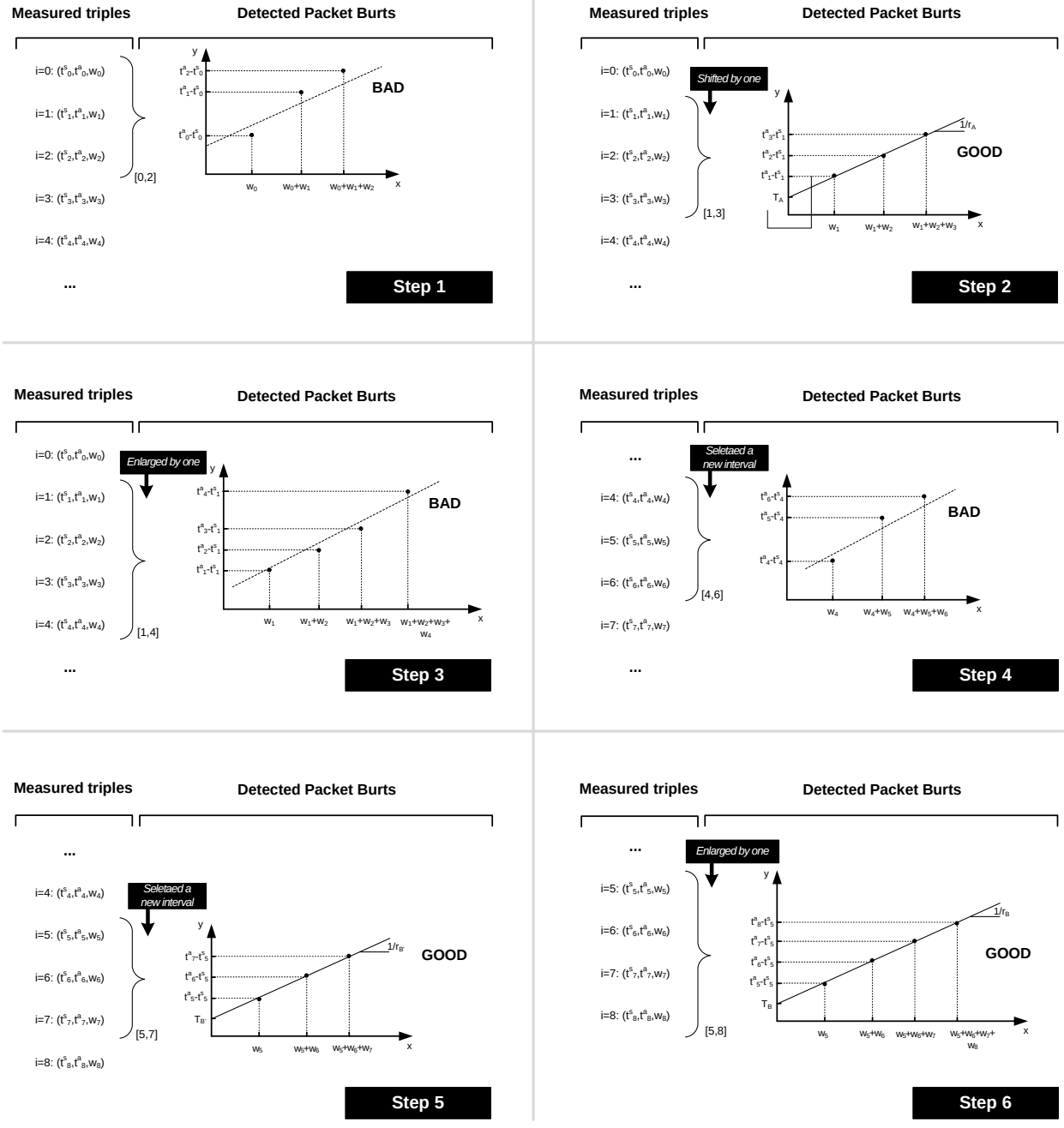

Figure 4.8: Capacity estimation algorithm: sample execution

4.2.2 Goodness of fit

The capacity estimation algorithm needs to discriminate between a good and a bad fit of the measured points to the linear model. To do so it is necessary to calculate the coefficient of determination R^2, that is defined as follows:

$$R^2 \equiv 1 - \frac{\sum_{j=i}^{i+m}(y_j - \hat{y_j})^2}{\sum_{j=i}^{i+m}(y_j - \bar{y})^2}$$

where $\hat{y_i}$ is the value predicted by the linear model (i.e. $\hat{y_j} \equiv T + \frac{1}{r}x_j$) and $\bar{y}$ is the mean of the y_j ($\bar{y} \equiv \frac{\sum_{j=i}^{i+m} y_j}{m}$). The coefficient of determination value is between 0 and 1, where 1 means that the fit line passes exactly through the measured points. Thus, it is necessary to set a threshold R_t^2, and considering a regression with a value greater or equal to the threshold to be a 'good' regression. The value of the threshold depends on the application, and must be appropriately tuned.

Chapter 5

Analysis: interfering traffic

This chapter presents an analysis of the effects of the interfering traffic on the uplink and on the downlink of the access line considered in the scenario presented in Chapter 2. The interfering traffic on the downlink access queue, analyzed in Section 5.1 is the traffic that arrives to the downlink access queue without passing through the TMS. This type of interfering traffic can cause a disruptive effect on the detection of the packet burst, or a so-called 'false positive', i.e. a pattern of measured traffic and interfering traffic that leads to a capacity underestimation. The interfering traffic on the uplink access queue, analyzed in Section 5.2 is composed by the data traffic generated on the CPG, that can cause the queueing of the ACK packet on the uplink access queue, namely the 'ACK compression' phenomenon.

5.1 Interfering traffic on the access downlink queue

In this section we analyze the effect of the interfering traffic on the access downlink queue, i.e. the effect of the TCP forward segments that arrive to the access downlink queue without passing through the TMS. As a consequence, the access queue could contain interleaved traffic coming from different paths, possibly invalidating the capacity estimation method described in Section 4.2.1. Consider for example, at a given moment, the queue contains the traffic pattern

$$\texttt{M M I M I I I M M} \ldots$$

where M denotes a segment that passed from the measurement node, and I denotes an interfering traffic segment. Such a pattern is likely to destroy the linear relationship described by Equation (4.7), invalidating the capacity estimation method. The interfering traffic on the downlink has two possible outcomes:

- The measurements on a given segment sub-sequence show a burst condition. This can be due to two causes:

  - There is no interfering traffic and the measured traffic produces by itself a burst on the downlink queue. In this case, the capacity estimation method measures the correct access link capacity.
  - The interfering traffic and the measured traffic are shaped to cause a false positive, i.e. a pattern of measured and interfering traffic that leads to a burst condition with the wrong capacity. We will discuss how to detect false positives in the following.

- The measurements on the TMS during a given segment sub-sequence do not show a burst. This can be due:
  - to measured traffic not producing by itself a packet burst on the downlink queue, or
  - to interfering traffic destroying the linearity on the downlink queue.

  In both cases, the fact that we do not observe a burst condition leads us to discard the measurements.

Summarizing, in order to perform a correct capacity estimation in the presence of interfering traffic on the downlink, it is necessary to detect the false positives in order to filter them.

5.1.1 False positives detection

We now provide an analysis of the false positives, i.e. of the pattern of measured traffic and interfering traffic that cause a wrong estimation of the link capacity.

The Equation (4.7), that provides an expression for the acknowledgment passing times during a burst, can be modified to take into account the interfering traffic on the downlink. Using v_j as the total size of the interfering traffic that arrives to the downlink queue between the arrival of $(j-1)$-th measured segment and the j-th measured segment, we obtain:

$$t^a_{i+m} = t^s_i + (T + \xi^d_i) + \xi^u_{i+m} + \frac{1}{r}\sum_{j=i}^{i+m}(w_j + v_j)$$

$$\forall m : 0 < m < l$$

However, as we stated before, the capacity estimation algorithm has the possibility to monitor only the traffic that passes through the monitoring system. So, the system equation seen by the algorithm is the following:

$$t^a_{i+m} = t^s_i + (T + \xi^d_i) + \xi^u_{i+m} + \frac{1}{r^*}\sum_{j=i}^{i+m}(w_j)$$

Packet pattern	r^*
M I M I M I	$\frac{1}{2}r$
M I I M I I M I I	$\frac{1}{3}r$
M I I I M I I I M I I I	$\frac{1}{4}r$
...	...

Table 5.1: False positive patterns

$$\forall m : 0 < m < l$$

with a capacity (r^*) different than the real one. It is easy to devise the necessary condition for a false positive:

$$\frac{V_i}{w_i} = \frac{V_{i+1}}{w_{i+1}} = \frac{V_{i+1}}{w_{i+2}} = \dots \tag{5.1}$$

The estimated (wrong) capacity will be:

$$r^* = \underbrace{\frac{w_i}{V_i + w_i}}_{<1} \; r$$

Moreover, as we know the vast majority of TCP data segments are about 1500 bytes long [38], the denominator of the aforementioned formula can assume only a value that is an integer multiple of 1500. The segment patterns that can cause a false positive and their corresponding wrong rates are summarized Table 5.1.

In summary, the probability of false positives depends on the segment pattern of the measured and interfering traffic on the access downlink queue. However, as shown before, a false positive always causes a capacity under-estimation, with a capacity less than or equal to half of the real one. Additionally, as we will show in Chapter 7, the delayed acknowledgement mechanism implemented by TCP can raise the maximum wrong capacity to $\frac{2}{3}r$. The capacity underestimation can be exploited in order to filter out the false positives, as we will show in Chapter 6.

5.2 Interfering traffic on the access uplink queue

The analysis carried out in Chapter 4 is based on the assumption that the acknowledgement segments are never queued on the uplink access link. This means, if we ignore the effect of the upstream path noise, the acknowledgment segments arrive to the TMS at the same rate they were sent by the receiver. For that reason, such an analysis ignores the role of the access uplink queue's possible congestion, in particular regarding the possible interfering traffic on the uplink, i.e. data segments flowing through the uplink. There are two possibilities:

- The interfering traffic on the uplink modifies the ACK rate destroying the linearity. In this case, the segment sequence is discarded by the linear regression method, without causing a wrong capacity estimate.

- The interfering traffic on the uplink causes the ACK-compression phenomenon.

The TCP ACK-compression, widely studied in literature (see, for example [40]) consists of a reduction of the time spacing between successive ACKs due to a congested uplink access queue. In the typical ADSL scenario the ACK compression is often triggered by the data traffic on the uplink queue, that can interfere with the ACK traffic (for example during a file upload, or when a file sharing application is active).

In order to understand the impact of ACK compression on the capacity estimation method described in Chapter 4, we consider an example. Let us consider a burst composed by n segments $[0, 1, 2, \ldots, n-1]$ having the following properties:

- The access uplink queue contains w_{int} bits at the arrival of the first ACK to the uplink queue.

- w_{int} is large enough to remain in the uplink queue for sufficient time to cause the buffering of all the burst acknowledgment segments in the uplink queue. This condition is frequent on asymmetric access links in the presence of uplink data traffic due to the fact that the uplink capacity is usually large enough for the reverse-flow ACK packets, but limited for the uplink data traffic.

For simplicity, let us suppose that $w_0 = w_1 = w_2 = \ldots = w$

Let r^u be the uplink capacity and w_{ack} the size of the acknowledgement packets. From Equation (4.1) we obtain:

$$\begin{aligned} t_0^a &= t_0^s + T + \frac{w}{r} + \frac{w_{int}}{r^u} \\ t_1^a &= t_0^s + T + \frac{w}{r} + \frac{w_{int}}{r^u} + \frac{w_{ack}}{r^u} \\ t_2^a &= t_0^s + T + \frac{w}{r} + \frac{w_{int}}{r^u} + 2\frac{w_{ack}}{r^u} \\ &\ldots \end{aligned}$$

The time interval between ACKs will be:

$$\Delta_a = t_1^a - t_0^a = t_2^a - t_1^a = \ldots = \frac{w_{ack}}{r^u}$$

The capacity obtained by the estimation method (r^*) will be:

$$r^* = \frac{w}{t_1^a - t_0^a} = \frac{w}{\Delta_a} = \frac{w}{w_{ack}} r^u \tag{5.2}$$

that does not depend on the real r, so it is not correct.

On the other hand, the estimation of the network delay T will be:

$$T_0^* = t_0^a - t_0^s - \frac{w}{r^*} = t_0^a - t_0^s - \frac{w_{ack}}{r^u}$$
$$T_1^* = t_1^a - t_0^s - \frac{2w}{r^*} = t_1^a - t_0^s - \frac{2w_{ack}}{r^u}$$
$$\ldots$$

The error on T will be:

$$\Delta_T = T_0^* - T = \frac{w}{r} + \frac{w_{int}}{r^u} - \frac{w_{ack}}{r^u}$$

For example, if we consider a TCP connection ($w \simeq 12000$ bit, $w_{ack} \simeq 320$ bit) on a typical residential ADSL with $r = 7$ Mbps, $r^u = 384$ kbps, we obtain a wrong capacity estimation:

$$r^* = 14.4 \text{ Mbps}$$

In general, the downlink/uplink capacity ratio of the ADSL lines is not bigger than 20. So, it is easy to see the r^* is always bigger than r, i.e. the ACK compression phenomenon always causes a capacity over-estimation on asymmetric access lines. Now, suppose the interfering traffic on the uplink queue that causes the ACK compression is composed by a single full data packet ($WS_{int} = 12000$). We obtain:

$$\Delta_T \simeq 40 \text{ ms}$$

In other words, the ACK compression phenomenon causes an overestimation on r and a large over-estimation on T. It can be noted that the estimated capacity in the presence of ACK compression is always r^*, irrespective of the size of the interfering traffic. This fact can be exploited to detect the ACK compression, as we will show in Chapter 6.

Chapter 6

Model driven data analysis

The traditional approach to capacity estimation usually consists of packet timing analysis and various statistical approaches to data filtering and refining. It is believed that a model driven data analysis can be very effective in order to filter out the wrong capacity estimations, taking advantage of the knowledge of the TCP/IP network environment characteristics.

In section 4.2.1 we provided a method based on the linear regression aimed to extract a set of bursts from a sequence of measured triples (t_i^s, t_i^a, w_i), obtaining a set of quintuples (t^s, r, T, n, R^2) each one representing a burst. Figure 6.1 shows a histogram of the r values of the quintuples obtained by a typical outcome of the linear regression method performed in the presence of a busy access link. The histogram shows a number of samples around the true capacity value, an overestimation peak due to the ACK compression phenomenon (around the r^* value provided by Equation (5.2)) and a number of samples that underestimate the true capacity, due to the interfering traffic on the downlink.

In this chapter we will provide a set of heuristics that appropriately combine such results in order to obtain the access capacity estimation. In particular, the *Maximum delay heuristics* is aimed to filter out the overestimation samples, while the *Minimum rate heuristics* and the *Maximum rate heuristics* are aimed to filter out the underestimation.

In summary, we propose the following procedure aimed at obtaining a capacity estimation in the presence of interfering traffic.

1. Apply the capacity estimation algorithm based on the linear regression, obtaining a set of quintuples representing a packet burst.

2. Use the heuristics described in this chapter in order to filter out the quintuple that could have been affected by a interfering traffic.

Phenomenon	Action
Packet sequences that do not form a Packet Burst	Filtered by the regression
Interf. traffic on the downlink access queue that destroys the linearity	Filtered by the regression
Interf. traffic on the downlink access queue causing a false positive	Filtered by the min. rate heuristics and by the max. rate heuristics
Additive noise on the downlink	Cancelled by the FIFO queue
Interf. traffic on the access uplink queue that destroys the linearity	Filtered by the regression
Interf, traffic on the access uplink queue causing an ACK compression	Filtered by the maximum delay heuristics (T_{max})
Additive noise on the uplink that destroys the linearity	Filtered by the regression

Table 6.1: Summary of the different issues that can cause wrong capacity estimations

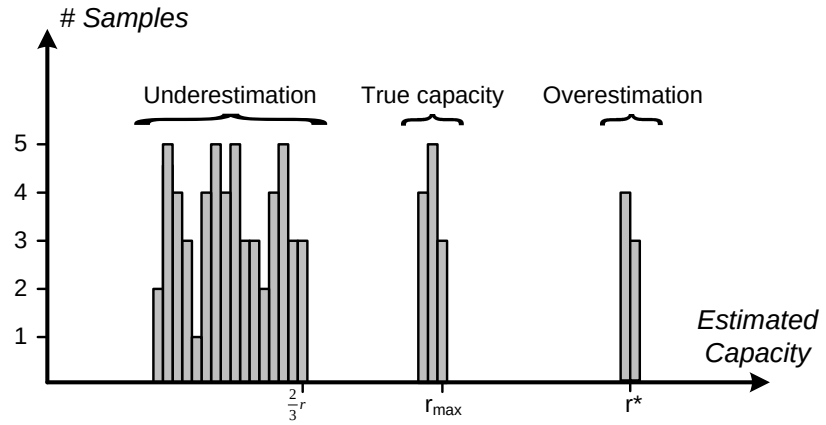

Figure 6.1: Outcome of the linear regression method on a busy link

6.1 Maximum delay heuristics

We define T_{max} as the smallest round trip time $(t_i^a - t_i^s)$ during the measurement period:

$$T_{max} \equiv \min_{0 \leq i \leq n-1} (t_i^a - t_i^s)$$

As shown in Equation (4.1a), the round trip time is the sum of the fixed network delay (T), the queuing delay, and the noise. So, the round trip times are always greater than T, and T_{max} represents an upper bound on a T estimate.

In Section 5.2 we have shown that the traffic on the uplink can cause the ACK compression phenomenon on the uplink access queue, leading to a wrong capacity estimate that depends only on the uplink queue capacity (r^u). We have also showed that on a typical access link scenario the ACK compression also causes a large overestimation of T. This fact permits filtering out the quintuples that incurred in the ACK compression, i.e. the quintuples that show an estimated T value bigger than T_{max}; it can be noted that such quintuples will form a peak on the histogram of estimated rate.

As the T estimations are affected by the path noise, we observed that it is safer to consider a threshold that is 10% more than the T_{max}. So, the *maximum delay heuristics* consists of:

Discarding every quintuple having a T value bigger than $(T_{max} \cdot 1.1)$.

6.2 Minimum rate heuristics

Equation (3.1) in Chapter 3 defines the quantity $\hat{r}$ as follows

$$\hat{r} \equiv \max_{i\geq 1, j\leq n;\, i\leq j} \left(\frac{\sum_{k=i}^{j} w_k}{t_j^a - t_i^s} \right) = \max_{i\geq 1, j\leq n;\, i\leq j} \left(\frac{\sum_{k=i}^{j} w_k}{\Delta_{i,j}^s} \right)$$

In general, we have that $\hat{r} \leq r$, i.e. $\hat{r}$ is a lower bound on r. So, $\hat{r}$ can be a good approximation for the rate only if the measured traffic saturates the downlink access queue for a long time interval. However, being $\hat{r}$ a lower bound of r, it can be used to filter out some quintuples. The *minimum rate heuristics* consist of:

> *Discarding every quintuple having an r value smaller than $\hat{r}$.*

6.3 Maximum rate heuristics

As reported in Section 5.1, the interfering traffic on the downlink can have two possible outcomes:

- The interfering traffic destroys the linearity. So, the sub-sequence is discarded by the linear regression method.
- The interfering traffic causes a false positive, i.e. a linear relationship with an estimated capacity lesser than the real one.

The aim of the maximum rate heuristics is to filter out the false positives.

In Section 6.1, we proposed the maximum delay heuristics, aimed to discard all the capacity over-estimations. So, after having applied such heuristics, we expect that no other phenomenon can cause a rate overestimation. Additionally, as we will show in Chapter 7, the underestimation samples have a maximum capacity of $\frac{2}{3}r$. So, it is possible to determine the capacity estimation by selecting the higher capacity mode on the histogram (tagged 'True capacity' in Figure 6.1). So, the *maximum rate heuristics* consist of:

> *After having applied the* maximum delay heuristics *and the* minimum rate heuristics*, the true capacity is associated with the maximum rate mode on the rate histogram.*

In order to do so we propose the following algorithm:

1. Let r_{max} be the capacity of the maximum capacity quintuple.
2. We consider as *valid* all the quintuples having $r \geq (r_{max} \cdot 0.8)$.
3. The final estimation of the capacity will be the average of all the valid quintuple capacity values.

6.4 Summary

Table 6.1 summarizes the phenomena that can be an issue on the capacity estimation process, and the mechanism that can be exploited to filter them out in order to obtain a capacity estimation.

Chapter 7

Delayed acknowledgement

The TCP specifications offer the possibility of employing a technique called 'delayed ACK'; using such technique, "a host that is receiving a stream of TCP data segments can increase efficiency in both the Internet and the hosts by sending fewer than one ACK (acknowledgment) segment per data segment received" [3]. The specifications allow a host to send an ACK at most, every two full sized incoming TCP segments and the host can not delay an ACK more than 500 ms [1] (most implementations lower this limit to 200 ms. Our tests show that the delayed ACK technique is implemented on the majority of modern operating system TCP stacks.

In the presence of delayed ACK, the rate estimation algorithm can not measure the acknowledgment time of every TCP segment, but possibly one of every two. However, as we will see, the properties of the FIFO queue make it possible to overcome this problem. Suppose that six successive TCP segments form a burst on the downlink queue, and that, due to delayed ACKs, only the second, the fourth and the sixth generate an ACK. The equations that describe the system are:

[1] For the exact meaning of 'full sized segment', see [1]

$$\begin{cases} q_1 = w_1 \\ t_1^a = t_1^s + T + \frac{w_1}{r} \end{cases}$$
$$\begin{cases} q_2 = w_2 + w_1 - (t_2^s - t_1^s)r \\ t_2^a = t_1^s + T + \frac{w_2+w_1}{r} \end{cases}$$
$$\begin{cases} q_3 = w_3 + w_2 + w_1 - (t_3^s - t_1^s)r \\ t_3^a = t_1^s + T + \frac{w_3+w_2+w_1}{r} \end{cases}$$
$$\begin{cases} q_4 = w_4 + w_3 + w_2 + w_1 - (t_4^s - t_1^s)r \\ t_4^a = t_1^s + T + \frac{w_4+w_3+w_2+w_1}{r} \end{cases}$$
$$\begin{cases} q_5 = w_5 + w_4 + w_3 + w_2 + w_1 - (t_5^s - t_i^s)r \\ t_5^a = t_1^s + T + \frac{w_5+w_4+w_3+w_2+w_1}{r} \end{cases}$$
$$\begin{cases} q_6 = w_6 + w_5 + w_4 + w_3 + w_2 + w_1 - (t_6^s - t_1^s)r \\ t_6^a = t_1^s + T + \frac{w_6+w_5+w_4+w_3+w_2+w_1}{r} \end{cases}$$

As we don't know t_1^a, t_3^a, t_5^a, we have to rely only on the three following equations:

$$\begin{aligned} t_2^a - t_1^s &= T + \tfrac{w_2+w_1}{r} \\ t_4^a - t_1^s &= T + \tfrac{w_4+w_3+w_2+w_1}{r} \\ t_6^a - t_1^s &= T + \tfrac{w_6+w_5+w_4+w_3+w_2+w_1}{r} \end{aligned}$$

In other words, in order to apply the rate estimation to a TCP stream containing delayed ACKs, it is necessary to apply the rate estimation method to the following triples:

$$\begin{aligned} &(w_1 + w_2, t_2^s, t_2^a) \\ &(w_3 + w_4, t_4^s, t_4^a) \\ &(w_5 + w_6, t_6^s, t_6^a) \end{aligned}$$

So, a minimum of six successive TCP segments forming a packet burst are necessary to perform the capacity estimation method in such a case.

The delayed ACK mechanism influences also the effect of the interfering traffic on the downlink as described in Section 5.1. In fact, in presence of delayed ACKs, there are more combinations of interfering traffic and measured traffic that can lead to a false positive. In particular, the maximum rate for a false positive in the presence of delayed ACK is caused by the following traffic pattern on the downlink queue:

$$\mathtt{M\ I\ M^*\ M\ I\ M^*\ M\ I\ M^*\ M\ \ldots}$$

where `M*` represents a measured packet that does not receive an ACK, `M` represents a measured packet that does receive an ACK, and `I` is an interfering traffic packet. It is easy to show that in this case the estimated rate will be $r^* = \frac{2}{3}r$.

Chapter 8

Capacity estimation architecture

The capacity estimation algorithm presented in Section 4.2.1 is based on the capture and analysis of the TCP packets travelling through an Internet node (router or switch), tagged MN in Figure 2.1. Monitoring one node permits estimation of the capacity of all the access links placed downstream of that node. So, placing the TMS on a high capacity node make it possible to estimate simultaneously the capacity of a high number of access links. Each access link is usually terminated by a residential router, which can be identified by its IP address.

The first step of such a process consists of capturing all the frames travelling on the MN, and this has to be done at wire speed. The second step consists of analyzing the captured packets. For every access link it is necessary to correlate the forward TCP packets with the corresponding acknowledgement packets (using the TCP session identifier[1] and the TCP sequence number) in order to extract a set of triples. Finally, the triples are used to feed the estimation algorithm presented in Section 4.2.1.

Performing such tasks effectively is not trivial, in particular if the MN is placed on a high capacity link (e.g. a link with a capacity greater than or equal to 1 Gbps). For example, the backbone of an Internet Service Provider or the connection links between a provider and a Neutral Access Point (NAP) are usually built on Gigabit links.

Section 8.1 presents the main issues related to packet capture on a high-capacity link. Section 8.2 suggests some architecture to perform packet analysis by means of the capacity estimation algorithm.

[1] Source address, source port, destination address, destination port.

8.1 Packet capture

The packet capture at wire speed can be a challenging task on the ever increasing link speeds available on the backbones of the Internet Service Providers. The main concerns are to avoid packet losses and to obtain a precise timestamping for the captured packets.

Enterprise-grade network equipment (router and switches) offer the possibility of collecting traffic information and exporting them in a compressed format (see for example sFlow [32], NetFlow [43]). However, such a task can be computationally expensive for the device hardware, so in general it is only possible to extract a sample of the packets, looking at a packet every N. Additionally, in some cases the activation of the such traffic collector can slow down the device performance. While in general a packet sample can be useful to perform traffic characterization (see for example [8]), it is not usually useful for the purpose of capacity estimation; this is true also for the capacity estimation algorithm presented in Section 4.2.1, that relies on the capture of all the packets that involve a given access link.

Due to the problems of automated collection, in order to perform a full capture at wire speed it is necessary to replicate all the traffic passing through a router port and send it to a network interface. This can be done in two ways:

1. Exploiting a dedicated mirror port on the monitored device. (See Figure 8.1).
2. Using a fiber-optic splitter, namely a 'passive tap'. (See Figure 8.2).

Mirror ports are dedicated ports present on high-end switches and routers that provide the replication of a regular port. However the replication task can load the CPU, degrading the performance of the devices, and leading in some cases to packet losses on the mirror port. Additionally, the timestamp information can be altered by the mirroring process. On the other hand, a fiber optic splitter is a passive device that allows the redirection of a small fraction of the fiber light of a port without impacting the traffic on that port. So, the splitter solution is usually the most convenient one.

After having obtained the traffic replication by means of mirroring or fiber tapping, there are two possibilities regarding the actual packet capture phase.

1. Using a standard network card, discussed in section 8.1.1.
2. Using a dedicated acquisition card, discussed in section 8.1.2.

8.1.1 Packet capture using standard Network Interface Cards (NIC)

Packet capture of high-data rate links using standard Network Interface Cards (NIC) can be a challenging task. The limiting factors resides in the hardware performances (PCI bus throughput, memory bandwidth, disk system data rate) and

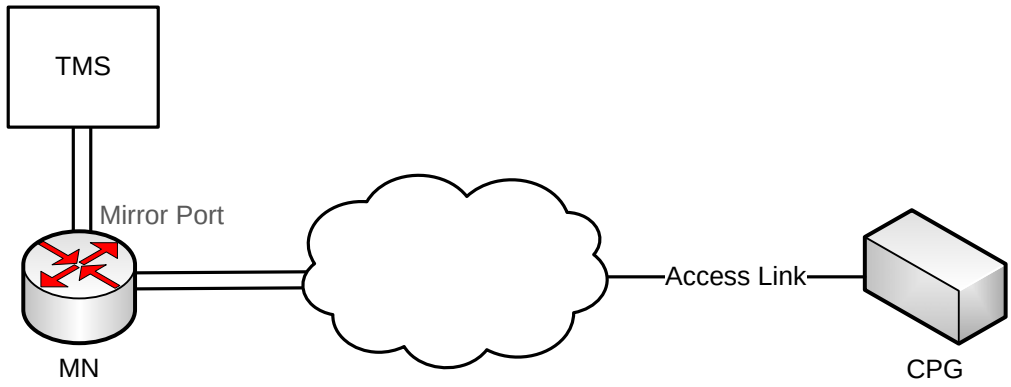

Figure 8.1: Traffic capture using a mirror port

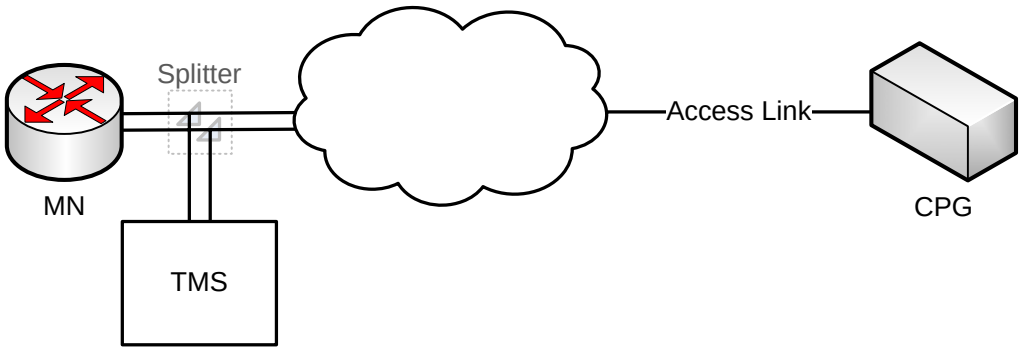

Figure 8.2: Traffic capture using a fiber-optic splitter

software performances. Mogul [28] noted that the capture on high capacity links can cause a bad performance of the interrupt handling architecture of the operating system: if the packet arrives too fast the CPU spends all its time to process the related interrupts. Several solutions have been proposed in order to mitigate that problem (smart device polling instead of interrupt-driver operation [35], optimization of the packet passing between kernel-space and user-space by means of a memory mapped approach [10, 39]). Taking advantage of such enhancements, the authors of [36] report that a PC with standard network interfaces can capture full packet traces to disk up to a rate of 600 to 700 Mbps. However, no claims are made about the precision of the packet timestamps.

8.1.2 Packet capture using dedicated cards

A different approach to packet capture is to use dedicated hardware (card). Several solutions are present in the market (e.g. [44], [45]), reaching a capture speed of 10 Gbps. A dedicated board is characterized by efficient on-board packet filtering and data reduction, i.e. it is possible to extract only the packet headers in hardware, passing to the software layer a reduced quantity of information. Additionally, the dedicated hardware can guarantee a precise timestamping, that is crucial to apply the capacity estimation algorithm. The main drawback of such an approach is the sensible price of dedicated cards[2].

8.2 Packet analysis

The output of the packet capture hardware consists of only the headers of each captured packet. This permits the reduction of the amount of information. For example, a 1500 byte TCP frame over Ethernet is composed by around 60 bytes of headers (MAC+IP+TCP), leading to a compression factor of 1/25. The rate estimation algorithm needs to work on all the packets that come or go to a given host, identified by its IP address. This fact means that the computation of a large amount of packets can be easily parallelized over a set of different machines by dividing the packets based on IP address range. This scenario, that we call 'real time analysis' is depicted in Figure 8.3: the output of the capture hardware (only headers) is sent to a set of machines for computation.

Another possibility, depicted in Figure 8.4, is the deferred analysis: the only-headers information is dumped on a disk (or disk-system) to be analyzed off-line later.

[2] A dedicated card costs about $5000.

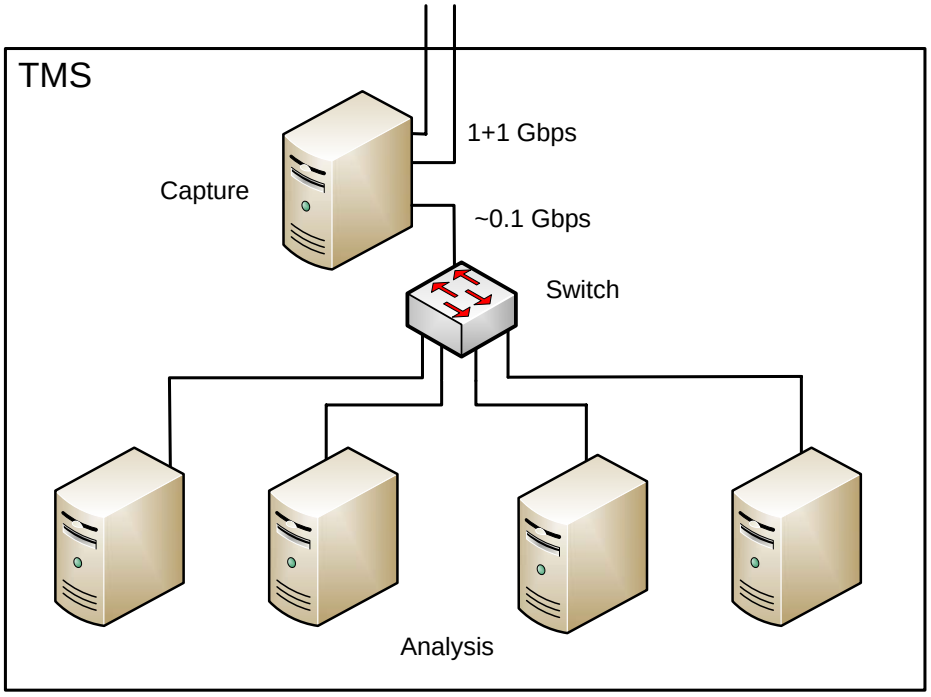

Figure 8.3: Traffic Monitoring System architecture, real time analysis

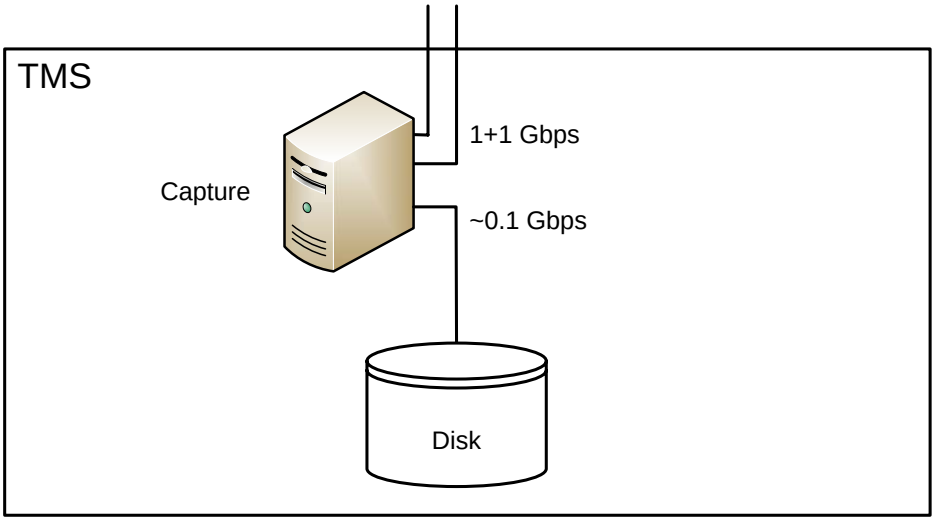

Figure 8.4: Traffic Monitoring System architecture, deferred analysis

Chapter 9

Experiments

Chapter 8 presented an overview of the issues related to packet capture and analysis in high-speed network links. On the other hand, this chapter presents some small-scale experiments aimed at validating the approach presented in this thesis.

The tests have been performed on two residential ADSL lines under different interfering traffic conditions. The measurements arrangement is shown in Figure 9.1. A web server has been placed in a university laboratory, attached to a well-provisioned link, acting as measurement node. A script has been installed on a PC placed in the customer premises, in order to create various traffic conditions.

9.1 Test 1A

The first test has been performed on a 3.5 Mbps downstream / 384 kbps upstream ADSL line, with the TMS placed on a web server 20 hops away; the results are reported in Table 9.1, page 79. The 3.5 Mbps is the average capacity previously obtained by means of large download from well-provisioned websites.

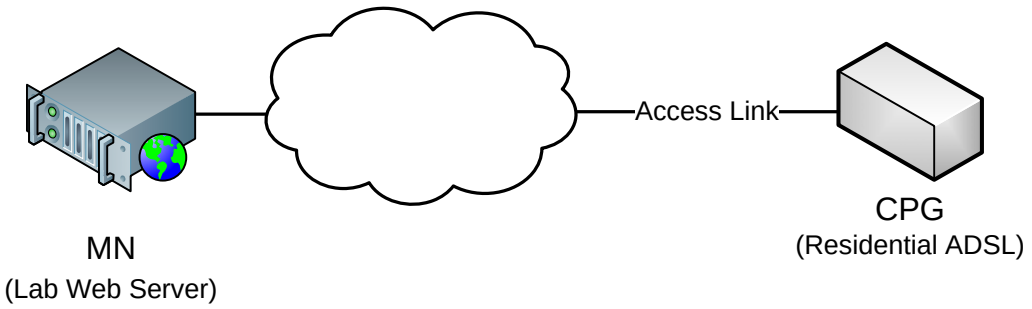

Figure 9.1: ADSL experiments arrangement

Operating conditions:

- No sources of interfering traffic
- Measured traffic composed by some HTTP GETs to a website, with a total of 160 measured triples.

This test has been without interfering traffic sources. For this reasons, the algorithm located two long bursts ($n = 64$ and $n = 67$). The estimated capacity is very good, being 3.44 Mbps (2% error).

9.2 Test 1B

The second test has been set up on the same ADSL line of Test 1B. In this case a large file download was active on the ADSL host, with the file being downloaded placed on a public web server. This caused high interfering traffic on the downlink access queue. The results are summarized in Table 9.2, page 79.

Operating conditions:

- Large file download active on the ADSL host (meaning high probability of interfering traffic on the downlink access queue)
- Measured traffic composed by some HTTP GETs to a website, with a total of 120 captured TCP data segments.

It can be noted that only three short bursts ($n = 3$) passed the heuristics, with an average capacity of 3.28 Mbps (6% error).

9.3 Test 2

Test 2 has been performed on a 7 Mbps downstream / 512 kbps upstream ADSL line, with the TMS placed on a web server 20 hops away; the results are reported in Table 9.3, page 80. Operating conditions:

- File sharing application active on the ADSL host (meaning high probability of interfering traffic on the uplink).
- Measured traffic composed by some HTTP GETs to a website, with a total of 130 captured TCP data segments.

It can be noted (first line of Table 9.3) that the interfering traffic on the uplink caused an over estimation, correctly filtered by the maximum delay heuristics. The estimated capacity is 6.94 Mbps (1% error).

160 triples $R_t^2 = 0.99$
$T_{max} = 43.8$ ms $r_{min} = 3.3$ Mbps

n	r [Mbps]	T [ms]	R^2	Action
3	**3.59e+06**	40.919	0.994	OK
5	**3.44e+06**	40.890	0.998	OK
64	**3.42e+06**	41.177	0.991	OK
3	**3.37e+06**	39.815	0.999	OK
5	**3.40e+06**	40.232	0.998	OK
67	**3.43e+06**	37.174	0.991	OK

$r = 3.44$ Mbps

Table 9.1: Test 1A

60 triples $R_t^2 = 0.99$
$T_{max} = 46.8$ ms $\hat{r} = 1.81$ Mbps

n	r [Mbps]	T [ms]	R^2	Action
3	0.15e+06	58.193	0.999	Filtered by max delay
3	2.46e+06	42.589	0.997	Filtered by max rate
5	2.60e+06	37.651	0.998	Filtered by max rate
3	**3.20e+06**	43.226	0.996	OK
3	1.86e+06	42.059	0.990	Filtered by max rate
5	3.33e+06	59.310	0.999	Filtered by max delay
4	3.39e+06	62.508	0.999	Filtered by max delay
4	1.49e+06	41.241	0.995	Filtered by min rate
3	**3.20e+06**	46.275	0.999	OK
3	**3.43e+06**	47.837	0.999	OK
3	1.40e+06	44.910	0.996	Filtered by max rate

$r = 3.28$ Mbps

Table 9.2: Test 1B

64 triples $R_t^2 = 0.99$
$T_{max} = 33.4$ ms $r_{min} = 5.22$ Mbps

n	r [Mbps]	T [ms]	R^2	Action
3	14.80e+06	45.159	1.000	Filtered by max delay
5	**6.85e+06**	33.976	0.996	OK
3	**6.62e+06**	34.107	0.999	OK
3	13.62e+06	38.887	0.998	Filtered by max delay
6	**6.93e+06**	34.278	0.998	OK
7	7.23e+06	38.834	0.993	Filtered by max delay
11	**7.35e+06**	36.695	0.997	OK

$\boldsymbol{r = 6.94}$ **Mbps**

Table 9.3: Test 2

Chapter 10

Conclusion

A novel approach to passive downlink access capacity has been proposed. The advantage of such an approach is the possibility of exploiting the existing TCP connections passing from a backbone network node in order to estimate the access capacity of a large number of hosts placed downstream of that node.

The method should be useful for the purpose of service level agreement compliance verification, in the development of new capacity-aware network protocols, and makes possible large-scale studies aimed at estimating the access capacities of a large number of access links, due to its passiveness and scalability.

Future work may include a large-scale study aimed at characterizing the access capacities of a large number of residential and commercial end users without the need of injecting traffic on the network.

Appendix A

Sign of Δ

This Appendix demonstrates that the Δ term in Equation (4.7) is positive or zero. In fact, for $m = l$ the max term of Equation (4.1) is zero, so the argument of the max is less than or equal to zero:

$$\begin{aligned} &\max\left(q_{i+l-1} - (t^s_{i+l} - t^s_{i+l-1})r\right) = 0 \;\Rightarrow \\ &q_{i+l-1} - (t^s_{i+l} - t^s_{i+l-1})r \leq 0 \end{aligned}$$

for the term q_{i+l-1} it is possible to use the form of Equation (4.5):

$$\sum_{j=i}^{i+m-1} w_j - (t^s_{i+m-1} - t^s_i)r - (t^s_{i+m} - t^s_{i+m-1})r \leq 0$$

Dividing both members by r we have:

$$\underbrace{\frac{1}{r}\sum_{j=i}^{i+m-1} w_j - (t^s_{i+m} - t^s_i)}_{-\Delta} \leq 0 \quad \Longrightarrow \quad \Delta \geq 0$$

Bibliography

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999. Updated by RFC 3390.

[2] J.C. Bolot. Characterizing end-to-end packet delay and loss in the internet. *Journal of High Speed Networks*, 2(3):289–298, 1993.

[3] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379.

[4] R.L. Carter and M.E. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27(28):297–318, 1996.

[5] L.J. Chen, T. Sun, G. Yang, M.Y. Sanadidi, and M. Gerla. End-to-end asymmetric link capacity estimation. In *IFIP Networking*. Springer, 2005.

[6] P. Chimento and J. Ishac. Defining Network Capacity. RFC 5136 (Informational), February 2008.

[7] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki, and Akira Kato. The impact and implications of the growth in residential user-to-user traffic. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 207–218, New York, NY, USA, 2006. ACM.

[8] Kimberly C. Claffy, George C. Polyzos, and Hans-Werner Braun. Application of sampling methodologies to network traffic characterization. *SIGCOMM Comput. Commun. Rev.*, 23(4):194–203, 1993.

[9] M. Claypool, R. Kinicki, M. Li, J. Nichols, and H. Wu. Inferring Queue Sizes in Access Networks by Active Measurement. In *Passive And Active Network Measurement: 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004; Proceedings*. Springer, 2004.

[10] L. Deri et al. Improving passive packet capture: Beyond device polling. In *Proceedings of SANE*, 2004.

[11] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing residential broadband networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 43–56, New York, NY, USA, 2007. ACM.

[12] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, 2001.

[13] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Trans. Netw.*, 12(6):963–977, 2004.

[14] N.R. Draper and H. Smith. *Applied Regression Analysis*. New York, 1967.

[15] T. En-Najjary and G. Urvoy-Keller. PPrate: A Passive Capacity Estimation Tool. In *End-to-End Monitoring Techniques and Services, 2006 4th IEEE/IFIP Workshop on*, pages 82–89, 2006.

[16] K. Harfoush, A. Bestavros, and J. Byers. Measuring bottleneck bandwidth of targeted path segments. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 3, 2003.

[17] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard), December 2002. Updated by RFC 5343.

[18] V. Jacobson. Congestion avoidance and control. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 314–329, New York, NY, USA, 1988. ACM.

[19] Kang, Liu, Dai, and Loguinov. Packet-pair bandwidth estimation: stochastic analysis of a single congested node. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 316–325, 2004.

[20] Rohit Kapoor, Ling-Jyh Chen, Li Lao, Mario Gerla, and M. Y. Sanadidi. CapProbe: a simple and accurate capacity estimation technique. *SIGCOMM Comput. Commun. Rev.*, 34(4):67–78, 2004.

[21] Sachin Katti, Dina Katabi, Charles Blake, Eddie Kohler, and Jacob Strauss. Multiq: automated detection of multiple bottleneck capacities along a path.

In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 245–250, New York, NY, USA, 2004. ACM.

[22] Srinivasan Keshav. A control-theoretic approach to flow control. In *SIGCOMM '91: Proceedings of the conference on Communications architecture & protocols*, pages 3–15, New York, NY, USA, 1991. ACM.

[23] K. Lai and M. Baker. Measuring bandwidth. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, 1999.

[24] Kevin Lai and Mary Baker. Nettimer: a tool for measuring bottleneck link bandwidth. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, pages 11–11, Berkeley, CA, USA, 2001. USENIX Association.

[25] Karthik Lakshminarayanan and Venkata N. Padmanabhan. Some findings on the network performance of broadband hosts. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 45–50, New York, NY, USA, 2003. ACM.

[26] Karthik Lakshminarayanan, Venkata N. Padmanabhan, and Jitendra Padhye. Bandwidth estimation in broadband access networks. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 314–321, New York, NY, USA, 2004. ACM.

[27] X. Liu, K. Ravindran, and D. Loguinov. What signals do packet-pair dispersions carry? In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, 2005.

[28] Jeffrey C. Mogul and K. K. Ramakrishnan. Eliminating receive livelock in an interrupt-driven kernel. *ACM Trans. Comput. Syst.*, 15(3):217–252, 1997.

[29] A. Pasztor and D. Veitch. The packet size dependence of packet pair like methods. In *Quality of Service, 2002. Tenth IEEE International Workshop on*, pages 204–213, 2002.

[30] Vern Paxson. End-to-end internet packet dynamics. *SIGCOMM Comput. Commun. Rev.*, 27(4):139–152, 1997.

[31] A. Persson, C.A.C. Marcondes, L.J. Chen, MY Sanadidi, and M. Gerla. TCP Probe: A TCP with built-in Path Capacity Estimation. In *IEEE Global Internet Symposium*, 2005.

[32] P. Phaal, S. Panchen, and N. McKee. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational), September 2001.

[33] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[34] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, 17(6):27–35, 2003.

[35] L. Rizzo. Device Polling support for FreeBSD. In *Proceedings of the Main European BSD Conference (EuroBSDCon 2001), Brighton, UK*, 2001.

[36] F. Schneider, J. Wallerich, and A. Feldmann. Packet Capture in 10-Gigabit Ethernet Environments Using Contemporary Commodity Hardware. *LECTURE NOTES IN COMPUTER SCIENCE*, 4427:207, 2007.

[37] Fabian Schneider and Jörg Wallerich. Performance evaluation of packet capturing systems for high-speed networks. In *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 284–285, New York, NY, USA, 2005. ACM.

[38] Rishi Sinha, Christos Papadopoulos, and John Heidemann. Internet packet size distributions: Some observations. Technical Report ISI-TR-2007-643, USC/Information Sciences Institute, May 2007. Orignally released October 2005 as web page `http://netweb.usc.edu/~rsinha/pkt-sizes/`.

[39] P. Wood. libpcap-mmap. Los Alamos National Lab.

[40] Lixia Zhang, Scott Shenker, and Daivd D. Clark. Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(4):133–147, 1991.

[41] `http://oss.oetiker.ch/mrtg/`.

[42] `http://oss.oetiker.ch/rrdtool/`.

[43] `http://www.cisco.com/go/netflow`.

[44] `http://www.endace.com`.

[45] `http://www.napatech.com`.