UNIVERSITY OF OSLO
Department of Physics

Path Capacity Estimation for Measurement-based Admission Control in Military IP Networks

Master Thesis

Espen Flydahl

May 29, 2012

Abstract

In this thesis, the performance of a state-of-the-art path capacity estimation algorithm is evaluated in terms of its suitability for being part of a measurement-based admission controller in a military Internet Protocol (IP) network.

The strive towards Network Enabled Capability (NEC) in miltary organizations drives the need for interconnecting all the actors taking part in an operation. The resulting network is an IP-based, heterogeneous Wide Area Network (WAN), comprising of a variety of fixed and mobile communication links with different capacities.

In military IP networks, Quality of Service (QoS) cannot be guaranteed when there is congestion. This is due to the use of IPsec, which forms a cryptographic boundary between the traffic source and forwarding routers, thus rendering end-to-end resource reservation impossible. This calls for the implementation of a congestion *avoidance* policy through Measurement-based Admission Control (MBAC).

Based on a literature study, the estimation algorithms combining the use of packet-pair dispersion and delay analysis, were found to be the most suited for providing low-intrusive, fast and reliable measurements of the path capacity. One of these algorithms, Ad Hoc Probe, was extensively evaluated in a test bed based on link technologies typically found in military networks.

A number of performance limiting factors were identified, including a lack of support for Time Division Multiple Access (TDMA)-based links, restrictions on the number of hops in contention-based Mobile Ad Hoc Networks (MANETs) and a minimum required capacity that made the algorithm unfit for use in networks containing narrowband, low-capacity links.

Acknowledgements

This thesis concludes my master's degree in Electronics and Computer Technology at the University of Oslo, Faculty of Mathematics and Natural Sciences, Department of Physics.

The work was funded by Forsvaret (Norwegian Armed Forces), and it was carried out in the period January 23 - May 29, 2012 at Forsvarets Forskningsinstitutt (Norwegian Defence Research Establishment) under the supervision of dr. scient. Mariann Hauge (FFI) and professor Josef Noll (UNIK).

First and foremost, I would like to thank my main supervisor Mariann Hauge and her coworker Erlend Larsen for valuable discussions and feedback, and for providing me with an interesting and challenging assignment. I would also like to extend my gratitude to Josef Noll for taking on the job as internal supervisor, and for his guidance in the writing process.

Furthermore, a great thanks goes to all the other people at FFI for showing interest in my work and for their assistance in providing test bed equipment.

Finally, I would especially like to thank my beloved Kari Elise for her support and patience during the last months.

Acronyms

ARP Address Resolution Protocol

ARQ Automatic Repeat reQuest

BPSK Binary Phase Shift Keying

CDMA Code Division Multiple Access

COMSEC Communication Security

CSMA Carrier Sense Multiple Access

CSMA/CA Carrier Sense Multiple Access Collision Avoidance

CSMA/CD Carrier Sense Multiple Access Collision Detection

CT Ciphertext

CTS Clear-to-send

DCF Distributed Coordination Function

DiffServ Differentiated Services

DSCP Differentiated Services Code Point

DSSS Direct Sequence Spread Spectrum

EF Expedited Forwarding

ESP Encapsulating Security Payload

FDMA Frequency Division Multiple Access

FEC Forward Error Correction

FFI Forsvarets Forskningsinstitutt (Norwegian Defence Research Establishment)

FIFO First In, First Out

IBSS Independent Basic Service Set

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IP Internet Protocol

ISM Industrial, Scientific and Medical

kbps Kilobits per second

MAC Medium Access Control

MANET Mobile Ad Hoc Network

MBAC Measurement-based Admission Control

Mbps Megabits per second

MGEN Multi-Generator

MTU Maximum Transmission Unit

NATO Northern Atlantic Treaty Organization

NEC Network Enabled Capability

OS Operating System

OWD One-Way Delay

PAN Personal Area Network

PER Packet Error Rate

PPD Packet-Pair Dispersion

PT Plaintext

QoS Quality of Service

QPSK Quadrature Phase Shift Keying

RTS Ready-to-send

RTT Round Trip Time

TDMA Time Division Multiple Access

TMC Theoretical Maximum Capacity

TOPP Trains of Packet-Pairs

TRANSEC Transmission Security

TSC Time Stamp Counter

TTL Time to live

UCLA University of California, Los Angeles

UDP User Datagram Protocol

UHF Ultra-High Frequency

UNIK Universitetssenteret på Kjeller

USB Universal Serial Bus

VoIP Voice over IP

VPS Variable Packet Size

WAN Wide Area Network

WLAN Wireless Local Area Network

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

During the last decade, there has been a lot of focus on Network Enabled Capability (NEC) in military operations. One of the key concepts in this strategy is to strive for *Information Superiority* which is defined by the Northern Atlantic Treaty Organization (NATO) as

> "... the operational advantage derived from the ability to collect, process, and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same." [38]

Information superiority enables the decision-makers to optimize the use of available resources. Moreover, it leads to a better ability to synchronize the units taking part in an operation, thus decreasing the time needed for mission execution. The net effect is a greater chance of accomplishing the mission. In order to achieve this operational advantage, it is first and foremost vital that operations are planned and executed based on the "need to share" principle. This is in contrast to the traditional "need to know" approach. Secondly, technical aids must be available to facilitate the process of collecting, processing and disseminating information.

The communication systems play a key role among these technical aids, and they must have the ability to interconnect arbitrary units in the battlefield. Furthermore, it is important that these units are able to communicate with upper echelons situated outside of the battlefield. The resulting network is a heterogeneous Wide Area Network (WAN), comprising of a variety of fixed and mobile communication links with different capacities and technologies for Medium Access Control (MAC) and physical transmission.

The preferred approach for achieving efficient routing in such a network is to implement packet switching combined with a common logical addressing scheme that is independent of the technologies employed for physical transmission and MAC. The obvious candidate protocol for logical addressing is the Internet Protocol (IP) [42, 13], given its scalability and widespread deployment in military core networks.

A fundamental challenge in all IP networks is to find ways to share the finite network resources in a way that makes the traffic flow smoothly, preventing the occurrence of congestion and subsequent severe loss of network throughput. The available resources in a military network are

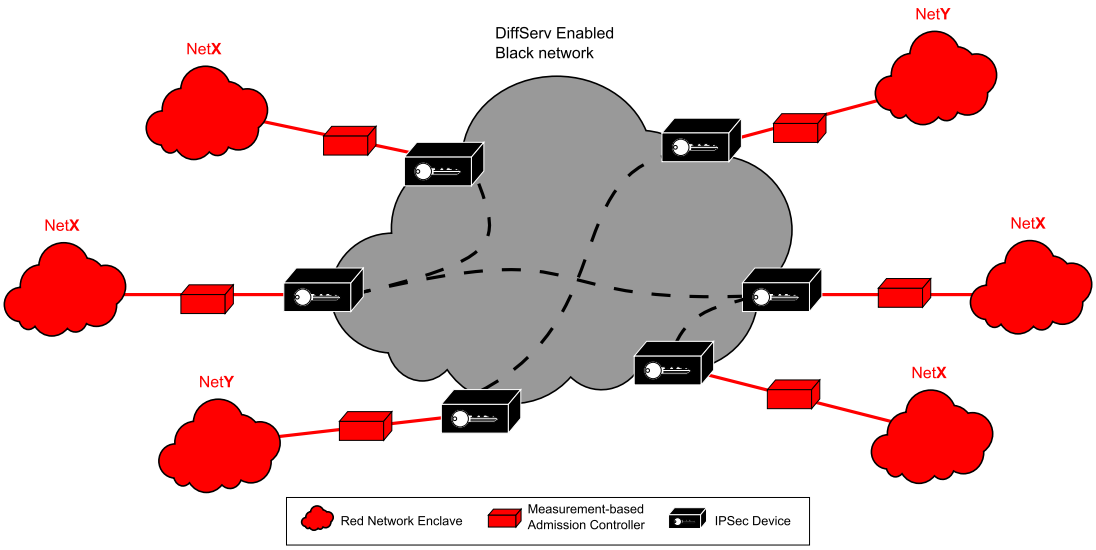


Figure 1.1: Illustration of the thesis scenario

scarce and possibly time varying, thus making it particularly difficult to avoid congestion, driving the need for mechanisms that prevent the network from collapsing when it is overloaded. Furthermore, there is the added problem of allocating the currently available network resources in a way that optimize the combat-effectiveness. The main motivation behind the work presented in this thesis is to contribute to the efficiency of such mechanisms.

1.2 Scenario

Communication Security (COMSEC) is required in military communications and this is normally provided by encryption using the Encapsulating Security Payload (ESP) protocol [29], standardized by the Internet Engineering Task Force (IETF) as part of the IPsec suite. However, this added security comes with a price.

Military users tend to apply ESP in *tunnel mode*, where the whole plaintext packet, including IP header, is encrypted and thereafter encapsulated by a new IP-header. This effectively divides the network into two routing domains with separate address spaces. No two-way communication is allowed between the two domains due to the risk of compromising classified information. The routing domain for the encapsulated ciphertext packets is traditionally termed the *red* network while the routing domain for the encapsulating plaintext header is termed the *black* network. The black network functions as a transit network, and is in most cases a resource that is shared by many. The red network enclaves may vary in size, ranging from a single soldier's Personal Area Network (PAN) to a large network within the fence of a large military base.

When there is need to communicate between enclaves of the red network, most user applications are designed to just start sending traffic. Some applications employ a certain flow control where they, in retrospect, see if the data got through the network at the required rate, delay, jitter, etc. If the observed behavior met the requirements, all is well and the flow is sustained. Else, the flow is terminated or the quality is lowered using another codec bit rate, etc. Even if flow control is used, the "just-send" behavior will in many cases lead to severe instability in the network

performance, since temporary network overloads cause congestion and degraded throughput for already established flows in the network.

The operational consequences of this congestion problem can be alleviated by using Differentiated Services (DiffServ) [40, 4], thus marking the traffic coming from different services using the Differentiated Services Code Point (DSCP)-field in the IP header. Simply copying[†] the DSCP-field from the red to the black header then provides the black network routers with information to prioritize the forwarding of mission-critical services during congestion, and to ensure that real-time traffic experiences minimum queuing delay.

Unfortunately, the introduction of traffic prioritization does not solve all the challenges. In a military heterogeneous network, the maximum achievable capacity from a source to a destination, the *path capacity*, will be time-varying due to the presence of mobile nodes. In many cases, the network cannot possibly provide the performance that is required for typical high priority, capacity demanding traffic (e.g., real-time video). When relocating a node, the application that worked in the previous position may not work in the new position due to a reduction in the path capacity. If the application follows the typical "just send" approach, the network will waste its scarce resources by prioritizing the forwarding of a flow that never can achieve the required Quality of Service (QoS). Additionally, this will have a negative effect on already established flows of same or lower priority.

As an approach to alleviate this problem, DiffServ can be augmented by an *admission control* mechanism to ensure that all flows exceeding the path capacity are not allowed into the network. The admission controller has to be able to communicate with user applications in order to inform the traffic sources whether or not new flows are admitted, thus providing applications with the alternative to adjust their required bit rate to the available network resources. Therefore, in a military network, the admission controllers need to be deployed inside the red networks. Since the IPsec partitioning removes the possibility to obtain network resource information from nodes within the black network, the admission controller has to be *measurement-based*.

Ideally, the measurement-based admission controller should measure the *available* capacity in the network path, thus ensuring that already established flows of same or higher priority are not disturbed. In this thesis, the focus is kept on the estimation of the maximum achievable path capacity, i.e. the capacity that can be achieved for a single flow over a path if there was no other traffic in the network.

Measuring the path capacity is non-trivial in a military network with scarce resources. The obvious approach of flooding the network path with data and measure the average bit rate cannot be applied since this would congest the network. A lightweight approach is needed.

1.3 Scope

Path capacity estimation has been a research topic for some time, resulting in a variety of techniques. However, most of the techniques were designed for wired links. With the advent of

[†]This opens a covert channel since a compromised host could use the DSCP-field as information bits and by doing so disclose secret information into the black network. Nevertheless, the benefits from a better availability of mission-critical services are in most cases found to outweigh the increased security risk.

Wireless Local Area Networks (WLANs) and mobile broadband data services, several new methods for path capacity estimation were proposed. The aim of this thesis is to gain a deeper understanding of how the path capacity can be estimated in a lightweight manner, and further to evaluate if current state-of-the-art estimation techniques can provide satisfactory performance for Measurement-based Admission Control (MBAC) in military IP networks.

1.4 Methods

The thesis starts with a study of the available literature within the field of path capacity estimation. Based on the findings, a set of requirements for a path capacity estimator fitting the thesis scenario is developed. A state-of-the-art path capacity estimation algorithm is selected for performance evaluation in a test bed. By varying the algorithm's parameters and the test bed configuration, the performance limitations of the algorithm are found. These results are used to evaluate the algorithm's compliance to the set of requirements.

1.5 Outline

Chapter 2 provides the thesis background by covering terminology and definitions, basic theory, the development of the set of estimator requirements, a review of path capacity estimation techniques and published algorithms, and finally, the selection and detailed description of the algorithm selected for performance evaluation. Chapter 3 and 4 are dedicated to an in-depth performance analysis of the selected algorithm, based on test bed experiments. Lastly, the thesis is concluded along with suggestions for further work in chapter 5.

Chapter 2

Background

In this chapter, the reader is first introduced to the terminology and definitions that will be used throughout the thesis. Then follows a review of different medium access techniques. Next, there is a discussion of the path capacity estimator performance requirements. Thereafter, the path capacity estimation techniques and state-of-the-art algorithms found in litterature are treated. Finally, the chapter is concluded by selecting one of the published path capacity estimation algorithms for further in-depth analysis and performance evaluation.

2.1 Terminology and Definitions

2.1.1 Network Capacity Metrics

Over the years, various terms and definitions have been used in literature for metrics describing network capacity. RFC 5136 [8] proposes a unified nomenclature as a framework for discussion and analysis within this field. In this thesis, an effort is made to adopt to the proposed nomenclature. However, the RFC 5136 framework does not take into account some critical physical layer nuances. Therefore, a few additional definitions from [45] are provided.

Representing information [45] Every communication system transfers information by the use of symbols which have been given a prearranged meaning. The size of the set of different symbols, the alphabet size M, determine how much information that can be transferred per symbol. Typical M-values are powers of 2 since, in digital communication systems, the information is represented with binary digits, *bits*, having only two possible values. If one bit is to be transmitted per symbol, then an alphabet size of two is needed. If two bits are to be transmitted per symbol, four different symbols are needed since two bits can be combined in four different ways. Generally, if l bits per symbol are to be transmitted then

$$M = 2^l$$

Equivalently, the number of bits per symbol is given by

$$l = \lfloor \log_2 M \rfloor \tag{2.1}$$

where $\lfloor \cdot \rfloor$ is the floor operator.

The number of symbols transmitted per second, is denoted R_s

Error correction coding [45] Since wireless channels are unstable and prone to bit errors, most wireless communication links transmit some extra information which allows the erroneous bits to be recovered from the correctly received bits. This is referred to as Forward Error Correction (FEC) in literature. The extra information is introduced by a process commonly referred to as *coding*, which adds redundancy bits to the information bits.

If a block of k information-bits results in n encoded bits, then the *code rate* is k/n. The resulting *information bit rate*, R_b, that is the upper bound on the amount of bits that can be delivered per second from the physical to the data link layer, is given by

$$R_b = l \cdot R_s \frac{k}{n} = \lfloor \log_2 M \rfloor \, R_s \frac{k}{n} \tag{2.2}$$

In many wireless communication systems, the alphabet size (i.e., modulation) and code rate can be dynamically adapted to the prevailing channel conditions, thus making the information bit rate a time-varying parameter.

Nodes, links and paths [8] A *node* N is a device where the input and output data rate can differ (e.g. hosts, switches, routers), a *link* L is a connection between two nodes and a *path* P is a series of n links $(L_1, L_2, \ldots, L_n)$ connecting a sequence of $n+1$ nodes $(N_1, N_2, \ldots, N_{n+1})$.

Link capacity [8] The *link capacity*, $C(L, T, I)$, is in RFC 5136 defined as the maximum number of IP layer bits of a single traffic type that can be transmitted from the source S and correctly received by the destination D over the link L during the time interval $[T, T+I]$, divided by I.

It is important to take into consideration that this metric is an average measure, hence the size of I has a significant impact. Additionally, the link capacity depends on the type of traffic. The protocol overhead ratio will vary with packet size and there could be Quality of Service (QoS) mechanisms that limit the capacity offered to different classes of traffic. Moreover, the layer 2 Medium Access Control (MAC) scheduling algorithm is important to take into account. Finally, one should keep in mind that wireless communication links can provide a time-varying information bit rate.

Path capacity [8] Expanding the rationale started above, RFC 5136 defines the *path capacity*, $C(P, T, I)$, as the minimum capacity of the links constituting a path P.

However, as pointed out in [7], this is not an optimal definition since the different links of a path might share the same communication channel in multi-hop wireless networks, thus making the achievable performance lower than the RFC 5136 definition implies. Considering the key role of multi-hop communication in tactical military networks, another definition is needed. Modifying the RFC 5136 definition of *link* capacity by substituting the link L with the path P, is a good starting point. However, in order to make the definition complete, the parameters T and I have to be defined.

As discussed in section 1.2, the path capacity estimate will be used to determine whether or not to try out a new flow in the network. Herein lies the assumption that the current path capacity estimate represents the future state of the path.

No real life network path satisfies the assumption of a indefinite steady-state, and this is especially important to note in mobile wireless communication where the nodes move and radio channel conditions fluctuate, causing the links in a path to break or change information bitrate, resulting in a change of the number of wireless hops in the path. The counter-measure for this behavior is to conduct new measurements for each admission decision, and to monitor the performance of already accepted flows, terminating these when they no longer satisfy the requirements. However, even if the topology is static and the information bit rates are constant, the possibly random MAC-protocol do not necessarily provide a stable instantaneous flow of bits to the IP layer. This, and the unknown flow duration, calls for a long averaging period.

In this thesis, the *path capacity* is defined as the limit of the maximum number of IP layer bits of a single traffic type that can be transmitted from the source S and correctly received by the destination D over the path P in a static network with stable information bit rates, during the time interval $[T, T + I]$, divided by I, as I approaches infinity.

Note that T is the arrival time of a new flow, and furthermore that this definition makes it necessary to conduct separate measurements for different types of traffic. I.e., flows of different packet sizes and/or Differentiated Services (DiffServ)-class are expected to have different path capacities.

2.1.2 Common Capacity-related Terminology in Literature [8]

The link with the smallest capacity along a path is often termed the *narrow* link.

Bandwidth is frequently used in literature to refer to what has been described as capacity in the definitions above. However, bandwidth is an overloaded term within the field of communication technology since many define bandwidth as the amount of spectral resources occupied by the communication link, measured in Hertz. These two measures are correlated, but they do not mean the same. For example, if the modulation method is changed from Quadrature Phase Shift Keying (QPSK) to Binary Phase Shift Keying (BPSK), the occupied spectral resources may remain the same, but the bit rate provided to the IP layer is at least halved.

2.1.3 Theoretical Maximum Capacity (Throughput)

The Theoretical Maximum Capacity (TMC) is an analytical upper bound for the link capacity. It does not take into account the effects of real-world performance degrading effects, such as bit-errors. Theoretical Maximum *Throughput* is the term most commonly used to describe this calculation in literature [26]. Nevertheless, following the course laid out in RFC 5136 [8], *capacity* will be the term in this thesis. The TMC is defined as the following relation:

$$\text{TMC} = \frac{\texttt{PSize}}{\text{E}\left\{\text{Transfer time}(\texttt{PSize})\right\}}$$

where $\text{E}\{\cdot\}$ is the expectation operator, `PSize` the layer 3 packet size and the transfer time denotes the time period that is needed to complete the transfer of one layer 3 packet. The reason for the use of the expectation operator is the fact that some link technologies have a random transfer time. Note that the TMC is a function of the packet size.

Appendix A contains the TMC-calculation for two commonly found link technologies, IEEE802.11b [20] and IEEE802.3 [19].

2.2 Medium Access Control

When two or more nodes in a network share a common channel[†] for transmission of information, the scheduling of the transmissions must follow certain rules in order to avoid the destructive interference that occurs if nodes transmit simultaneously. MAC-protocols are formalized versions of these rules.

Conceptually, there are four different approaches for designing MAC-protocols:

- Carrier Sense Multiple Access (CSMA)
- Time Division Multiple Access (TDMA)
- Frequency Division Multiple Access (FDMA)
- Code Division Multiple Access (CDMA)

CSMA, TDMA and FDMA are the most relevant approaches for this thesis since CDMA is very rarely used for multiple access in military networks[‡].

2.2.1 Carrier Sense Multiple Access

The basic principle behind CSMA is that the individual nodes listen to the channel before they attempt a transmission. If a busy channel is detected, they defer from transmission until the

[†]For example, the same radio frequency band or physical wire

[‡]In fact, the techniques used for CDMA are employed for Communication Security (COMSEC) and Transmission Security (TRANSEC) in military networks, i.e. for the *protection*, and not the *sharing* of channels

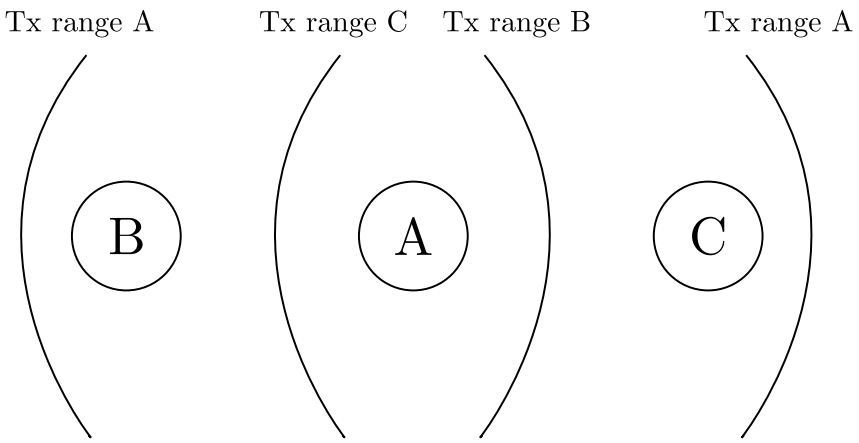


Figure 2.1: The "hidden node" problem that arises when sharing a wireless channel

medium is detected idle. Upon detection of an idle medium, the nodes do not transmit immediately as this would result in collision if more than one node has traffic to transmit. In stead, upon sensing the medium as idle, each node sets a backoff timer to a random value. The node that draws the lowest backoff time "wins" this contention and transmits first.

The nodes that lose the contention stop decrementing their backoff timers until they sense the medium idle again. At this time, they continue to decrement the timers from the value that they were at before the last transmission. If the node that won the previous contention has more traffic to send, it sets its backoff timer to a new random value and joins the contention.

Using this scheme, a fair sharing of the transmission channel is achieved since the nodes that have been waiting are, on average, more likely to have a lower value in their backoff timer than the node that just transmitted. The risk of collision is minimized, but there is still a probability for collision between the if two nodes have picked backoff times resulting in almost coinciding transmission times. Due to the propagation delay in the physical medium, one of the nodes may falsely detect the medium as idle even though the other has commenced transmission.

Since the risk of collision is non-negligible in CSMA, it is necessary to implement measures that reduces the chance of packet loss due to collisions. In a wireless setting, collisions are hard to detect. Additionally, the so called "hidden node" problem comes into effect. As shown in figure 2.1, the hidden node problem occurs when node B and C cannot hear each other, while A hears both. If B is transmitting to A, C will not hear this and assume that the channel is available when it is in fact busy. A transmission from C at this time will cause collisions at A.

In order to mitigate the problem of collisions in a wireless CSMA scenario, it is common to use a scheme termed Carrier Sense Multiple Access Collision Avoidance (CSMA/CA). Collision Avoidance is a preventive approach. The main idea is to transmit very short Ready-to-send (RTS) and Clear-to-send (CTS) messages to alert nodes within the transmission range of both the sending and receiving node of a future transmission. Of course, there is still the chance of colliding RTS messages. Nevertheless, the cost in terms of link efficiency is substantially higher if a long data frame is lost compared to a short RTS message. In CSMA/CA, the random backoff time concept is still used. However, if an RTS message is not followed by a CTS message due to interference or "node out of range", a timeout occurs and the sender node has to reset its random backoff timer to a new random value. In most cases, the maximum value of the interval

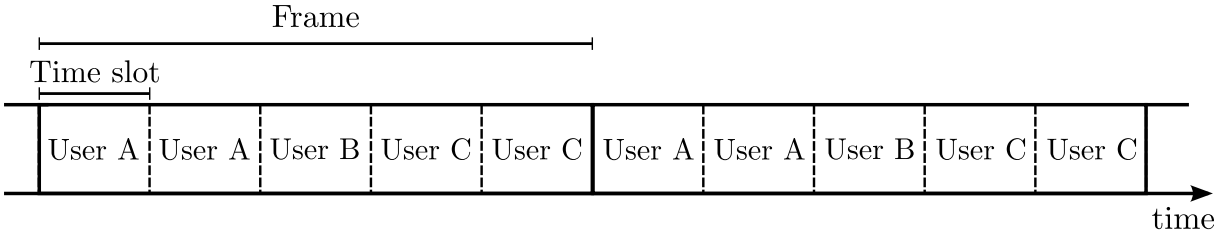


Figure 2.2: Illustration of the TDMA concept

from which the node's random backoff timer is drawn, the so called *Contention Window*, is set to a higher value after such a timeout. This ensures that nodes with no problem achieving the RTS/CTS handshake are prioritized, and it thus reduces the amount of transmission time that is wasted in attempts to reach nodes that are out of range.

Mobile Ad Hoc Networks Mobile Ad Hoc Network (MANET) is an umbrella term for a wireless mobile communication system that can be deployed in an ad hoc manner, i.e. with little or no need for configuration or existing infrastructure. These networks are commonly used by military personnel and first responders in areas where there is no infrastructure, or when the existing infrastructure is overloaded or cannot be relied upon due to security reasons.

In a packet switched MANET, all hosts have routing capability, thus making it possible to extend the network coverage area by multi-hop communication. In most cases, all nodes share the same channel using CSMA/CA for medium access. Hence, the range extension achieved through multi-hop communication comes at the cost of reduced capacity, since no more than a single node can transmit at a time within its interference range with respect to other nodes.

2.2.2 Time-Division Multiple Access

In contrast to the *random* approach of CSMA, the concept of TDMA is to time-share the channel in a *deterministic* manner. As shown in figure 2.2, the channel access schedule is organized into *frames* and *time slots*, where each user is allocated a certain number of time slots per frame. In most cases the time slots are allocated dynamically, based on which users that currently need to transmit and their capacity needs. This requires a time slot reservation scheme to ensure that there is no destructive interference between the users. The coordination can be handled by a central entity (e.g. a base station or master node) or in a distributed manner, which is the most common approach in tactical military networks. Either way, the users first have to communicate that they need channel access. Therefore, TDMA is in most cases combined with a CSMA signaling channel for access requests/reservations.

2.2.3 Frequency-Division Multiple Access

As the name suggests, FDMA is based on separating the different users in frequency (Hz), enabling the users to transmit concurrently. However, they do not have access to the entire available bandwidth as was the case for TDMA and CSMA. With FDMA, the bandwidth is

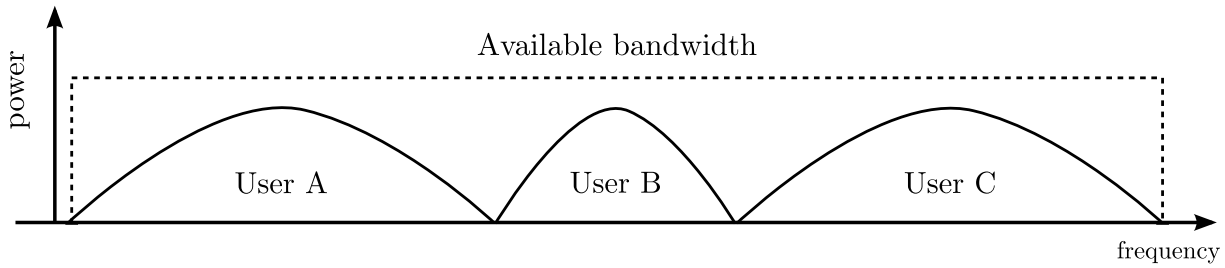


Figure 2.3: Illustration of the FDMA concept

often shared in a prearranged way, offering the users a dedicated channel. This makes the scheme less effective in terms of bandwidth utilization, and is the reason to why its application has become less widespread over the years. Nevertheless, it is still the preferred method for most military satellite links.

2.3 Path Capacity Estimator Performance Requirements

A path capacity estimator must have certain properties in order to qualify for use in a military scenario as described in section 1.2. In [14], a list of requirements for a wireless home network path capacity estimation tool is presented as follows:

> "The tool must:
>
> 1. require adaptation/upgrade only of the server-side of the e2e path. In a home network that is often the home gateway: services from commercial service providers enter the home network via the home gateway. This makes the tool applicable to the current plethora of existing (thin) clients. For most use cases we can therefore assume the predicting tool to be a service running on the home gateway, serving various clients in the home network which only need to have a regular IP stack.
> 2. be non-intrusive. It should not disrupt other traffic in the home noticeably. The other traffic in the home may be important to the user.
> 3. have a short measurement time, i.e. it should have a low convergence time from an end-user perspective, and it should be fast enough to react to major changes in the home network traffic pattern. We assume this to be in the order of a few seconds. This assumption is based on the performance of current Internet speed tests accepted in the market, and the real-time performance of discovery protocols of UPnP.
> 4. not require pre-knowledge of the link-layer network topology. Home networks can be very heterogeneous and support many different link-layer topologies, of which some even may not be standardized or widely known.
> 5. be accurate enough to make educated predictions about the admission of delay- and jitter-critical applications. In the case of IPTV and IP telephony that means an accuracy of 1 Mbit/s and 50 kbit/s respectively."

However, these requirements cannot be applied directly in a military scenario. The next subsections describe the requirements for a *military* path capacity estimator, where the requirements provided in [14] are modified to fit into the thesis scenario, and some new requirements are added. A set of qualitative requirements are presented first, followed by the quantitative requirements.

2.3.1 Qualitative Requirements

Topology independence Due to the partitioning of the network into two routing domains, the red and the black network, and the fact that the measurement-based admission controller needs to be placed in the red network; the estimator *must not* rely on communication with the forwarding nodes in the black network.

Since the black network topology is generally regarded as unknown, the path capacity estimator should ideally be compatible with all MAC protocols. However, such a requirement is impossible to verify. Still, the requirement can be narrowed down: Since the most common MAC schemes found in military networks are based on CSMA, TDMA or FDMA, the path capacity estimator *must* be compatible with at least one of these methods if it is to be of any use.

Furthermore, the estimator *should* be compatible with wireless multi-hop communication where the same channel is shared between multiple nodes using CSMA/CA, since this topology is very relevant in military networks. This is a *should*-requirement due to the fact that the estimator would still be useful in networks where there is no multi-hop communication.

Path asymmetry In [14], it is stated that the estimator "must require adaptation/upgrade only of the server-side of the e2e path". However, the forward and reverse path capacities of a military network are in general not equal. For instance, it is very common that satellite communication systems provide an asymmetric uplink/downlink capacity. Therefore, the path capacity estimator *must not* rely on this assumption - the measurements have to be one-way.

Rate-limiters In order to make a packet switched network satisfy the military requirements for QoS, it is paramount that there are mechanisms in place that ensure that jitter-sensitive flows like voice and video communication services are subject to minimum queuing delay along the forwarding path.

For DiffServ enabled networks, there is a per-hop-behavior definition called Expedited Forwarding (EF) [12] that is specifically designed to minimize the queuing delay. It is typically implemented as a high-priority queue in which a newly arrived packet at most should wait the time it takes to transfer one packet before being serviced. Rate-limiters are commonly deployed in order to ensure that the EF-queue does not completely starve the other traffic transiting through a node [12]. rate-limiters typically ensure that the EF-queue arrival bit rate never exceeds a given fraction of the link capacity.

In order to avoid the futile admission of a flow that exceeds such a rate limit, the path capacity estimator *should* be compatible with rate-limited paths. This is a should-requirement because

the path capacity estimate would still be useful even if it did not take into account the rate-limiters. An estimate based on links utilizing their full capacity will still provide an upper bound that can be used to reject flows.

IP packet fragmentation The presence of IPsec devices makes it impossible to detect the black network path Maximum Transmission Unit (MTU) from the red network, since the ciphertext packets have to be reassembled before they can be deciphered. Due to the variety of link technologies that exist in military networks, it is expected that the MTU will differ along a path. This will result in frequent occurrence of fragmentation in the black network. The path capacity estimator therefore *must* work even if the estimation traffic is subject to packet fragmentation.

2.3.2 Quantitative Requirements

Accuracy In, [14], it is stated that the estimator

> "must be accurate enough to make educated predictions about the admission of delay- and jitter-critical applications. In the case of IPTV and IP telephony that means an accuracy of 1 Mbit/s and 50 kbit/s respectively."

The approach of setting the accuracy limit to an exact figure (e.g. 50 kbps), is not optimal. For instance, there are very few cases— if any —where the difference between the capacity requirements of two Mbps-flows is as low as 50 kbps. However, this will very likely be the case between flows with bit rates in the order of 100 kbps. Therefore, the accuracy requirement should be a relative figure, and this is the chosen approach in this thesis.

By requiring that the path capacity estimate is within $\pm\alpha$ % of the true path capacity, C, the reported path capacity estimate, $\widehat{C}$, will have a maximum value of

$$\max\{\widehat{C}\} = \left(1 + \frac{\alpha}{100}\right) C$$

The objective is to avoid allowing flows with bit rate requirements higher than C into the network. Hence, the admission decision should be based on $\widehat{C}$ normalized to $\left(1 + \frac{\alpha}{100}\right)$ since

$$\widehat{C} < \max\{\widehat{C}\} = \left(1 + \frac{\alpha}{100}\right) C \qquad \Rightarrow \qquad \frac{\widehat{C}}{1 + \frac{\alpha}{100}} < C$$

However, the penalty by doing so is that the decision could potentially be based on a far too low estimate:

$$\frac{\min\{\widehat{C}\}}{1 + \frac{\alpha}{100}} = \frac{1 - \frac{\alpha}{100}}{1 + \frac{\alpha}{100}} C$$

The value of α determines the how much of the path capacity that— in the worst case —will be left unused. The value chosen in this thesis is

$$\alpha = 20 \qquad \Rightarrow \qquad \frac{1-\frac{\alpha}{100}}{1+\frac{\alpha}{100}}C = \frac{1-0.2}{1+0.2}C = \frac{2}{3}C$$

with the result that $\frac{1}{3}C$ is left unused in the worst underestimation case and $0.2C$ if the estimation is correct, while the entire capacity is being put to use in the worst case of overestimation. The "expected" underutilization of 20 % is not necessarily a negative consequence since the path should not be operating at full load, leaving no margin to recover from packet bursts or short-time capacity degradations.

The accuracy of an estimator is often quantified by its bias and standard deviation, formally defined as

$$\text{Bias}\{\widehat{C}\} = \text{E}\{\widehat{C}\} - C = \mu_{\widehat{C}} - C \qquad \sigma_{\widehat{C}} = \sqrt{\text{E}\left\{\left(\widehat{C} - \mu_{\widehat{C}}\right)^2\right\}}$$

where $\widehat{C}$, is the random variable representing the estimated path capacity, C is a constant representing the true path capacity and $\text{E}\{\cdot\}$ is the expectation operator. The lower bias and standard deviation, the better the estimator performs. Ideally, the estimator is unbiased and has a very low standard deviation.

It is hard to generally define maximum standard deviation and bias due to the fact that the probability density function (pdf) of the estimator value is unknown. Nevertheless, by assuming that the estimator value follows a Gaussian distribution it is possible to derive maximum values that are reasonable.

Formally, one can state that there must be a 95 % chance that a single measurement results in an estimate that is within 20 % of true path capacity. I.e.,

$$P(0.8C < \widehat{C} < 1.2C) = 0.95 \tag{2.3}$$

Since $\widehat{C}$ is assumed to have a Gaussian pdf

$$P(\mu_{\widehat{C}} - 1.96\sigma_{\widehat{C}} < \widehat{C} < \mu_{\widehat{C}} + 1.96\sigma_{\widehat{C}}) = 0.95$$

and

$$\mu_{\widehat{C}} = \text{Bias}\{\widehat{C}\} + C$$

Then, in order for equation (2.3) to hold

$$\text{Bias}\{\widehat{C}\} + C + 1.96\sigma_{\widehat{C}} < 1.2C \quad \Rightarrow \quad \sigma_{\widehat{C}} < \frac{0.2C - \text{Bias}\{\widehat{C}\}}{1.96} \tag{2.4}$$

and, in addition

$$\text{Bias}\{\widehat{C}\} + C - 1.96\sigma_{\widehat{C}} > 0.8C \quad \Rightarrow \quad \sigma_{\widehat{C}} < \frac{0.2C + \text{Bias}\{\widehat{C}\}}{1.96} \tag{2.5}$$

Taking into account that the standard deviation is a positive number or zero:

$$-0.2C \leq \text{Bias}\{\widehat{C}\} \leq 0.2C$$

This results in the following accuracy requirements for the path capacity estimator:

$$-0.2 \leq \frac{\text{Bias}\{\widehat{C}\}}{C} \leq 0.2 \tag{2.6}$$

$$\frac{\sigma_{\widehat{C}}}{C} < 0.102 - 0.5102 \cdot \left| \frac{\text{Bias}\{\widehat{C}\}}{C} \right| \tag{2.7}$$

Measurement Time Ideally, the admission control entity would proactively perform a real-time estimation of the path capacity, enabling it to make decisions on-the-fly at the arrival of a new flow. However, this is not scalable since separate flows generally follow separate paths, i.e. not all traffic is destined for the same red enclave. Taking this into account, the number of parallel estimations from a potentially large amount of admission controllers would overload the network if a real-time proactive approach was chosen. The most scalable and precise method is therefore the reactive approach of only performing estimations at the arrival of a new flow. As a consequence, the path capacity measurement time should be low in order to reduce the flow set-up time.

Since many new flows can arrive in a short time interval, there should be a minimum time between measurements to minimize the amount of measurement traffic in the network. How long this estimate "lifetime" should be is difficult to decide since this depends on the rate of topology changes in the network. A network with a high amount of mobile nodes is expected to have more frequent topology changes and should therefore employ a short lifetime, while a more stationary topology could use a long lifetime. This design issue is out of scope for this thesis and is left for further study.

In [22], the mean call set-up time is required to be below 6 seconds for 95 % of all local connections. This is a reasonable upper bound that corresponds to the time needed for synchronization using legacy tactical voice encryption devices[†] that are used in coalition operations today. Using this as a starting point, and further taking into account that Voice over IP (VoIP) packet sizes most likely are less than 100 bytes, the requirement chosen in this thesis is that it *must* take less than 6 seconds to estimate the path capacity for a flow of 100 byte packets. The specification of the packet size is necessary, since

$$\text{Measurement time} = \frac{\text{Amount of traffic needed per measurement}}{\text{Measurement traffic transferred per second}}$$

where the amount of traffic per measurement is expected to be proportional to the packet size of the traffic type in question.

It is important to note that the equation above is only valid if all the sent measurement traffic

[†]For example, the VINSON familiy of tactical voice encryption devices

actually arrive at the measurement point. Hence, this requirement is based on the assumption that the network there is no packet loss during the measurement.

Intrusiveness The degree of intrusiveness is determined by the estimator's influence on the performance of already established flows. This depends on the percentage of the path capacity that is required by the estimator, and how much of the capacity that is currently in use for the traffic type in question. In this thesis, it is chosen to require that the estimation traffic *must* be kept below 10 % of the path capacity.

2.3.3 Summary of the Requirements

Qualitative requirements The path capacity estimator:

- *must not* rely on communication with the forwarding nodes in the black network
- *must* be compatible with CSMA and/or TDMA and/or FDMA
- *should* be compatible with wireless multi-hop communication where the same channel is shared between multiple nodes using CSMA/CA
- *must not* rely on the assumption of symmetric path capacities
- *must* work even if the estimation traffic is subject to packet fragmentation
- *should* be compatible with rate-limited paths

Quantitative requirements

- It *must* take less than 6 seconds to complete a measurement for packet size 100 bytes, given that there is no packet loss during the measurement.
- Assuming that the path capacity estimator follows a Gaussian distribution, there *must* be a 95 % chance that the estimation error is less than ± 20 % of the true path capacity. I.e., $-0.2 \leq \frac{\text{Bias}\{\widehat{C}\}}{C} \leq 0.2 \quad \bigwedge \quad \frac{\sigma_{\widehat{C}}}{C} < 0.102 - 0.5102 \cdot \left| \frac{\text{Bias}\{\widehat{C}\}}{C} \right|$
- The estimation traffic layer 3 bit rate *must* not exceed 10 % of the path capacity.

2.4 Path Capacity Estimation Techniques

When performing an input/output analysis in a communication network, there is the possibility of refraining from input generation, and only observe the already existing flows in the network. This *passive* approach has the advantage of being non-intrusive, but comes with the risk of very long measurement times. By chosing the *active* approach of injecting traffic into the network (probing), short measurement times can be achieved at the cost of possibly intruding on established flows in the network. Considering the requirement for a short measurement time in section 2.3.2, only techniques and algorithms supporting an active approach are treated in this section.

There are first and foremost two quantities that are measured by path capacity estimation techniques [28]:

Dispersion The time interval (dispersion) between receiving the last bits of two packets. Also termed the *interarrival* time.

Delay The time interval from when the first bit of a packet leaves the sender until the last bit of the packet has arrived at the receiver

The path capacity estimation techniques described in literature can therefore be divided into three categories, depending on whether they rely on: 1) packet dispersion analysis, 2) packet delay analysis or 3) a hybrid approach, using both packet dispersion and delay analysis

The rest of this section is dedicated to presenting the different models that are used within the different categories of estimation techniques, along with a presentation of the state-of-the-art algorithms belonging to each category.

2.4.1 Packet Dispersion Analysis

The packet dispersion based techniques described in literature can be divided into three categories:

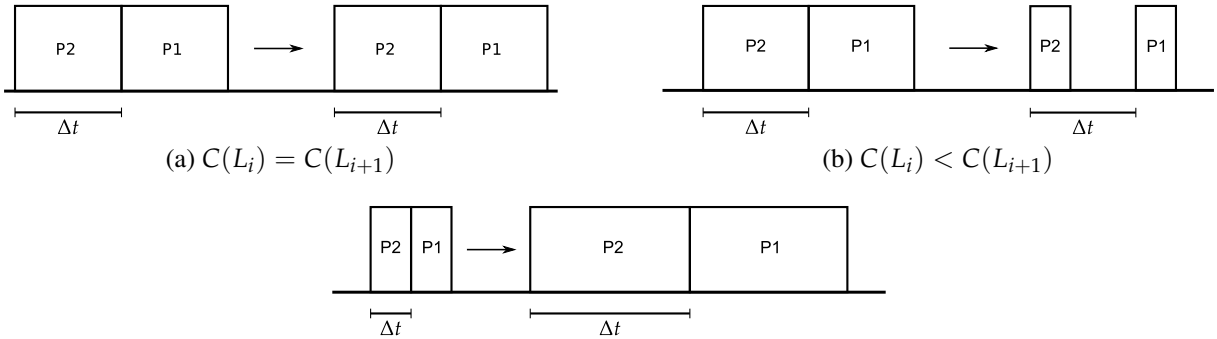
1. If the technique is based on measuring the dispersion between two back-to-back[†] packets, the technique is said to be based on *packet-pairs* and a model termed the *Packet-Pair Dispersion (PPD)-model* is used to calculate the capacity.

2. If the dispersion between the first and last package of three or more back-to-back packets is measured, the technique is based on *packet-trains* and the capacity is calculated using a relation termed the *dispersion rate*.

3. Finally, there are techniques based on analyzing the dispersion of *Trains of Packet-Pairs (TOPP)*, where the packet-pairs are not sent back-to-back, but with a varying initial dispersion. A model commonly termed the *fluid model* is used to calculate the path capacity.

[†]Back-to-back: Initial time spacing between the packets is ~0

Packet-Pair Dispersion model When applying packet dispersion analysis for end-to-end path capacity estimation, a common approach is to probe the network by sending pairs of packets to a receiver at the end of the path in question. The packets are sent back-to-back, i. e. the dispersion between the packets is set as small as possible at the time of transmission. Moreover, the packets are assumed to follow the same, single path through the network. The PPD model is used to calculate the path capacity on the basis of packet interarrival time measurements at the end node. Figure 2.4 illustrates the effects that can occur according to the model when a packet-pair is forwarded from the i-th to the $(i+1)$-th link in a path.



(a) $C(L_i) = C(L_{i+1})$ (b) $C(L_i) < C(L_{i+1})$

(c) $C(L_i) > C(L_{i+1})$

Figure 2.4: Packet dispersion effects

If the $(i+1)$-th link capacity is the same as the i-th, no effects occur. If the $(i+1)$-th link capacity is greater than the i-th, the transmission time needed to forward the individual packets in a pair will be shorter, but the time difference between the last bits of the first and second packet will be the same as upon reception. Furthermore, if the $(i+1)$-th link capacity is lower than the i-th, the required transmission time will be longer and consequently the packet dispersion will increase. The packet dispersion may yet increase further if an even slower link is encountered later in the path, but it can never be decreased.

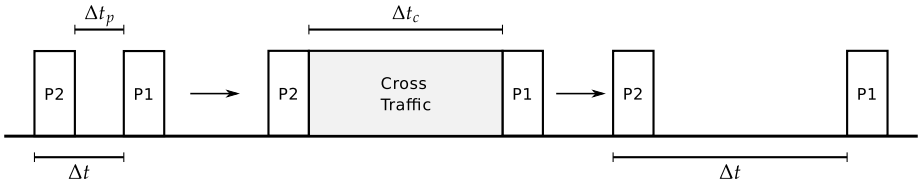
As a result, the interpacket arrival time measured at the end node is the time that it took to transmit the second packet over the path's narrow link and the path capacity can be calculated using the following relation:

$$\widehat{C} = \frac{\texttt{PSize}}{\Delta t} \tag{2.8}$$

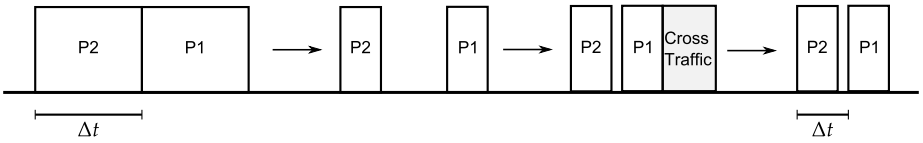
where $\widehat{C}$ is the path capacity estimate, `PSize` is the packet size and Δt is the observed packet-pair dispersion which corresponds to the transmission time of a single packet over the link with the minimum capacity in the path.

As pointed out in [43], a key assumption for the PPD model is the absence of other traffic on the path. This is not a realistic assumption, and the effects of cross traffic has to be taken into consideration in order to provide accurate estimates. Figure 2.5 shows how cross traffic can influence the accuracy of estimates based on the PPD model.

In the first case shown in figure 2.5a, one or more cross traffic packets can be multiplexed between the packets in a pair. If $\Delta t_c > \Delta t_p$, the effect is an expansion of the PPD. In accordance with equation (2.8), this will lead to an underestimation of the path capacity.



(a) The expansion effect



(b) The compression (post-narrow link) effect

Figure 2.5: The packet dispersion effects of cross traffic

The second cross traffic effect, the compression effect, is illustrated in figure 2.5b. If cross traffic causes the transmission queue to be non-empty when the last bit of the first packet has been received, it has to wait before being forwarded. This leads to a compression of the packet-pair dispersion and, consequently, an overestimation of the path capacity. The compression effect is also called the *post-narrow link* effect [15] since it only will result in overestimation if the compression occurs after the packet-pair has passed the path's narrow link.

Taking into account the expansion and compression effect, it is clear that not all packet-pairs will be dispersed according to the minimum link capacity. As a consequence, sending and measuring the dispersion of a single packet-pair should not be the preferred approach for path capacity measurements. In practice, multiple packet-pairs need to be sent under the assumption that some of these probes make it through the path without being expanded or compressed. The challenge is to pick out these "good" samples from the rest.

Dispersion rate When using the dispersion between the last received bits of a sequence of M packets sent back-to-back, the path capacity is calculated using the relation

$$\widehat{C} = \frac{(M-1)\texttt{PSize}}{\Delta t}$$

where $\widehat{C}$ is the path capacity estimate, `PSize` is the packet size and Δt is the observed dispersion between the last bit of the 1st and M-th packet. Packet-trains will also be subject to cross-traffic induced compression and expansion, causing the model to over- or underestimate the path capacity. In fact, since a packet train is in the path for a longer time than a packet-pair, it is more likely coincide with cross-traffic, resulting in expansion and underestimation. However, the impact of compression or expansion will have a smaller influence on the path capacity estimate, since the relative change in dispersion will be less than in the case of packet-pairs. The trade-off is clear: Packet-trains deliver lower variance than packet-pairs at the cost of increased measurement time/overhead.

Fluid model The fluid model was introduced by Melander et al. in [37] in the context of measuring *available* capacity, and is based on modeling the cross traffic as a continuous flow of

data, and not discrete packets. The cross traffic is assumed be of constant rate, R_c. I.e., during a time interval of length τ, the amount of arrived cross traffic is given by $R_c\tau$.

When the first packet of a packet-pair with initial dispersion $\Delta t_i > 0$ and packet size $\texttt{PSize}$—corresponding to an instantaneous rate of $R_i = \texttt{PSize}/\Delta t_i$ —arrives at a transmission queue with an outgoing capacity $C > R_c$, the cross traffic will accumulate in the queue during the time period it takes to forward this first packet, $\texttt{PSize}/C$, corresponding to an amount of $R_c \cdot \texttt{PSize}/C$. If this accumulated amount cannot be serviced— in addition to the continuous flow —before the second packet in the pair arrives, the packet dispersion will increase. I.e, if

$$C\left(\Delta t_i - \frac{\texttt{PSize}}{C}\right) < R_c \Delta t_i$$

which is equivalent to

$$\Rightarrow C < R_c + R_i$$

then the second packet in the pair will be queued for the time it takes to send the "excess" cross traffic, $R_c\Delta t_i - C\left(\Delta t_i - \frac{\texttt{PSize}}{C}\right)$, resulting in the relation

$$\begin{aligned}
\Delta t_o &= \Delta t_i + \frac{R_c\Delta t_i - C\left(\Delta t_i - \frac{\texttt{PSize}}{C}\right)}{C} \\
&= \Delta t_i + \frac{R_c\Delta t_i}{C} - \Delta t_i + \frac{\texttt{PSize}}{C} \\
&= \frac{R_c\Delta t_i}{C} + \frac{\texttt{PSize}}{C} \\
&= \frac{\Delta t_i R_c + \Delta t_i R_c}{C} \\
\Rightarrow \frac{\Delta t_o}{\Delta t_i}(R_i) &= \frac{R_c + R_i}{C}
\end{aligned} \tag{2.9}$$

Note that equation (2.9) is a linear function of R_i, corresponding to a straight line.

By contrast, if the "excess" amount of cross traffic has been serviced when the second packet arrives, i.e if

$$C > R_c + R_i$$

then the packet-pair dispersion will remain unchanged, $\Delta t_i = \Delta t_o$. Summarized, this means that the relation between the output and input dispersion is given by

$$\frac{\Delta t_o}{\Delta t_i}(R_i) = \begin{cases} \frac{R_c+R_i}{C} & C < R_c + R_i \\ 1 & C > R_c + R_i \end{cases} \tag{2.10}$$

By sending trains of packet-pairs with gradually decreasing initial dispersions, the link with capacity and cross traffic satisfying $C < R_c + R_o$ will impose the dispersion that results in the linear relationship of equation (2.9). The capacity of this link can be found by observing the slope of the linear increase in the ratio between the output and input dispersion.

A fair question is whether this capacity represents the minimum link capacity of the path. Again,

the cross traffic has the potential to disturb the accuracy of the measurement since the link where the sum of the cross traffic rate and the packet-pairs instantaneous rate not necessarily is the same as the link with the minimum capacity. The estimate could represent the capacity of a heavily loaded link of high capacity.

Algorithms In [16], Dovrolis et al. proposed *pathrate* is a tool based on the combined use of packet-pairs and packet trains. The algorithm is founded on the observation that a path capacity estimates based on the PPD-model will follow a multi-modal distribution where one of the modes is very close to the true path capacity. However, this is not necessarily the dominant mode since cross traffic may have resulted in that most of the measurements were over- or underestimations. By further realizing that large packet trains will follow a uni-modal distribution due to the reduced impact of cross traffic, and that the mode of this distribution is equal or less than the path capacity, an interval in which the true path capacity resides can be reported. Pathrate is first and foremost designed for accuracy, not speed. It sends a large amount of measurement traffic, occupying a large part of the path capacity.

DietTOPP [25] is based on the fluid model and the dispersion rate approach. First, an appropriate range to vary the probing rate is found by calculating the dispersion rate of a single *back-to-back* packet train. Next, several trains of packets[†], are sent with a gradually increasing rate, starting at the dispersion rate. The ratio between the input and output rate is assumed to follow a straight line, as described by the fluid model. Finally, the capacity estimate is based on using linear regression to find the slope of this line.

2.4.2 Delay Analysis

Delay-based path capacity estimation techniques were first introduced by S. Bellovin [3] and V. Jacobson [23], and are collectively termed *Variable Packet Size (VPS) probing* in literature. The network path is assumed to conform to the *one-packet model* where the Round Trip Time (RTT) of a single packet is modeled to be

$$\mathrm{RTT}_i(\texttt{PSize}) = X + \sum_{i=1}^{N} \frac{\texttt{PSize} + I}{C_i} + d_i \tag{2.11}$$

where

- X is a random variable representing queuing delay
- `PSize` is the size of a probe packet
- I the size of an Internet Control Message Protocol (ICMP) Time Exceeded Message
- C_i is the link capacity of the i-th link of a path of N hops.
- d_i is the constant two-way propagation delay of the i-th link

The capacity of the links constituting the path is recursively estimated, starting by sending single packets of different sizes with the Time to live (TTL) field set to 1 and measure the time interval between the departure of the packets and the arrival of the ICMP Time Exceeded

[†]Not trains of packet-pairs (TOPP) as was described to be used in the fluid model

Messages. These RTT measurements for different packet sizes cannot be directly inserted into equation (2.11) due to the stochastic variable X which depends on the amount of cross traffic in the transmission buffers at the router. However, by observing the fact that the *minimum* RTT measurement for each packet size is very likely to represent a case where $X = 0$, a deterministic variant of equation (2.11) for $i = 1$ can be used to estimate C_1:

$$\min\{\mathrm{RTT}_1(\texttt{PSize})\} = \frac{\texttt{PSize} + I}{C_1} + d_1$$

This equation contains two unknowns, C_1 and d_1, which can be found by forming a set of equations based on two or more minimum RTTs for different packet sizes.

When C_1 and d_1 is determined, the TTL can be set to 2, and the capacity of the second link can be calculated by the relation

$$\min\{\mathrm{RTT}_2(\texttt{PSize})\} = (\texttt{PSize} + I)\left(\frac{1}{C_1} + \frac{1}{C_2}\right) + d_1 + d_2$$

This process is continued until $i = N$, and the path capacity is defined as $\min_{i=1\to N}\{C_i\}$.

Delay-based capacity estimation relies on that all routers along a path behave correctly in terms of sending ICMP messages. This is not the case in all of today's networks. Furthermore, the model does not take into account store-and-forward layer 2 switches which contribute significantly to the transmission delay [43]. Finally, since the algorithm is recursive, errors made in the first estimations will have degrade the accuracy for all subsequent estimates.

Algorithms *Pathchar*[23] is was the first tool that found widespread use, and it was further optimized by Downey in his tool *clink* [17] and by Mah in the tool *pchar* [36].

2.4.3 Hybrid Approach

Considering that packet dispersion techniques are accurate, but sensitive to cross traffic, while the delay-based techniques are inaccurate, but less sensitive to cross traffic - the concept of combining these two techniques is a promising strategy for providing accurate and cross traffic insensitive estimates. By using Bellovin and Jacobson's stochastic approach of assuming that in a data set of many probes, the probe with the *minimum* delay has not experienced queuing in the path, there is a very good chance that applying packet dispersion analysis on this probe will result in a correct and cross traffic-robust estimate. This is indeed the main idea behind the latest publications within the field of capacity estimation.

Algorithms *CapProbe* [28] is based on the concept of combining dispersion and delay analysis by sending back-to-back packet-pairs of ICMP echo messages to the destination and measuring the arrival times of the ICMP echo-reply messages. The algorithm keeps track of the

departure times of each of the packet-pairs, and when the two corresponding ICMP echo-reply[†] packets arrive, the RTT of the individual packet in a pair is calculated. The packet-pair with the *minimum RTT sum* is selected as the "good" sample, and the path capacity is calculated based on the packet size and the difference between the arrival times of the ICMP echo-replies. CapProbe is freely available for download from the University of California, Los Angeles (UCLA) Network Reserch Laboratory website [30].

Ad Hoc Probe [7, 30] is a one-way variant of the CapProbe algorithm based on sending User Datagram Protocol (UDP)-datagrams in a client-server fashion, requiring specific software in both ends of the path. Instead of measuring RTTs, it measures One-Way Delays (OWDs). As the name suggests, it is specifically designed to provide estimates in wireless Ad Hoc networks.

Allbest [14] is a packet-pair based algorithm specifically designed for one-hop IEEE802.11 Wireless Local Area Networks (WLANs) [20]. Similar to CapProbe, it also employs the concept of minimum RTTs for filtering out queued packets, but in stead of basing the capacity calculation on the minimum RTT *sum* packet-pair it constructs a "virtual" packet-pair by using the minimum RTT of the first packets and the minimum RTT of the second packets. I.e., if $\mathrm{RTT1}_i$ denotes the RTT based on the i-th received ICMP echo-reply corresponding to the first packet of a packet-pair of ICMP echos, and $\mathrm{RTT2}_i$ represents the same for the second packet of the pair, the path capacity calculation based on n packet-pairs is performed using the following relation:

$$\widehat{C} = \frac{2 \cdot \texttt{PSize}}{\min_{i=1\rightarrow n}\{\mathrm{RTT2}_i\} + \min_{i=1\rightarrow n}\{\mathrm{RTT1}_i\}} \tag{2.12}$$

Furthermore, Allbest forms its packet-pairs by sending single packets that are twice the path MTU, forcing the the packet to fragmented in two parts. The advantage of this approach is that the receiver node has to wait until the second packet arrives before sending the two fragments of the ICMP echo-reply message, thus avoiding the CSMA contention that occurs between the transmissions of the first ICMP echo-reply packet and the second ICMP echo packet. Since the two fragments of the ICMP echo-reply message are reassembled before they are delivered to the probing application, only RTT2 is found this way. RTT1 is found by probing with single packets.

In [1], Alzate et al. introduced an unnamed path capacity estimation algorithm. Based on a stochastic analysis, the mean path capacity was proposed to conform to the following model under the conditions of no cross traffic:

$$C(\texttt{PSize}) = \frac{\texttt{PSize}}{\alpha \cdot \texttt{PSize} + \beta} \tag{2.13}$$

where α is a unknown constant representing how much service time[‡] the path uses to transmit a single bit from source to destination, and β is the unknown *expected value* of the added delay introduced by lower layer overhead[†]. By recognizing that $\alpha \cdot \mathrm{PSize} + \beta$ is a linear function of `PSize`, representing a straight line, and that the dispersion of a packet-pair that has experi-

[†]ICMP echo-replies are of same packet size as the corresponding ICMP echo message [41]

[‡]The “on-air” and/or “on-wire” time

[†]This overhead can be random

enced no queuing delay is a point on this line; α and β can be found by obtaining two of these points. This can be achieved through sending multiple packet-pairs of two different sizes, and subsequently using the dispersions of the minimum OWD sum packet-pairs. When α and β is found, the path capacity for other packet sizes can be found through interpolation.

2.5 Selecting an Algorithm for Experimental Evaluation

Since all of the algorithms presented in the previous section were designed for a civilian context, it was necessary to evaluate their performance in a military scenario before any conclusions could be made about their suitability for use in a military measurement-based admission controller. Due to the limited time frame for this thesis, there was only time to evaluate one of the algorithms. Therefore, the choice of the algorithm most likely to yield the best results had to be made.

Based on the literature review, the most promising algorithms were based on the hybrid approach of using both dispersion and delay analysis. I.e., the choice stood between the following candidates:

- CapProbe [28]
- Ad Hoc Probe [7]
- Allbest [14]
- The algorithm by Alzate et al. [1]

Based on their published results, all of these algorithms had the potential of meeting the quantitative requirements from section 2.3. However, all did not meet the qualitative requirements. As a foundation for further discussion, the qualitative requirements derived in section 2.3.1 are relisted here:

The path capacity estimator:

- *must not* rely on communication with the forwarding nodes in the black network
- *must* be compatible with CSMA and/or TDMA and/or FDMA
- *should* be compatible with wireless multi-hop communication where the same channel is shared between multiple nodes using CSMA/CA
- *must not* rely on the assumption of symmetric path capacities
- *must* work even if the estimation traffic is subject to packet fragmentation
- *should* be compatible with rate-limited paths

2.5.1 Discussion

CapProbe was based on two-way measurements, thus relying on symmetric path capacities. However, the one-way variant of CapProbe, Ad Hoc Probe, was indeed promising. In [7], strong simulation and test bed results were reported, even for multi-hop wireless communication. Furthermore, the Ad Hoc Probe source code was publicly available, eliminating the need to use a lot of time on implementing the algorithm.

Allbest seemed like a very promising approach, but it had to be modified to be one-way. Moreover, the published results only showed the algorithm's performance in a single-hop topology. In contrast to Ad Hoc probe, there was no publicly available source code.

The algorithm by Alzate et al. was based on taking two measurements based on two different packet sizes, and further interpolate the path capacity for all other packet sizes. This would for sure not work in the presence of rate-limiting QoS mechanisms that enforce different path capacities for different traffic types. If the algorithm was modified to make more measurements, it would converge to correspond to Ad Hoc Probe. Furthermore, in [1] only results from simulations were presented and not any test bed implementations.

Based on these considerations and the limited time frame for the thesis work, Ad Hoc Probe was chosen as the candidate for further, in-depth performance evaluation.

2.5.2 The Ad Hoc Probe Algorithm

In this section, a detailed walkthrough of the Ad Hoc Probe algorithm is given to provide a basis for later analysis of the algorithm's performance.

Basic concept Ad Hoc Probe [7] uses *one-way* measurements, requiring specific software at both ends of the path. The algorithm is based on packet-pair dispersion and the observation that compressed or expanded packet-pairs have been subject to queuing delays somewhere in the path. The packet-pair with the *minimum sum of OWDs* is therefore assumed to be the correct sample, even though its relative frequency may be low compared to other samples if the path is loaded with heavy cross traffic.

The path capacity estimate calculated using the relation

$$C = \frac{\texttt{PSize}}{\Delta t}$$

where `PSize` is the probe packet size and Δt is the dispersion of the packet-pair with the minimum OWD sum.

Clock synchronization Since the algorithm is depending on delay measurements, one may object that this requires clock synchronization between the sender and the receiver. This is however not the case. Following the reasoning in [7], let T_{snd_i} be the local time that is stamped on the i-th packet-pair when it is put into the sender's transmission queue. Further, let the packet-wise local arrival times at the receiver be T_{rcv1_i} and T_{rcv2_i} for the first- and second-arriving packet, respectively. The *observed* packet OWD sum at the receiver for the i-th packet-pair is then given by

$$S_i' = \left(T_{rcv1_i} - T_{snd_i}\right) + \left(T_{rcv2_i} - T_{snd_i}\right)$$

Assuming that the sender and receiver are incrementing their clock at the same rate, the local time at the sender will be at a constant offset δ from the receiver's local time. I.e., the "actual"

departure time from the receiver's point of view is

$$T_{snd_i} + \delta$$

Hence, the *true* packet delay sum is

$$\begin{aligned} S_i &= \left(T_{rcv1_i} - \left(T_{snd_i} + \delta\right)\right) + \left(T_{rcv2_i} - \left(T_{snd_i} + \delta\right)\right) \\ &= S'_i - 2\delta \end{aligned}$$

which clearly shows that if δ is constant, the packet-pair with the minimum *observed* OWD sum is the pair with the *true* minimum OWD sum.

Clock skew issue Unfortunately, it turns out that the assumption of the same clock incrementation rate is invalid. The clock at both ends of the path will in most cases drift independently of each other, resulting in a time-varying clock offset $\delta(t)$, where t here is the local time at the receiver. Hence, the resulting OWD sum measurements can be expressed as

$$S'_i = S_i + 2\delta(T_{snd_i})$$

If the clock offset function is assumed to be linear of the form $\delta(t) = a + bt$, the receiver's clock is incrementing at a faster rate than the sender clock. The measured OWD sum would show an increasing trend, and it could occur that one of the earliest received packet-pairs consequently was chosen by Ad Hoc Probe as the minimum OWD sample. In the opposite case, if $\delta(t) = a - bt$, the trend would be negative, and the same could happen except that it would be the latest samples that were chosen.

This issue could result in serious measurement errors; the worst-case being that the path capacity was calculated on the basis of the dispersion of a compressed or expanded packet-pair. Ad Hoc Probe deals with this issue by detecting an increasing or decreasing trend in the OWD sum measurements. Pseudo-code describing the detection method is given in algorithm 2.1

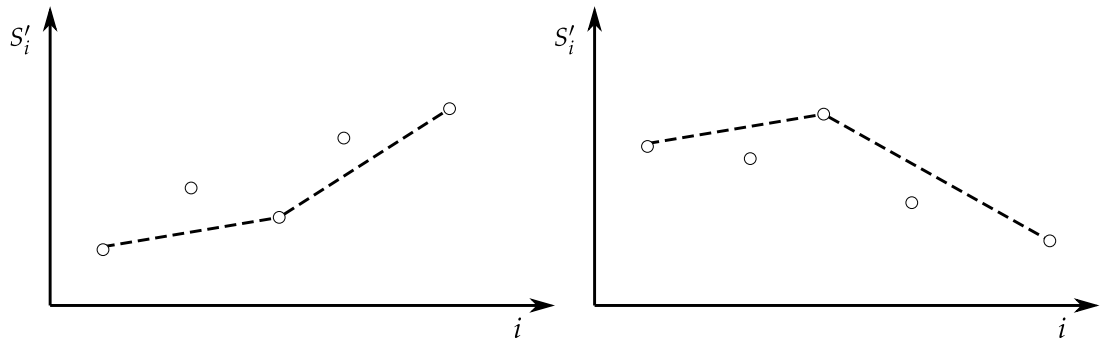
For each of the N received packet-pairs, the counter *trend* is incremented if the packet-pair that has a larger OWD sum than the previous, and decremented in the opposite case. This way, a decreasing trend results in a large negative counter value, while an increasing trend would result in a large positive value. The counter *trend* is compared to a predetermined threshold value K and the trend is reported as increasing, decreasing, or none of these. In the latter case, no clock skew correction is necessary.

If an increasing or decreasing trend is determined, the algorithm tries to find a set of packet-pairs that are likely to be minimum delay sum samples. A method based on the convex hull approach in [47] is used to do this selection. Figure 2.6 shows the concept. The samples that lie on the upper/lower convex hull are selected according to the reported trend. Since all the points lying on the convex hull of the data set $\Omega_{S'} = (T_{rcv2_i}, S'_i)$ not necessarily are minimum OWD sum samples, the same algorithm is applied to the data set containing the first-packet OWDs; $\Omega_1 = (T_{rcv2_i}, \mathrm{OWD1_i})$. Taking the intersection of Ω'_S and Ω_1 will filter out some

Algorithm 2.1 The Ad Hoc Probe algorithm for detecting trends in the OWD sum measurements

$trend \leftarrow 0$
for $i = 2$ to N **do**
  if $S'_i > S'_{i-1}$ **then**
    $trend \leftarrow trend + 1$
  else if $S'_i < S'_{i-1}$ **then**
    $trend \leftarrow trend - 1$
  end if
end for
if $trend > K$ **then**
  Report $\delta(t)$ is with increasing trend
else if $trend < -K$ **then**
  Report $\delta(t)$ is with decreasing trend
end if

of the compressed packet-pairs, however there is still a significant chance that "bad" packet-packet-pairs are present in the data set. The final path capacity estimate is therefore given based on the mean dispersion of the packet-pairs in the set $\Omega_{S'} \cap \Omega_1$ in an effort to average out the effect of bad samples.



(a) Increasing trend (b) Decreasing trend

Figure 2.6: Selection of the samples constituting the convex hull

Chapter 3

Estimating Path Capacity under Ideal Conditions

In the previous chapter, Ad Hoc Probe [7] was chosen as the best candidate for further evaluation. This and the next chapter documents the experiments that were performed to gain a deeper understanding of how Ad Hoc Probe works, and whether or not it is suitable for being part of a military Measurement-based Admission Control (MBAC)-scheme.

This chapter will focus on how Ad Hoc Probe performs under the ideal conditions of no cross traffic. The goal is to evaluate under what constraints— if any —it conforms to the performance requirements derived in section 2.3. In other words, this chapter is dedicated to evaluating under what constraints the following hypothesis holds:

> Ad Hoc Probe can provide path capacity estimates that comply to the requirements in section 2.3 when there is *no other traffic present in the network*.

The chapter is organized as follows: First, a review of the experiment set-up is given. Next, the obtained results are presented in a dedicated section in order to lay the foundation for the following discussion. Finally, the chapter is concluded with an assessment of the hypothesis above, in the light of the knowledge acquired from the experiments.

3.1 Experimental Set-up

3.1.1 Test Bed Network Topologies

The experiments were conducted at Forsvarets Forskningsinstitutt (Norwegian Defence Research Establishment) (FFI) using the test bed network topologies as shown in figures 3.1 and 3.2. In the first topology, several different technologies were used for the link L in order to test the algorithm's compatibility and accuracy.

In the topology illustrated in figure 3.2, the test bed was spread out in different rooms of a large office environment in an effort to achieve the transmission range pattern as illustrated in the

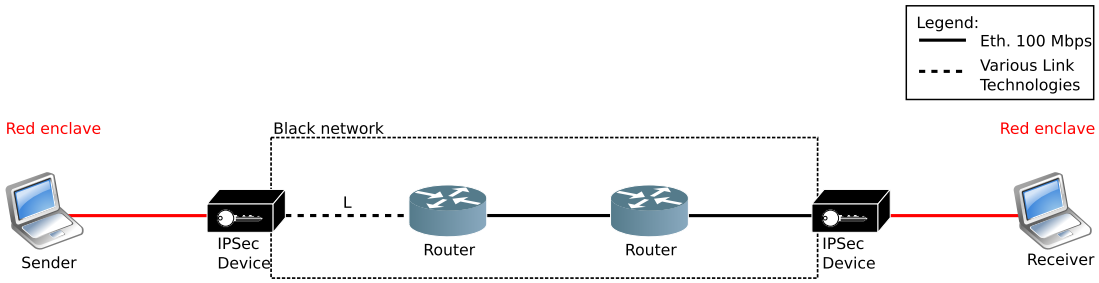


Figure 3.1: Test bed topology for measuring the performance of Ad Hoc Probe [9]

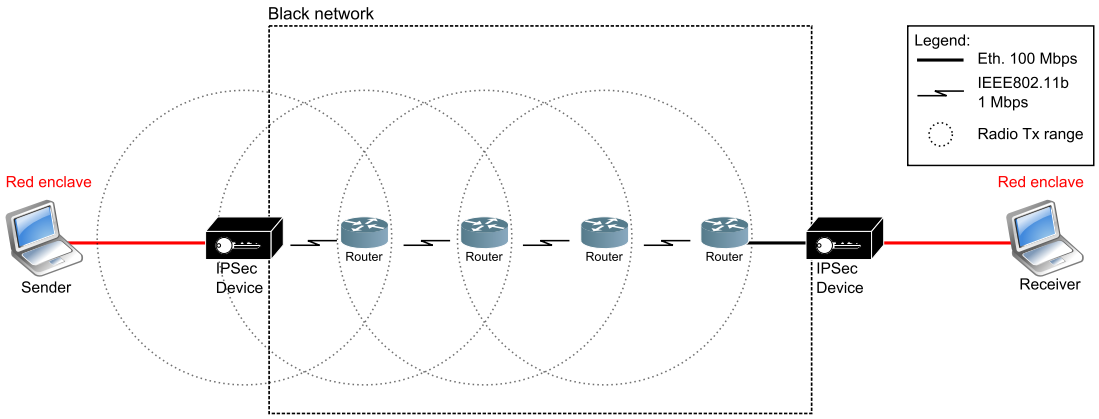


Figure 3.2: Test bed topology for measuring the performance of Ad Hoc Probe in a wireless multi-hop network [9]

figure. The wireless links were sharing a single channel. Most of the routers were separated by concrete walls that attenuated the signal significantly. The end-to-end distance in the figure was approximately 150 m. In such a scenario, there are many strong multipath components that vary with the position of surfaces like doors and windows, resulting in a time-varying transmission range. Therefore, a static routing configuration was employed to keep the number of wireless hops under control. Figure 3.2 shows the case of four wireless hops. In order to vary the number of wireless hops in the experiments, the physical connection point of the receiving IPsec device was changed within the wireless chain.

3.1.2 Link Technologies

With the intention of verifying the requirements in section 2.3.1 for compatibility with different Medium Access Control (MAC)-schemes, the following link technologies were selected:

- IEEE802.3 [19]
- IEEE802.11b [20]
- Commercial satellite modem [10]
- Tactical UHF personal radio [11]

IEEE802.3 This wired link technology is in widespread civilian and military use, often simply referred to as "Ethernet". The main reason for including IEEE802.3 in the experiments was to evaluate whether Ad Hoc Probe could be used for admission control decisions between headquarters[†], where the path capacity typically is limited by the Ethernet connections.

The IEEE802.3 MAC-protocol is based on Carrier Sense Multiple Access Collision Detection (CSMA/CD), but since most Ethernet networks are deployed in a star topology where every connected node has a dedicated physical wire, there will never be any collisions triggering the random access mechanism built into the MAC-protocol.

IEEE802.11b IEEE802.11b has limited military application due to its physical layer specifications. The use of the unlicensed 2.4 and 5 GHz Industrial, Scientific and Medical (ISM) frequency bands results in poor long range propagation properties. Additionally, since these bands are unlicensed, destructive interference from other communication systems or electrical devices is a significant problem. However, the IEEE802.11b MAC protocol is based on Carrier Sense Multiple Access Collision Avoidance (CSMA/CA), which makes it very interesting to evaluate the estimators performance over this link technology, considering that contention based medium access is widely deployed in military Mobile Ad Hoc Networks (MANETs).

The experiments were conducted using Alfa Network AWUS036NEH Wireless USB adapters [39] in IEEE802.11b 1 Mbps Ad Hoc[†] mode using the Distributed Coordination Function (DCF) variant of the MAC protocol with Ready-to-send (RTS)/Clear-to-send (CTS) enabled.

Commercial Satellite Modem Considering the key role of satellite communication in military networks, and the frequent use of Frequency Division Multiple Access (FDMA) in these types of links, it was natural to include a satellite link in the experiments.

The employed satellite modems were based on proprietary technology. Consequently, the access to technical documentation was limited. However, it was clear that the modems operated as an Ethernet bridges, relaying traffic that was destined for nodes situated on the other side of the satellite link. In the experiments, two of these modems were connected via a satellite emulator that imposed a propagation delay corresponding to what would be the case if the signals passed through a geostationary satellite transponder. The emulated satellite's transponder bandwidth was shared through the use of a static FDMA approach. I.e, the link was allocated a dedicated part of the transponder bandwidth. The information data rate of the link was set to 2048 kbps, and the channel was practically noise-free.

Tactical UHF personal radio A battery-powered tactical personal radio operating in the Ultra-High Frequency (UHF)-band was used for evaluating Ad Hoc Probe over a typical Time Division Multiple Access (TDMA)-based link found in a military network. There was little documentation providing the tactical radio specifications. However, a user manual containing some parameters was available, and by conferring with personnel at FFI the most important properties

[†]I.e., office environments

[†]Independent Basic Service Set (IBSS)

of the radio were clarified. The radio supports bandwidths of 500 and 1200 Hz; providing an expected layer 1 information bit rate, R_b, of respectively 135 and 338 kbps. The medium access scheme is based on TDMA using frames of 125 ms with 8 time slots of equal lengths, of which 6 time slots can be allocated for user data while the other two are reserved for signaling and a voice break-in channel. The user time slots can be set-up in different combinations to share the radio capacity between dedicated voice service and packet data transfer.

In these experiments, the radio was used in two configurations. The first a low-capacity mode using a bandwidth of 500 Hz and 2 of 6 time slots for packet data transfer, resulting in an *average* information bit rate of $135/8 \cdot 2 \approx 30$ kbps. In the second configuration, the radio was operating at full speed - using 6 of 6 time slots over a bandwidth of 1200 Hz with an *average* information bitrate of $338/8 \cdot 6 \approx 250$ kbps.

The radio was connected to the involved computers via Universal Serial Bus (USB) revision 2.0.

3.1.3 Software Configuration

The measurements were made using the CapProbe project's C-implementation of Ad Hoc Probe, downloaded from the University of California, Los Angeles (UCLA) Network Reserch Laboratory website [30]. Only minor changes were made to the source code in order to change parameters without needing to recompile, and to generate output in a more processing friendly format. The source code that was compiled and installed on the probing and receiving machine is available in appendix B.5.

In order to use this algorithm for admission decisions, there has to be a feedback connection from the receiver to the sender. This was not implemented in these experiments since the main focus was to evaluate the performance of the algorithm, and not to provide a complete implementation of an admission control scheme.

All computers that were used in the experiments had at least 1.7 GHz CPU, 512 MB RAM. The end hosts were running Ubuntu Linux [35]. All the routers, including the ones configured as IPsec end-points, were running a router software based on Vyatta [21]. Static routing and address configuration was employed. The router configuration files are of some length and needed to be changed on a scenario basis. Due to space consideration, only the configuration files for one of the scenarios is enclosed in appendix B.3. The Vyatta default Quality of Service (QoS) configuration is based on placing packets into different First In, First Out (FIFO) queues based on the Differentiated Services Code Point (DSCP)-field in the IP header. In these experiments, the DSCP-field was the same on all the traffic that was sent, and the Vyatta routers thus acted as single FIFO queues.

In military networks, the Maximum Transmission Unit (MTU) is not necessarily set to the IEEE802.3 standard of 1500 bytes. It was therefore interesting to evaluate the effect of fragmentation on the accuracy of Ad Hoc Probe. The network path MTU was set to 1200 bytes; letting ciphertext IP packets up to and including 1200 bytes pass through unfragmented.

Rate limiters are commonly employed QoS mechanisms in civilian and military networks. In one of the experiments, a token bucket 500 kbps rate limiter with maximum burst size set to

0, was configured on one of the routers in the path in order to find out if Ad Hoc Probe could detect the maximum capacity of a rate limited path.

The Address Resolution Protocol (ARP) caused some cross traffic in the test bed. The capacity needed for this traffic was however insignificant when the path capacity was in the order of Mbps. In the case of low capacity channels, the ARP was disabled and the layer 2 addresses were configured manually.

3.1.4 Ad Hoc Probe Parameters

Ad Hoc Probe has four adjustable parameters; the size of the packets in a packet-pair, `PSize`, the number of packet-pairs that constitute an estimate, `Numprobes`, the length of the packet-pair interdeparture time[†], `Int`, and the clock skew correction threshold. The quantifiable estimator requirements from section 2.3 are threefold, including bounds on the estimator's accuracy, intrusiveness and measurement time.

Since the path capacity inherently varies by `PSize` due to the varying relative overhead, this was a parameter that had to be included to allow for considerations of the estimator accuracy. In military networks, small packets are often used for tactical voice or messages, and it is very interesting to evaluate Ad Hoc Probe's accuracy for this kind of traffic.

Considering the path capacity dependence on `PSize`, this was regarded as a fixed parameter when it came to evaluating the intrusiveness of the estimator. A key question was how many packet-pairs of this fixed packet size it was necessary to transmit. Therefore, the estimator performance for different settings of `Numprobes` was included in the analysis.

When the network is uncongested, the queuing and transmission delay of a packet pair will be small compared packet-pair interdeparture time. Therefore, it is a valid assumption that the measurement time is given by the product of `Numprobes` and the interdeparture time, given that the probe rate is much less than the path capacity. In these measurements, the probing rate was always kept well below the path capacity and the minimum measurement time could therefore be inferred after determining the minimum `Numprobes`-value.

The effect of varying the clock skew correction threshold was not investigated in order to limit the degrees of freedom in the experiment. The default value of 30 from [7] was used, effectively disabling the clock skew correction algorithm for `Numprobes`< 30.

3.1.5 Measuring the True Path Capacity

For each test bed configuration, the path capacity was estimated for different IP packet sizes using a "brute-force" technique based on generating UDP traffic with the Multi-Generator (MGEN)[18]. For each IP packet size, `PSize`, a flow of packets, lasting `MGENlength`, was sent towards the receiver with a constant packet interdeparture time corresponding to the data rate `MGENrate`, set greater than the path capacity. In order to avoid basing the measurement on data containing transient effects due to transmission buffer emptying/filling, there was a pause

[†]Time between sending each packet-pair

of 10 seconds between flows of different packet sizes and the test script counted received data for a period of `MGENlength`-10 seconds, starting 5 seconds into each received flow. The path capacity was calculated by dividing the count of received bits by (`MGENlength`-10). See the BASH and AWK scripts enclosed in appendix B.4 for implementation details.

Following the definition of path capacity in section 2.1.1, `MGENlength` should have been infinitely long. Nevertheless, it is assumed in this thesis that the results from these measurements were close to the true path capacity, using `MGENlength` in the order of tens of seconds.

3.1.6 Summary

Tables 3.1 and 3.2 summarize the most important parameters that were introduced in the previous subsections. In addition, all the different values that were assigned to these parameters during the experiments, are listed.

Table 3.1: Overview over parameters that were varied during the experiments

Parameter	Description	Values used in measurements
`L`	Test bed link technologies	IEEE802.11b $R_b = 1$ Mbps IEEE802.3 10BASE-T $R_b = 10$ Mbps Commercial satellite modem $R_b = 2048$ kbps Tactical UHF personal radio 500 kHz 2 of 6 time slots for packet data transfer $R_b \approx 30$ kbps Tactical UHF personal radio 1200 kHz 6 of 6 time slots for packet data transfer $R_b \approx 250$ kbps
`Ratelim`	Token bucket rate limit	500 kbps, Disabled
`PSize`	Plaintext (red) IP packet size	100-1500 bytes
`Numprobes`	Number of packet-pairs per Ad Hoc Probe estimate	10-100
`Int`	Ad Hoc Probe packet-pair interdeparture time	0.1-2 s
`MGENrate`	Average layer 3 bit rate generated by MGEN during brute-force measurement	0.5-12 Mbps
`MGENlength`	Duration of MGEN flow for each IP packet size during brute-force measurement	30 s, 250 s (for IEEE802.11b)

Table 3.2: Overview over the most relevant parameters that were kept constant during all the experiments

Description	**Value used in measurements**
Largest allowed size of layer 2 payload in path (MTU)	1200 bytes
Number of Ad Hoc Probe estimates per trial	20
Ad Hoc Probe clock skew correction threshold	30

3.2 Results

In this section, the results from all experiments are presented along with short comments describing the key observations. The results are interpreted and analyzed section 3.3.

3.2.1 The Effect of Varying the Packet Size

The results presented in this subsection are 95 % confidence intervals for the Ad Hoc Probe capacity estimator expected value, $\mathrm{E}\{\widehat{\mathrm{C}}\}$. Only packet sizes that did not result in packet fragmentation according to table B.1 in appendix B were used. The number of probes per measurement was fixed to 100 in order to provide a very good probability for obtaining a "good" minimum One-Way Delay (OWD) sample, thus providing Ad Hoc Probe with the best conditions possible, resulting in the highest achievable estimation accuracy. The 95 % confidence intervals were calculated using the sample mean and standard error of 20 observations of $\widehat{C}$ and the Student-t distribution with 19 degrees of freedom.

Capacity limited by single-hop Figure 3.3 shows the results obtained for the test bed topology illustrated in figure 3.1 where the path capacity was limited by two common open standard link technologies. The test bed configuration parameters are listed in table 3.3.

Table 3.3: Test bed configuration parameter values

(a) Configuration for results in figure 3.3a

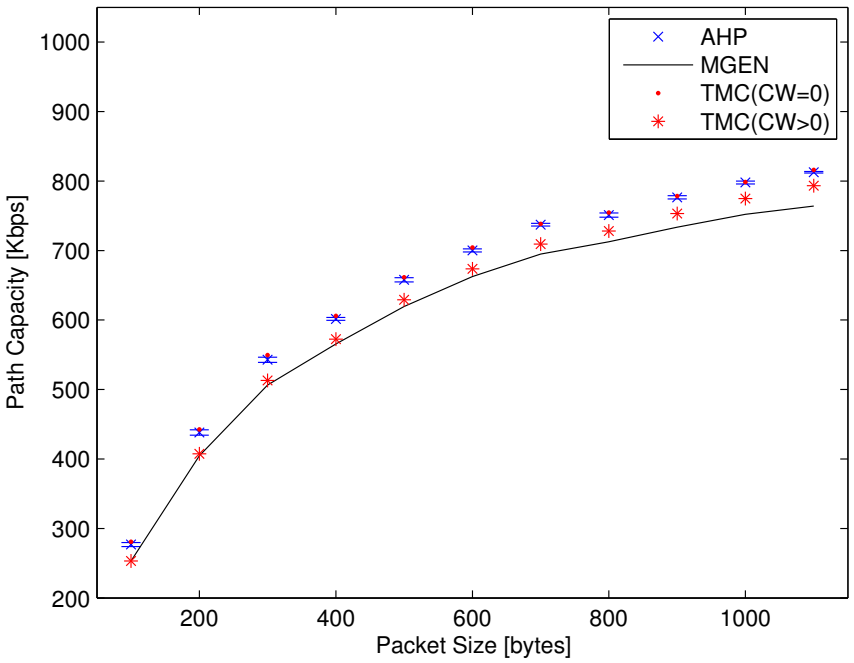
Parameter	Value
L	IEEE802.11b $R_b = 1$ Mbps
Ratelim	Disabled
PSize	100-1100 bytes
Numprobes	100
Int	0.1 s
MGENrate	3 Mbps
MGENlength	250 s

(b) Configuration for results in figure 3.3b

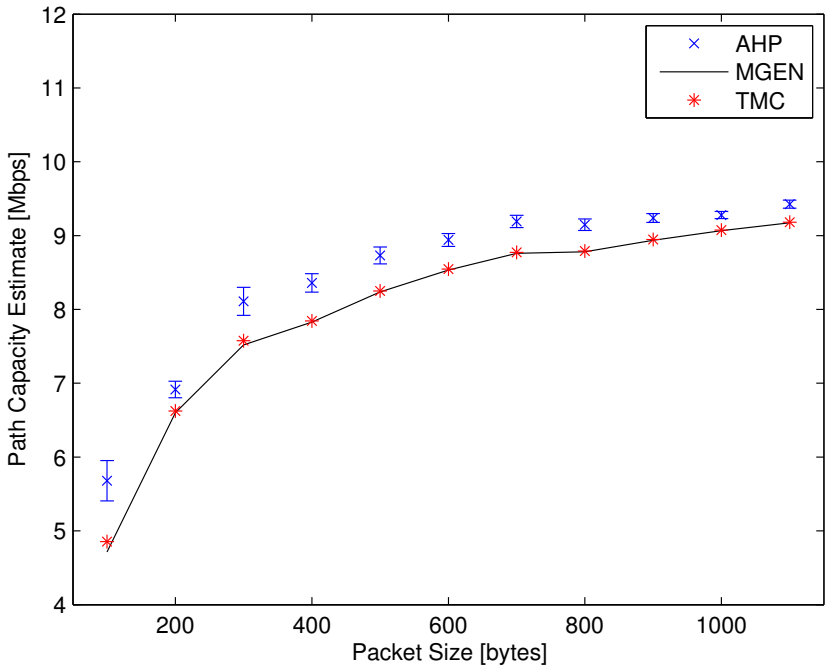
Parameter	Value
L	IEEE802.3 10BASE-T $R_b = 10$ Mbps
Ratelim	Disabled
PSize	100-1100 bytes
Numprobes	100
Int	0.1 s
MGENrate	12 Mbps
MGENlength	30 s

In addition to the path capacity estimations, the plots also show the links' TMCs; calculated using equations (A.1) and (A.2) from appendix A, including the IPsec overhead. The TMC in figure 3.3a is calculated in two variants; one where the average Contention Window size is included, and one where the Contention Window is set to zero. The MGEN brute-force results for the IEEE802.11b experiments were based on a flow duration of 250 s in order to reduce the measurement variance caused by the random medium access.

Common for both link technologies was the fact that Ad Hoc probe consistently overestimated the path capacity. In the case of IEEE802.11b, the estimates conformed to the TMC not including the Contention Window, and the true path capacity gradually diverged from the TMC including the average Contention Window as the packet size increased. The IEEE802.3 true path capacity conformed perfectly to the TMC.



(a) L: IEEE80211b $R_b = 1$ Mbps



(b) L: IEEE802.3 10BASE-T $R_b = 10$ Mbps

Figure 3.3: Path capacity estimation results. $\mathrm{E}\{\widehat{\mathrm{C}}\}$ with 95 % confidence intervals, MGEN brute-force method and TMC.

Capacity limited by multi-hop Figure 3.4 shows the results obtained for test bed topology illustrated in figure 3.2 where the path capacity was limited by a wireless shared-channel multi-hop topology. The test bed configuration parameters are listed in table 3.4. The TMC-calculations were based on the single-hop TMC divided by the number of hops [34] .

Table 3.4: Test bed configuration parameter values for results in figure 3.4

Parameter	Value
L	1-4 hop IEEE802.11b chain $R_b = 1$ Mbps
Ratelim	Disabled
PSize	400, 700 bytes
Numprobes	100
Int	0.2 s
MGENrate	1 Mbps
MGENlength	250 s

The key observations from the multi-hop results were that Ad Hoc Probe conformed to the TMC with no Contention Window up to and including two hops, thereafter following a "delayed" pattern; converging on the TMC divided by (number of hops-1). Furthermore, the TMC with no Contention Window converges towards the regular TMC as the number of hops increase.

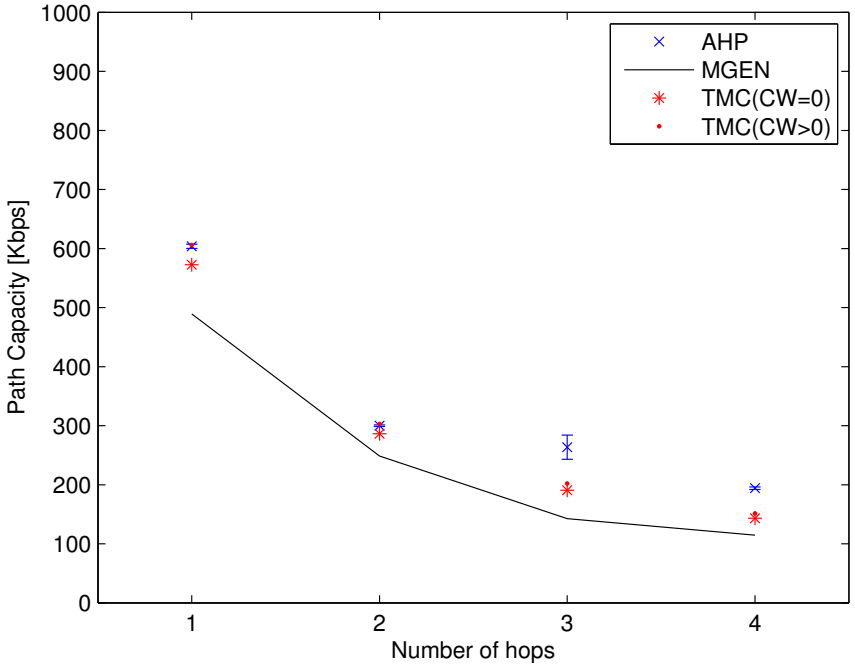
Tactical link Figure 3.5 shows the results from a scenario where the path capacity was limited by a link between two tactical UHF radios. The radios were used in two different configurations as listed in table 3.5; the first configuration providing a very low link capacity, the second the maximum link capacity of the radio. Only packet sizes in the range 100-700 bytes were used when the radio was in the low-capacity mode, since larger packet sizes could lead to a probing rate greater than the path capacity. By increasing the probe interdeparture time, congestion could have been avoided, but this would have made it impossible to complete the measurement script within the lifetime of a single battery.

In figure 3.5a, when the radio was configured in the low-capacity mode, Ad Hoc Probe severely overestimated the path capacity for packet size 100 bytes. For the other packet sizes, the measurement bias followed an increasing trend, starting out with a relatively large underestimation at packet size 200 bytes and ending up at a relatively large overestimation at 700 bytes.

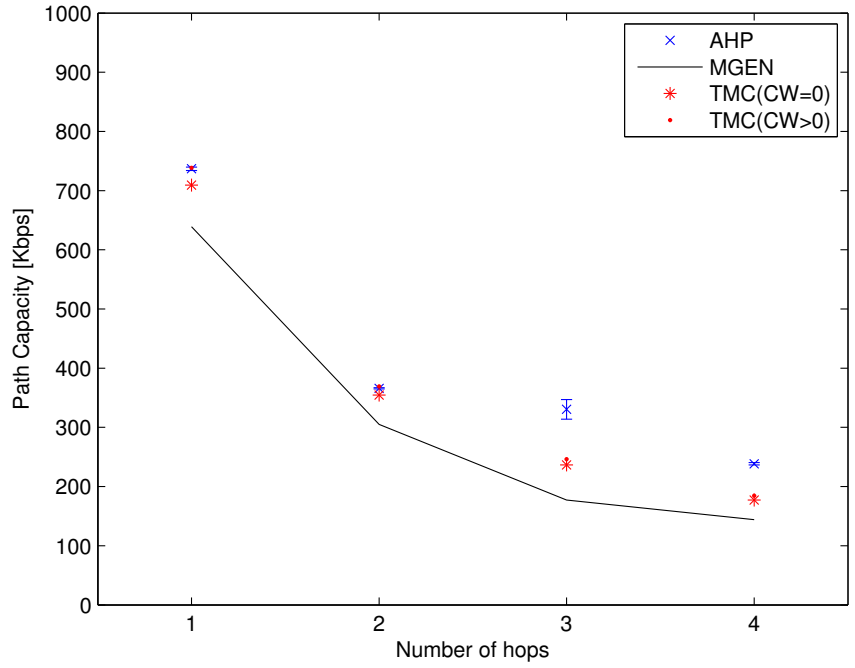
Severe overestimation was the case for all packet sizes in figure 3.5b. In addition, the estimates followed a linearly increasing trend until 600 bytes. For the larger packet sizes, there was no clear pattern.

Satellite link The results for a test bed configuration including a satellite link are presented in figure 3.6. Table 3.6 contain the text bed configuration parameters that were used.

Even though a relatively small but consistent overestimation was observed, Ad Hoc Probe conformed well to the true path capacity.

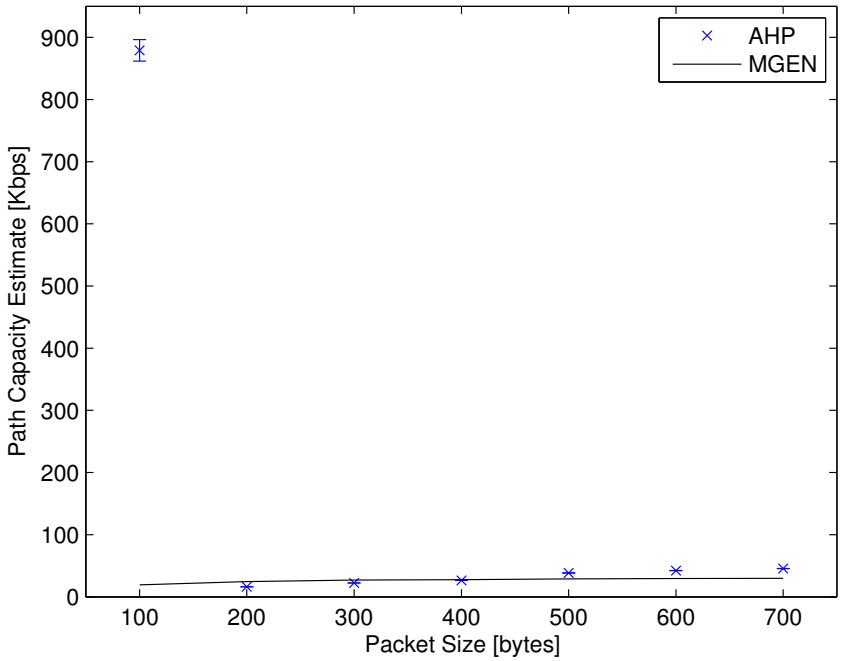


(a) `PSize`: 400 bytes

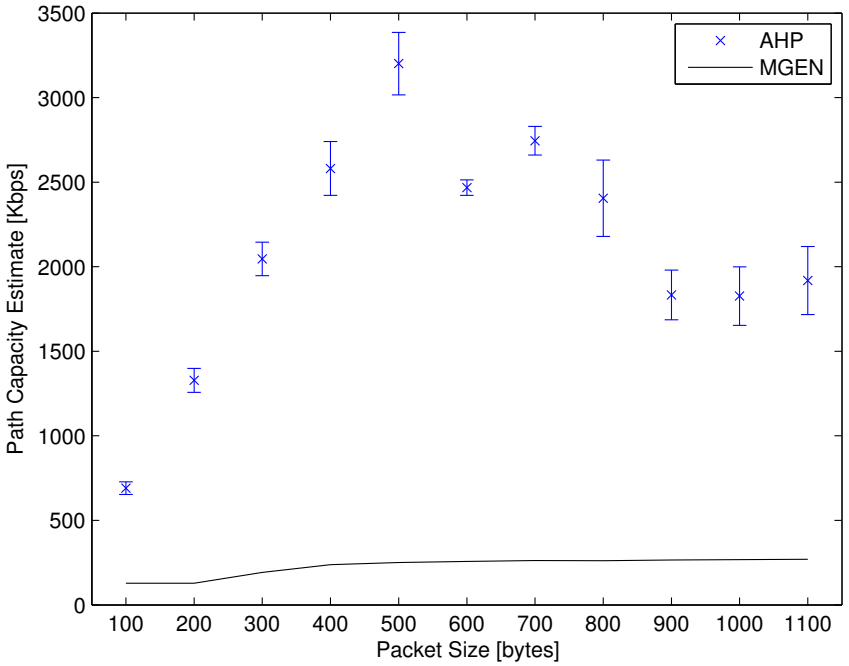


(b) `PSize`: 700 bytes

Figure 3.4: Path capacity estimation results for multi-hop test bed topology. $\mathrm{E}\{\widehat{\mathrm{C}}\}$ with 95 % confidence intervals, MGEN brute-force method and TMC divided by number of hops.



(a) L: Tactical UHF radio 500 kHz bandwidth 2/6 time slots



(b) L: Tactical UHF radio 1200 kHz bandwidth 6/6 time slots

Figure 3.5: Capacity estimation results for a path including a tactical UHF radio link. $\mathrm{E}\{\widehat{\mathrm{C}}\}$ with 95 % confidence intervals and MGEN brute-force measurements

Table 3.5: Test bed configuration parameter values

(a) Configuration for results in figure 3.5a

Parameter	Value
L	Tactical UHF personal radio 500 kHz 2 of 8 time slots used for packet data $R_b = 135$kbps *Average* information bit rate: $135/8 \cdot 2 \approx 30$ kbps
Ratelim	Disabled
PSize	100-700 bytes
Numprobes	100
Int	2 s
MGENrate	500 kbps
MGENlength	30 s

(b) Configuration for results in figure 3.5b

Parameter	Value
L	Tactical UHF personal radio 1200 kHz 6 of 8 time slots used for packet data $R_b = 338$kbps *Average* information bit rate: $338/8 \cdot 6 \approx 250$ kbps
Ratelim	Disabled
PSize	100-1100 bytes
Numprobes	100
Int	1 s
MGENrate	500 kbps
MGENlength	30 s

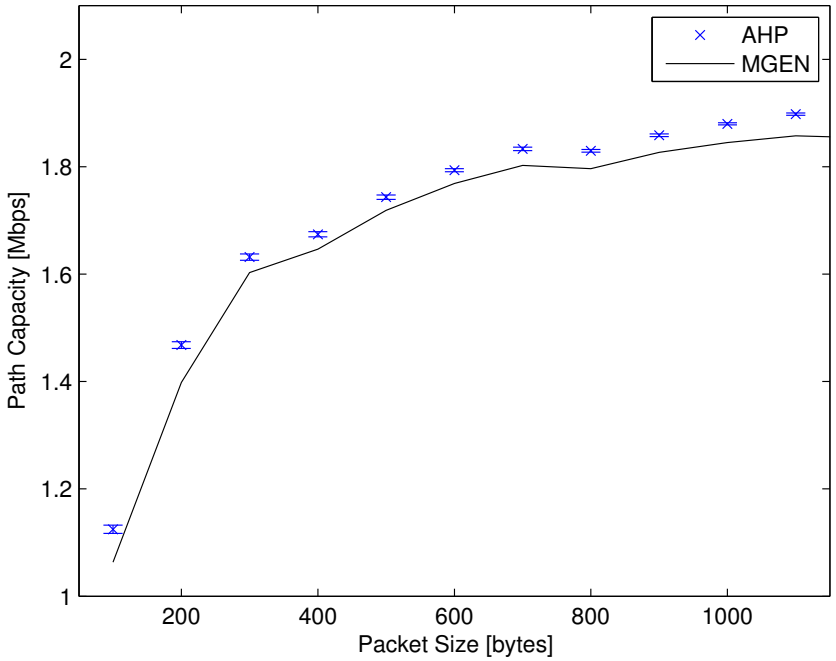


Figure 3.6: Path capacity estimation results for various plaintext packet sizes over a path including a satellite link. $\mathrm{E}\{\widehat{\mathrm{C}}\}$ with 95 % confidence intervals and MGEN brute-force measurements.

Table 3.6: Test bed configuration parameter values for results in figure 3.6

Parameter	Description
`L`	Commercial satellite modem $R_b \approx$ 2048kbps
`Ratelim`	Disabled
`PSize`	100-1100 bytes
`Numprobes`	100
`Int`	0.2 s
`MGENrate`	5 Mbps
`MGENlength`	30 s

IP Packet Fragmentation Figure 3.7 shows the results from Ad Hoc Probe, MGEN brute-force measurements and the TMC for the packet sizes that were observed to result in packet fragmentation (see table B.1). For packet sizes 1200-1500 bytes, the TMC takes into account the increased layer 3 overhead for due to fragmentation, and the layer 2 overhead of two Ethernet frames. Table 3.7 contains the test bed configuration.

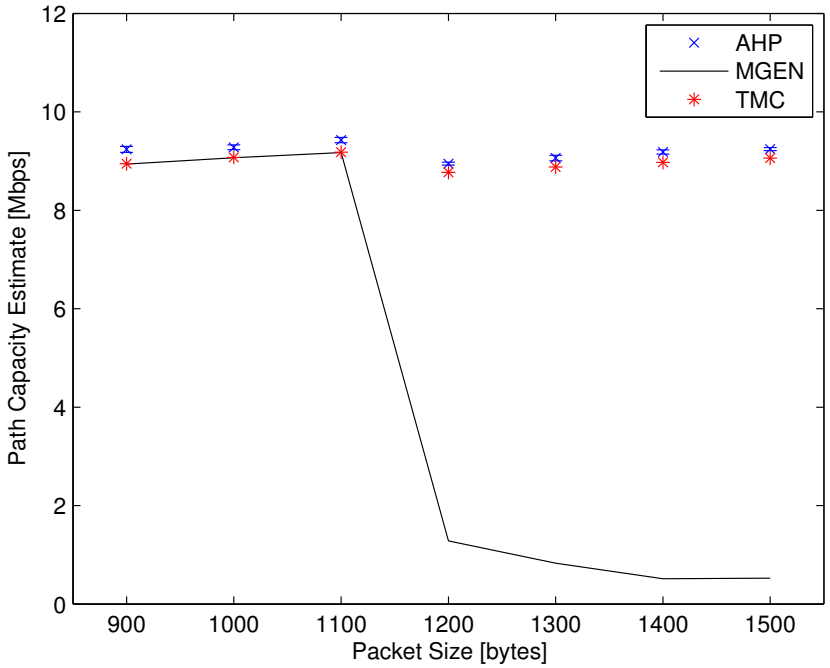


Figure 3.7: Path capacity estimation results. $\mathrm{E}\{\widehat{\mathrm{C}}\}$ with 95 % confidence intervals, MGEN brute-force measurements and TMC for various packet sizes that resulted in fragmentation.

The Ad Hoc Probe estimates were in accordance with the TMC, and did capture the expected drop in capacity due to the increased overhead imposed on fragmented packets. By contrast, the MGEN brute-force estimates did not conform to the TMC, reporting a very low path capacity for fragmented packets.

Table 3.7: Test bed configuration parameter values for results in figure 3.7

Parameter	Description
`L`	IEEE802.3 10BASE-T $R_b = 10$ Mbps
`Ratelim`	Disabled
`PSize`	900-1500 bytes
`Numprobes`	100
`Int`	0.1 s
`MGENrate`	12 Mbps
`MGENlength`	30 s

Rate limiter In the results presented in figure 3.8, one of the Vyatta routers in the path was configured to use a token bucket 500 kbps rate limiter with burst size set to zero. Table 3.8 lists the test bed configuration.

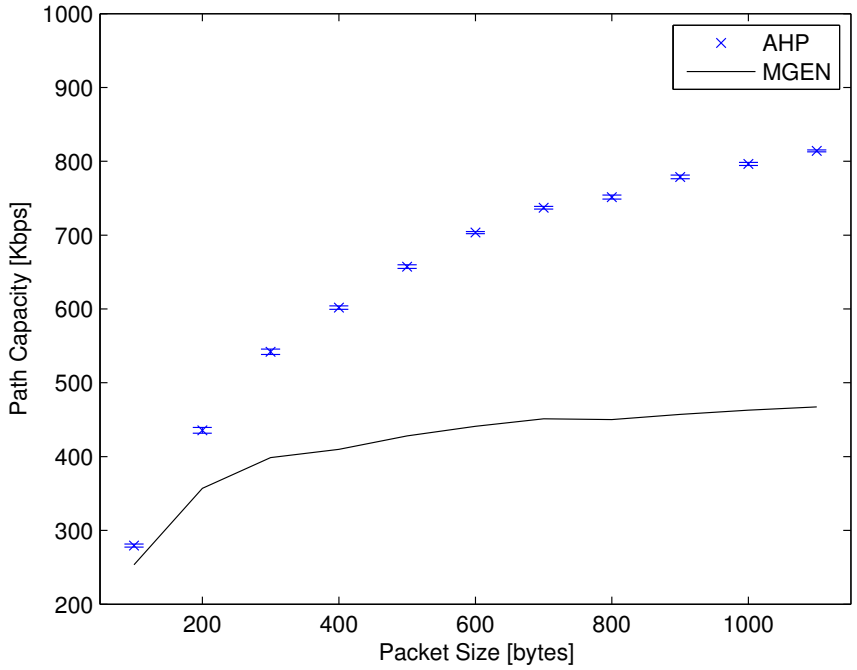


Figure 3.8: Path capacity estimation results for rate limited path. $\mathrm{E}\{\widehat{\mathrm{C}}\}$ with 95 % confidence intervals and MGEN brute-force measurements.

Ad Hoc probe failed to comply with the true path capacity for the rate-limited path. In stead, the estimates were equal to the results in figure 3.3a where the same test bed configuration was used, except that the rate-limiter was disabled.

3.2.2 The Effect of Varying the Number of Probes Per Measurement

Figure 3.9 shows estimates of the Ad Hoc Probe bias and standard deviation as a function of the number of probes per measurement. The figure shows the results obtained for various links in

Table 3.8: Test bed configuration parameter values for results in figure 3.8

Parameter	Description
L	IEEE802.11b $R_b = 1$ Mbps
Ratelim	500 kbps (set on first router after L)
PSize	100-1100 bytes
Numprobes	100
Int	0.1 s
MGENrate	3 Mbps
MGENlength	250 s

the topology illustrated in figure 3.1 and the four-hop wireless Carrier Sense Multiple Access (CSMA) chain illustrated in figure 3.2. The test bed configurations are listed in table 3.9.

Moreover, the tactical UHF radio TDMA-link was not included in these measurements since—for this link type —Ad Hoc Probe did not provide reasonable estimates for the optimal case of 100 probes per measurement, and improved performance was not expected if fewer probes were used. Considering this, it was regarded as not worth-while to vary the number of probes per measurement over the TDMA tactical radio.

The bias estimates are presented normalized to the true path capacity. I.e.,

$$\frac{\text{Bias}\{\widehat{\text{C}}\}}{\text{C}} = \text{E}\left\{\frac{\widehat{\text{C}}-\text{C}}{\text{C}}\right\} = \frac{\text{E}\{\widehat{\text{C}}\}}{\text{C}} - 1$$

where $\widehat{C}$ is the random Ad Hoc Probe path capacity estimator and C is the true path capacity, assumed to be the estimate provided by the MGEN brute-force method. The same normalization is applied to the standard deviation estimate.

$\text{E}\{\widehat{\text{C}}\}$ was unknown, and had to be estimated. Therefore, 95 % confidence intervals are presented, based on the sample mean and standard error of 20 observations of $\widehat{C}$ and the Student-t distribution with 19 degrees of freedom.

Since the only one sample was taken of the standard deviation, and the distribution of $\widehat{C}$ is unknown, a confidence interval is not provided for the standard deviation estimate. However, the estimate is calculated by taking the square root of an unbiased estimator of the variance of $\widehat{C}$, so it is expected to be close to the true value.

The results presented in figure 3.9 show no obvious coupling between the number of probes per measurement and the bias for the single-hop cases. A small improvement can be read from 10 to 30 number of probes per measurement in figures 3.9b and 3.9c, and from 80 to 100 in figure 3.9c. The standard deviation in the IEEE802.11b case displayed a decreasing trend until 60 probes per measurement, where there was a sudden increase.

In the multi-hop case of figure 3.9d, the bias was increasing with more number of probes per measurements, while the standard deviation showed a decreasing trend.

Table 3.9: Test bed configuration parameter values

(a) Configuration for results in figure 3.9a

Parameter	Value
L	IEEE802.11b $R_b = 1$ Mbps
Ratelim	Disabled
PSize	700 bytes
Numprobes	10-100
Int	0.1 s
MGENrate	3 Mbps
MGENlength	250 s

(b) Configuration for results in figure 3.9b

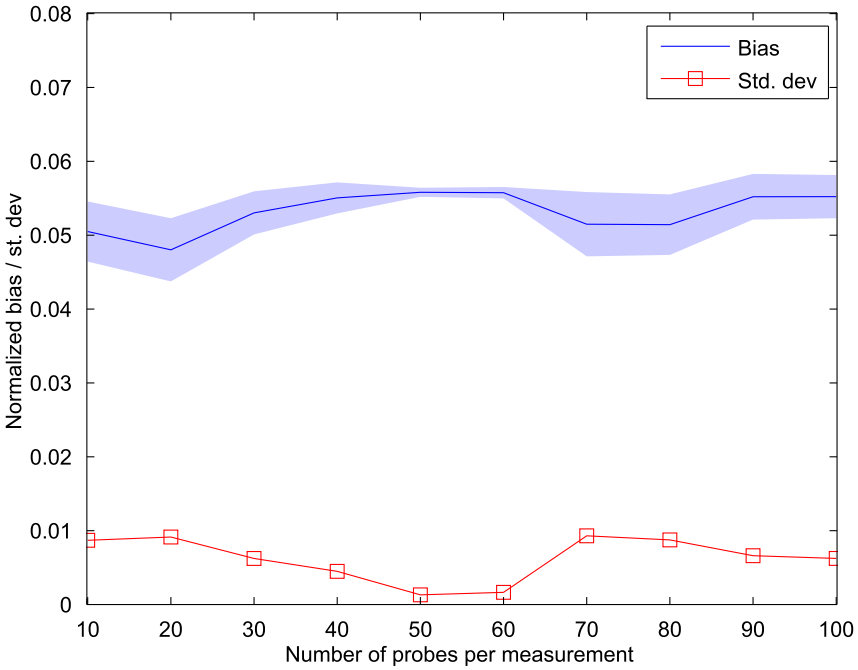
Parameter	Value
L	IEEE802.3 10BASE-T $R_b = 10$ Mbps
Ratelim	Disabled
PSize	700 bytes
Numprobes	10-100
Int	0.1 s
MGENrate	12 Mbps
MGENlength	30 s

(c) Configuration for results in figure 3.9c

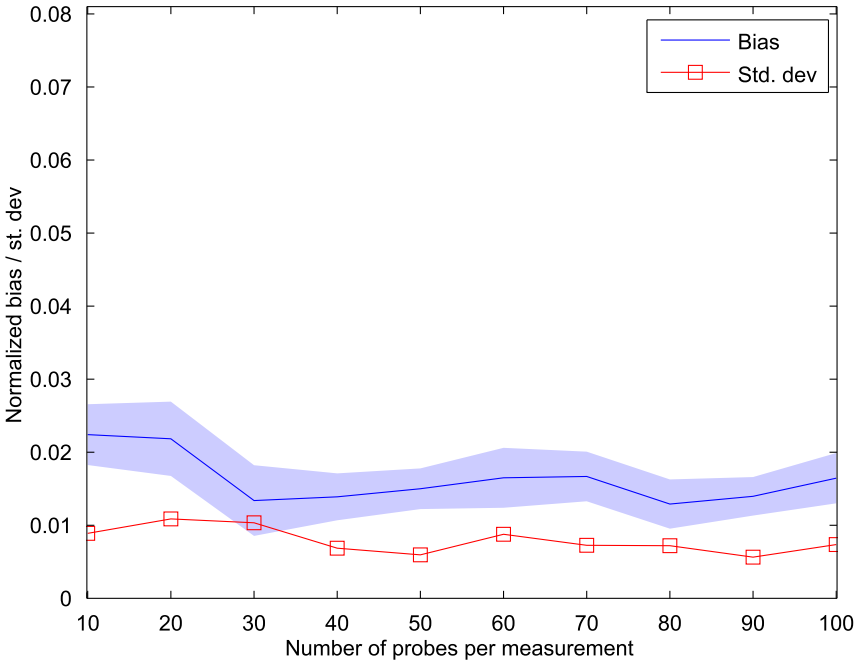
Parameter	Value
L	Commercial satellite modem $R_b = 2048$ kbps
Ratelim	Disabled
PSize	700 bytes
Numprobes	10-100
Int	0.2 s
MGENrate	5 Mbps
MGENlength	30 s

(d) Configuration for results in figure 3.9d

Parameter	Value
L	4-hop IEEE802.11b chain $R_b = 1$ Mbps
Ratelim	Disabled
PSize	700 bytes
Numprobes	10-100
Int	0.2 s
MGENrate	1 Mbps
MGENlength	250 s

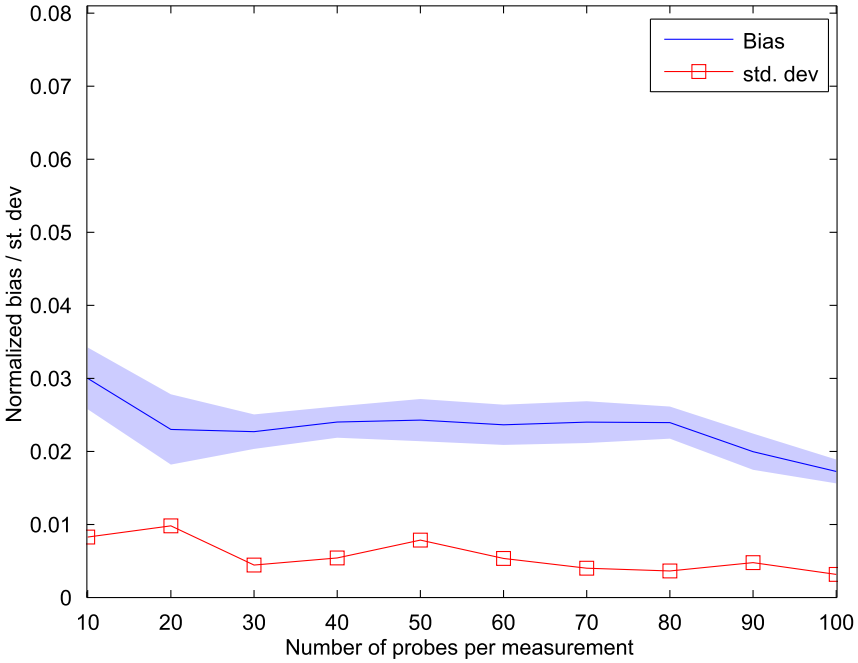


(a) L: IEEE802.11b $R_b = 1$ Mbps

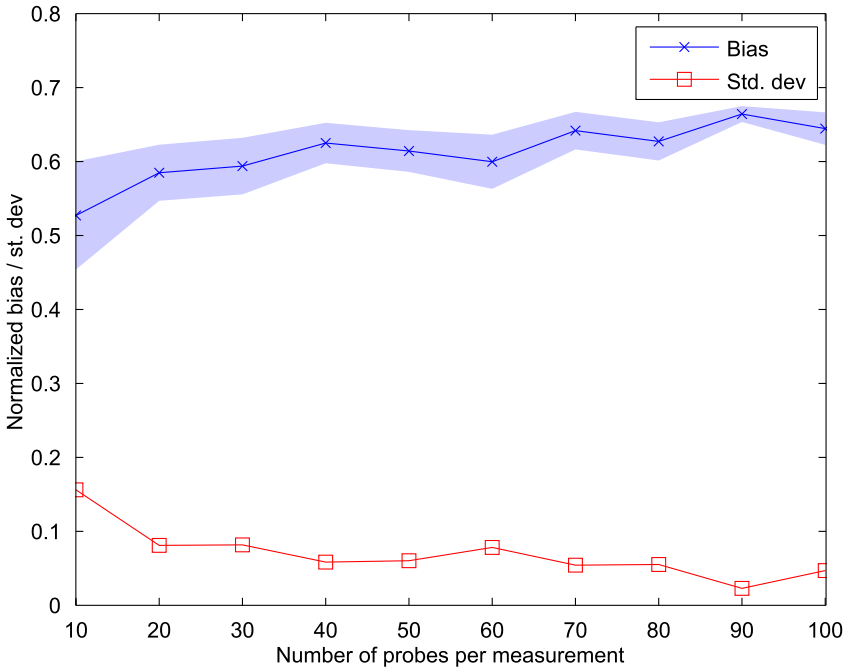


(b) L: IEEE802.3 10BASE-T $R_b = 10$ Mbps

Figure 3.9: Ad Hoc Probe normalized bias and variance versus number of probes per measurement for various test bed configurations.



(c) L: Commercial satellite modem $R_b = 2048$ kbps



(d) L: IEEE802.11b 4-hop wireless chain. Note the larger scale.

Figure 3.9: Ad Hoc Probe normalized bias and variance versus number of probes per measurement for various test bed configurations.

3.3 Discussion

3.3.1 Accuracy over CSMA-link

Single-hop In figure 3.3a, the MGEN brute-force estimates followed the TMC-curve that took into account the average backoff time value, while the Ad Hoc Probe estimates were in accordance with the TMC that did not take this into account. The reason for this was that Ad Hoc Probe selected the packet-pair with the minimum OWD sum. Hence, the packet-pair that had the lowest random backoff time sum was chosen as the "good" sample. Since the value zero is a legal value for the backoff timer [20], the Ad Hoc Probe estimates were very close to the TMC with the Contention Window set to zero.

An analytical expression giving a lower bound of the measurement bias, $\text{Bias}[\widehat{C}]$, normalized to the true path capacity, C, can be derived by observing that the true path capacity never exceeded the TMC, while the mean Ad Hoc Probe estimates complied to the TMC with the Contention Window set to zero.

$$\min\left(\frac{\text{Bias}[\widehat{C}]}{C}\right) \approx \frac{\text{TMC}_{\text{noCW}} - \text{TMC}}{\text{TMC}} = \frac{\text{TMC}_{\text{noCW}}}{\text{TMC}} - 1 \tag{3.1}$$

Figure 3.10 is a plot of equation (3.1) for various data rates using the IEEE802.11b TMC formula given in equation (A.1), including the extra overhead from IPsec. The observed normalized bias is also plotted. Judging from these findings, the lower bound of the bias is substantial for high data rates, resulting in overestimation errors in the order of 20 %. The measured bias corresponds well with the theoretical lower bound. Moreover, the theoretical lower bound is approximately equal to the real bias for small packet sizes, but is much lower than the measured bias for large packets.

Note that it is the size of the Contention Window that governs the relation $\text{TMC}/\text{TMC}_{\text{noCW}}$. Ad Hoc Probe may display an improved or worsened bias when used with other CSMA-based links.

The path capacity achieved by MGEN brute forcing diverged from the TMC at large packet sizes. Since the TMC assumes that no retransmissions occur, the difference was most likely caused by the fact that the layer 2 frame error rate will increase with packet size [32], given a constant bit error rate.

Multi-hop In a wireless multi-hop chain scenario, it is expected that the path capacity is inversely proportional with the number of wireless hops for chains with four or less hops [34]. The MGEN brute-force results in figure 3.4 comply to this expectation, while Ad Hoc Probe diverged from this behavior when there was more than two wireless hops. For these cases, Ad Hoc Probe reported the path capacity as the no-Contention-Window TMC for (number of hops-1).

A likely explanation for this can be found by analyzing the possible sequence of transmission events that can take place in a FIFO-based wireless CSMA chain. The binary trees in figure

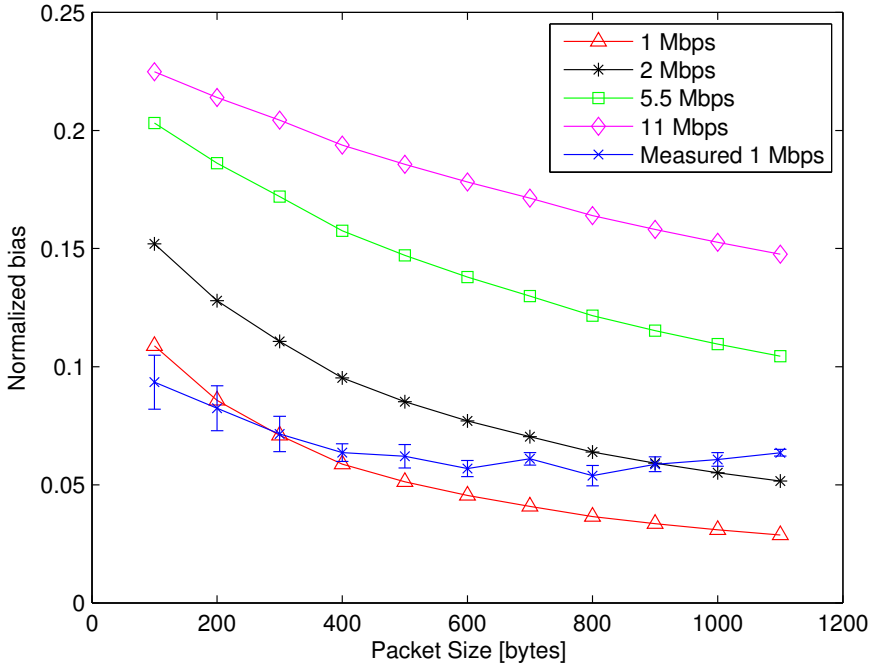


Figure 3.10: Theoretical lower bound and observed Ad Hoc Probe IEEE802.11b bias, normalized to the path capacity

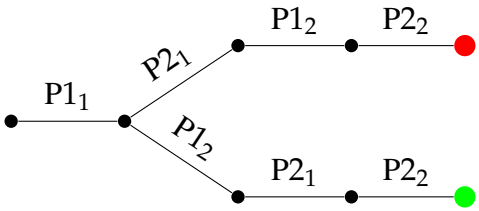
3.11 show that there are several cases that will lead to overestimation of the path capacity - each having the same minimum OWD sums as the correct case. Ad Hoc Probe will only provide the correct path capacity if the second packet is held back until the first packet has been relayed through all nodes that cannot transmit at the same time without causing destructive interference[†]. The four-hop binary tree is not included due to its large size, but it is safe to conclude that there are many more branches representing overestimation in that case. This issue is also briefly mentioned in [2].

Figure 3.11 does not provide the probabilities connected with each sequence of events. However, the results in figure 3.4 indicate that the error branch of the two-hop case is very unlikely. Considering the results from the three- and four-hop case, it is evident that the probability of the correct branch is small compared to the branches that respectively end with $\{P2_2, P2_3\}$ and $\{P2_2, P2_3, P2_4\}$.

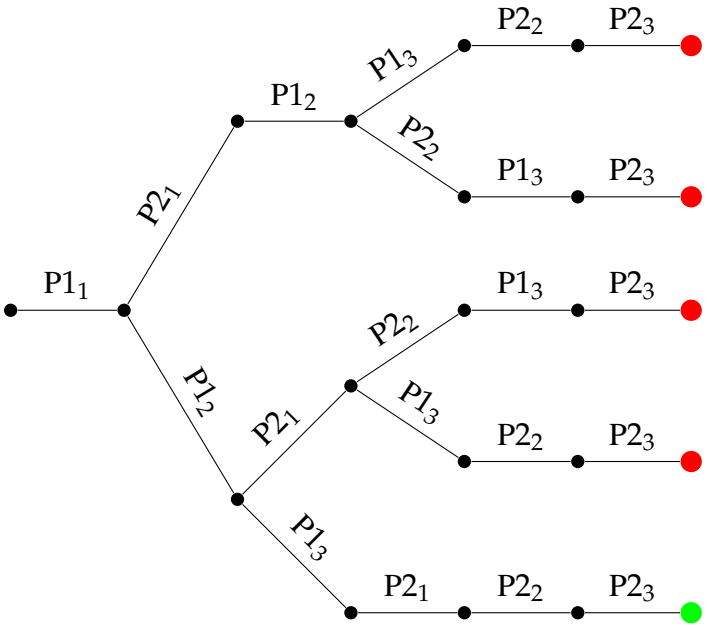
The occurrence of this overestimation problem can most likely be avoided if the algorithm is modified to always select the packet-pair with the least first-packet OWD *and* the least OWD sum. In Ad Hoc Probe, this is only a requirement if the clock skew correction algorithm is triggered, and not in the "regular" cases. In [7], this behavior is not reported. This could be due to that the authors consequently used 200 probes per measurement, most likely always triggering the clock skew correction algorithm.

If this behavior could be avoided, Ad Hoc Probe would converge towards the TMC without the Contention Window, and considering that this again converges towards the regular TMC as the

[†]Based on the transmission ranges illustrated in fig. 3.2 this is three hops, but due to the fact that the interference range is longer than the transmission range, it can be as many as seven hops [34].



(a) Two hops



(b) Three hops

Figure 3.11: Binary tree showing the possible transmission sequences for a packet-pair over a wireless CSMA multihop chain. The branches that are terminated with a red dot correspond to cases of overestimation. P1$_i$: First packet transmitted over i-th hop, P2$_i$: Second packet transmitted over i-th hop

number of hops increase, the performance in terms of bias would be improved. The reason for the convergence between the TMC without and with Contention Window is that the difference between these are divided by the number of hops as well since

$$\frac{A}{i} - \frac{B}{i} = \frac{1}{i}(A - B)$$

3.3.2 More Causes of Overestimation

Figures 3.9b and 3.9c clearly suggest that Ad Hoc Probe also is biased for links not employing a random backoff time. In addition, three other interesting observations can be made out of figure 3.3:

- In figure 3.3b, the Ad Hoc Probe expected value was considerably higher than the TMC and MGEN brute-force results.
- There is a trend in the expected value of figure 3.3b which indicates that large packet-pairs provides better accuracy than small packet-pairs

- Taking into account the analysis in section 3.3.1, then the results in figures 3.3a and 3.3b suggest that a large true path capacity leads to greater inaccuracy than a small true path capacity.

In the case of the wired IEEE802.3 10BASE-T scenario of figure 3.3b, the packet-pairs have not suffered any compression or expansion along the path since no random backoff time is employed when the medium is sensed idle [19]. Therefore, Ad Hoc Probe's minimum OWD sum algorithm should be unnecessary, since every packet-pair should arrive at the receiver with a dispersion in compliance with the Packet-Pair Dispersion model

$$\frac{\texttt{PSize}}{\Delta t} = \frac{\texttt{PSize}}{T_{rcv2} - T_{rcv1}}$$

The packet size, `PSize`, is a constant which always will be measured correctly by the receiver, but the measured packet arrival times, T_{rcv2} and T_{rcv1}, will in most cases come with errors

$$\widehat{C} = \frac{\texttt{PSize}}{(T_{rcv2} + e(T_{rcv2})) - (T_{rcv1} + e(T_{rcv1}))}$$

where $e(t)$ is a random process representing the time measurement error at time t, receiver time. For convenience, hereafter the notation e_1 and e_2 is respectively used for $e(T_{rcv1})$ and $e(T_{rcv2})$.

The measurement bias, Bias$\{\widehat{C}\}$, normalized to the true path capacity C can be expressed as follows

$$\frac{\text{Bias}\{\widehat{C}\}}{C} = \text{E}\left\{\frac{\widehat{\text{C}} - \text{C}}{\text{C}}\right\} = \text{E}\left\{\frac{\texttt{PSize}}{\text{C}(\text{T}_{\text{rcv2}} + \text{e}_2) - \text{C}(\text{T}_{\text{rcv1}} + \text{e}_1)} - 1\right\}$$

Since $T_{rcv2} = T_{rcv1} + \texttt{PSize/C}$,

$$\begin{aligned} &\text{E}\left\{\frac{\texttt{PSize}}{\text{C}(\text{T}_{\text{rcv1}} + \texttt{PSize}/\text{C} + \text{e}_2) - \text{C}(\text{T}_{\text{rcv1}} + \text{e}_1)} - 1\right\} \\ =&\frac{\texttt{PSize}}{\texttt{PSize} + C\,(\text{E}\{\text{e}_2\} - \text{E}\{\text{e}_1\})} - 1 \end{aligned} \tag{3.2}$$

By taking a closer look at equation (3.2), it is clear that it describes the effect of a local packet-pair compression or expansion at the receiver. The case $e_1 > e_2$ leads to overestimation while the opposite case leads to underestimation. If $e_1 = e_2$, the packet dispersion is unaffected and the estimate will be correct. A large packet size will reduce the impact of timing error, while a high true path capacity will make the accuracy very sensitive to timing errors.

In general, the time measurement errors e_1 and e_2 can be caused by different reasons; the most important being finite clock resolution and/or Operating System (OS) overhead due to scheduling/context-switches.

Clock resolution The Ad Hoc Probe implementation used for these measurements conducts time measurements by reading the Time Stamp Counter (TSC) CPU-register; a 64-bit counter that is incremented every clock cycle. Further, the counter value is divided by the CPU clock frequency to obtain the measurement in seconds. I.e., the clock resolution is dependent on the speed of the CPU. In this case, the CPU clock frequency was 1.7 GHz, providing a clock resolution of $1/1.7$ ns. Assuming that the oscillator circuits provide perfect frequency stability, $e(t)$ will be stationary in the mean, and the per packet time stamp error would be uniformly distributed in the interval $[0, 0.6]$ ns. This is a very accurate time reading and it should not lead to a biased path capacity estimate as shown in figure 3.3b since

$$\mathrm{E}\{\mathrm{e}_2\} - \mathrm{E}\{\mathrm{e}_1\} = 0.3 - 0.3 \text{ ns} = 0 \text{ ns}$$

OS Overhead As analyzed in [27], the OS overhead can have significant impact on packet-pair based capacity estimates. Operating systems such as Linux share CPU resources between user space applications in order to make multitasking possible. This is done by the use of a scheduling algorithm implemented in the OS kernel. The algorithm swaps processes and threads in and out of execution in accordance with a priority scheme, and the act of swapping a process or thread is referred to as a *context switch*. Since the Ad Hoc Probe receiver is a user space application, it will be swapped out of execution at time intervals determined by the OS scheduling. If a packet arrives during the time interval when the receiver application is swapped out of execution, the result will be an error in the measured arrival time since this measurement is performed first after the application is allowed to resume execution. The error introduced by this will be at least the time it takes to perform a context switch. In [33], measurements of the context switch overhead was performed on hardware with similar specifications as the test bed used for experiments in this thesis. The time needed for a context switch was found to vary in the range of a few to over a thousand μs, depending on the need to swap data in and out of the Level 1 and 2 cache and the data access pattern.

Figure 3.3b indicates that the path capacity tends to be consistently overestimated by Ad Hoc Probe. The same tendency was reported for a slow computer in [27]. This could be due to the fact that context switching, in most cases, cause error in the first packet arrival time measurement since the receiver application is very likely to be swapped out of execution during the relatively long packet-pair interdeparture time. In cases where the true path capacity is several Mbps, the second packet should in most cases not be affected by context switching due to the high likelihood of that the receiver application still "owns" the CPU when second packet arrives very shortly after the first. This reasoning leads to the conclusion that $\mathrm{E}\{\mathrm{e}_1\} > \mathrm{E}\{\mathrm{e}_2\}$ for high capacity paths, and consequently a measurement bias resulting in a consistent overestimation for these cases.

The operating system scheduler is in most cases a complex algorithm, resulting in that the context switch delay depends on many factors, e.g. user activity, the number of processes running, process priorities, the CPU clock frequency, available memory etc. In general, the time measurement error process is non-stationary in the mean. It is therefore very hard to generally determine the *bias* of Ad Hoc Probe due to context switching, since this also consequently will be time varying.

It is important to point out that the Ad Hoc Probe minimum OWD sum algorithm mitigates the problem of packet-pair compression and expansion caused by context switch delay since packets that have been subject to waiting by a context switch will be measured with an increased OWD. However, the results in figure 3.3b suggest that the context switching in many cases affected all of the 100 packet-pairs that made up a single measurement. This is not unrealistic considering that corresponding behavior was reported for 600 probes per measurement in [27].

Figure 3.12 shows the maximum observed Ad Hoc Probe estimates of the data sets presented in figure 3.3, normalized to the true path capacity. The theoretical measurement error was obtained by substituting equations (A.1) and (A.2)[†] into equation (3.2) using $e_2 - e_1 = -55\mu$s. This value seems to roughly correspond to the maximum context switch delay error made by Ad Hoc Probe for these two particular data sets. Comparing the two link technologies, it is clear that context switching has the greatest impact if the true path capacity is in the order of several Mbps.

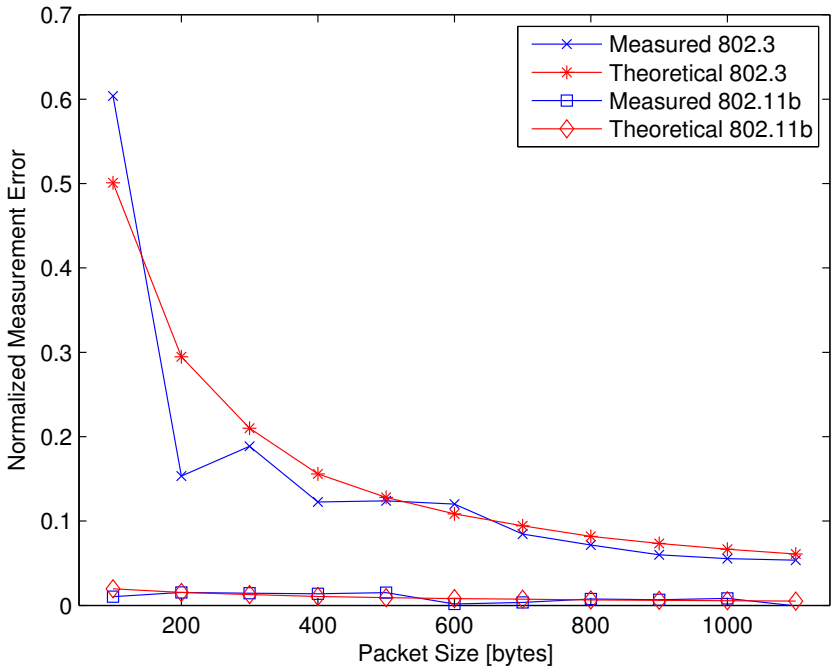


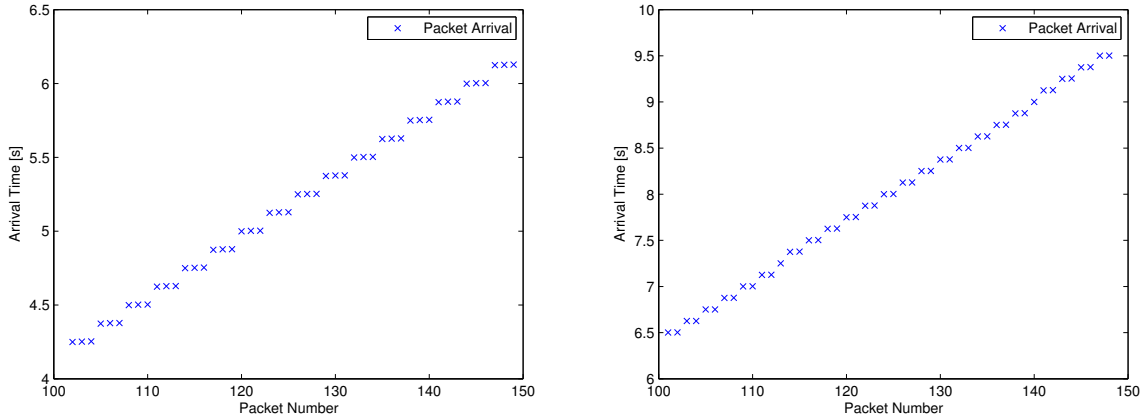
Figure 3.12: Maximum Ad Hoc Probe estimates compared to theoretical measurement error for IEEE802.3 10BASE-T and IEEE802.11b. $e_2 - e_1 = -55\ \mu$s

The observed overestimation error for high capacity non-CSMA paths have now been given a likely explanation. The theory could be evaluated by modifying the Ad Hoc Probe source code so that the kernel performed the time-stamping, using the SO_TIMESTAMP socket option [44]. The expected results from such an experiment would be a significant reduction in the measurement bias for links with dedicated medium access, since the context switch delay no longer would play a role for the time measurement error.

[†]Contention Window set to zero. This is also used as the true path capacity when calculating the observed normalized measurement error for IEEE802.11b in order to separate the effect of the bias caused by the algorithm and this implementation-caused bias.

3.3.3 Accuracy over a TDMA-link

As can be observed in figure 3.5, Ad Hoc Probe did not provide accurate estimates when the TDMA-based tactical UHF radio was limiting the path capacity. In figure 3.5a, the estimates were completely wrong for probe packet size 100 bytes, and biased for the other packet sizes. Furthermore, in the case of the 1200 kHz mode of operation shown in figure 3.5b, Ad Hoc Probe failed to provide any reasonable estimates.



(a) Packet size 100 bytes

(b) Packet size 200 bytes

Figure 3.13: A selection of packet arrival times from MGEN brute-force measurements when the path capacity was limited by the tactical UHF radio.

In order to understand the results in figure 3.5a, a comparison of the packet arrival times from the MGEN brute-force measurement is beneficial. Figure 3.13 shows that there was a special pattern in the receive times. The 100 bytes packets were consistently delivered three at a time, while the 200 bytes packets were mainly delivered two at a time with occasional single-packet deliveries. The time difference between packet arrivals within the triplets or doublets was 1-5 ms; much smaller than the time that passed between the arrival of each triplet or doublet.

The explanation of this behavior lies in the fact that the tactical radio used TDMA. If the sum of the packet size `PSize` and the overhead H from IPsec and lower layer headers, is less than the amount of data D that can be fitted into a single time slot of length T at layer 1 information bit rate R_b, then two or more consecutively sent packets will often be delivered to layer 3 at nearly the same time, since the radio seem to complete processing of all the received data in a single time slot before delivering data to higher layers.

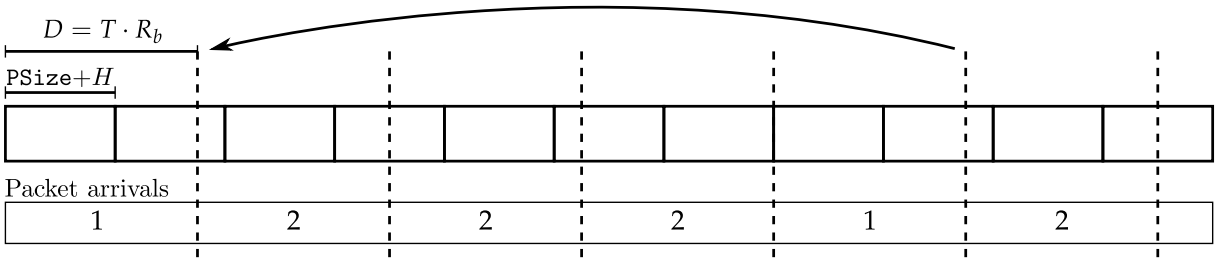


Figure 3.14: Illustration of the packet arrival pattern for $\frac{\texttt{PSize}+H}{D} = \frac{4}{7}$

Figure 3.14 shows an example of how two "last bit received" events can occur in a data block corresponding to a single time slot. The number of packets completely received during processing of a single time slot will follow a cyclic pattern whose period in bits is given by the least common multiple of $(\texttt{PSize} + H)$ and D. The period of the cyclic pattern can potentially be large, especially if one or both of these numbers are primes.

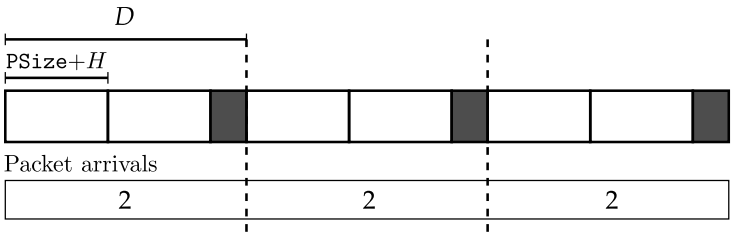


Figure 3.15: Illustration of the Ad Hoc Probe packet-pair arrival pattern if $2(\texttt{PSize} + H) < D$.

This insight into the packet delivery process can be used to explain the Ad Hoc Probe results. Figure 3.15 illustrates the arrival pattern in the case where a packet-pair fits into a single time slot. When the packet-pair is delivered at the same time to higher layers, there is no correlation between the resulting packet dispersion and the link capacity, but it most likely represents the radio's processing speed or the capacity of the USB connection between the radio and the computer. This explains the large Ad Hop Probe estimate for packet size 100 bytes in figure 3.5a, and shows that $\texttt{PSize} + H$ must be greater than half the size of D for Ad Hoc Probe to provide results that are correlated to the link capacity of a TDMA-based link. In the 1200 kHz mode of operation of figure 3.5b, the radio operated at a higher bit rate and used all time slots for data transfer, making D larger than $2(\texttt{PSize} + H)$ for all packet sizes, resulting in unusable estimates.

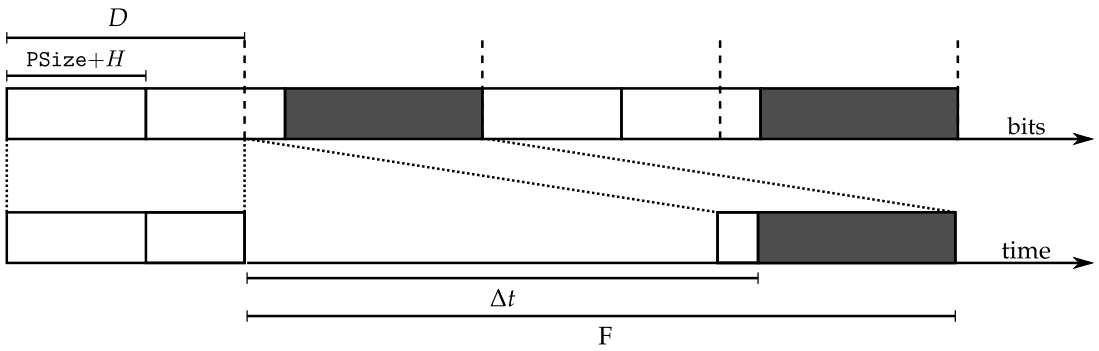


Figure 3.16: Illustration of the Ad Hoc Probe arrival pattern if $2(\texttt{PSize} + H) > d$

As shown in figure 3.16, if $2(\texttt{PSize} + H) > D$, the measured packet dispersion will be a multiple of the length of a time frame F minus the length of the unused part of the last time slot. Since Ad Hoc Probe sends packet-pairs with a relatively large time spacing in between, and taking into account that there was no cross traffic present, the last time slot used to send a packet-pair was non-full[†]. Consequently, Ad Hoc Probe can be expected to deliver estimates according to equation (3.3) when there is no other traffic than the packet-pairs going through the radio.

[†]In the case of the MGEN brute-force measurements, the transmission queue was always full at the radio, causing it to never "lack" data to fill an entire time slot

$$\widehat{C} = \frac{\texttt{PSize}}{\Delta t} = \frac{\texttt{PSize}}{\left(\left\lceil \frac{2(\texttt{PSize}+H)}{D} \right\rceil - \left\lceil \frac{(\texttt{PSize}+H)}{D} \right\rceil\right) F - \left(\frac{D - 2(\texttt{PSize}+H) - D\left(\left\lceil \frac{2(\texttt{PSize}+H)}{D} \right\rceil - 1\right)}{R_b} \right)}$$

$$= \frac{\texttt{PSize}}{\left\lceil \frac{2(\texttt{PSize}+H)}{D} \right\rceil (F-T) - \left\lceil \frac{(\texttt{PSize}+H)}{D} \right\rceil F + \frac{2(\texttt{PSize}+H)}{R_b}}, \quad (\texttt{PSize}+H) > \frac{D}{2} \tag{3.3}$$

where $\lceil \cdot \rceil$ is the ceiling operator.

By defining the MGEN brute-force estimates as the true path capacity, C, an expression for the measurement bias can be found. Let N be the number of packets that have been received by the MGEN application, and $e < F$ be the time from the last received frame to the end of the data counting period. Then,

$$C = \frac{N \cdot \texttt{PSize}}{\left\lceil \frac{(\texttt{PSize}+H)\cdot N}{D} \right\rceil F + e} \tag{3.4}$$

Since

$$\left\lceil \frac{(\texttt{PSize}+H)N}{D} \right\rceil \approx \frac{(\texttt{PSize}+H)N}{D}$$

for a large N, and considering that

$$e < F \ll \frac{(\texttt{PSize}+H)N}{D}$$

then e can be disregarded. As a result, equation (3.4) can be simplified

$$C \approx \frac{N \cdot \texttt{PSize}}{\frac{(\texttt{PSize}+H)\cdot N}{D} F} = \frac{N \cdot \texttt{PSize}}{\frac{(\texttt{PSize}+H)\cdot N}{D} F} = \frac{\texttt{PSize} \cdot D}{(\texttt{PSize}+H)F} \tag{3.5}$$

The bias, normalized to the Ad Hoc Probe estimate is given by

$$\frac{\text{Bias}\{\widehat{C}\}}{C} = \frac{\widehat{C} - C}{C} = \frac{\widehat{C}}{C} - 1 \tag{3.6}$$

In figure 3.17, equation (3.6) is plotted using equations (3.3) and (3.5) for respectively $\widehat{C}$ and C , taking into account the overhead caused by IPsec and the layer 2 Ethernet header. The measured normalized bias is also included in the plot and it corresponds well to the theoretical value. The small deviation is most likely caused by the fact that the layer 1 overhead (e.g., synchronization preamble) is not included.

In accordance with figure 3.5a, plaintext packet sizes 200-400 represent cases of underestimation, while plaintext packet sizes 500-700 results in overestimation. When the ciphertext packet size plus the layer 2 header add up to $D = 125/8 \cdot 2 \cdot 135 \approx 530$ bytes, somewhere between plaintext packet size 400 and 500 bytes, the bias is zero.

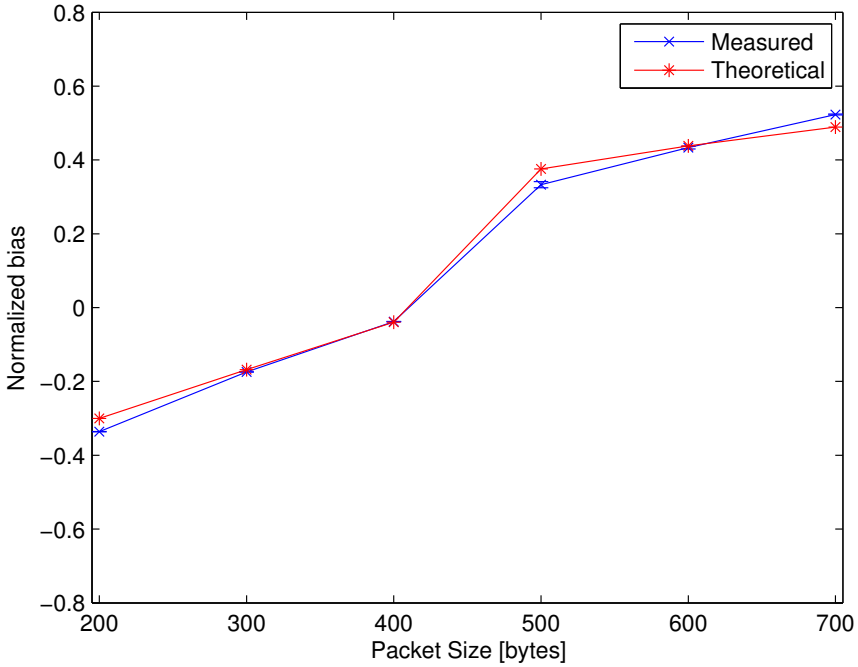


Figure 3.17: Normalized Ad Hoc Probe bias for the TDMA-based tactical radio. $F = 125$ ms, $T = 125/8 \cdot 2 = 31.25$ ms, $R_b = 135$ kbps

3.3.4 Accuracy over FDMA Link

As can be observed in figure 3.6, Ad Hoc Probe provides reasonably accurate estimations when used over a FDMA-based link. The reason for this is that there is no need for buffering or collision avoidance mechanisms such as in TDMA and CSMA links. The transmission of a packet-pair can therefore be initiated immediately after the first packet has arrived at the modem as long as the transmission queue is empty. The second-packet is sent directly after the first, causing the link to comply with the Packet-Pair Dispersion model.

Ad Hoc Probe was accurate within the overestimation error that can be expected due to context switching. The maximum overestimation errors in these measurements correspond to $e_2 - e_1 \approx -75$ μs, which is not too far from the maximum delay experienced in the IEEE802.3 10BASE-T case.

3.3.5 IP Fragmentation

When considering the results presented in figure 3.7, a key observation is that Ad Hoc Probe was in far better accordance with the TMC than the MGEN brute-force estimates during the packet fragmentation that took place for packet sizes 1200-1500 bytes. The reason for this is that the latter method overloaded the narrow link by sending traffic at a rate exceeding the narrow link capacity, causing congestion to occur at the outgoing 10BASE-T interface. The layer 2 transmission queue associated with this interface works in a droptail fashion; discarding packets that arrive when the queue is full. Since each packet was divided into two fragments before being placed in this queue, there were many cases where the first fragment was dropped,

but not the second and vica versa. Consequently, a great part of the path capacity was wasted on sending IP fragments that never could be reassembled and delivered to the MGEN receiver application. These unassembled fragments were therefore never accounted for, even though they were correctly transferred.

In today's wired military networks, an MTU of 1500 bytes is normal both in the red and black parts of the network. Since the IPsec devices introduce overhead that makes the packets larger than the black network's MTU, a considerable amount of fragmentation is occurring in the black network. Considering this, one should be careful trusting path capacity estimates based on UDP flooding in such settings since this method potentially results in serious underestimation.

In the case of Ad Hoc Probe, the packet fragmentation results in an increase in the packet-pair dispersion observed at the receiver due to the extra overhead. The resulting estimate therefore takes into account the reduction in capacity that occur due to the fragmentation.

3.3.6 Accuracy over a Rate Limited Path

Figure 3.8 shows that Ad Hoc Probe overestimated the path capacity. In fact, it provided the same estimations as in figure 3.3a where no rate limiter was in use. The results indicate that the packet-pair probes were sent straight through the Vyatta rate limiter even though the token bucket allowed burst size was set to zero. It seems that the Vyatta rate limiter does not operate on a per packet basis. Given the fact that most rate limiters are configured with a non-zero allowed burst size, it is very likely that packet-pairs, in general, will not be influenced by such mechanisms; causing Ad Hoc Probe to consistently overestimate the capacity of rate limited paths. This is also reported to be the case for another packet dispersion based tool in [31].

3.3.7 Number of Probes Per Measurement

The results presented in figure 3.9 indicate that the bias of Ad Hoc probe under these ideal conditions is nearly independent on the number of probes per measurement. A small improvement can be read from 10 to 30 number of probes per measurement in figures 3.9b and 3.9c, and from 80 to 100 in figure 3.9c. These improvements are most likely caused by a higher probability for the occurrence of a low context switch error when increasing the number of probes per measurement.

There is a steady decrease in the IEEE802.11b standard deviation curve up to 60 probes per measurement, where there is a sudden increase. This could be caused by the clock skew correction algorithm, which is likely to be activated in the vicinity of the standard deviation increase. Since the variance of the minimum CSMA backoff time is expected to decrease as the number of probes increase, it should be very small for 70 or more probes per measurement. However, the variance of the average dispersion of the packet-pairs picked out by the clock skew correction algorithm[†] is not necessarily as small.

In the multi-hop case of figure 3.9d, the bias is showing an increasing trend with an increasing number of probes per measurement. By contrast, the variance is showing a decreasing trend.

[†]The clock skew correction algorithm is explained in section 2.5.2

Judging from figure 3.4, the Ad Hoc Probe estimate is expected to converge on the zero contention window TMC divided by 3 in this case. This corresponds to a normalized bias of 0.71, which is higher than the observed bias.

Considering that each packet draws four random backoff times, the probability that the minimum sum of these samples is approximately zero is much less than in the single-hop case where there is only one random backoff time sample per packet. However, as the number of probes per measurement is increased, there is an increased probability for a lower delay sum and a smaller chance for high delay sums. Consequently, the variance is reduced when increasing the number of probes, since most minimum OWD sums will be small in this case.

Although the Ad Hoc Probe average measurement error (bias) is under the upper bound of 0.71 for 10-100 probes per measurements, it is still expected that a few cases of a very low delay sum will occur even though this is unlikely. The *maximum* observed estimates in the 4 hop chain experiment are plotted in figure 3.18, and they do indeed approach the upper bound of 0.71.

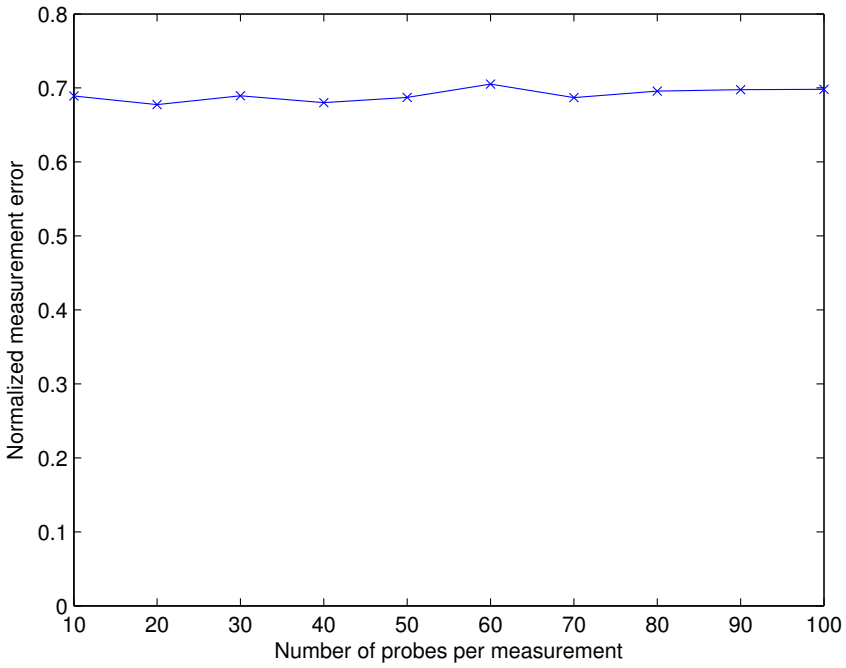


Figure 3.18: Maximum measurement error for 4 hop IEEE802.11b chain

3.4 Evaluation

In this section the validity of the hypothesis stated in the beginning of the chapter will be evaluated in the light of the knowledge obtained by experiments and analysis. The hypothesis was as follows:

> Ad Hoc Probe [7] can provide path capacity estimates that comply to the requirements in section 2.3 when there is *no other traffic present in the network*.

First, the qualitiative requirements from section 2.3 are treated, followed by the quantitative requirements. Lastly, a brief summary of the evaluation is given.

3.4.1 Compliance to Qualitative Requirements

The qualitative requirements developed in section 2.3.1, were that the path capacity estimator:

- *must not* rely on communication with the forwarding nodes in the black network
- *must* be compatible with CSMA and/or TDMA and/or FDMA
- *should* be compatible with wireless multi-hop communication where the same channel is shared between multiple nodes using CSMA/CA
- *must not* rely on the assumption of symmetric path capacities
- *must* work even if the estimation traffic is subject to packet fragmentation
- *should* be compatible with rate-limited paths

Ad Hoc Probe is based on end-to-end probing, not relying on communication with the forwarding nodes.

During the course of the experiments presented in this chapter, it became clear that Ad Hoc Probe was incompatible with TDMA links. If both packets in a pair fitted into a single time slot, there was no correlation between the packet dispersion and the path capacity. If the packet-pair could not fit into a single time slot, there was correlation, but the estimates were biased to severely over- or underestimate the path capacity for packet sizes that did not correspond to the size of a time slot. In general, the amount of data that fits into a time slot will be link-dependent, making the Ad Hoc Probe estimate completely unreliable for TDMA-links. By contrast, the algorithm provided reasonably accurate estimates for FDMA and CSMA links.

Ad Hoc Probe was able to capture the path capacity reduction in wireless multi-hop networks sharing a single channel.

The algorithm is based on one-way measurements, thus not relying on a symmetric path capacity.

In the case of packet fragmentation, Ad Hoc Probe outperformed the MGEN brute-force method, and provided estimates that took into account the reduced capacity due to the extra overhead.

Ad Hoc Probe is not compatible with paths where the capacity is limited by a rate-limiter due to the fact that these in most cases allow the forwarding of small bursts at a higher rate than the limit.

3.4.2 Compliance to Quantitative Requirements

The quantitative requirements developed in section 2.3.2, were as follows:

- It *must* take less than 6 seconds to complete a measurement for packet size 100 bytes, given that there is no packet loss during the measurement.
- Assuming that the path capacity estimator follows a Gaussian distribution, there *must* be a 95 % chance that the estimation error is less than ± 20 % of the true path capacity. I.e., $-0.2 \leq \frac{\text{Bias}\{\widehat{C}\}}{C} \leq 0.2 \quad \wedge \quad \frac{\sigma_{\widehat{C}}}{C} < 0.102 - 0.5102 \cdot \left| \frac{\text{Bias}\{\widehat{C}\}}{C} \right|$
- The estimation traffic layer 3 bit rate *must* not exceed 10 % of the path capacity.

Measurement time and intrusiveness The IP layer bit rate needed by Ad Hoc Probe is tightly coupled to the measurement time. A fast measurement requires the sending of many packet-pairs in a short period of time, thus increasing the chance of disturbing other flows. The results presented in figure 3.9 show that— in this ideal scenario with no cross traffic —there is very little to gain by increasing the number of probes probes from the minimum value of 10.

Since

$$\text{Measurement time} \approx \frac{2 \cdot \texttt{PSize}}{\text{Probing rate}} \cdot \texttt{Numprobes} = \frac{2 \cdot 100 \cdot 8}{\text{Probing rate}} \cdot 10 < 6 \text{ s}$$

then

$$\min\{\text{Probing rate}\} = \frac{2 \cdot 100 \cdot 8}{6} \cdot 10 = 2.67 \text{ kbps}$$

Additionally, since

$$\text{Probing rate} < 0.1C$$

then

$$\min\{C\} = \min\left\{\frac{\text{Probing rate}}{0.1}\right\} = \frac{2.67}{0.1} = 26.67 \text{ kbps} \tag{3.7}$$

The minimum requirement for path capacity given by equation (3.7) corresponds to requiring a information bit rate in the order of[†]

$$R_b \approx \frac{26.67}{\text{IPsec overhead ratio}} = \frac{26.67}{0.60} = 44.4 \text{ kbps}$$

for the case where the path capacity is equivalent to the path's minimum link capacity. If the path capacity is limited by a multi-hop wireless network, then the required information bit rate of each wireless link in the chain needs to be approximately 90 kbps for two-hop, and 135 kbps for three-hop. The one- and two-hop cases are within the range of what can be expected of a medium-range tactical narrowband link dedicated to data traffic [24, 6], while three or more hops requires wideband links.

[†]Using the entry for plaintext packet size 100 bytes in table B.1 and excluding layer 2 overhead

Accuracy By extending the use of the assumption that the accuracy is approximately the same for 100 and 10 probes per measurement, the results in figures 3.3, 3.4 and 3.6 are valid for evaluating whether the bias and variance for the CSMA and FDMA links meet the requirements. Figure 3.19 shows the maximum bias inferred from the calculated confidence intervals. Furthermore, the observed standard deviation of the estimator and the accuracy requirements are included. All curves are normalized to the true path capacity given by the results from the MGEN brute-force method.

Ad Hoc Probe provided sufficient accuracy for all plaintext packet sizes when the path capacity was limited by a single-hop *1 Mbps* IEEE802.11b link. However, considering the calculations in figure 3.10, the bias is expected to be increased for higher information bit rates. This leads to the conclusion that the maximum bit rate for CSMA links using the IEEE802.11b Contention Window size, should not be greater than 2 Mbps for Ad Hoc Probe to meet the accuracy requirements.

The accuracy was insufficient for flows of 100 byte packets when the path was limited by an IEEE802.3 10BASE-T link. Taking into account equation (3.2), this inaccuracy is expected to be worse for higher path capacities. For the FDMA based satellite link, the accuracy met the requirements, however the inaccuracy is expected to increase with higher capacities in this case as well.

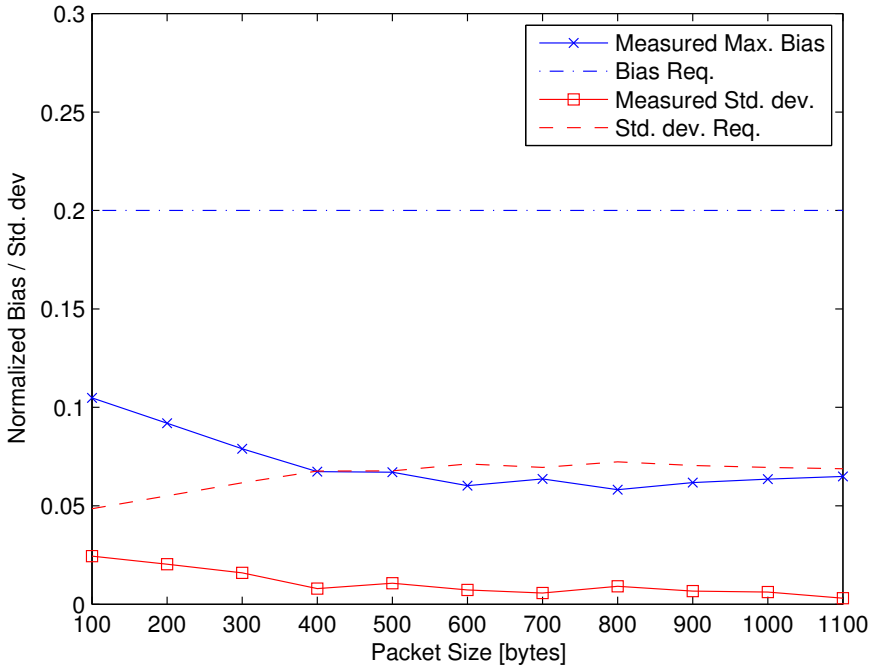
Finally, the accuracy of Ad Hoc probe was not acceptable when the number wireless hops exceeded two in a multi-hop IEEE802.11b scenario.

3.4.3 Conclusive Remarks

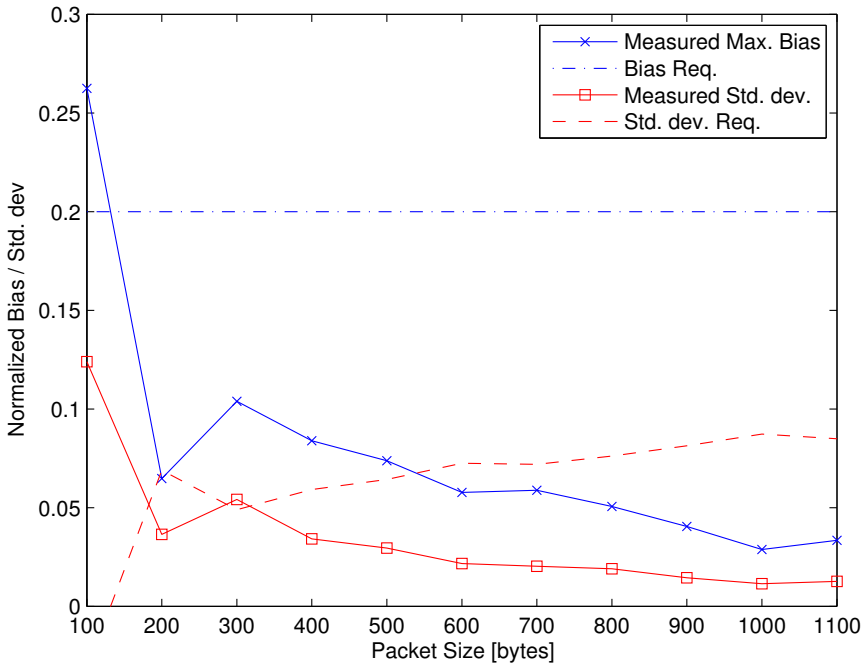
Based on the experimental results and analysis provided in this chapter, it can be concluded that Ad Hoc Probe meets the performance requirements in section 2.3 if:

a) the network consists of only CSMA- and/or FDMA-based links
b) the path capacity is not limited by a wireless chain of more than two hops
c) the information bit rate of the slowest link is at least 44.4 kbps in cases where the path capacity is limited by the path's minimum link capacity, and minimum 90 kbps per link if the path capacity is limited by a two-hop wireless chain
d) the path capacity is not limited by a faster IEEE802.3-link than 10BASE-T, or a CSMA-link operating at a information bit rate higher than 2 Mbps, using the IEEE802.11b minimum Contention Window size

However these results are only valid in the case of no other traffic in the network. Clearly, this is not the conditions in which an admission controller will operate. Therefore, the performance of Ad Hoc Probe under more realistic conditions is treated in the next chapter.

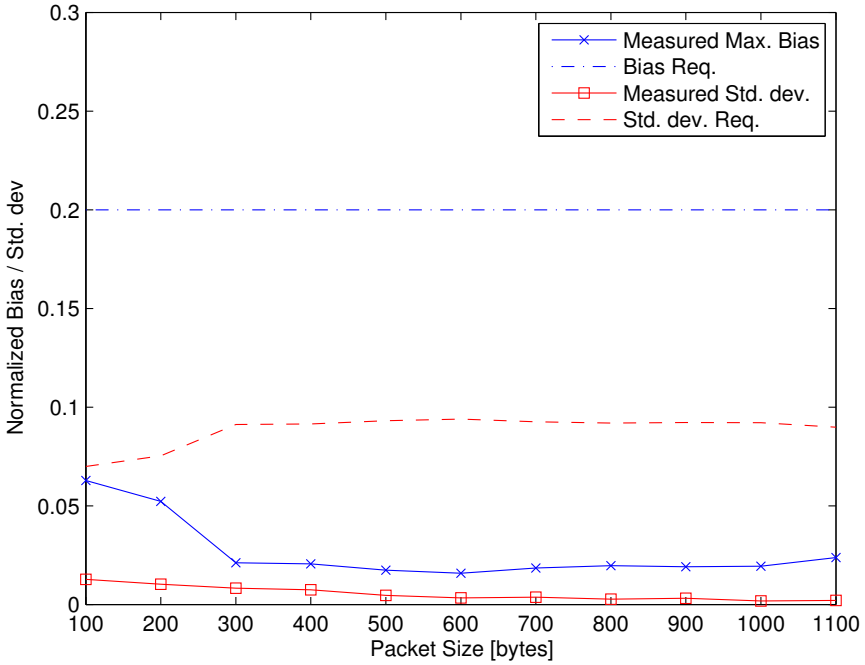


(a) L: IEEE802.11b $R_b = 1$ Mbps

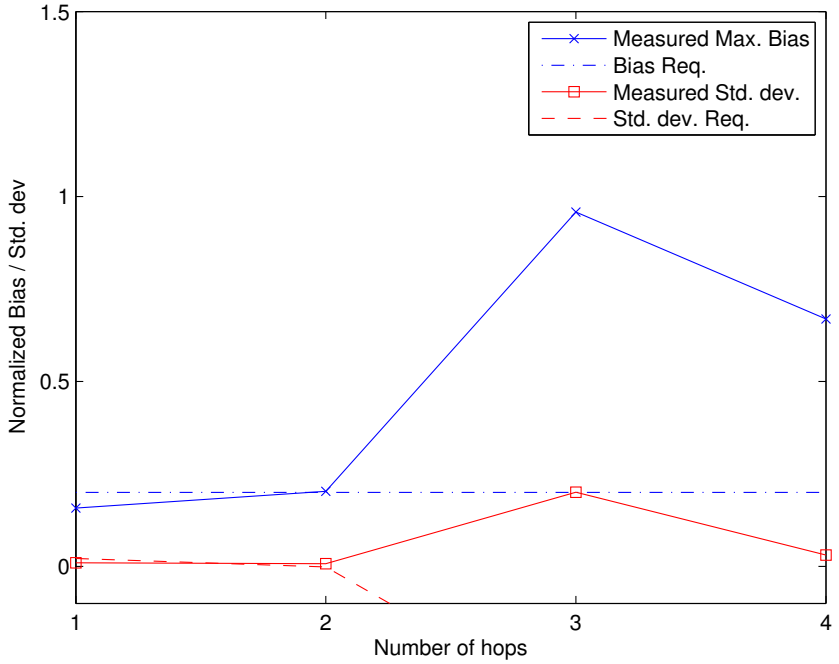


(b) L: IEEE802.3 10BASE-T $R_b = 10$ Mbps

Figure 3.19: Achieved and required performance for Ad Hoc Probe over various link technologies



(c) L: Commercial satellite modem $R_b = 2048$ kbps



(d) L: Multihop IEEE802.11b $R_b = 1$ Mbps. Packet size 700 bytes. Note the scale.

Figure 3.19: Achieved and required performance for Ad Hoc Probe over various link technologies

Chapter 4

Estimating Path Capacity under Non-ideal Conditions

When there was no other traffic present in the network, Ad Hoc Probe was found to achieve an acceptable performance under certain constraints. In this chapter, the focus is on how much these constraints change when other traffic sources are introduced. The performance is expected to worsen, but the challenge is to find out if the algorithm still can be used for Measurement-based Admission Control (MBAC), and, if so, under what constraints?

Stated otherwise, the goal of the experiments presented in this chapter was to evaluate the validity of the following hypothesis:

> Ad Hoc Probe [7] can provide path capacity estimates that comply to the requirements in section 2.3, even when the network is under load from other traffic sources.

The chapter follows the same structure as chapter 3, starting with a presentation of the experimental set-up before continuing with the obtained results, the discussion of these and, lastly, the chapter is concluded with an evaluation of the hypothesis based on the findings.

4.1 Experimental Set-Up

4.1.1 Test Bed Network Topologies

The experiments were conducted at Forsvarets Forskningsinstitutt (Norwegian Defence Research Establishment) (FFI) and were based on the same test beds that were presented in chapter 3, with the added complexity of a cross traffic source and sink as shown in figures 4.1 and 4.2.

In the experiments using the topology in figure 4.1, the links `L1` and `L2` were varied in order to create different scenarios.

Considering that two-hop configuration was the only one providing satisfactory results without cross traffic, the topology illustrated in figure 4.2 was the only multi-hop topology evaluated in

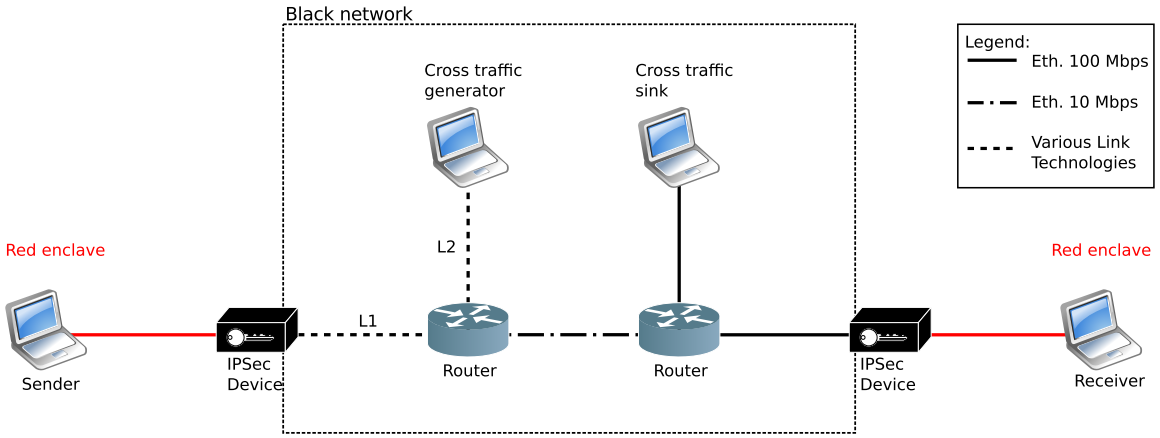


Figure 4.1: Test bed topology for measuring the performance of Ad Hoc Probe with cross traffic (CT) present in the path

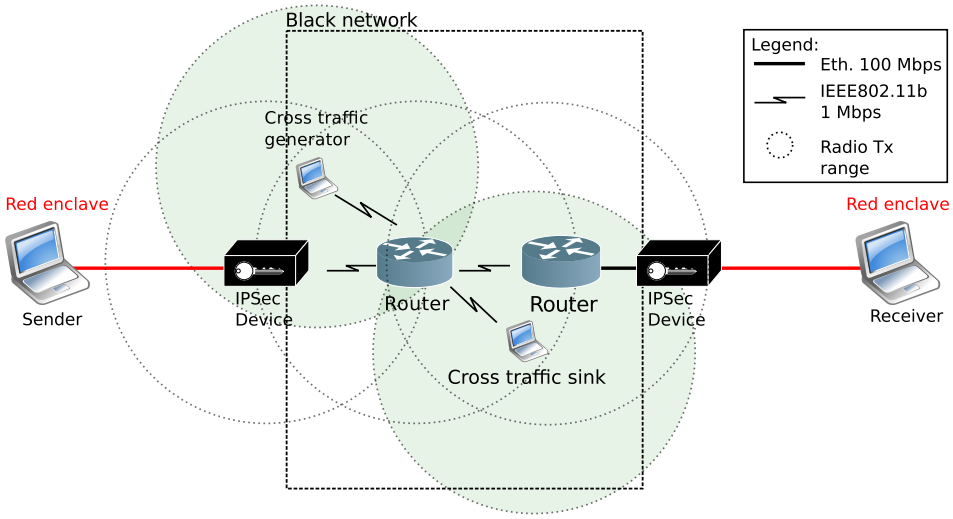


Figure 4.2: Test bed topology for measuring the performance of Ad Hoc Probe over multi-hop topology with cross traffic (CT) within the contention domain

these experiments. The transmission ranges that are indicated in the figure cannot be guaranteed due to the unpredictable channel conditions in an office environment. Nevertheless, since all the wireless transmitters were within two-hops, the Ready-to-send (RTS)/Clear-to-send (CTS) mechanism most likely prevented the occurrence of on-air collisions.

4.1.2 Link Technologies

The following link technologies were included in the experiments

- IEEE802.3 [19]
- IEEE802.11b [20]
- Commercial satellite modem [10]

The link technologies where used in the same configuration as in chapter 3. Only Carrier Sense Multiple Access (CSMA) and Frequency Division Multiple Access (FDMA) links were included since Ad Hoc Probe was proven to be incompatible with Time Division Multiple Access (TDMA) links in chapter 3.

4.1.3 Sofware Configuration

The software configuration was the same as in the experiments presented in chapter 3, with the exception that no rate limiters where configured.

4.1.4 Ad Hoc Probe Parameters

Of the four Ad Hoc Probe parameters, only the number of probes per measurement, `Numprobes` was varied in these experiments. This reduction was necessary since the introduction of a varying cross traffic increased the degree of freedom in the experiments, and thus one of the other parameters had to be set constant in order to adapt the number of required measurements to the thesis time frame. This raised the question of which parameter to set constant, and to what value?

The cross traffic was expected to cause compression or expansion of the packet-pairs and thus influence the estimator variance, but not the bias, which was expected to be the same as observed in the ideal case of no cross traffic in chapter 3. Most likely, the variance would increase with the packet size since small packet-pairs spend less time in the path than large packet-pairs, and thus will have a lower probability for coinciding with other traffic [16]. However, taking into consideration the effect of context-switch delays on the small packet-pairs, the variance would not necessarily have its minimum at the smallest packet size. Nevertheless, packet sizes much greater than 100 bytes were expected to result in a "worst-case" variance which could be used to find the minimum number of probes per measurement that was necessary to meet the accuracy requirements.

Having found that the packet size could be set constant, its value was yet to be decided. Considering that large packets have a greater Packet Error Rate (PER) in wireless networks [32], and consequently, that content codecs should operate sending slices in separate, small packets [5], it is a fair assumption that most non-best-effort flows use a packet size of less than 700 bytes, and that it is reasonable to use this packet size to provide a worst-case performance.

4.1.5 Cross Traffic Generation

Cross Traffic was generated by Multi-Generator (MGEN) [18] using the script enclosed in appendix B.6. The cross traffic was 1200 byte packets with Poisson-distributed interdeparture times. The mean interdeparture time was set according to the desired layer 3 bit rate, `CrossRate`. In most of the experiments, this rate was varied in steps of 20 % of the minimum information bit rate on the path between the cross traffic generator and sink.

Ideally, the effect of the cross traffic packet size should have been evaluated by varying this parameter in the experiments. However, it was yet again neccessery to reduce the degrees of freedom to meet the thesis timeframe. Therefore, a worst-case behavior approach was chosen in this case as well. A large cross traffic packet size was expected to result in a large estimator variance, since such packets would have the greatest effect in terms of compression or expansion of the packet-pairs [16].

4.1.6 Measurement Procedure

The measurements conducted during cross traffic followed the procedure given in algorithm 4.1.

Algorithm 4.1 Procedure for measurements with cross traffic

for `CrossRate`$\leftarrow$ 0 to $0.8R_{b_{min}}$ in steps of $0.2R_b$ **do**
  Start Ad Hoc Probe packet-pair generator(`PSize`$\leftarrow$ 700 bytes:, `Int`$\gg 2$`PSize`$/R_b$)
  Start MGEN cross traffic(packet size$\leftarrow$1200 bytes, `CrossRate`)
  for `Numprobes`$\leftarrow$ 10 to 100 in steps of 10 **do**
    for $i \leftarrow$ 1 to 20 in steps of 1 **do**
      Start Ad Hoc Probe packet-pair receiver(`Numprobes` , clock skew detection thresh$\leftarrow$30)
    end for
  end for
  Stop MGEN cross traffic generator
  Stop Ad Hoc Probe packet-pair generator
end for

4.1.7 Measuring the True Path Capacity

The results from the MGEN-based brute-force method provided in chapter 3 were reused.

4.1.8 Summary

Tables 4.1 and 4.2 summarize the most important parameters that were introduced in the previous subsections. In addition, all the different values that were assigned to these parameters during the experiments, are listed.

Table 4.1: Overview over parameters that were varied during the experiments

Parameter	Description	Values used in measurements
`L1`	Test bed link technologies	IEEE802.11b $R_b = 1$ Mbps IEEE802.3 10BASE-T $R_b = 10$ Mbps Commercial satellite modem $R_b = 2048$ kbps
`L2`	Test bed cross traffic link technologies	IEEE802.11b $R_b = 1$ Mbps IEEE802.3 10BASE-T $R_b = 10$ Mbps
`CrossRate`	Average layer 3 bit rate of cross traffic	0-80 % of minimum R_b of path from cross traffic generator to sink in steps of 20 %
`Numprobes`	Number of packet-pairs per Ad Hoc Probe estimate	10-100
`Int`	Ad Hoc Probe packet-pair interdeparture time	0.1, 0.2 s

Table 4.2: Overview over the most relevant parameters that were kept constant during the experiments

Description	Value used in measurements
Plaintext probe IP packet size	700 bytes (1400 bytes per packet-pair)
Largest allowed size of layer 2 payload in path (MTU)	1200 bytes
Distribution of interdeparture times between cross traffic packets	Poisson
IP packet size of cross traffic	1200 bytes
Number of Ad Hoc Probe estimates per trial	20
Ad Hoc Probe packet-pair interdeparture time	0.1 s
Ad Hoc Probe clock skew correction threshold	30

4.2 Results

In the following subsections, the observed Ad Hoc Probe bias and standard deviation is presented normalized to the true path capacity given by the MGEN brute force estimate. The plots also show the accuracy requirements that were derived in section 2.3:

$$-0.2 \leq \frac{\text{Bias}\{\widehat{C}\}}{C} \leq 0.2 \qquad \bigwedge \qquad \frac{\sigma_{\widehat{C}}}{C} < 0.102 - 0.5102 \cdot \left| \frac{\text{Bias}\{\widehat{C}\}}{C} \right| \tag{4.1}$$

$\text{E}\{\widehat{C}\}$ was unknown, and had to be estimated. Therefore, 95 % confidence intervals are presented, based on the sample mean and standard error of 20 observations of $\widehat{C}$ and the Student-t distribution with 19 degrees of freedom. In order to consider the worst-case performance, the presented results are the maximum values of these confidence intervals.

Since the only one sample was taken of the standard deviation, and the distribution of $\widehat{C}$ is unknown , a confidence interval is not provided for the standard deviation estimate. However, the estimate is calculated by taking the square root of an unbiased estimator of the variance of $\widehat{C}$, so it is expected to be close to the true value.

4.2.1 IEEE802.11b Contention

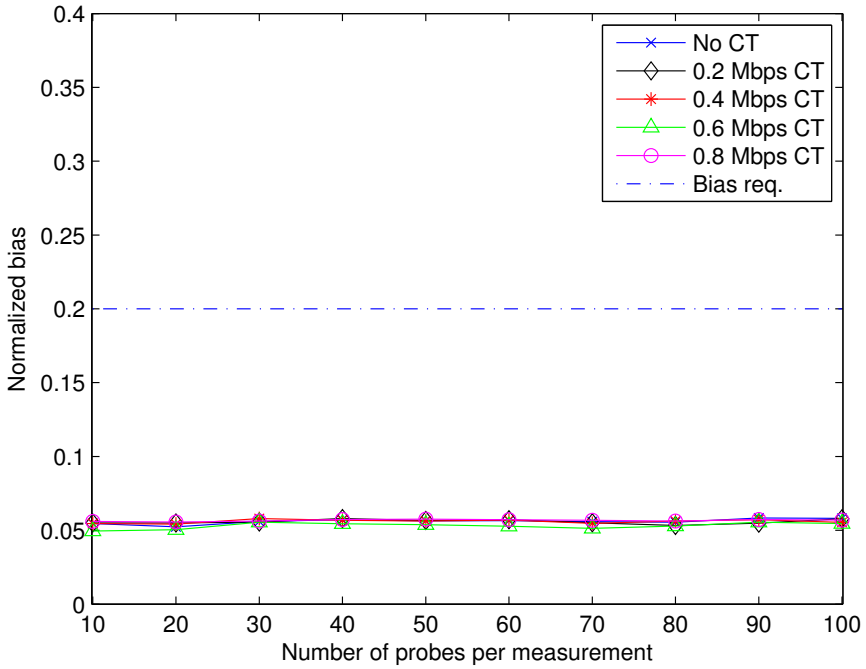
Single-hop Figure 4.3 shows the Ad Hoc Probe performance where `L1` and `L2` in figure 4.1 were IEEE802.11b-links. The cross traffic was sent on the same wireless channel as the packet-pairs, and the cross traffic generator and the Ad Hoc Probe packet-pair generator were within transmission range of each other. Table 4.3 lists the test bed configuration. The MGEN brute force path capacity estimate for packet size 700 bytes was 699 kbps.

Table 4.3: Configuration for results in figure 4.3

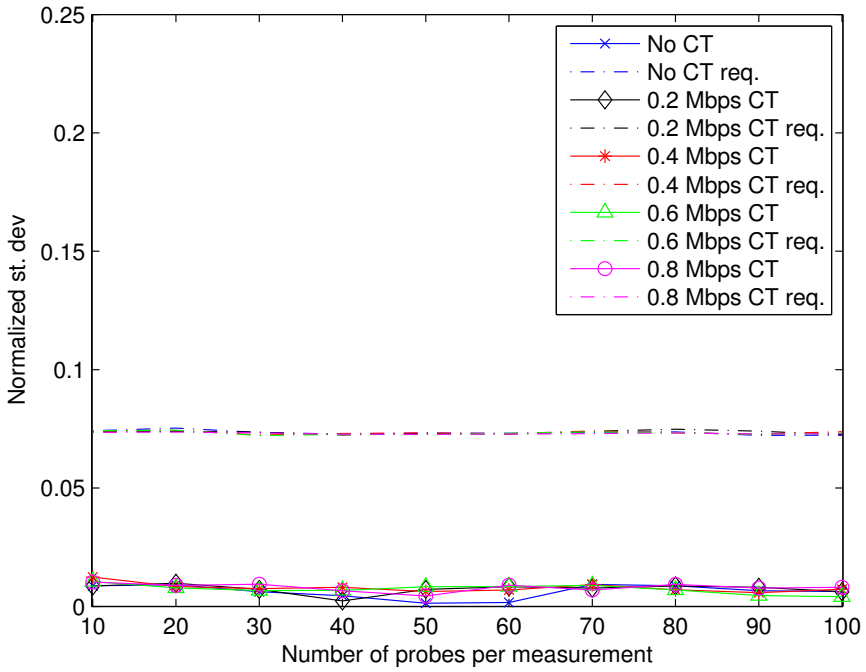
Parameter	**Values used in measurements**
`L1`	IEEE802.11b $R_b = 1$ Mbps
`L2`	IEEE802.11b $R_b = 1$ Mbps
`CrossRate`	0-80 % of 1 Mbps in steps of 20 %
`Numprobes`	10-100

Ad Hoc probe was well within the performance requirements for all cross traffic intensities and number of probes per measurement. In fact, Ad Hoc Probe showed the same accuracy as when there was no other traffic in the network[†].

[†]See the results for packet size 700 bytes in section 3.4.1



(a) Poisson cross traffic - bias



(b) Poisson cross traffic - standard deviation

Figure 4.3: Achieved and required performance for Ad Hoc Probe over IEEE802.11b with cross traffic (CT) on the same channel

Two-hop Figure 4.4 shows Ad Hoc Probe's performance for the two-hop scenario illustrated in figure 4.2. The cross traffic was sent on the same channel as the Ad Hoc Probe traffic. Considering the that the Theoretical Maximum Capacity (TMC) of the path was expected to be half of the single-hop case, the cross traffic intensity was varied in the interval 0.1-0.4 Mbps in stead of steps of 20 % of the minimum link capacity. The MGEN brute force path capacity estimate for packet size 700 bytes was 305 kbps.

Table 4.4: Configuration for results in figure 4.4

Parameter	**Values used in measurements**
`L`	IEEE802.11b $R_b = 1$ Mbps (for all links)
`CrossRate`	0.1-0.4 Mbps in steps of 0.1 Mbps
`Numprobes`	10-100

With 0.1 Mbps Poisson cross traffic, the accuracy requirements were met with very little margin for 30 or more probes per measurement. In the case of 0.2 Mbps cross traffic, Ad Hoc Probe needed 90 probes per measurements to perform within the requirements.

The bias at 0.3 Mbps cross traffic was close the requirement, while in the 0.4 Mbps case, the estimator was biased to overestimate the path capacity. However, the standard deviation always exceeded the requirements[‡] for both these cases.

4.2.2 Pure IEEE802.3

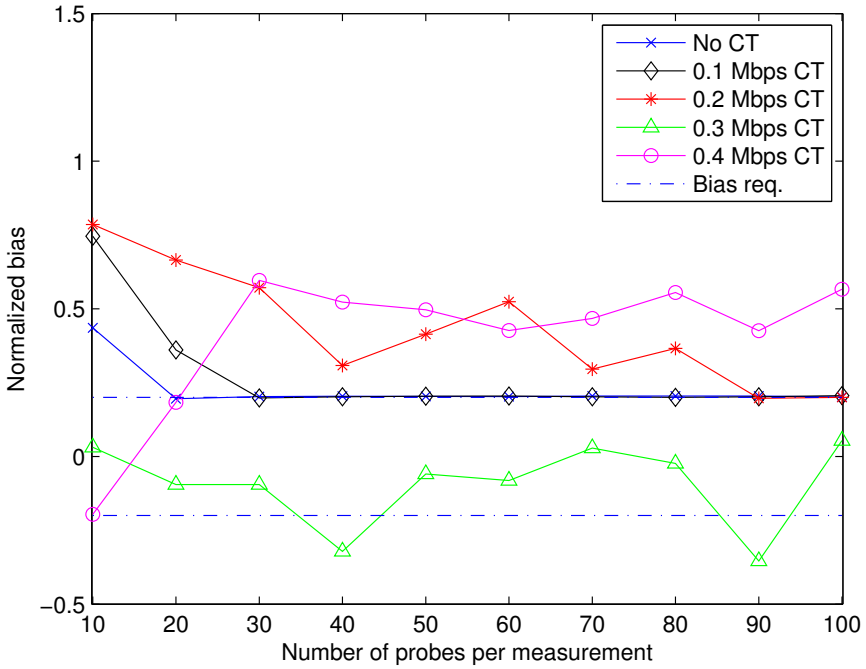
In this scenario, the path capacity was limited by IEEE802.3 10 Mbps links. The results are shown in figure 4.3 and the specific test bed configuration are given in table 4.5. The MGEN brute force path capacity estimate for packet size 700 bytes was 8.76 Mbps in this case.

Table 4.5: Configuration for results in figure 4.5

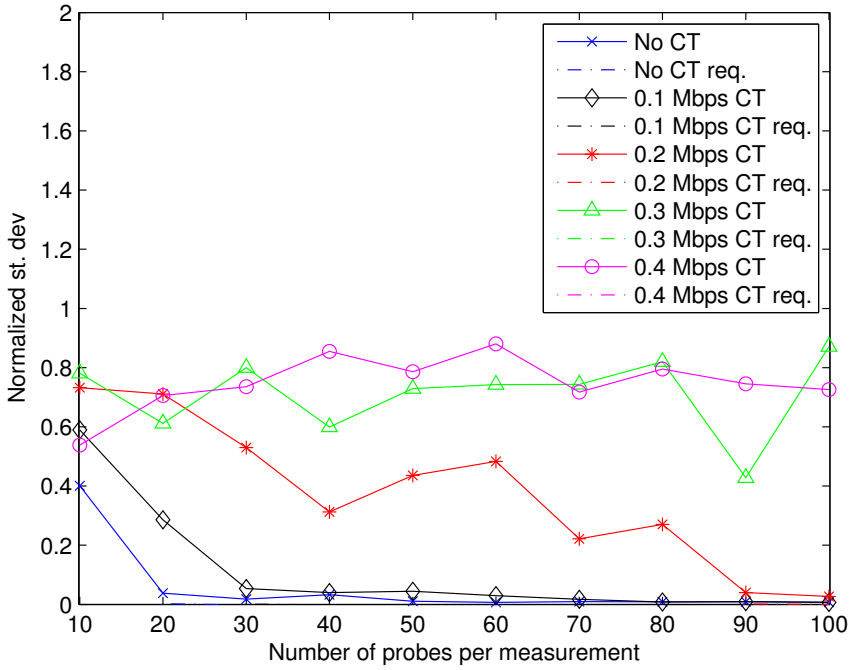
Parameter	**Values used in measurements**
`L1`	IEEE802.3 10BASE-T $R_b = 10$ Mbps
`L2`	IEEE802.3 10BASE-T $R_b = 10$ Mbps
`CrossRate`	0-80 % of 10 Mbps in steps of 20 %
`Numprobes`	10-100

Ad Hoc Probe complied to the accuracy requirements except for the case of 10 probes per measurement with 8 Mbps cross traffic, where the standard deviation exceeded the required

[‡]According to equation 4.1, the std. dev. requirement is ~0 if the bias requirement is barely satisfied

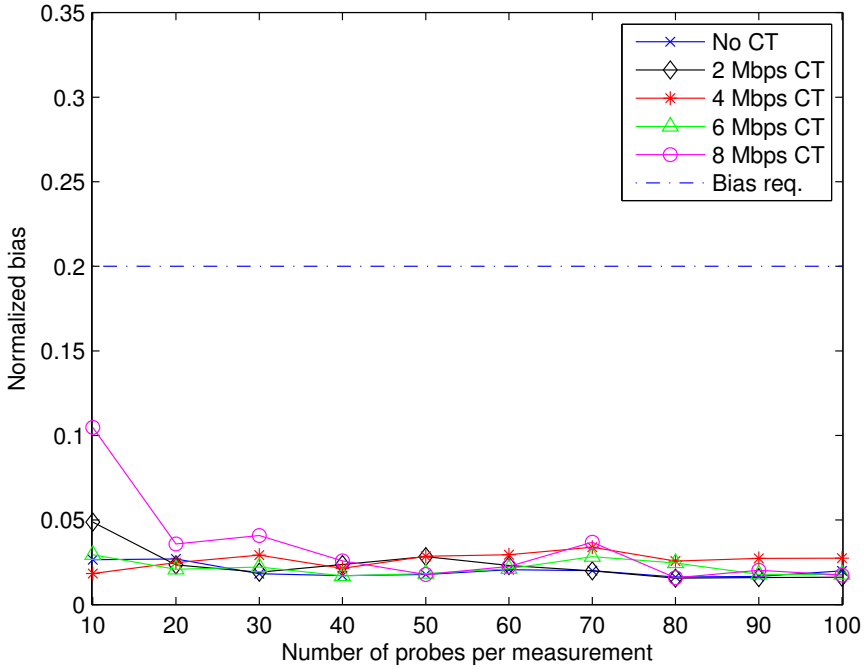


(a) Poisson cross traffic - bias

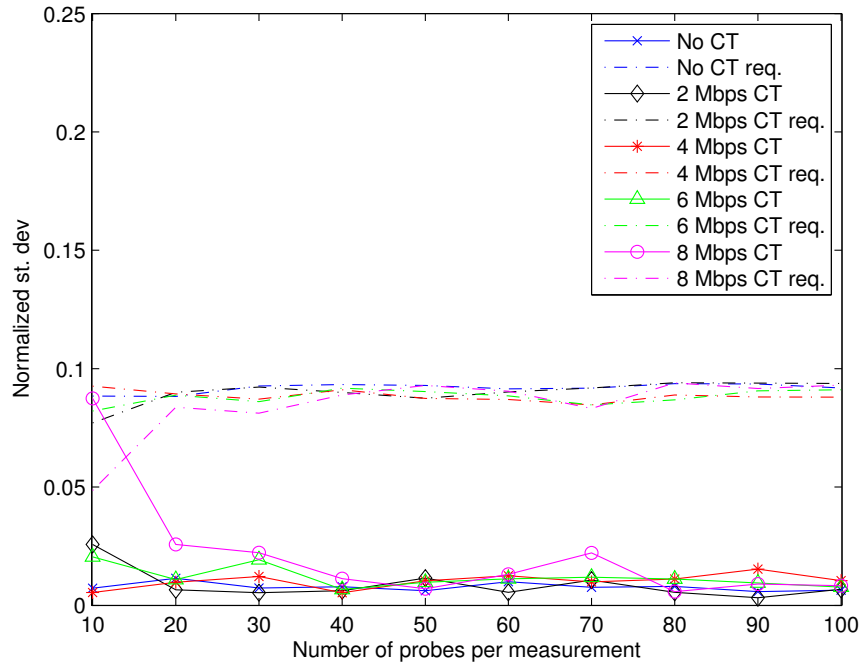


(b) Poisson cross traffic - standard deviation

Figure 4.4: Achieved and required performance for Ad Hoc Probe over two-hop IEEE802.11b with cross traffic (CT) on the same channel



(a) Poisson cross traffic - bias



(b) Poisson cross traffic - standard deviation

Figure 4.5: Achieved and required performance for Ad Hoc Probe over IEEE802.3 path with cross traffic (CT)

maximum value. The standard deviation quickly decayed as the number of probes were increased.

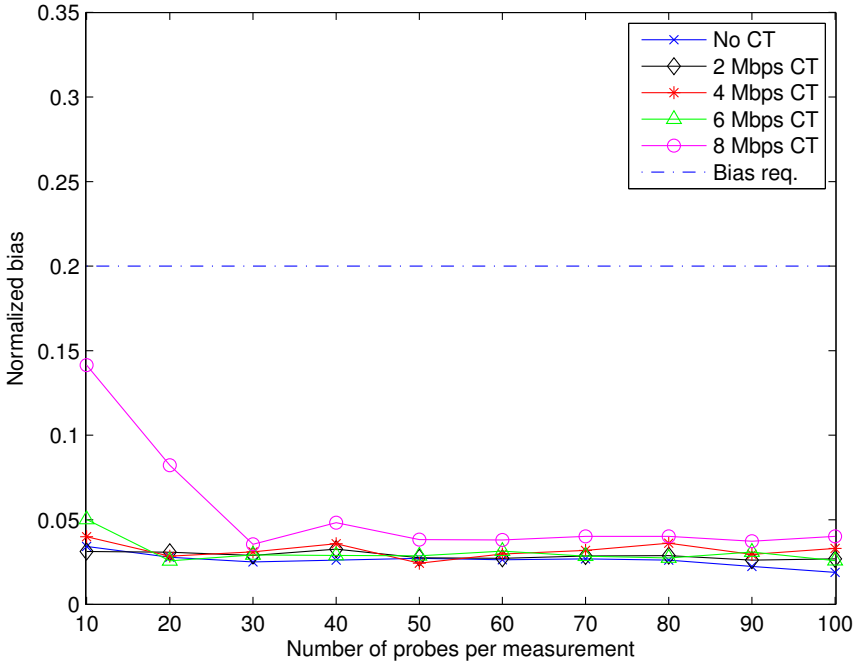
4.2.3 Satellite Link

Figure 4.6 shows the observed Ad Hoc Probe accuracy when the path was limited by a satellite link. The test bed configuration details are listed in table 4.6. The MGEN brute force path capacity estimate for packet size 700 bytes was 1.8 Mbps.

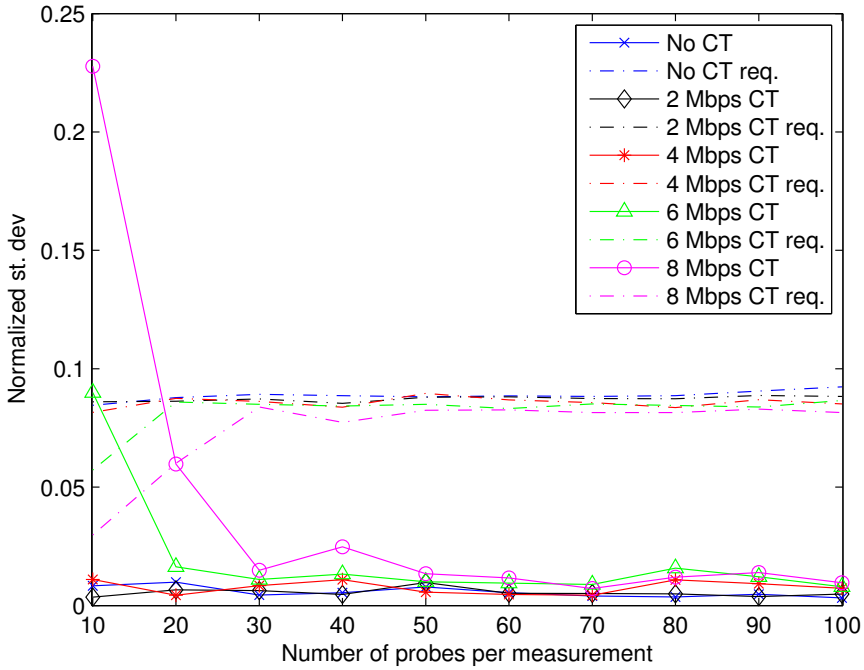
Table 4.6: Configuration for results in figure 4.6

Parameter	**Values used in measurements**
`L1`	Commercial satellite modem $R_b = 2048$ kbps
`L2`	IEEE802.3 10BASE-T $R_b = 10$ Mbps
`CrossRate`	0-80 % of 10 Mbps in steps of 20 %
`Numprobes`	10-100

The Ad Hoc Probe bias was within the requirements in all cases, while the standard deviation was within the requirements for 20 or more probes per measurement.



(a) Poisson cross traffic - bias



(b) Poisson cross traffic - standard deviation

Figure 4.6: Achieved and required performance for Ad Hoc Probe over satellite link path with cross traffic (CT)

4.3 Discussion

4.3.1 Accuracy with CSMA Contention

Capacity limited by single-hop The results from the single-hop case in figure 4.3 shows that Ad Hoc Probe performed remarkably well in this scenario. One might have expected that the performance decreased as the cross traffic intensity increased, leaving less and less "room" for the probes. However, the accuracy was nearly unaffected by the amount of cross traffic that was generated.

When the cross traffic intensity was high, Ad Hoc Probe needed to win the contention two times in a row to perform with the same accuracy as in the case of no cross traffic. In other words, the packet-pair sender needed to consecutively draw two backoff times whose sum was less than the contending node's backoff time counter at the time of the drawings. Judging from these results, the probability for this to occur within 10 attempts is very high.

Capacity limited by multi-hop As shown in figure 4.4, Ad Hoc Probe's performance was not as impressing when the path capacity was limited by a two-hop CSMA-link. It is safe to conclude that the algorithm's accuracy was strongly correlated to the cross traffic intensity. As long as the combined rate of the cross traffic and probes[†] stayed under the path capacity of 300 kbps, the standard deviation decreased as the number of probes increased and the estimates converged to the same accuracy as in the case of no cross traffic. By contrast, the estimator accuracy was severely reduced when the network was overloaded.

Why was the performance so degraded with the introduction of an additional hop? Much like the single-hop scenario, Ad Hoc Probe always had to contend for medium access when the collective load on the network exceeded the capacity. To get the correct estimate, it was necessary to gain access to the medium four times, in this specific order:

1. first packet first hop (always the case)
2. first packet second hop
3. second packet first hop
4. second packet second hop (always the case)

where it is especially important that the no other nodes win the contention between steps 2 to 3 and 3 to 4 as this would cause packet-pair expansion and consequently underestimation. At step 2, the first packet also has to contend with the first transmission of the second packet, introducing the possibility of interchanging step 2 and 3— a case which Ad Hoc Probe fails to detect unless the clock skew correction algorithm has been triggered — with the result that the minimum One-Way Delay (OWD) sum packet-pair is very likely to be compressed. In sum, Ad Hoc Probe dependent on a very unlikely sequence of events when every medium access was based on contention, thus resulting in the poor performance observed for high cross traffic intensities.

In the case of 0.3 Mbps cross traffic, the results indicate that the minimum OWD sum packet-pair was equally likely to suffer from packet expansion as compression since the bias was close

[†]The probing rate was ~120 kbps

to zero in many cases. With 0.2 and 0.4 Mbps cross traffic, compression of the minimum OWD sum packet-pair appears to have been the most likely case considering the large positive bias observed in the majority of the measurements.

4.3.2 Accuracy when Sharing a FIFO Queue

Capacity limited by IEEE802.3 Figure 4.5 shows that Ad Hoc Probe performed solidly over the IEEE802.3 limited path. Nevertheless, there was a deviating behavior at 10 probes per measurement during 8 Mbps cross traffic. Under these conditions, Ad Hoc Probe varied too much to be reliable and, on average, overestimated the path capacity.

This behavior was unexpected since there was no cross traffic destined for the red network which could have caused packet-pair compression at the second router. Again, the most likely reason behind this error is the context switch delay, which is the time between the arrival of a packet at the receiver's network interface and the timestamping that is first performed when the user space receiver application is switched into execution.

From chapter 3, the bias imposed by context switching is expected to be according to

$$\frac{\text{Bias}\{\widehat{C}\}}{C} = \frac{\texttt{PSize}}{\texttt{PSize} + C\left(\text{E}\{e_2\} - \text{E}\{e_1\}\right)} - 1 \tag{4.2}$$

where e_1 and e_2 are the random time measurement errors caused by context switching. In this case, a bias of 10 % of the 8.76 Mbps path capacity was observed. Based on equation (4.2), this would correspond to

$$\begin{aligned}
\text{E}\{e_2\} - \text{E}\{e_1\} &= \frac{\texttt{PSize}}{C}\left(\frac{1}{\text{Bias}\{\widehat{C}\}/C + 1} - 1\right) \\
&= \frac{700 \cdot 8}{8.76\text{e}6}\left(\frac{1}{0.10 + 1} - 1\right) \\
&= -58\ \mu\text{s}
\end{aligned}$$

which matches the maximum value of ~55 μs that was observed in the case of no cross traffic in figure 3.12.

Why is this not a problem for the other cross traffic intensities? The answer is that in the other cases, there most likely was a larger supply of packet-pairs that did not experience delay in the shared First In, First Out (FIFO)-queue, and that one of these were likely to obtain the same *minimum* context switch delay as in the case of no cross traffic.

In contrast, when the cross traffic intensity was high, more packet-pairs were likely to experience delay[†] in the FIFO-queue, thus the number of "good" packet-pairs was reduced, making it more likely that a packet-pair with a severe[‡] context switch delay was the minimum OWD sum sample used to calculate the path capacity.

[†] The time needed to transmit a 1200 bytes packet on a 10 Mbps link is~1 ms

[‡] In the order of tens of *micro*seconds, thus a great deal less than what is imposed on a packet-pair through queuing behind a 1200 byte packet

Capacity limited by satellite link The satellite link scenario satisfied the measurement requirements for 20 or more probes per measurement for all cross traffic cases. As shown in figure 4.6, the deviating cases were for the two highest cross traffic intensities, 6 and 8 Mbps, where the standard deviations exceeded the requirement for 10 probes per measurement. The bias curve shows that the estimator had the potential to overestimate the path capacity in both these cases, and this overestimation was larger than in the case of the pure IEEE802.3 path.

The context switch error's dependence on the true path capacity C makes it very unlikely that it should cause greater overestimation than in the case of a IEEE802.3 path with a capacity of 8.76 Mbps. Indeed, the bias curve in figure 4.6 require that the measurement error would have to be

$$\begin{aligned} \mathrm{E}\{e_2\} - \mathrm{E}\{e_1\} &= \frac{\texttt{PSize}}{C}\left(\frac{1}{\mathrm{Bias}\{\widehat{C}\}/C+1} - 1\right) \\ &= \frac{700 \cdot 8}{1.8e6}\left(\frac{1}{0.15+1} - 1\right) \\ &= -405\ \mu\mathrm{s} \end{aligned}$$

which is far more than the maximum error observed in chapter 3, indicating that the context switch delay is unlikely to be the only source of error here.

Compared to the pure IEEE802.3 scenario, the packet-pairs were far more "stretched out" time, leading to an increase in the probability of that the second packet in a pair was queued under the same cross traffic conditions. Therefore, the number of unqueued packet-pairs arriving at the receiver was likely to be less in this case. When using only 10 probes per measurement, the results suggest that there were many cases where all packet-pairs in a measurement had experienced some degree of either expansion or compression, and consequently the standard deviation was high.

Since the minimum OWD sum packet-pair was equally likely to be compressed or expanded, one might have expected that the bias was centered around zero, However, by considering the fact that $\lim_{\Delta t \to 0} \frac{\texttt{PSize}}{\Delta t} = \infty$, the observed bias is a natural result of that the packet-pair dispersion model is more sensitive to severe compression errors than expansion.

4.4 Evaluation

In this section the validity constraints of the hypothesis stated in the beginning of the chapter will be evaluated based on the conclusions that can be drawn in the light of the performed experiments and discussions.

Hypothesis

> Ad Hoc Probe [7] can provide path capacity estimates that comply to the requirements in section 2.3, even when the network is under load from other traffic sources.

Validity constraints - ideal conditions When there was no other traffic sources in the network, Ad Hoc Probe was found to satisfy the performance requirements if[†]:

a) the network consisted only of CSMA- and/or FDMA-based links
b) the path capacity was not limited by a wireless chain of more than two hops
c) the information bit rate of the slowest link was at least 44.4 kbps in cases where the path capacity was limited by the path's minimum link capacity, and minimum 90 kbps per link if the path capacity was limited by a two-hop wireless chain
d) the path capacity was not limited by a faster IEEE802.3 link than 10BASE-T, or a CSMA-link operating at a information bit rate higher than 2 Mbps, using the IEEE802.11b Contention Window size.

The question is how much more stringent these constraints became when there was other traffic sources in the network.

4.4.1 Modification of the Validity Constraints

MAC technique Restriction a) from the list above can be left unmodified since Ad Hoc Probe still provided usable results for both CSMA- and FDMA-based links.

Measurement time and intrusiveness Ad Hoc Probe provided very impressing results in the IEEE802.11b single-hop topology when contending with another node for medium access. The same accuracy as in the case of no cross traffic was observed. For two-hops, 90 probes per measurement were necessary to satisfy the accuracy requirements when the the network was on the verge of congestion, while 30 probes per measurement sufficed when the load was about 80 % of what the network could handle.

In the cases of IEEE802.3 and the commercial FDMA satellite link, the required number of probes per measurement were 20, and the same performance as in the case of no cross traffic was observed for 30 or more probes per measurement.

[†]See section 3.4.3

By accepting that the requirements are not strictly met for the two-hop topology when the network load is very high, these results suggest that 30 probes per measurement will result in a performance that is equivalent to the case of no cross traffic. This result makes it possible to calculate a minimum required probing rate based on the requirements in section 2.3, where it is stated that the measurement time for a flow of 100 byte packets must not exceed 6 seconds.

$$\begin{aligned}\min\{\text{Probing rate}\} &= \frac{2 \cdot \texttt{PSize}}{\max\{\text{Measurement time}\}} \cdot \min\{\texttt{Numprobes}\} \\ &= \frac{2 \cdot 100 \cdot 8}{6} \cdot 30 \\ &= 8 \text{ kbps}\end{aligned}$$

Furthermore, since the probing rate was required to at most be equal to 10 % of the path capacity, a requirement for a minimum path capacity can be found.

$$\min\{C\} = \frac{\min\{\text{Probing rate}\}}{0.1} = 80 \text{ kbps}$$

Taking into account the IPsec overhead ratio of 0.68 for 100 byte packets, a lower bound on the information bit rate of the slowest link in the network can be derived for the single-hop case.

$$\min\{R_b\} = \frac{\min\{C\}\}}{0.68} = 117.6 \text{ kbps}$$

For the two-hop case, double the information bit rate is needed on the individual links.

Accuracy Since Ad Hoc Probe converged to the same performance as the no cross traffic case, there is no reason to modify the constraint d) in the list.

4.4.2 Conclusive Remarks

Based on the experimental results and analysis provided in this chapter, it can be concluded that Ad Hoc Probe meets the performance requirements in section 2.2 if:

a) the network consists only of CSMA- and/or FDMA-based links
b) the path capacity is not limited by a wireless chain of more than two hops
c) the information bit rate of the slowest link is at least 117 kbps in cases where the path capacity is limited by the path's minimum link capacity, and minimum 234 kbps per link if the path capacity is limited by a two-hop wireless chain
d) the path capacity is not limited by a faster IEEE802.3 link than 10BASE-T, or a CSMA-link operating at a information bit rate higher than 2 Mbps, using the IEEE802.11b minimum Contention Window size.

The minimum requirements for information bit rate are very unlikely to be achieved in narrowband communication links [24]. If MBAC is to be deployed in a scenario where such low-capacity links are part of the black network, the probing rate has to be lowered in order to avoid congesting the network with probing traffic. This would increase the measurement time to way above the requirement of 6 seconds.

However, if the black network is based on short-to-medium range wideband communication links (e.g., [46]), the constraints are not "showstoppers", and Ad Hoc Probe can successfully be implemented as part of a measurement-based admission controller.

Chapter 5

Conclusions and Future Work

The aim of this thesis was to gain a deeper understanding of how the path capacity can be estimated in a lightweight manner, and further to evaluate if current state-of-the-art estimation techniques could provide satisfactory performance for Measurement-based Admission Control (MBAC) in military IP networks.

5.1 Conclusions

During the course of this thesis, the problem of finding a satisfactory path capacity estimation algorithm has been dealt with thoroughly. A set of requirements for a path capacity estimator in a military context was derived, after which a candidate estimation algorithm, Ad Hoc Probe [7], was selected for in-depth evaluation. The selection was based on a literature review of estimation techniques and current state-of-the-art algorithms.

The performance of Ad Hoc Probe was analyzed by conducting experiments in a test bed. During the analysis of these results, several insights into the performance limitations of the algorithm were gained. By filtering queued packet-pairs through the use of the minimum One-Way Delay (OWD) sum approach, the algorithm failed to capture the average behavior of unqueued packet-pairs, resulting in overestimation in cases where the path capacity was limited by Carrier Sense Multiple Access (CSMA)-based links.

Furthermore, the operating system descheduled the receiver application between the arrival of packet-pairs, causing time measurement errors that especially degraded the performance for small packet sizes in combination with a true path capacity in the order of several Mbps.

Ad Hoc Probe was incompatible with Time Division Multiple Access (TDMA)-based links. If both packets in a pair fitted into a single time slot, there was no correlation between the packet dispersion and the path capacity. If the packet-pair could not fit into a single time slot, there was correlation, but the estimates were biased to severely over- or underestimate the path capacity for packet sizes that did not correspond exactly to the size of a time slot. In general, the amount of data that fits into a time slot will be link-dependent, making the Ad Hoc Probe estimator unfit for use in networks where the path capacity is limited by TDMA-based links.

Ad Hoc Probe failed to detect the presence of a rate-limiter, since the packet-pairs were too short to exceed the burst size that was allowed to pass through unshaped. By contrast, in the case of IP fragmentation along the path, Ad Hoc Probe performed much better than the traditional brute-force method, capturing the effect of the added overhead.

The results from the wireless multi-hop experiments showed that the Ad Hoc Probe algorithm failed to filter the packet-pairs that had been scheduled for transfer in a non-ideal order, causing it to overestimate the path capacity for Carrier Sense Multiple Access Collision Avoidance (CSMA/CA)-based multi-hop networks with more than three hops.

When cross traffic was introduced into the experiments, the number of packet-pairs needed to obtain an unqueued packet-pair increased. Ad Hoc Probe converged to the same performance as in the case of no cross traffic when the number of packet-pairs per measurement was increased to 30.

The evaluation of these findings versus the set of requirements resulted in the conclusion that Ad Hoc Probe can be used as a path capacity estimator for MBAC in a military IP network if:

a) the network consists only of CSMA- and/or Frequency Division Multiple Access (FDMA)-based links

b) the path capacity is not limited by a wireless chain of more than two hops

c) the information bit rate of the slowest link is at least 117 kbps in cases where the path capacity is limited by the path's minimum link capacity, and minimum 234 kbps per link if the path capacity is limited by a two-hop wireless chain

d) the path capacity is not limited by a faster IEEE802.3-link than 10BASE-T, or a CSMA-link operating at a information bit rate higher than 2 Mbps, using the IEEE802.11b minimum Contention Window size.

Based on these insights and results, it is fair to conclude that the aim of the thesis was achieved.

5.2 Future Work

5.2.1 Implementing the Measurement-based Admission Controller

Although a candidate algorithm for path capacity estimation has been found, the job of implementing it as part of an admission controller still remains. Even though the discussions in this thesis provide some direction in the design choices that must be made, there are still unanswered questions. Among these are:

- How long is the lifetime of a single path capacity measurement?
- How to deal with drops in the path capacity. Is preemption the responsibility of the admission controller or user application?
- Where in the red network should the admission controller be situated?

5.2.2 Estimation of Other Parameters

The path capacity is not the ideal parameter for MBAC. The "holy grail" is the *available* capacity. Moreover, other parameters such as propagation delay and jitter are also of interest for MBAC.

5.2.3 Time-Division Multiple Access

The challenge of estimating the capacity of paths limited by TDMA-based links remains unsolved. Packet-pair dispersion analysis cannot be used alone due to the risk of not "filling" more than a single time slot. It would be interesting to evaluate whether or not minimum OWD sum packet-*trains* could be used a supplement to the packet-pairs for detecting and calculating the capacity of TDMA-based links.

5.2.4 Increase the Complexity

The results obtained in this thesis are based on a series of assumptions. Among these, the most important are that:

- all packets in a flow follows the same path through the network
- all flows were unicast, and not multi- or broadcast
- within each Differentiated Services (DiffServ)-class, the transmission queues are First In, First Out (FIFO)
- the path capacity does not decrease during the time it takes to complete an estimation

Removing one or more these preconditions is suggested for future work. This will increase the complexity of the path capacity estimation problem, and Ad Hoc Probe will not necessarily perform as well.

5.2.5 Evaluate the Performance of a One-Way variant of Allbest

When selecting an algorithm for in-depth analysis, Allbest [14] was found to be promising, but Ad Hoc Probe was chosen instead since a lot of time would be saved by not having to implement the estimation algorithm from scratch.

However, during the last days of the thesis work, there was found time to modify the Ad Hoc Probe source code into a one-way variant of Allbest. Moreover, the time stamping at the receiver was moved from user to kernel space by using the SO_TIMESTAMP socket option. Some preliminary results from this implementation can be found in appendix C. Unfortunately there was no time for further evaluation of this implementation, and this is therefore suggested as future work.

Bibliography

[1] M.A. Alzate, M.P. Salamanca, N.M. Pena, and M.A. Labrador. End-to-end mean bandwidth estimation as a function of packet length in mobile ad hoc networks. In *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pages 415 –420, july 2007.

[2] M.A. Alzate, M.P. Salamanca, N.M. Pena, and M.A. Labrador. End-to-end mean bandwidth estimation as a function of packet length in mobile ad hoc networks. In *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pages 415 –420, july 2007.

[3] Steven M. Bellovin. A best-case network performance model. Technical report, AT&T Bell Laboratories, February 1992.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), December 1998. Updated by RFC 3260.

[5] M. Brown, D. Bushmitch, K. Kerpez, D. Waring, and Y. Wang. Low-bit rate video codec parameter evaluation and optimization. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–20. IEEE, 2009.

[6] E. Casini, M. Street, P. Vigneron, and R. Barfoot. SDR-Ready Standardized Waveforms for Tactical VHF and UHF Communications for NATO. In *Information Systems and Technology Panel (IST) Symposium*, 2010.

[7] Ling-Jyh Chen, Tony Sun, Guang Yang, M. Sanadidi, and Mario Gerla. AdHoc Probe: end-to-end capacity probing in wireless ad hoc networks. *Wireless Networks*, 15:111–126, 2009. 10.1007/s11276-007-0047-4.

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

[9] Everaldo Coelho, YellowIcon, George Shuklin, and Ford prefect. Respectively creators of Laptop icon (License: LGPL v2.1), Router icon (License: GFDL v1.2) and IPSec device icon (License: CC-BY). `http://commons.wikimedia.org`.

[10] Comtech EF Data Corporation. DMD20 Universal Satellite Modem. `http://www.comtechefdata.com/files/datasheets/ds-DMD20.pdf`.

[11] Harris Corporation. RF-7800S Secure Personal Radio. http://rf.harris.com/capabilities/tactical-radios-networking/rf-7800s/default.asp.

[12] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246 (Proposed Standard), March 2002.

[13] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437.

[14] A. Delphinanto, T. Koonen, Shuang Zhang, and F. den Hartog. Path capacity estimation in heterogeneous, best-effort, small-scale ip networks. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 1076 –1083, oct. 2010.

[15] 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, pages 905 –914 vol.2, 2001.

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

[17] Allen B. Downey. Using pathchar to estimate internet link characteristics. *SIGCOMM Comput. Commun. Rev.*, 29(4):241–250, August 1999.

[18] Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) Research Group. Multi-Generator (MGEN). http://cs.itd.nrl.navy.mil/work/mgen/.

[19] IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks–Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Section One. *IEEE Std 802.3-2008 (Revision of IEEE Std 802.3-2005)*, pages 49,56 and 91, 26 2008.

[20] IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, page 270 and 547, 12 2007.

[21] Vyatta Inc. Vyatta Core software (VC). http://www.vyatta.org.

[22] Network grade of service parameters and target values for circuit-switched services in the evolving ISDN. *ITU-T Recommendation E.721*, page 5, 1999.

[23] Van Jacobson. Pathchar: A tool to infer characteristics of internet paths. ftp://ftp.ee.lbl.gov/pathchar/, April 1997.

[24] V. Jodalen, B. Solberg, and S. Haavik. NATO Narrowband Waveform (NBWF)-overview of link layer design. Technical Report 2009/01894, Forsvarets Forskningsinstitutt (Norwegian Defence Research Establishment), 2011.

[25] A. Johnsson, B. Melander, and M. Björkman. Diettopp: A first implementation and evaluation of a simplified bandwidth measurement method. In *Second Swedish National Computer Networking Workshop*, page 5, 2004.

[26] Jangeun Jun, P. Peddabachagari, and M. Sichitiu. Theoretical maximum throughput of ieee 802.11 and its applications. In *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on*, pages 249 – 256, april 2003.

[27] R. Kapoor, L.J. Chen, MY Sanadidi, and M. Gerla. Accuracy of link capacity estimates using passive and active approaches with CapProbe. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 2, pages 1085–1090. IEEE, 2004.

[28] 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.*, 34:67–78, August 2004.

[29] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), December 2005.

[30] UCLA Network Research Laboratory. Website for downloading CapProbe and Ad Hoc Probe. `http://www.cs.ucla.edu/~nrl/CapProbe/download.htm`.

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

[32] P. Lettieri and M.B. Srivastava. Adaptive frame length control for improving wireless link throughput, range, and energy efficiency. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 564 –571 vol.2, mar-2 apr 1998.

[33] Chuanpeng Li, Chen Ding, and Kai Shen. Quantifying the cost of context switch. In *Proceedings of the 2007 workshop on Experimental computer science*, ExpCS '07, New York, NY, USA, 2007. ACM.

[34] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of Ad Hoc wireless networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 61–69, New York, NY, USA, 2001. ACM.

[35] Canonical Ltd. Ubuntu operating system. `http://www.ubuntu.com`.

[36] B.A. Mah. Pchar: Child of pathchar. In *DOE NGI testbed workshop, berkeley, ca*, volume 21, 1999.

[37] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, volume 1, pages 415 –420 vol.1, 2000.

[38] NATO. What is NNEC? `http://nnec.act.nato.int/Test/NNEC_WhatisNNEC_TextOverview.pdf`. Retrieved 26. Jan 2012.

[39] Alfa Networks. AWUS036NEH Wireless USB adapter. `http://www.alfa.com.tw/in/front/bin/ptdetail.phtml?Part=AWUS036NEH&Category=105483`.

[40] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), December 1998. Updated by RFCs 3168, 3260.

[41] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), September 1981. Updated by RFCs 950, 4884.

[42] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

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

[44] A. Shipalov, C.D. Guerrero, M.A. Labrador, and M. Alzate. On the implementation of a capacity estimator for wireless ad hoc networks. In *Southeastcon, 2009. SOUTHEASTCON '09. IEEE*, pages 242 –247, march 2009.

[45] B. Sklar. *Digital communications: fundamentals and applications*. Prentice Hall Communications Engineering and Emerging Technologies Series. Prentice-Hall PTR, 2001.

[46] Kongsberg Defence Systems. TacLAN. `http://www.kongsberg.com/en/kds/products/defencecommunications/taclan`.

[47] L. Zhang, Z. Liu, and C. Honghui Xia. Clock synchronization algorithms for network measurements. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 160–169. IEEE, 2002.

Appendices

Appendix A

Theoretical Maximum Capacity

A.1 The TMC of IEEE802.11b

The following relation can be used to find the Theoretical Maximum Capacity (TMC) of IEEE802.11b [20] links using Ready-to-send (RTS)/Clear-to-send (CTS) for Collision Avoidance:

$$\text{TMC} = \frac{\texttt{PSize}}{\texttt{DIFS} + \frac{\texttt{CW}_{\texttt{min}}}{2} + 4(\texttt{Pre} + \texttt{PLCP}) + \frac{\texttt{RTS+CTS+ACK}}{\texttt{BR}} + 3 \cdot \texttt{SIFS} + \frac{\texttt{L2H+PSize}}{\texttt{R}}} \quad \text{(A.1)}$$

where

- DIFS is the minimum time the medium has to be sensed idle before transmission
- $\texttt{CW}_{\texttt{min}}$ is the minimum contention window size, and the expected value of the backoff time is one half of this time since it is a uniformly distributed random variable from the range $[0 - \text{CW}_{min}]$
- Pre is the size in bits of the synchronization sequence before the layer 1 header
- PLCP is the size in bits of the layer 1 header
- RTS, CTS, ACK is the number of bits in the respective layer 2 control frames
- SIFS is the time to wait between RTS/CTS/ACK
- L2H is the number of bits in the layer 2 data frame header
- BR is the information data rate used to transmit the Pre, layer 1 header and layer 2 control frames.
- R_b is the information data rate used to transmit the layer 2 data frame

It is important to note that this calculation indeed provides an upper bound since some of the channel capacity will be used for signaling traffic; for instance the beacon signal that advertizes

the presence and capabilites of the network or handshakes for associating/disassociating. More important, since Automatic Repeat reQuest (ARQ) is part of the IEEE802.11b Medium Access Control (MAC)-protocol, there will always be time consuming retransmissions due to bit errors caused by varying channel conditions.

In this thesis, the Direct Sequence Spread Spectrum (DSSS) variant of IEEE802.11b with data rate will be used. The values for the parameters above are listed in table A.1.

Table A.1: Parameter values for calculating the Theoretical Maximum Throughput for IEEE802.11b

DIFS	50 μs
CW_{min}	620 μs
Pre	144 μs
PLCP	48 μs
RTS	160 bits
CTS	136 bits
ACK	136 bits
SIFS	10 μs
L2H	272 bits
BR	1 Mbps
R_b	1,2,5.5,11 Mbps

A.2 The TMC of IEEE802.3 Ethernet

The following relation can be used to find the TMC of an IEEE802.3 [19] link

$$\text{TMC} = \frac{\texttt{PSize}}{(\texttt{IPG} + \texttt{L2H} + \texttt{PSize})/R_b} \tag{A.2}$$

where IPG is the minimum interpacket gap measured in bits, L2H is the number of bits in the layer 2 data frame header and R_b is the information data rate used to transmit the layer 2 data frame. Equation equation (A.2) is purely deterministic and is valid for cases where there is no contention for medium access.

Table A.2 lists the parameter values that are used in this thesis for calculation of the TMC for Ethernet 10BASE-T links.

Table A.2: Parameter values for calculating the Theoretical Maximum Throughput for IEEE802.3

IPG	96 bits
L2H	208 bits
R	10 Mbps

Appendix B

Details Regarding the Path Capacity Experiments

B.1 IPsec Overhead

IPsec and packet fragmentation introduce a packet size dependent overhead. Since calculation of the TMC for various link technologies require that the size of the layer 2 payload is known, it was necessary to find the mapping between the Plaintext (PT) and Ciphertext (CT) packet sizes. Table B.1 shows the observed difference in packet size between PT and CT IP packets.

Table B.1: The layer 3 overhead (OH) for various PT packet sizes. All values except the overhead ratio are in bytes

Red PSize	Black PSize	Black IP Header	ESP const. OH	ESP Padding	$\frac{\text{PT PSize}}{\text{(PT PSize+OH)}}$
100	168	20	10	38	0.60
200	264	20	10	34	0.76
300	358	20	10	28	0.84
400	472	20	10	42	0.85
500	568	20	10	38	0.88
600	664	20	10	34	0.90
700	760	20	10	30	0.92
800	872	20	10	42	0.92
900	968	20	10	38	0.93
1000	1064	20	10	34	0.94
1100	1160	20	10	30	0.95
1200	P1: 1196 P2: 96	40	20	32	0.93
1300	P1: 1196 P2: 192	40	20	28	0.94
1400	P1: 1196 P2: 288	40	20	24	0.94
1500	P1: 1196 P2: 384	40	20	20	0.95

Small packets were especially exposed to overhead. IPsec overhead varied in size since the

plaintext data had to fit specific block sizes given by the implementation-specific cryptographic algorithm. Fragmentation took place for 1200-1500 bytes plaintext packets.

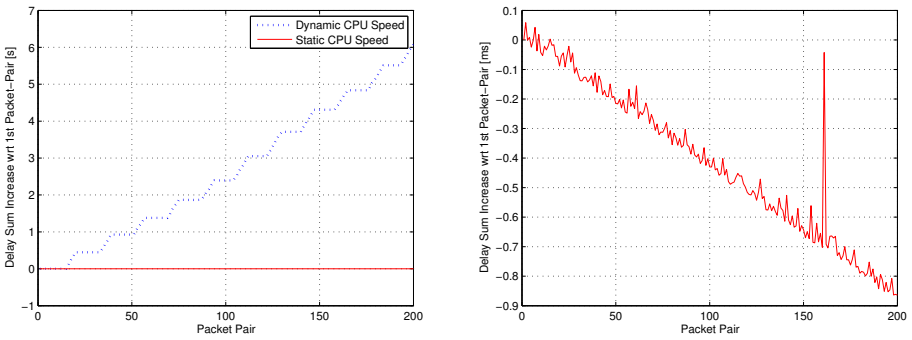
B.2 Specific Settings in Operating System

The system default Path MTU Discovery was disabled at the sender using the command

```
echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc
```

due to the fact that the Vyatta IPsec router copied the Don't Fragment bit from the inner to the outer IP header, thereby making the black network routers discard all packets of length larger than the path's minimum Maximum Transmission Unit (MTU). I did not find a way to disable this behavior in Vyatta.

Dynamic CPU Speed Throttling proved to cause severe clock skew problems in the order of several seconds. This was due to the fact that the timing in the Ad Hoc Probe implementation depends on the x86 CPU-register Time Stamp Counter that is incremented once per clock cycle. The tool `rcconf` was used to to disable the `ondemand` CPU Frequency Scaling governor in Ubuntu Linux. Figure B.1 shows the resulting improvement in clock skew performance.



(a) Comparison

(b) Zoomed in on "Static CPU speed" curve. Note the millisecond time scale.

Figure B.1: The effect of dynamic CPU Throttling on the clock skew

B.3 Vyatta Sample Configuration

B.3.1 Vyatta1

```
interfaces {
    ethernet eth0 {
        address 192.168.0.1/24
        duplex auto
        hw-id 00:21:91:8c:ca:b2
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 10.0.2.1/29
        duplex auto
        hw-id 00:13:21:68:01:ed
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.0.1/32
    }
}
protocols {
    static {
        route 0.0.0.0/0 {
            next-hop 10.0.2.2 {
            }
        }
    }
}
service {
    ssh {
        port 22
        protocol-version v2
    }
}
system {
    config-management {
        commit-revisions 20
    }
    console {
        device ttyS0 {
            speed 9600
        }
    }
    host-name vyatta1
    login {
        user vyatta {
            authentication {
                encrypted-password $1$XQyI9wAj$WzxA6mn/jZ3GpkyNmlkPk0
                public-keys Generert_23.01.2012_Espen_Flydahl {
```

```
49                      key AAAAB3NzaC1yc2EAAAABJQAAAIEApa76gP+4
                            xjOP7Pqf53e8uDpqjDnDZRye5Yqg0e222ES2G/pKcnX6jgMRG7B+
                            U6OnOUFjp0VPYa9+
                            Lo8l5rzGLVO3oP3JVB7U67XighLifW4NULvueESaj/
                            ssUeC8j7ShpAPGSGxso7TDktVngOxJBaXThukhZtnmi6EMWVz5PUc
                            =
                        type ssh-rsa
51                  }
                }
53              level admin
            }
55      }
        ntp {
57          server 0.vyatta.pool.ntp.org {
            }
59          server 1.vyatta.pool.ntp.org {
            }
61          server 2.vyatta.pool.ntp.org {
            }
63      }
        package {
65          auto-sync 1
            repository community {
67              components main
                distribution stable
69              password ""
                url http://packages.vyatta.com/vyatta
71              username ""
            }
73      }
        syslog {
75          global {
                facility all {
77                  level notice
                }
79              facility protocols {
                    level debug
81              }
            }
83      }
        time-zone GMT
85  }
    vpn {
87      ipsec {
            esp-group ESP {
89              compression disable
                lifetime 1800
91              mode tunnel
                pfs enable
93              proposal 1 {
                    encryption aes128
95                  hash sha1
                }
```

```
97          }
            ike-group IKE {
99              lifetime 3600
                proposal 1 {
101                 encryption aes128
                    hash sha1
103             }
            }
105         ipsec-interfaces {
                interface eth1
107         }
            site-to-site {
109             peer 10.0.2.26 {
                    authentication {
111                     mode pre-shared-secret
                        pre-shared-secret test
113                 }
                    connection-type initiate
115                 default-esp-group ESP
                    ike-group IKE
117                 local-ip 10.0.2.1
                    tunnel 1 {
119                     allow-nat-networks disable
                        allow-public-networks disable
121                     local {
                            subnet 192.168.0.0/24
123                     }
                        remote {
125                         subnet 192.168.1.0/24
                        }
127                 }
                }
129         }
        }
131 }

133
    /* Warning: Do not remove the following line. */
135 /* === vyatta-config-version: "vrrp@1:content-inspection@3:ipsec@3:dhcp-
        relay@1:zone-policy@1:qos@1:dhcp-server@4:config-management@1:webproxy@1
        :quagga@2:wanloadbalance@3:system@5:firewall@4:cluster@1:webgui@1:nat@3:
        conntrack-sync@1" === */
    /* Release version: 999.mtnapa.10181129 */
```

B.3.2 Vyatta2

```
    interfaces {
2       ethernet eth0 {
            address 10.0.2.2/29
4           duplex auto
            hw-id 00:0e:7f:60:b9:b2
```

```
6           smp_affinity auto
            speed auto
8       }
        ethernet eth1 {
10          address 10.0.2.9/29
            duplex auto
12          hw-id 00:30:4f:6d:a5:e8
            smp_affinity auto
14          speed auto
        }
16      ethernet eth2 {
            address 10.0.2.17/29
18          duplex full
            hw-id 00:30:4f:6d:94:72
20          mtu 1200
            smp_affinity auto
22          speed 10
        }
24      loopback lo {
            address 10.0.0.2/32
26      }
    }
28  protocols {
        static {
30          route 0.0.0.0/0 {
                next-hop 10.0.2.18 {
32              }
            }
34          route 10.0.0.1/32 {
                next-hop 10.0.2.1 {
36              }
            }
38      }
    }
40  service {
        ssh {
42          port 22
            protocol-version v2
44      }
    }
46  system {
        config-management {
48          commit-revisions 20
        }
50      console {
            device ttyS0 {
52              speed 9600
            }
54      }
        domain-name vyatta1
56      host-name vyatta2
        login {
58          user vyatta {
```

```
            authentication {
60              encrypted-password $1$4XHPj9eT$G3ww9B/pYDLSXC8YVvazP0
                public-keys Generert_23.01.2012_Espen_Flydahl {
62                  key AAAAB3NzaC1yc2EAAAABJQAAAIEApa76gP+4
                        xjOP7Pqf53e8uDpqjDnDZRye5Yqg0e222ES2G/pKcnX6jgMRG7B+
                        U6OnOUFjp0VPYa9+
                        Lo8l5rzGLVO3oP3JVB7U67XighLifW4NULvueESaj/
                        ssUeC8j7ShpAPGSGxso7TDktVngOxJBaXThukhZtnmi6EMWVz5PUc
                        =
                    type ssh-rsa
64              }
            }
66          level admin
        }
68  }
    ntp {
70      server 0.vyatta.pool.ntp.org {
        }
72      server 1.vyatta.pool.ntp.org {
        }
74      server 2.vyatta.pool.ntp.org {
        }
76  }
    package {
78      auto-sync 1
        repository community {
80          components main
            distribution stable
82          password ""
            url http://packages.vyatta.com/vyatta
84          username ""
        }
86  }
    syslog {
88      global {
            facility all {
90              level notice
            }
92          facility protocols {
                level debug
94          }
        }
96  }
    time-zone GMT
98 }

100
    /* Warning: Do not remove the following line. */
102 /* === vyatta-config-version: "cluster@1:config-management@1:conntrack-
       sync@1:content-inspection@3:dhcp-relay@1:dhcp-server@4:firewall@4:
       ipsec@3:nat@3:qos@1:quagga@2:system@5:vrrp@1:wanloadbalance@3:webgui@1:
       webproxy@1:zone-policy@1" === */
    /* Release version: 999.mtnapa.10181129 */
```

B.3.3 Vyatta3

```
interfaces {
    ethernet eth0 {
        address 10.0.2.25/29
        duplex auto
        hw-id 00:1a:4b:36:f6:c8
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 10.0.2.33/29
        duplex auto
        hw-id 00:21:91:8c:c8:9c
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 10.0.2.18/29
        duplex full
        hw-id 00:21:91:8c:c7:ad
        mtu 1200
        smp_affinity auto
        speed 10
    }
    loopback lo {
        address 10.0.0.3/32
    }
}
protocols {
    static {
        route 0.0.0.0/0 {
            next-hop 10.0.2.17 {
            }
        }
        route 10.0.0.4/32 {
            next-hop 10.0.2.26 {
            }
        }
    }
}
service {
    ssh {
        port 22
        protocol-version v2
    }
}
system {
    config-management {
        commit-revisions 20
    }
    console {
        device ttyS0 {
```

```
            speed 9600
        }
    }
    host-name vyatta3
    login {
        user vyatta {
            authentication {
                encrypted-password $1$zREsATH7$/tHbr1jBjP6CdjLnGlKwf/
                public-keys Generert_23.01.2012_Espen_Flydahl {
                    key AAAAB3NzaC1yc2EAAAABJQAAAIEApa76gP+4
                        xjOP7Pqf53e8uDpqjDnDZRye5Yqg0e222ES2G/pKcnX6jgMRG7B+
                        U6OnOUFjp0VPYa9+
                        Lo8l5rzGLVO3oP3JVB7U67XighLifW4NULvueESaj/
                        ssUeC8j7ShpAPGSGxso7TDktVngOxJBaXThukhZtnmi6EMWVz5PUc
                        =
                    type ssh-rsa
                }
            }
            level admin
        }
    }
    ntp {
        server 0.vyatta.pool.ntp.org {
        }
        server 1.vyatta.pool.ntp.org {
        }
        server 2.vyatta.pool.ntp.org {
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    syslog {
        global {
            facility all {
                level notice
            }
            facility protocols {
                level debug
            }
        }
    }
    time-zone GMT
}
traffic-policy {
    rate-control narrow_link {
```

```
        bandwidth 64kbit
        burst 15k
        description "Narrowband link"
        latency 50ms
    }
    rate-control wide_link {
        bandwidth 2mbit
        burst 15k
        description "Wideband link"
        latency 50ms
    }
}

/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "quagga@2:conntrack-sync@1:dhcp-relay@1:
    webgui@1:zone-policy@1:firewall@4:webproxy@1:qos@1:dhcp-server@4:
    system@5:content-inspection@3:config-management@1:ipsec@3:
    wanloadbalance@3:nat@3:cluster@1:vrrp@1" === */
/* Release version: 999.mtnapa.10181129 */
```

B.3.4 Vyatta4

```
interfaces {
    ethernet eth0 {
        address 192.168.1.1/24
        duplex auto
        hw-id 00:26:5a:81:a1:04
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 10.0.2.26/29
        duplex auto
        hw-id 00:11:85:79:f1:8d
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 10.0.0.4/32
    }
}
protocols {
    static {
        route 0.0.0.0/0 {
            next-hop 10.0.2.25 {
            }
        }
    }
}
service {
```

```
    ssh {
        port 22
        protocol-version v2
    }
}
system {
    config-management {
        commit-revisions 20
    }
    console {
        device ttyS0 {
            speed 9600
        }
    }
    host-name vyatta4
    login {
        user vyatta {
            authentication {
                encrypted-password $1$uP8V1n6Y$SS.wfEGlH0cl8ybPmxBA5/
                public-keys Generert_23.01.2012_Espen_Flydahl {
                    key AAAAB3NzaC1yc2EAAAABJQAAAIEApa76gP+4
                        xjOP7Pqf53e8uDpqjDnDZRye5Yqg0e222ES2G/pKcnX6jgMRG7B+
                        U6OnOUFjp0VPYa9+
                        Lo8l5rzGLVO3oP3JVB7U67XighLifW4NULvueESaj/
                        ssUeC8j7ShpAPGSGxso7TDktVngOxJBaXThukhZtnmi6EMWVz5PUc
                        =
                    type ssh-rsa
                }
            }
            level admin
        }
    }
    ntp {
        server 0.vyatta.pool.ntp.org {
        }
        server 1.vyatta.pool.ntp.org {
        }
        server 2.vyatta.pool.ntp.org {
        }
    }
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    syslog {
        global {
            facility all {
```

```
                level notice
            }
            facility protocols {
                level debug
            }
        }
    }
    time-zone GMT
}
vpn {
    ipsec {
        esp-group ESP {
            compression disable
            lifetime 1800
            mode tunnel
            pfs enable
            proposal 1 {
                encryption aes128
                hash sha1
            }
        }
        ike-group IKE {
            lifetime 1800
            proposal 1 {
                encryption aes128
                hash sha1
            }
        }
        ipsec-interfaces {
            interface eth1
        }
        site-to-site {
            peer 10.0.2.1 {
                authentication {
                    mode pre-shared-secret
                    pre-shared-secret test
                }
                connection-type initiate
                default-esp-group ESP
                ike-group IKE
                local-ip 10.0.2.26
                tunnel 1 {
                    allow-nat-networks disable
                    allow-public-networks disable
                    local {
                        subnet 192.168.1.0/24
                    }
                    remote {
                        subnet 192.168.0.0/24
                    }
                }
            }
        }
```

```
130         }
    }
132
134 /* Warning: Do not remove the following line. */
    /* === vyatta-config-version: "webproxy@1:nat@3:quagga@2:conntrack-sync@1:
        ipsec@3:zone-policy@1:content-inspection@3:dhcp-server@4:system@5:
        firewall@4:config-management@1:cluster@1:dhcp-relay@1:qos@1:vrrp@1:
        wanloadbalance@3:webgui@1" === */
136 /* Release version: 999.mtnapa.10181129 */
```

B.4 Scripts for Measuring the Correct Path Capacity

B.4.1 Traffic Generation

```
#!/bin/bash

# E. Flydahl - Script for testing the path capacity with mgen

maxpsize=100  # Maximum IP packet size (bytes)
minpsize=100  # Minimum IP packet size (bytes)
psizeint=100  # Increase in packet size per run (bytes)
runtime=30    # How long each run lasts (seconds)
wait=5        # How long to wait between runs (seconds)
bitrate=12    # Layer 3 output bit rate (Mbps)

receiverip="192.168.1.2"
dstport=5001
srcport=5001
overhead=28 # IP header 20 bytes + UDP header 8 bytes

control_c()
{
  echo "Enabling Path MTU Discovery"
  echo 0 > /proc/sys/net/ipv4/ip_no_pmtu_disc

  echo "Terminating"

  if [ $1 -eq 0 ]
  then
    exit 0
  fi
  ssh $2 "kill $1"

  if [ $? -ne 0 ]
  then
    echo "Could not kill sink process" >&2
    exit 1
  fi
  exit 0
}

if [ $EUID -ne 0 ]
then
  echo "Please run as root"
  exit 1;
fi

if [ ! -x mgen ]
then
  echo "Cannot find/execute mgen in current directory" >&2
  exit 1
fi
```

```
50 if [ ! -x /usr/bin/bc ]
   then
52   echo "Cannot find/execute bc" >&2
     exit 1
54 fi

56 echo "Disabling Path MTU Discovery"
   echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc
58
   echo "Starting sink process at $receiverip"
60 # Start sink, "return" PID
   sinkpid=$(ssh $(echo $receiverip' nohup mgen event "LISTEN UDP '$dstport'"
      > mgenout.drc 2> /dev/null < /dev/null & echo $!'))
62 if [ $? -ne 0 ]
   then
64   echo "Could not start sink process at $receiverip" >&2
     exit 1
66 fi
   trap "control_c $sinkpid $receiverip" SIGINT
68
   echo "Starting traffic generation"
70
   psize=$minpsize
72 while [ $psize -le $maxpsize ]
   do
74   echo "Sending IP packets of size $psize bytes at rate $bitrate Mbps for
        $runtime seconds"
     #echo "$bitrate*10^6/($psize-$overhead)/8"|bc -l
76   #echo "($psize-$overhead)*8"|bc -l
     ./mgen event "$wait ON $psize UDP SRC $srcport DST $receiverip/$dstport
        PERIODIC \
78     [`echo "$bitrate*10^6/($psize*8)"|bc -l` `echo "$psize-$overhead"|bc -l
          `]" \
      event "$[$runtime+$wait] OFF $psize"
80
     psize=$[$psize+$psizeint]
82
   done
84 echo "Traffic generation complete"

86 echo "Enabling Path MTU Discovery"
   echo 0 > /proc/sys/net/ipv4/ip_no_pmtu_disc
88
   echo "Killing sink process"
90 ssh $receiverip "kill $sinkpid"
   if [ $? -ne 0 ]
92 then
     echo "Could not kill sink process at receiver" >&2
94   exit 1
   fi
96 sinkpid=0
   trap "control_c $sinkpid $receiverip" SIGINT
```

```
98
echo "Compressing trace file at $receiverip"
100 ssh $receiverip "gzip -f mgenout.drc"
if [ $? -ne 0 ]
102 then
  echo "Could not compress output data at receiver" >&2
104   exit 1
fi
106
filename=$(date "+mgenout_%Y%m%d_%H%M%S.drc.gz")
108 echo "Copying trace file to $(pwd)/$filename"
scp $receiverip:mgenout.drc.gz $filename
110 if [ $? -ne 0 ]
then
112   echo "Could not copy trace file from receiver" >&2
  exit 1
114 fi

116 echo "Deleting trace file from $receiverip"
ssh $receiverip "rm mgenout.drc.gz"
118 if [ $? -ne 0 ]
then
120   echo "Could not delete trace file at receiver" >&2
  exit 1
122 fi


124
filename2=$(date "+path_cap_$(echo $bitrate)_%Y%m%d_%H%M%S")
126 echo "Processing tracefile and saving final result in $(pwd)/$filename2"
zcat $filename | gawk -f process.awk > $filename2
128 if [ $? -ne 0 ]
then
130   echo "Could not process trace file" >&2
  exit 1
132 fi
echo
134 cat $filename2
echo
136
echo "Exiting"
138 echo -e "\a" >&2
exit 0;
```

The receiver must have a SSH-server with the sender's public key installed for passwordless login in order to make the script work as intended.

B.4.2 Trace File Analysis

```
1 # E. Flydahl - Awk script to process mgen trace file. Delivers psize,
    capacity, MB, transferred, packet loss percentage, packet count
```

```
BEGIN{
  startdelay = 5; # How far into the flow before starting measurements (seconds)
  duration = 20; # How long to measure the flow
}

{

  if ($2=="RECV")
  {
    split($1, tmp,":");
    trcv = tmp[1]*60*60+tmp[2]*60+tmp[3];
    split($4, tmp, ">");
    flow=tmp[2];
    split($5, tmp, ">");
    seq=tmp[2];

    # Register flow
    found = 0
    for(f in rcvdflows)
    {
      if (flow == f) {found = 1;}
    }
    if (found == 0)
    {
      rcvdflows[flow]=1;
      tmin[flow] = trcv+startdelay;
      tmax[flow] = tmin[flow]+duration;
    }

    # Count received and lost packets
    if (flow in rcvdflows && trcv >= tmin[flow] && trcv < tmax[flow])
    {
      if(!rcv[flow]) {nextseq[flow]=seq;}
      rcv[flow]++;
      if(seq>=nextseq[flow])
      {
        ploss[flow] = ploss[flow]+seq-nextseq[flow];

        #if(seq-nextseq[flow]>0)
        #{
        # printf("When rcving seqnr %d,%d packets lost at time %f from flow %d\n",seq, seq-nextseq[flow],trcv, flow) > "/dev/stderr" ;
        #}
        nextseq[flow]=seq+1;
      }
      else
      {
        ploss[flow]--;
        #printf("seq %d from flow %d rcvd out of order\n", seq,flow) > "/dev/stderr";
      }
    }
```

```
53
    }
55
  }
57 END{
    # Sort indices in separate array ind
59  j=1;
    for (i in rcv)
61  {
      ind[j]=i;
63    ind[j]++;ind[j]--; # "Cast" to integer
      j++;
65  }
    n=asort(ind);
67
    # Print out data
69  for(i=1;i<=n;i++)
    {
71    printf("%d %.2f %.1f %.3f %d\n", ind[i], rcv[ind[i]]*ind[i]*8/duration,
          rcv[ind[i]]*ind[i]*1024^(-2), ploss[ind[i]]/(nextseq[ind[i]]-1), rcv
          [ind[i]]);
    }
73 }
```

B.5 Ad Hoc Probe Code

B.5.1 Sender Source Code

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <string.h>
   #include <unistd.h>
5  #include <sys/types.h>
   #include <sys/shm.h>
7  #include <sys/time.h>
   #include <sys/select.h>
9  #include <sys/param.h>
   #include <sys/types.h>
11 #include <sys/socket.h>
   #include <assert.h>
13 #include <netinet/in.h>
   #include <arpa/inet.h>
15 #include <netdb.h>
   #include <signal.h>
17 #include <time.h>

19 #define PORT_F 15000
   #define PORT_B 15000
21 #define MAXBUFLEN 100000

23 // IP: 20 bytes  UDP: 8 bytes
   #define HEADER_SIZE 28
25
       __inline__ unsigned long long rdtsc(){
27             unsigned long long int x;
                    __asm__ volatile (".byte 0x0f, 0x31" : "=A" (x));
29                       return x;
       }
31
   struct sockaddr_in addr_f; // connector's address information
33
   int main(int argc, char *argv[]) {
35   char *addr = NULL;
     int port = -1;
37   int send_socket;
     struct hostent *he;
39
     int i;
41   int *DATA;
     int PACKET_SIZE;
43   int numbytes;

45   unsigned long long CPU_HZ = 0;
     struct timespec req, rem;
47   if (argc != 4) {
```

```
    fprintf(stderr,"usage: %s hostname PACKET_SIZE(bytes) INTERVAL(ms)\n",
        argv[0]);
49    exit(1);
  }
51

53  int interval_ms = atol(argv[3]);
  int interval_s = interval_ms/1000;
55  interval_ms -= interval_s*1000;

57  req.tv_sec = interval_s;
  req.tv_nsec = 1000000 * interval_ms;  // Interval between packet-pairs
59



63  if ((he=gethostbyname(argv[1])) == NULL) {  // get the host info
    perror("gethostbyname");
65    exit(1);
  }
67
  PACKET_SIZE = atol(argv[2]);
69  PACKET_SIZE -= HEADER_SIZE;
//        PACKET_DELAY = atol(argv[2]);
71 //        TEST_TIME = atol(argv[3]);

73  if ((send_socket = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
    perror("socket");
75    exit(1);
  }
77  addr_f.sin_family = AF_INET;      // host byte order
  addr_f.sin_port = htons(PORT_F); // short, network byte order
79  addr_f.sin_addr = *((struct in_addr *)he->h_addr);
  memset(&(addr_f.sin_zero), '\0', 8); // zero the rest of the struct
81

83  //-----------------------------------
  //  read /proc/cpuinfo information
85  //

87  FILE *Proc;
  char buf[300];
89  char buf2[300];

91
  Proc = fopen("/proc/cpuinfo","r");
93  while(fgets(buf,255,Proc)){
    if (strstr(buf,"cpu MHz")){
95      buf[strlen(buf)-1]='\0';
      CPU_HZ = atof(buf+11)*1000000;
97      break;
    }
99  }
```

```
    fclose(Proc);
101 //-----------------------------------

103
    double t;
105 double *dd;
    int sn = 1;
107 DATA = (int *)malloc(PACKET_SIZE);
    memset(DATA, '0',PACKET_SIZE);
109 dd = (double *)(DATA + 1);

111 while(1){
        DATA[0] = sn;
113   t = rdtsc()/(double)CPU_HZ;
        dd[0] = t;
115   if ((numbytes=sendto(send_socket, DATA, PACKET_SIZE, 0, (struct
        sockaddr *)&addr_f, sizeof(struct sockaddr))) == -1) {
      printf("Err: send sn=%d\n",sn*2-1);
117   }

119     DATA[0] = sn+1;
      if ((numbytes=sendto(send_socket, DATA, PACKET_SIZE, 0, (struct
        sockaddr *)&addr_f, sizeof(struct sockaddr))) == -1) {
121     printf("Err: send sn=%d\n",sn*2);
      }
123
      sn+=2;
125   sn %= 10000;
      nanosleep(&req, &rem);
127 }
    free(DATA);
129   exit(0);
}
```

B.5.2 Receiver Source Code

```
/*
2 ** listener.c -- a datagram sockets "server" demo
  */
4
#include <stdio.h>
6 #include <stdlib.h>
  #include <string.h>
8 #include <unistd.h>
  #include <errno.h>
10 #include <string.h>
  #include <sys/time.h>
12 #include <sys/types.h>
  #include <sys/socket.h>
14 #include <netinet/in.h>
  #include <arpa/inet.h>
```

```
16 #include <time.h>

18 #define MYPORT 15000    // the port users will be connecting to

20 #define MAXBUFLEN 100000

22 #define HEADER_SIZE 28

24 //#define CPU_HZ 501125000

26 __inline__ unsigned long long rdtsc(){
       unsigned long long int x;
28         __asm__ volatile (".byte 0x0f, 0x31" : "=A" (x));
               return x;
30 }

32 double getcpuspeed()
   {
34   //  read /proc/cpuinfo information
     //
36   FILE *Proc;
     char buf[300];
38
     Proc = fopen("/proc/cpuinfo","r");
40   while(fgets(buf,255,Proc)){
       if (strstr(buf,"cpu MHz")){
42       buf[strlen(buf)-1]='\0';
         return atof(buf+11)*1000000;
44       break;
       }
46   }
     fclose(Proc);
48   return -1;
   }
50

52 double CPU_HZ;

54 double cap_rtt1[10000];
   double cap_rtt2[10000];
56 double cap_C;
   double cap_RTT;
58 double px[5000];
   double py[5000];
60 double pd[5000];
   double pi[5000];
62
   double px1[5000];
64 double py1[5000];
   double pd1[5000];
66 double pi1[5000];

68 int num = 0;
```

```
70 int main(int argc, char *argv[])
   {
72     int sockfd;
       struct sockaddr_in my_addr;     // my address information
74     struct sockaddr_in their_addr; // connector's address information
       int addr_len, numbytes;
76     char buf[MAXBUFLEN];
         int *DATA;
78       int run = 1;

80       // E. Flydahl: Determine if number of runs is specified as argument

82       if(argc < 3){printf("Usage: %s NUMBEROFRUNS PROBESPERRUN\n", argv[0])
             ; exit(EXIT_FAILURE);}
         int numbruns = atoi(argv[1]);
84       int probesperrun = atoi(argv[2]);

86     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
     perror("socket");
88   exit(1);
       }
90
       my_addr.sin_family = AF_INET;          // host byte order
92     my_addr.sin_port = htons(MYPORT);      // short, network byte order
       my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my IP
94     memset(&(my_addr.sin_zero), '\0', 8); // zero the rest of the struct

96     if (bind(sockfd, (struct sockaddr *)&my_addr,
                sizeof(struct sockaddr)) == -1) {
98   perror("bind");
     exit(1);
100    }

102    addr_len = sizeof(struct sockaddr);

104    cap_C = 0;
       cap_RTT = 1000000000;
106
       CPU_HZ = getcpuspeed();
108    if(CPU_HZ == -1) {perror("Failed to read CPU speed in /proc/cpuinfo");
           exit(EXIT_FAILURE);}

110    struct timeval t_start,t_end;
       int i;
112
       while(run<=numbruns){
114
   if ((numbytes=recvfrom(sockfd,buf, MAXBUFLEN-1, 0,
116          (struct sockaddr *)&their_addr, &addr_len)) == -1) {
       perror("recvfrom");
118    exit(1);
   }
```

```
120   numbytes += HEADER_SIZE;
      DATA = buf;
122
      // E. Flydahl: Set start time
124   if(num==0) {gettimeofday(&t_start,NULL);}

126   i = (DATA[0] - 1)/2;

128   if (DATA[0]%2==1){
          double t = (double)rdtsc()/CPU_HZ;
130       double * dd = DATA + 1;
          cap_rtt1[i] = t - dd[0];
132   } else if (cap_rtt1[i]!=0) {
        double t = (double)rdtsc()/CPU_HZ;
134     double * dd = DATA + 1;
        double disp, rtt_sum, C, pre_rtt_sum, offset_x, offset_y;
136     int trend;
        cap_rtt2[i] = t - dd[0];
138     disp = cap_rtt2[i] - cap_rtt1[i];
        rtt_sum = cap_rtt1[i] + cap_rtt2[i];
140     if (disp>0) { // E. Flydahl: Comment to Capprobes implementation:
            Avoids calculating on the basis of reordered packets
            C = (double)(numbytes * 8) / disp;  // bps
142
          {
144       if (num>0) {
            if (rtt_sum>pre_rtt_sum) trend++;
146         else if (rtt_sum<pre_rtt_sum) trend--;
          } else {
148         trend = 0;
            offset_y = rtt_sum;
150         offset_x = t;
          }
152       pre_rtt_sum = rtt_sum;

154       // E. Flydahl: Comment to Capprobes implementation:
          // The set Omega_S' of (time,delay sum increase/decrease wrt 1st
              sample. packet-pair dispersion, index):
156       px[num] = t - offset_x;
          py[num] = rtt_sum - offset_y;
158       pd[num] = disp;
          pi[num] = num;
160
          // E. Flydahl: Comment to Capprobes implementation:
162       // The set Omega_1 of (time,first packet in pair OWD. packet-pair
              dispersion, index):
          px1[num] = t - offset_x;
164       py1[num] = cap_rtt1[i];
          pd1[num] = disp;
166       pi1[num] = num;
            }
168
            // E. Flydahl: Comment to Capprobes implementation:
```

```
170         // Increment packet-pair index
            num++;
172
            // E. Flydahl: Comment to Capprobes implementation:
174         // If this packet-pair's delay sum less than current minimum ,
                update current minimum delay sum and capacity estimate:
            if (rtt_sum<cap_RTT) {
176       cap_RTT = rtt_sum;
          cap_C = C;
178         }
        }
180
          cap_rtt1[i] = 0;
182       cap_rtt2[i] = 0;

184       // E. Flydahl: Comment to Capprobes implementation:
          // End-of-run procedure:
186       if (num>=probesperrun) {
            int i,j,k,k1,k_final;
188         k = 1;
            k1 = 1;
190
            // E. Flydahl: Comment to Capprobes implementation:
192         // If there is clock skew, find the data points that are part of
                bound curve
            if (trend>=30){  // increasing trend
194           for (i=2;i<num;i++) {
                if (py[i]<(py[k]+(py[k]-py[k-1])*(px[i]-px[k])/(px[k]-px[k-1]))
                    ) { // p[i] is under the line
196               px[k] = px[i];
                  py[k] = py[i];
198               pd[k] = pd[i];
                  pi[k] = pi[i];
200             } else {
                  k++;
202               px[k] = px[i];
                  py[k] = py[i];
204               pd[k] = pd[i];
                  pi[k] = pi[i];
206             }
              }
208           for (i=2;i<num;i++) {
                if (py1[i]<(py1[k1]+(py1[k1]-py1[k1-1])*(px1[i]-px1[k1])/(px1[
                    k1]-px1[k1-1]))) { // p[i] is under the line
210               px1[k1] = px1[i];
                  py1[k1] = py1[i];
212               pd1[k1] = pd1[i];
                  pi1[k1] = pi1[i];
214             } else {
                  k1++;
216               px1[k1] = px1[i];
                  py1[k1] = py1[i];
218               pd1[k1] = pd1[i];
```

```
            pi1[k1] = pi1[i];
          }
        }
      } else if (trend<=-30) { // decreasing trend
        for (i=2;i<num;i++) {
          if (py[i]>(py[k]+(py[k]-py[k-1])*(px[i]-px[k])/(px[k]-px[k-1]))
              ) { // p[i] is above the line
            px[k] = px[i];
            py[k] = py[i];
            pd[k] = pd[i];
            pi[k] = pi[i];
          } else {
            k++;
            px[k] = px[i];
            py[k] = py[i];
            pd[k] = pd[i];
            pi[k] = pi[i];
          }
        }
        for (i=2;i<num;i++) {
          if (py1[i]>(py1[k1]+(py1[k1]-py1[k1-1])*(px1[i]-px1[k1])/(px1[
              k1]-px1[k1-1]))) { // p[i] is above the line
            px1[k1] = px1[i];
            py1[k1] = py1[i];
            pd1[k1] = pd1[i];
            pi1[k1] = pi1[i];
          } else {
            k1++;
            px1[k1] = px1[i];
            py1[k1] = py1[i];
            pd1[k1] = pd1[i];
            pi1[k1] = pi1[i];
          }
        }
      } else trend = 0;

      if (trend!=0) {
        disp = 0;
        k_final = 0;
        // E. Flydahl: Comment to Capprobes implementation:
        // Choose measurements that are on the bound curve of both the
           minimum delay sum set Omega_S' and the  1st packet minimum one
           -way delay set
        for(i=1,j=1;i<=k,j<=k1;i++,j++){
          for (;j<=k1;j++){
            if (pi[i]==pi1[j]){
              disp += pd[i];
              k_final++;
              break;
            }
          }
        }
        if (k_final==0) { disp = pd[0]; }
```

```
268             else {
                  disp /= (double)(k_final);
270             }
                C = (double)(numbytes * 8) / disp;  // bps
272           } else {
                C = cap_C;
274             k_final = -1;
                k = -1;
276             k1 = -1;
              }
278
              if (trend!=0){
280             for(i=1,disp=0;i<=k;i++){
                  disp += pd[i];
282             }
                disp /= (double)(k);
284             C = (double)(numbytes * 8) / disp;  // bps
              }
286
              // E.Flydahl: Get end time
288           gettimeofday(&t_end,NULL);
              printf(" %.2f %.2f", C, t_end.tv_sec - t_start.tv_sec + (double)(
                  t_end.tv_usec-t_start.tv_usec)*0.000001);
290           //fprintf(in,"==> CAP = %3.3lf  k= %d  %d  %d\n",C,k_final,k,k1);

292           cap_C = 0;
              cap_RTT = 1000000000;
294           num = 0;
              run++;
296         }
    }
298 }
  close(sockfd);
300 return 0;
  }
```

B.5.3 Script for Measurements

```
  #!/bin/bash
2
  # E. Flydahl - Script to measure performance of Ad Hoc Probe
4


6 startpsize=100          # Start IP packet size [bytes]
  endpsize=1100          # End packet size [bytes]
8 psizeinc=100            # Number of bytes to increase the packet size between
      measurements
  runs=100             # The number of runs to perform for each psize,numprobes
     combination
10 startnumprobes=10      # Start number of packet-pairs
  endnumprobes=100        # End number of packet-pairs
```

```
12 numprobesinc=10        # Increase in number of probes
   int=1000           # Interval between packet pairs
14 receiverIP="192.168.1.2"
   senderIP="192.168.0.2"
16 pathtosender="/home/labuser/efly/probing_tools/adhocprobe/sender"

18 psize=$startpsize
   numprobes=$startnumprobes
20
   mtudisc=$(ssh $senderIP cat /proc/sys/net/ipv4/ip_no_pmtu_disc)
22 if [ $? -ne 0 ]
   then
24   echo "Could not check if MTU Discovery was disabled at sender" >&2
     exit 1
26 fi
   if [ $mtudisc -ne 1 ]
28 then
     echo "Please disable MTU Discovery at sender" >&2
30   echo "Run \"echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc\" as root" >&2
     exit 1
32 fi

34 # Interrupt routine
   control_c()
36 {
     if [ $1 -eq 0 ]
38   then
       exit 0
40   fi
     echo "Terminating" >&2
42   ssh $2 "kill $1"

44   if [ $? -ne 0 ]
     then
46     echo "Could not kill packet-pair generator at sender" >&2
       exit 1
48   fi
     exit 0
50 }

52 echo "Starting test routine between sender $senderIP and receiver
       $receiverIP" >&2
   echo "IP Packet Size in bytes: $startpsize - $endpsize in steps of
       $psizeinc" >&2
54 echo "Number of probes per measurement: $startnumprobes - $endnumprobes in
       steps of $numprobesinc" >&2
   echo "Number of runs to perform for each psize,numprobes combination: $runs
       " >&2
56
   while [ $psize -le $endpsize ]
58 do

60   # Start packet-pair generator with correct packet size, "return" PID
```

```
    echo "Starting packet generator. Psize: $psize bytes Int: $int" >&2
62  senderpid=$(ssh $(echo $senderIP' nohup '$pathtosender' '$receiverIP' '
        $psize' '$int' > /dev/null 2>&1 < /dev/null & echo $!'))
    if [ $? -ne 0 ]
64  then
      echo "Could not start packet-pair generator at sender" >&2
66    exit 1
    fi
68
    trap "control_c $senderpid $senderIP" SIGINT
70
    echo -e "\a" >&2
72  read -p "Start cross traffic and press a key"
    while [ $numprobes -le $endnumprobes ]
74  do
      echo -n "$psize $numprobes"
76    # Start receiver
      echo "Starting receiver. Numprobes: $numprobes" >&2
78    ./receiver $runs $numprobes
      if [ $? -ne 0 ]
80    then
        echo "Could not start packet-pair receiver" >&2
82      exit 1
      fi
84    numprobes=$[$numprobes+$numprobesinc]
      echo
86  done

88  echo -e "\a" >&2
    read -p "Stop cross traffic and press a key"
90  # Kill packet-pair generator
    echo "Killing packet generator" >&2
92  ssh $senderIP "kill $senderpid"
    if [ $? -ne 0 ]
94  then
      echo "Could not kill packet-pair generator at sender" >&2
96    exit 1
    fi
98  senderpid=0
    psize=$[$psize+$psizeinc]
100 numprobes=$startnumprobes
done
```

The sender must have a SSH-server with the receiver's public key installed for passwordless login in order to make the script work as intended.

B.6 Script for Generating Cross Traffic

```
#!/bin/bash

# E. Flydahl - Script to generate cross traffic

sinkIP="10.0.2.34"
dstport=5002
srcport=5002
overhead=28 # UDP and IP headers
psize=$1
bitrate=$2
type=$3

control_c()
{
  echo "Terminating"
  ssh $2 "kill $1"

  if [ $? -ne 0 ]
  then
    echo "Could not kill sink process" >&2
    exit 1
  fi
  exit 0
}


# Check input
if [ "$psize" == "" ] || [ "$bitrate" == "" ] || [ "$type" == "" ]
then
  echo "Usage: $0 IPPACKETSIZE(bytes) BITRATE(kbps) PERIODIC|POISSON"
  exit 1
fi

# Start sink, "return" PID

sinkpid=$(ssh $(echo $sinkIP' nohup mgen5 event "LISTEN UDP '$dstport'" > /
    dev/null 2>&1 < /dev/null & echo $!'))
if [ $? -ne 0 ]
then
  echo "Could not start packet-pair generator at sender" >&2
  exit 1
fi

trap "control_c $sinkpid $sinkIP" SIGINT

mgen5 event "0.0 ON 1 UDP SRC $srcport DST $sinkIP/$dstport $type [$(echo "
    $bitrate*1000/$psize/8"|bc -l) $[$psize-$overhead]]"
```

The sink must have a SSH-server with the traffic generator's public key installed for password-less login in order to make the script work as intended.

Appendix C

One-way Implementation of Allbest

C.1 Results

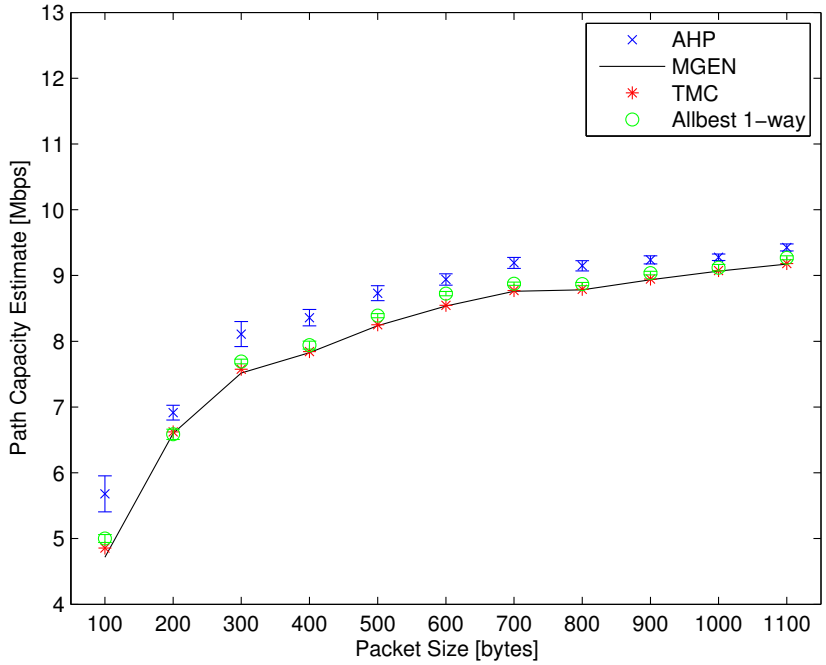
Figure C.1 shows the preliminary results obtained from a one-way implementation of Allbest [14]. The time stamping is performed by the kernel and, judging from the IEEE802.3 results, the problems with context switch delay are no longer an issue. The implementation does not have any clock skew correction, which most likely degrades its performance in this measurements since 100 hundred probes per measurement were used.

C.2 Allbest 1-Way Source Code

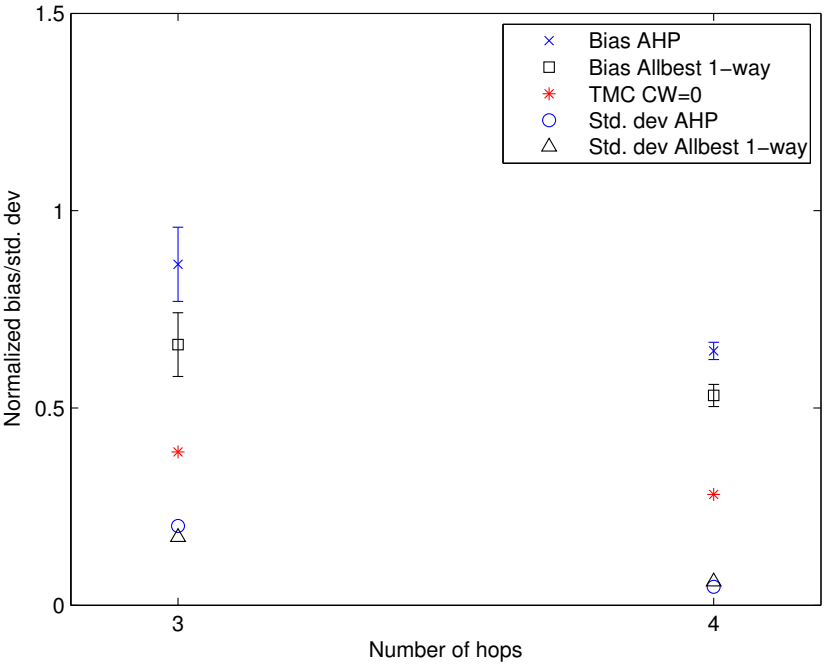
C.2.1 Sender Source Code

```
/*
 * sender.c - One-way delay version of Allbest
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/time.h>
#include <sys/select.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <assert.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <signal.h>
```



(a) IEEE802.3 results



(b) Multi-hop results

Figure C.1: Preliminary results for a one-way version of Allbest compared to Ad Hoc Probe. 95 % Student-t confidence interval.. Packet size: 700 bytes. Number of probes per measurement: 100. Number of measurements: 20 per data point

```
22 #include <time.h>

24 #define PORT_F 15000
   #define PORT_B 15000
26 #define MAXBUFLEN 100000

28 // IP: 20 bytes  UDP: 8 bytes
   #define HEADER_SIZE 28
30
   struct sockaddr_in addr_f; // connector's address information
32
   int main(int argc, char *argv[]) {
34   char *addr = NULL;
     int port = -1;
36   int send_socket;
     struct hostent *he;
38
     int i;
40   int *DATA;
     int PACKET_SIZE;
42   int numbytes;

44   unsigned long long CPU_HZ = 0;
     struct timespec req, rem;
46   if (argc != 4) {
       fprintf(stderr,"usage: %s hostname PACKET_SIZE(bytes) INTERVAL(ms)\n",
           argv[0]);
48     exit(1);
     }
50
     int interval_ms = atol(argv[3]);
52   int interval_s = interval_ms/1000;
     interval_ms -= interval_s*1000;
54
     req.tv_sec = interval_s;
56   req.tv_nsec = 1000000 * interval_ms;  // Interval between packet-pairs

58   if ((he=gethostbyname(argv[1])) == NULL) {  // get the host info
       perror("gethostbyname");
60     exit(1);
     }
62
    PACKET_SIZE = atol(argv[2]);
64  PACKET_SIZE -= HEADER_SIZE;
   //         PACKET_DELAY = atol(argv[2]);
66 //         TEST_TIME = atol(argv[3]);

68   if ((send_socket = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
       perror("socket");
70     exit(1);
     }
72   addr_f.sin_family = AF_INET;      // host byte order
     addr_f.sin_port = htons(PORT_F); // short, network byte order
```

```
74    addr_f.sin_addr = *((struct in_addr *)he->h_addr);
      memset(&(addr_f.sin_zero), '\0', 8); // zero the rest of the struct
76
      double t;
78    double *dd;
      struct timeval tsnd;
80    int sn = 1;
      DATA = (int *)malloc(PACKET_SIZE);
82    memset(DATA, '0',PACKET_SIZE);
      dd = (double *)(DATA + 1);
84
      while(1){
86          DATA[0] = sn;

88    gettimeofday(&tsnd, NULL);
      t = (double)tsnd.tv_sec + ((double)tsnd.tv_usec)*0.000001;
90          dd[0] = t;
        if ((numbytes=sendto(send_socket, DATA, PACKET_SIZE, 0, (struct
            sockaddr *)&addr_f, sizeof(struct sockaddr))) == -1) {
92        printf("Err: send sn=%d\n",sn*2-1);
        }
94
            gettimeofday(&tsnd, NULL);
96    t = tsnd.tv_sec+(double)tsnd.tv_usec*0.000001;
            DATA[0] = sn+1;
98      if ((numbytes=sendto(send_socket, DATA, PACKET_SIZE, 0, (struct
            sockaddr *)&addr_f, sizeof(struct sockaddr))) == -1) {
          printf("Err: send sn=%d\n",sn*2);
100     }

102     sn+=2;
        sn %= 10000;
104     nanosleep(&req, &rem);
      }
106   free(DATA);
        exit(0);
108 }
```

C.2.2 Receiver Source Code

```
/*
2 ** receiver.c One-way delay version of Allbest
  */
4
  #include <stdio.h>
6 #include <stdlib.h>
  #include <string.h>
8 #include <unistd.h>
  #include <errno.h>
10 #include <string.h>
  #include <sys/time.h>
```

```
12 #include <sys/types.h>
   #include <sys/socket.h>
14 #include <netinet/in.h>
   #include <arpa/inet.h>
16 #include <time.h>

18 #define MYPORT 15000    // the port users will be connecting to

20 #define MAXBUFLEN 100000

22 #define HEADER_SIZE 28

24 typedef struct{
     double t;
26   double owd;
   }point;
28
   int main(int argc, char *argv[])
30     {
           int sockfd;
32         struct sockaddr_in my_addr;    // my address information
           struct sockaddr_in their_addr; // connector's address information
34
     // E. Flydahl: Determine if number of runs is specified as argument
36   if(argc < 3){printf("Usage: %s NUMBEROFRUNS PROBESPERRUN\n", argv[0]);
       exit(EXIT_FAILURE);}
     int numbruns = atoi(argv[1]);
38   int probesperrun = atoi(argv[2]);

40   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
       perror("socket");
42     exit(1);
     }
44
           // E.Flydahl: Enable Kernel timestamping to avoid inaccuracy due to
                packet-pair compression or inflation caused by context switch
46   int timestampOn = 1;
     int error;
48   if ( (error=setsockopt(sockfd, SOL_SOCKET, SO_TIMESTAMP, (void *) &
       timestampOn, sizeof(timestampOn))) == -1)
     {
50     perror("setsockopt(SO_TIMESTAMP) failed");
       exit(EXIT_FAILURE);
52   }

54         my_addr.sin_family = AF_INET;          // host byte order
           my_addr.sin_port = htons(MYPORT);      // short, network byte order
56         my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my
               IP
           memset(&(my_addr.sin_zero), '\0', 8); // zero the rest of the
               struct
58
```

```
        if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct
            sockaddr)) == -1) {
            perror("bind");
            exit(1);
        }

// E. Flydahl: Declare and initialize buffers for reception of data and
    timestamp
char rcvbuf[MAXBUFLEN];
char msg_control[MAXBUFLEN];
struct iovec msg_iov = {&rcvbuf, MAXBUFLEN};
memset(rcvbuf, '0', MAXBUFLEN);
memset(msg_control, '0', MAXBUFLEN);
struct msghdr msg={
  &their_addr,
   sizeof(their_addr),
  &msg_iov, 1,
  &msg_control,
  MAXBUFLEN,
   0
};

// E. Flydahl: Variables for main loop
int *DATA;
int run = 1;
int num1 = 0;
int num2 = 0;
int seq=0;
int lastseq1=0;
int numbytes=0;
double t=0;
double* dd;
struct cmsghdr* cmsg;
struct timeval* trcv;
double cap_C=0;
point p1[10000]; // E. Flydahl: Array to save packet 1 data
point p2[10000]; // E. Flydahl: Array to save packet 2 data
struct timeval t_start,t_end;


while(run<=numbruns){
    // E. Flydahl: Set start time
    if(num1==0) {gettimeofday(&t_start,NULL);}

    if ((numbytes=recvmsg(sockfd, &msg, 0)) == -1) {
      perror("recvfrom");
      exit(1);
    }
    numbytes += HEADER_SIZE;

    // E. Flydahl: Read timestamp
    for(cmsg=CMSG_FIRSTHDR(&msg); cmsg!=NULL; cmsg=CMSG_NXTHDR(&msg, cmsg
        ))
```

```
        {
110       if(cmsg->cmsg_level==SOL_SOCKET && cmsg->cmsg_type==SO_TIMESTAMP)
          {
112     trcv = (struct timeval*)CMSG_DATA(cmsg);
        break;
114       }
        }
116     if(cmsg==NULL)
        {
118       perror("SO_TIMESTAMP not enabled or control buffer to small or I/O
              error");
          exit(EXIT_FAILURE);
120     }

122     t = (double)trcv->tv_usec*0.000001+trcv->tv_sec;

124     DATA = (int*)rcvbuf;
        dd = (double *)(DATA + 1);
126     seq = DATA[0];

128
      // E. Flydahl: Register OWD sums and increment packet counter
130   if (seq%2==1){
        p1[num1].owd = t - dd[0];
132     p1[num1].t = t;
        num1++;
134     lastseq1 = seq;
      } else if (seq%2 == 0 && seq-lastseq1==1) { // E. Flydahl: Only collect
          2nd packet if first packet of the pair has been received as last
          packet
136     p2[num2].owd = t - dd[0];
        p2[num2].t = t;
138     num2++;
      }
140
      // E.Flydahl: End-of-run procedure:
142   if (num2==probesperrun) {

144
        double min_owd1=10000000;
146     for(int k=0;k<num1;k++)
        {
148       if(p1[k].owd<min_owd1)
          {
150         min_owd1=p1[k].owd;
          }
152     }

154     double min_owd2=10000000;
        for(int k=0;k<num2;k++)
156     {
          if(p2[k].owd<min_owd2)
158       {
```

```
            min_owd2=p2[k].owd;
160         }
        }
162
        cap_C = numbytes*8/(min_owd2-min_owd1); // E.Flydahl: Capacity
            estimate
164
        // E.Flydahl: Get end time
166     gettimeofday(&t_end,NULL);

168     // E. Flydahl: Output
        printf(" %.2f %.2f", cap_C, t_end.tv_sec - t_start.tv_sec + (double)(
            t_end.tv_usec-t_start.tv_usec)*0.000001);
170


172
        // E.Flydahl: Reset for new run
174     cap_C=0;
        num1=0;
176     num2=0;
        run++;
178   }
  }
180   close(sockfd);

182       return 0;
    }
```