Adaptive Source Routing and Speculative Execution for Multi-homed Wireless Clients in Preclinical Medical Care

Bachelor Thesis

Oliver Michel

Medical University of Vienna
Center for Medical Statistics, Informatics and Intelligent Systems
Section for Medical Information Management and Imaging

University of Vienna
Faculty of Computer Science
Research Group "Future Communication"

oliver.michel@univie.ac.at

Vienna, November 2012

Co-advised by

Univ.-Prof. Dr. Kurt Tutschku
Dipl.-Ing. Georg Fischer
Dipl.-Inform. David Stezenbach

Abstract

While telemedicine is becoming a decisive part of modern health-care, it is not yet widely deployed in preclinical care, such as emergency medical services (EMS). This is mainly due to deficient and constrained mobile connectivity. Since a high degree of reliability and resilience is mandatory in this field of application, there is a need for enhanced and more failure-resistant mobile connectivity. We propose a transparently usable system enhancing mobile data access by using multi-homed wireless clients. Dynamic path selection techniques are used to improve end-to-end performance compared to a single wireless connection. Additionally, packet duplication can dramatically decrease latency and improve connection stability with potentially low overhead. Using off-line trace analysis we investigate reasonable algorithms for dynamic path selection and using a prototype implementation we show how mobile connectivity can be enhanced in different traffic-scenarios while considering the special requirements in a health-care setting.

Statement of Authenticity

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. I hereby declare that I have not submitted this material, either in whole or in part, for a degree at this or any other institution.

Oliver Michel, Vienna, November 2012

Contents

1 Introduction

While telemedicine is becoming a decisive part of modern health-care, it is not yet widely deployed in preclinical care, such as emergency medical services (EMS). This is mainly due to deficient and constrained mobile connectivity. Since a high degree of reliability and resilience is mandatory in this field of application, there is a need for enhanced and more failure-resistant mobile connectivity.

Based on previous results [35, 52], we propose a system designed to enhance mobile data access from multi-homed wireless clients. Given that a mobile client is connected to the Internet via multiple wireless interfaces, we propose an approach where the best estimated path is used as the primary data-path while continuous end-to-end probing is performed on all available paths in order to react to dynamics in the network as quickly as possible. As a consequence, in this setting multi-homing is used to improve both resilience and performance.

We use dynamic and intelligent path selection and path combination techniques between a mobile client and a stationary gateway in the Internet in order to improve end-to-end performance compared to a single wireless connection. Furthermore, depending on the available bandwidth, packet duplication can dramatically decrease latency, routing-related delay and improve connection stability. In the case of insufficient bandwidth, legacy concurrentmulti-path strategies can be used to increase the overall available bandwidth.

The proposed setup is transparently usable by any kind of application. Also, this strategy can be used in multiple areas of high-performance and high-availability mobile computing. However, in this work we focus on a deployment in health-care and especially emergency care where a secure and reliable connection between EMS staff and physicians in a hospital can improve a patient's outcome. The basic setup of our system is shown in Figure 1.

Using off-line trace analysis we investigate reasonable algorithms for dynamic path selection. Using a prototype implementation we show how mobile connectivity can be enhanced in different traffic-scenarios while considering the special requirements in a health-care setting.

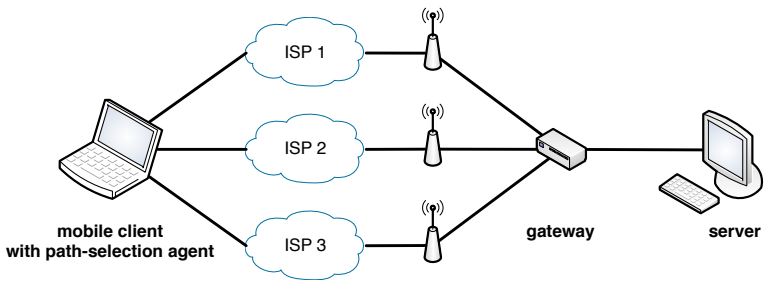


Figure 1: System Overview

The remainder of this thesis is structured as follows. First, related work from the field of Source Routing, Multi-Path Routing and Path Performance Estimation is presented and discussed in Section 2. Section 3 describes different path selection strategies and algorithms and elaborates on a parameter study we conducted in order to identify well-suited algorithms for different scenarios. After a detailed description of the implemented prototype, the system

is evaluated by its different features such as Handover, Path Selection, as well as Multi-Path policies. We are concluding this work with a discussion of possible areas of deployment in the health-care field.

2 Related Work

2.1 Source Routing

Although the work described in this thesis cannot be considered as a source-routing system since no multi-hop routing is performed but rather "good" paths are selected from a set of known routes. Still, we see this work somewhat related to the topic of source routing since we move control over the route selection away from the network's core and embed this functionality directly into the client in the form of a path-selection agent. Additionally our results about path selection and active probing are perfectly suited to be used in source routing protocols in order to improve connection performance, resilience and adaptability. At first an early citation of the source-routing term and it's concept is presented before more recent work on source-controlled routing is discussed.

Carl Sunshine described the concept of source-routing in an early article in 1977 [50] as a novel approach to internet routing where "the source of a packet specifies the complete internet route". Modern source-routing approaches usually define the route information in the packet header independently of a packet's source address. However, the key concept of the elimination of routing instead of simple forwarding at intermediate routers was already covered in Sunshine's elaboration. Beside the advantages of this approach in the form of reduction of complexity within the network, one of the main disadvantages of source routing i.e. the need for knowledge about the network's topology at the source is brought up. Building on a work by Farber and Vittal [15], he proposes a way for constructing return-paths for individual packets by appending the address of each traversed router to a return-path specification.

In order to perform path-selection not only between different interfaces at the client (like it is proposed in this thesis), but also in the context of an internet routing protocol, information about available paths, i.e. the topology of the network (abstracted and simplified to some extent) must be known to the selection agent. This means that also source-routing is dependent on a control-plane in order to retrieve and exchange information about available routes. There are several proposals for protocols enabling source-routing, such as MIRO [54], NIRA [55] and LISP [16]. Godfrey et al. presented a system called *Pathlet Routing* [18], a non-IP based routing protocol enabling source- and multipath routing while still allowing the definition of arbitrary types of policies that are currently available in BGP. In fact *Pathlet Routing* is able to fully emulate protocols like MIRO, NIRA, LISP, as well as BGP through its powerful policy-based concept. In this protocol, networks advertise so called pathlets (fragments of paths) which sources can concatenate into end-to-end source routes. While one-hop pathlets build the edges of the network graph, the vertices are called virtual nodes (vnodes). The combination of pathlets and virtual node build a routing overlay

which autonomous systems can advertise. Furthermore ingress and egress vnodes with local transit policies are defined in order to give an ISP control over traffic transiting through its autonomous system.

2.2 Multipath Routing

Multipath Routing approaches have gained interest over the past years as several systems (often closely related to source routing) have been proposed and deployed enabling multipath routing not only in multi-homing settings. Multipath Routing is primarily used for three different purposes. First, to improve reliability of a network path by allowing to adaptively switch paths in case of path failure or decreased path performance or even send copies of the same packet on different paths while only using the packet which arrives first and discarding any later arrivals. Second, to aggregate multiple paths in order to improve end to end capacity (concurrent multipath transmission), and third to perform load-balancing among multiple paths or resources.

Multipath Routing in settings where it is proposed for improved resilience and fault-tolerance usually give users control over the service they receive by choosing between multiple paths between a source and destination (cp. Section 2.1). Several systems for this application have been implemented and evaluated [18, 54, 56, 17, 37] for both intra- and inter-domain routing. However, most of these protocols do not provide guarantees on path diversity. That means that paths may not be disjunct enough to provide fault-tolerance and reliability in the case of link failure. *Yet Another Multipath Routing Protocol (YAMR)* [17] has been proposed to mitigate this problem. YAMR provably constructs sets of paths that are resilient to inter-domain link failures. This greatly improves reliability. Although, YAMR maintains more paths than BGB, it requires significantly less control traffic. Furthermore, although implemented in an overlay, Resilient Overlay Networks [3] make use of multipath routing in order to mitigate link failures in a highly adaptive manner based on end-to-end feedback and active probing. Another way of using a variety of available paths, as it may become customary in a future Internet, was recently proposed by Vulimiri et al. [52]. Although this paper primarily elaborates on the general idea and feasibility of duplicating packets or jobs and waiting for the first one to arrive or finish (ignoring later arriving packets), experiments in overlay routing were conducted showing that the latency tail can be radically cut using this simple but powerful technique to mitigate strugglers.

When using multipath techniques to increase bandwidth and therefore transmission performance, resources are virtually pooled forming a single resource or path. The underlying concept of this approach (independent of concurrent or selective transmission) is defined as Transport Virtualization (TV) [58, 57]. This approach comes with several challenges such as the need for packet-reordering at the destination which is inevitable as different paths most certainly do not transmit at the same speed and most like do not have the same length and delay. Concrete approaches allowing concurrent multipath transmission include [21, 20, 24]. In particular SCTP [38] offers an alternative for TCP supporting multistreaming which eliminates the need for concurrent, uncoordinated TCP sessions (e.g. in the case of HTTP) and multihoming enabling SCTP clients and servers to bind to multiple IP addresses and by that

use possibly diverse network resources in between for both better resilience and increased performance. Although multistreaming SCTP is able to reduce the occurance of Head of Line (HOL) blocking (i.e. an interuption in the TCP byte stream due to a lost packet and TCP's strict in-order delivery guarantees although the upcoming information could already be displayed by the browser). This problem nowadays is partly addressed by using parallel TCP connections which however implies other problems including additional time for connection establishment and increased server load.

2.3 Path Performance Estimation

Packet delay, loss and available bandwidth are fundamental performance metrics of network paths for service providers to meet SLA, as well as for protocol designers. Like the system in this thesis, many proposed systems and protocols in the networking community rely up to some degree on active measurements to predict the quality of some option like a path or network a packet should traverse. While latency and packet-loss is relatively straighforward to measure, the latter measure, bandwidth, is somewhat more complex.

Latency and packet loss (which we consider as the worst possible latency) is a powerful indicator for the overall status of a networking system. Many problems in packet networks such as congestion, routing misconfigurations or simply link failures can be detected through increased latency within the specific part of the network. While continous latency monitoring is relatively easy to conduct by active probing, there are a few caveats which should be taken into consideration. It is not completely clear how many probes need to be sent into the network to get accurate results without congesting the network with probes. Also, Internet path delay measurement is usually performed by measuring round trip times. As routing asymmetry is widly prevalent in the Internet, round trip times do not provide information about one-way delay. As a consequence, inaccurate measurement defeats the argument for performing networking measurement at all as wrong decisions may be made based on incorrect assumptions.

Effort has been taken to answer these questions in order to improve the correctness of network measurements and the performance and reliability of systems using these measurement results. [7] and [10] elaborate on reasonable approaches for latency measurement and probing schemes to send an amount of probing packets such that the probing performance represents the actual performance of regular traffic and the probes do not degrade the network's overall performance. [10] proposes a probing scheme measuring primarily the high quantiles of delay over period not longer than 30 minutes. They state that information about the latency tail is more valuable than measures of central tendency. Also due to frequent path changes in the Internet, measurements running significantly longer than the proposed duration may become extremely inaccurate as in essence distributions of different measurements simply get merged. While these broad guidelines may help to understand network probing and get an idea how to perform measurements optimally, the theoretical foundation of PASTA [53], which is applicable to numerous stochastic systems, should be considered when monitoring network characteristics. PASTA stands for Poisson Arrivals See Time Averages and states that if observations of a system made at times following a Poisson process, they converge

to the *true* value when averaged over time. Several systems [40, 41, 49] send out probes at times following a Poisson process implementing the PASTA principle. A work from 2006 [6] re-explores the applicability of PASTA to internet delay measurements to obtain unbiased path performance estimates. According to this work, the role of PASTA in network measurement is becoming less important because other sampling techniques also provide unbiased sampling. Additionally, PASTA requires different conditions to be satisfied in order to meet its guarantees, which must not apply in the case of active probing. We see that also probing for delay and loss can become a non-trivial matter when needing accurate results without biasing measurements by the probing itself.

As mentioned previously, another measure of interest for numerous systems is available bandwidth. Bandwidth information can be in particular useful when not optimizing for a real-time, low-latency setting but rather improving e.g. picture quality in a (non-live) video stream. First, available bandwidth must be distinguished from capacity. Available bandwidth can be seen as space capacity of a link at a certain point in time. This definition however is inaccurate as a link is either transmitting at full capacity or is idling at a time. According to this, available bandwidth could only be 0 or 1. Therefore, available bandwidth can only be seen in a meaningful way as an integral over time. [43] defines the average utilization $\bar{u}(t-\tau, t)$ for a time period $(t-\tau, t)$ as

$$\bar{u}(t-\tau, t) = \frac{1}{\tau} \int_{t-\tau}^{t} u(x)dx. \tag{1}$$

On the contrary, the capacity is defined as the maximum possible IP layer transfer rate of a link or path. Note that the capacity of a path consequently is the minimum of all link capacities. Different bandwidth-estimation techniques have been proposed in literature. Among those are *Packet Pair Probing*, *Self-Loading Periodic Streams (SLoPS)*, and *Trains of Packet Pairs* most widely used. Packet pair probing [25, 30, 9] sends multiple packet pairs back to back where each packet pair consists of two packets of the same size. The dispersion of a packet pair at a certain link can then be used to compute the estimated path capacity. This method is only able to measure capacity as it assumes no other traffic on a link. Self-Loading Periodic Streams [26] is a more recent technique for measuring end-to-end available bandwidth by sending a stream of equal-sized packets while observing variations in the packet's one-way delay. One-way delays would increase if packets queue up at a tight link meaning that the stream rate is higher than the available bandwidth. If no increasing latency is observed, packets go through the network without causing queueing. While this technique may for a period of time completely saturate the path, it is among the most accurate methods currently available. Trains of Packet Pairs is an approach proposed by Melander et al. in 2002 [33], also measuring available bandwidth of a network path by sending packet pairs at increasing rates. Processing of the dispersion of packet delays may then reveal an estimate for the available bandwidth.

As wide as the range of measurement techniques, is the variety of available measurement tools. Guerra et al. conducted a measurement study [19] comparing different tools. The performance of the particular tools highly depends on other network characteristics such as latency, cross traffic and packet loss rates. However, for certain scenarios reasonable tools

can be identified. For most scenarios ,but not in general, Spruce [49] (Trains of Packet Pairs (Probe Gap Model)), Pathchirp [44] (self-loading packet chirps) and Pathload [27] (SLopS) seem to be the best applicable tools.

3 Path Selection

Since this work is based on the idea of using multiple network connections simultaneously to improve resilience and performance, there is a need for a (to some extent) intelligent way of identifying well-performing connections or paths, respectively.

In order to collect data to apply various path selection algorithms we use the concept of continuous probing. As the probe packets can be very small (40 Bytes additional to Layer 2 PDU), probing multiple paths continously must not produce significant overhead on the wire. Once a path is selected for actual data transmission, there is no need for designated probe packets anymore since data packets may function as probe packets yielding a higher probing resolution and eliminating overhead. Once the frequency of actual data transmission over a selected path drops below the probing rate, probe packets are mixed into the data flow.

Based on probing data, algorithms can identify well-performing paths optimizing for delay, latency or loss. While in a real-world application this is done at runtime, we use post-hoc trace analysis based on previously measured data, as well as randomly sampled values for the following discussion.

3.1 Problem Definition

For the implementation of the different path selection techniques it seemed reasonable to formalize the underlying problem of path selection based on latency data.

All selection methods are round-based, such that every current inter-probe delay, which we sample from a memory-less exponential distribution causes the beginning of a new probing- and selection-round. Once the set of measured latencies for one round is fully or partly collected, an estimation algorithm identifies the best paths.

Given that r denotes the current round number and n different paths are available for data transmission, the vector of latencies at a point of time or round respectively is given by

$$\overrightarrow{l_r} = (l_{r,i}, l_{r,i+1}, ..., l_{r,n})\,. \tag{2}$$

Furthermore, for some strategies we introduce the concept of a sliding window, which is visible to the selection algorithm. However, additionally to this window, algorithms can keep additional state internally which is necessary for certain supplementary computation. The sliding window is defined by the sliding window size ω such that it can be represented as a $n \times \omega$ matrix

$$L_r = \begin{pmatrix} \overrightarrow{l_{r-\omega}} \\ \vdots \\ \overrightarrow{l_{r-1}} \\ \overrightarrow{l_r} \end{pmatrix}. \tag{3}$$

On this matrix after every round a selection function

$$s(L, \mu) = (p_0, p_1, ..., p_\mu), \quad \mu \leq n \tag{4}$$

is applied which returns a set of path-indices depending on the number of desired "best" paths μ. μ is called the multipath factor as it denotes what number of paths is considered reasonable for concurrent transmission. μ is to be chosen according to the desired level of resilience, keeping in mind that packet duplication over these μ paths might cause traffic overhead.

Since the actual best path in terms of latency is not yet known at the time of decision but one round later, we can subsequently rate the correctness of the chosen path selection method. A path selection for the case $\mu = 1$ was correct if the equation

$$\min(\overrightarrow{l_{r+1}}) = l_{r+1,s_r(L,\mu)} \tag{5}$$

is true for a given round r.

3.2 Algorithm Evaluation Methodology

We now present five different strategies for the path selection problem. These strategies can be divided into simple heuristics (SMA, TPMA, EWMA) and learning-based approaches (ϵ-greedy, UCB). While we do an extensive evaluation of the latency-based algorithms, we leave the evaluation of the two learning-based approaches and possibly several more for future work.

For reasons of complexity the following parameter study is limited to network latency as the performance metric which is to be minimized by the algorithm. For the analysis, two different real internet-latency data sets, as well as two groups of randomly sampled data-sets are used.

The real data sets were collected over several days using an overlay deployed on PlanetLab Central [48, 42], as well as on private MyPLC instances obtained through the GENI[1] Control Framework [12, 11]. The measurement setup was made from a tripartide graph with sending nodes (sources), forwarding nodes and receiving nodes (destinations). Two different data sets were collected. The first consisted of senders at the U.S. west-coast, intermediates mainly in the mid-west and other central regions of the states and receiving nodes at the east coast. We believe this topology is realistic in the context of internet-like environments. The other set used the same setup but the nodes were chosen randomly not representing any real geographical means. A measurement application was implemented in C++ sending

[1] Global Environment for Network Innovation

out probe packets with an inter-probe delay sampled from an exponential distribution with $\lambda = 1$. The packet's round trip time on layer 4 was measured and logged for each available path. Both datasets consist of approx. 12 mio. latency values spread over five different paths.

Additionally, the other two datasets were sampled from a Gaussian Distribution and from a Type-1 Pareto Distribution, respectively. The distribution's means were between 75 and 100 ms and the standard deviations between 2 and 6 ms. For each type of distribution 100 data-sets with each five paths were generated with the parameters uniformly sampled from the given intervals. While the Normal Distribution takes the mean and standard deviation as parameters directly, in the case of the Pareto Distribution the position parameter k and shape parameters α were determined solving

$$\mu = \frac{k\alpha}{\alpha - 1} \quad \wedge \quad \sigma = \frac{k\sqrt{\frac{\alpha}{\alpha-2}}}{\alpha - 1} \tag{6}$$

for k and α, repectively.

For the simple latency-based heuristics we investigate the mean and tail of the actually achieved latency distributions, the frequency of switching between paths, as well as the fraction of correct decisions depending on the strategy's parameters and underlying latency distributions. For the learning-based approaches we just elaborate very briefly on different parameters but rather present these techniques in general and in comparison to the simple heuristics.

3.3 Latency-Based Heuristics

We present three simple heuristics which we consider possible approximations to the path-selection problem. All heuristics are intended to minimize latency in the upcoming round by determining a good path based on previously seen latencies and then sending the actual payload speculatively over the selected μ paths. In this context, three parameters are critical for the selection. The first two parameters, namely the distribution's mean and variance are obvious as they give an elementary idea of the position and form of the distribution. While both mean and variance should be minimized, there is a third parameter of interest for the overall performance of the system. This parameter is the frequency of switching between paths. Because TCP may encounter out-of-order delivery or packet-loss and corresponding delays when the path-selection agent switches from one path to another, this measure needs to be minimized as well[2]. Therefore, we can see Internet path-selection as a multi-dimensional optimization problem.

[2]For reasons of space and complexity we do not investigate more appropriate metrics in this context such as buffer occupancy or average waiting time for packet reassembly but rather see the amount of path changes as an estimate for the severity of problems path switching comes with.

3.3.1 Simple Moving Average (SMA)

A Simple Moving Average with a window-size ω is calculated such that the last ω observations are weighed equally, namely by ω^{-1}. Therefore, recursively defined, a SMA takes the form

$$y_{t+1} = y_t - \frac{x_t - \omega}{\omega} + \frac{x_t}{\omega}. \tag{7}$$

The SMA is very easy to compute and tends to stabilize path-selection and converge to a reliable path over a series of seen values. It considers recent observations in the same way as older observations. In the case of a sudden shift in latency performance, the algorithm apparently reacts relatively slow. However, it turns out that in general the Simple Moving Average is performing surprisingly well especially with small window-sizes. When not considering the rate of path-changes, a window-size of only $\omega = 2$ yields best and extremely close to optimal results. Additionally to this, we refer to a SMA with a window-size of $\omega = 1$ as the *Previous Best Path Heuristic*. Although the performance of this heuristic in terms of actually experienced latency after applying it is very good, the algorithms tends to oscillate between two best paths. As a consequence, this heuristic is not taken into account for the remainder of this thesis.

3.3.2 Two-period Moving Average (TPMA)

Our findings about the satisfactory performance of the SMA-algorithm described above, leads to the idea of a different model of weighting for average-based heuristics. Obviously, the very recent results have the biggest impact on future performance. This means that recent observations may be weighted heavier while still considering older observations in order to reduce path-changes and therefore improve connection stability. While concepts like linearly and exponentially weighted averages exist, we tried the very simple idea of just weighting the recent two observations more than the following $\omega - 2$ latencies. Putting the half weight on the two recent measurements, the averages takes the form

$$y_{t+1} = \frac{x_t}{4} + \frac{x_{t-1}}{4} + \sum_{k=0}^{\omega-2} \frac{x_{t-k-2}}{2(\omega-2)}. \tag{8}$$

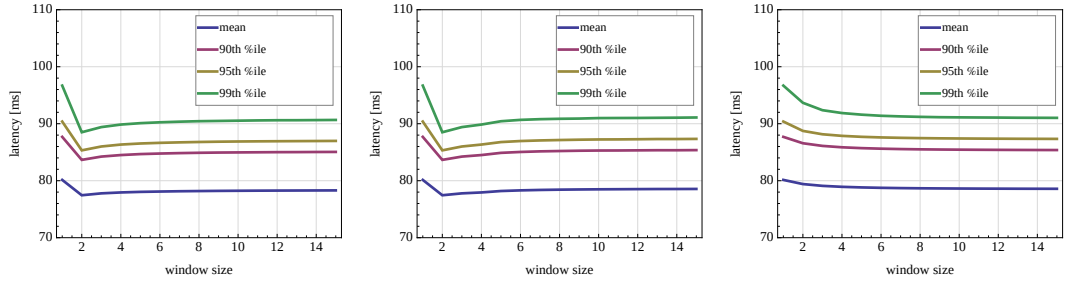
We refer to this type of moving average as the *Two Period Moving Average (TPMA)*. Although the shape of the sliding window (heavier weight on recent observations and faster recovery from extreme values) seems by intuition to be reasonable for Internet traffic (similar to the EWMA- algorithm), the algorithm performs extremely similarly to the Simple Moving Average.

3.3.3 Exponentially Weighted Moving Average (EWMA)

In other literature (e.g. [3]), selection based on a numeric objective such as latency is often performed using exponentially weighted moving average schemes (EWMA) [36, 2, 13]. An Exponentially Weighted Moving Average takes the form

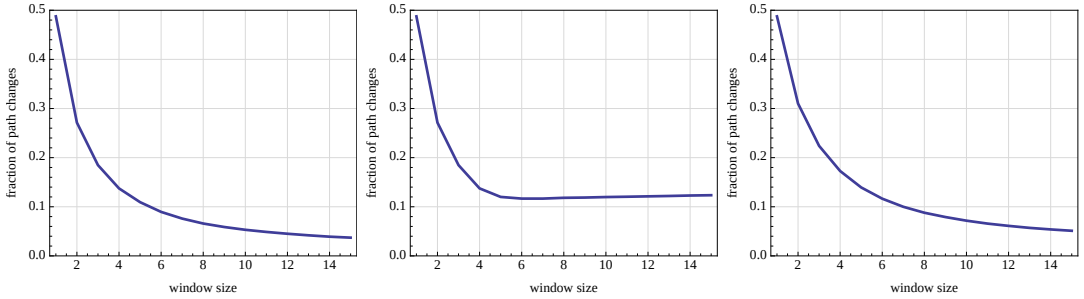
$$y_{t+1} = \alpha \cdot y_t + (1 - \alpha) \cdot x_t, \tag{9}$$

where α is the function's shape parameter. In regard to the number of recent observations to be considered (i.e. the window length ω), α can be calculated as $\alpha = \frac{2}{(\omega+1)}$.



(a) SMA (b) TPMA (c) EWMA

Figure 2: Latency results for different window-sizes $\omega = 1..15$ for normally distributed data



(a) SMA (b) TPMA (c) EWMA

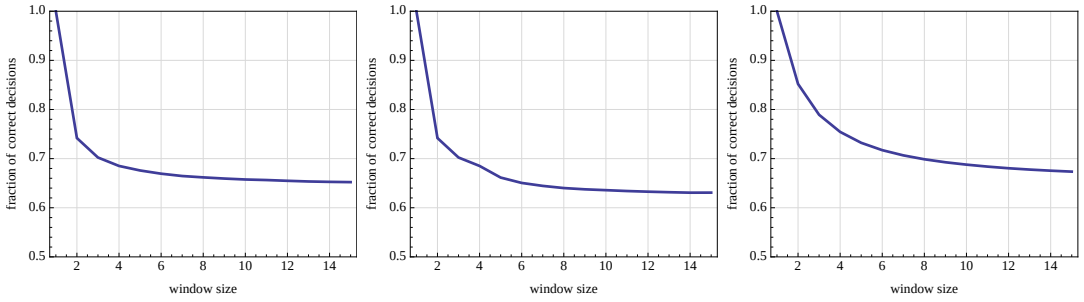
Figure 3: Path changes for different window-sizes $\omega = 1..15$ for normally distributed data

3.3.4 Evaluation

In order to furthermore investigate the differences between the different path-selection techniques, we tried to find out what window-size (which is the primary parameter) is favorable for the simple selection heuristics. In order to do this the shape of the achieved latency distribution, the amount of path switching, as well as the fraction of correct path decisions were studied.

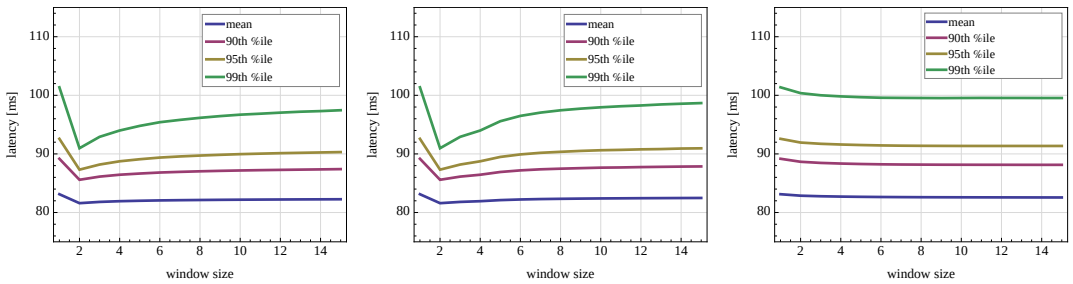
For the artificial datasets, figure 2 and 5 show that the very small window size of $\omega = 2$ performs best in terms of achieved latency. It is furthermore obvious that a window size of $\omega = 1$ which represents the earlier mentioned *previous best path*-heuristic is inferior to a sliding average. As shown in Figure 3 and 6 however, a small window size intuitively implicates more switching between paths as excursive observations are heavily weighted. The behavior of SMA, TPMA and EWMA values for a latency example is shown in Figure 8 and Figure 9, respectively.

We state therefore that a window-size between 4 and 6 is a reasonable compromise between pure latency outcome and switching frequency. Also the trivial SMA heuristic performs



(a) SMA (b) TPMA (c) EWMA

Figure 4: Fraction of correct path decisions for different window-sizes $\omega = 1..15$ for normally distributed data



(a) SMA (b) TPMA (c) EWMA

Figure 5: Latency results for different window-sizes $\omega = 1..15$ for Pareto-distributed data

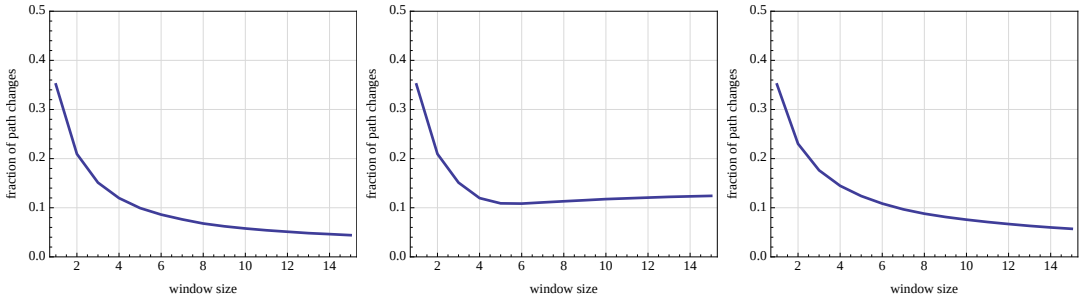
better than the very slightly more complex variants TPMA and EWMA.

3.4 Learning-based Approaches and the Multi-armed Bandit

In the past years, the relatively old model of multi-armed bandits (MAB) [45] gained increasing attention in machine-learning literature [51, 31, 32]. As the multi-armed bandit problem formalizes the problem of reward-optimization with unknown reward distributions (in our case the distribution of performance metrics such as latency, bandwidth or loss), this model can easily be applied to the Internet path selection problem.

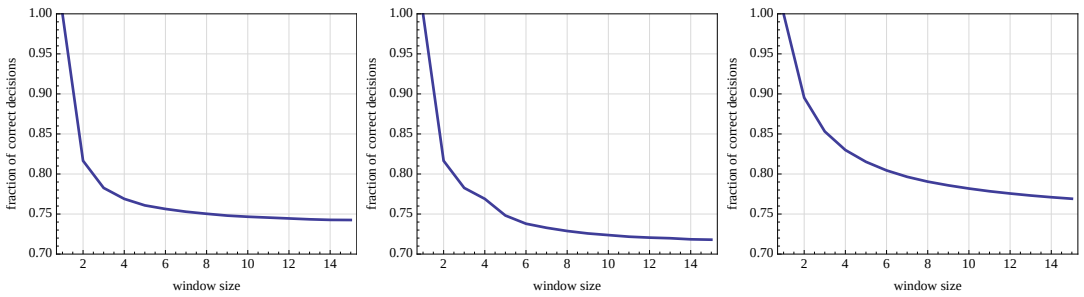
The multi-armed bandit basically describes a slot machine with multiple levers where the probability of a win (i.e. the reward distribution) when pulling a specific lever is not known by the player. Through repeated trials however (exploitation phase), the player might identify levers with a better outcome, eventually gaining knowledge of the underlying reward distributions. One of the key differences of the MAB-based approaches is that it is not needed to probe all available paths simultaneously to get a first estimate about the best path. This class of algorithms rather probe only one path per round (i.e. pulls a lever).

Formally the multi-armed or K-armed bandit consists of a set of reward distributions $\mathcal{D} = \{R_1, R_2, \ldots, R_K\}$ where each distribution has a mean $\mu_1, \mu_2, \ldots, \mu_K$. Subsequently, the performance of a certain solution strategy can be quantified by the regret ρ after T played



(a) SMA (b) TPMA (c) EWMA

Figure 6: Path changes for different window-sizes $\omega = 1..15$ for Pareto-distributed data



(a) SMA (b) TPMA (c) EWMA

Figure 7: Fraction of correct path decisions for different window-sizes $\omega = 1..15$ for Pareto-distributed data

rounds which is the difference between the best achievable reward sum and the actually achieved reward sum. ρ therefore is given by

$$\rho = T\mu^* - \sum_{t=1}^{T} \hat{r_t}, \tag{10}$$

where μ^* is the maximal reward mean $\max_k\{\mu_k\}$ and $\hat{r_t}$ the reward at time t [51]. For the following discussion of algorithms we also define $\hat{\mu}_i(t)$ as the empirical reward mean of arm i after t turns and $p_i(t)$ as the probability of picking arm i at time t as it is common in MAB literature.

We now very briefly mention two different algorithms for the MAB-problem.

3.4.1 ϵ-greedy strategy

The ϵ-greedy strategy is the probably most trivial algorithm to solve the multi-armed bandit problem. It uses the simple approach of pulling a random lever at a fixed probability ϵ and otherwise pulling the the lever with the highest empirical mean. Given initial empirical means $\hat{\mu}_1(0), \ldots, \hat{\mu}_K(0)$, the probability of pulling a lever is given by

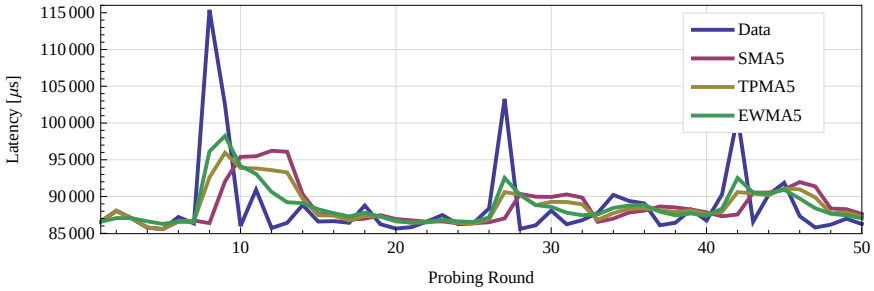


Figure 8: Convergence Behavior for latency-based heuristics with a window-size of $\omega = 5$

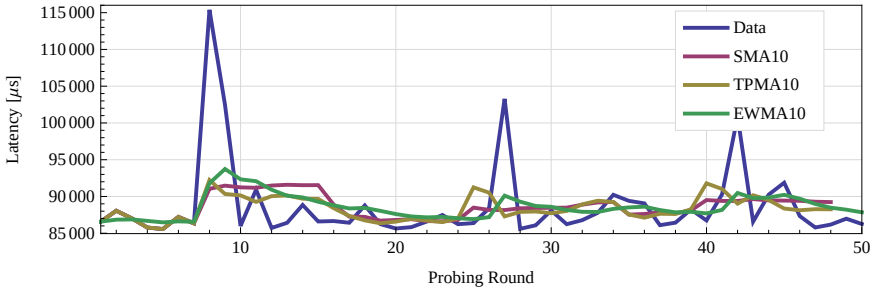


Figure 9: Convergence Behavior for latency-based heuristics with a window-size of $\omega = 10$

$$p_i(t+1) = \begin{cases} 1 - \epsilon + \epsilon/k & \text{if } i = \arg\max_{j=1,\ldots,K} \hat{\mu}_j(t), \\ \epsilon/k & \text{otherwise.} \end{cases} \tag{11}$$

While this strategy is one of the classic approaches for the problem, we doubt it is a reasonable solution for our scenario. Still, we leave this open for future research.

3.4.2 Upper Confidence Bounds (UCB)

Work from Auer, Cesa-Bianchi & Fisher [4, 5] from 2002 introduced a new class of algorithms for the Multi-armed Bandit, namely the UCB class of algorithms. The simplest version UCB keeps track of the number of pulls $n_i(t)$ of a certain arm i at time t, as well as its empirical mean $\hat{\mu}_i(t)$ at time t. In the first round every arm is played. Later on, the arm $j(t)$ to be played is picked by

$$j(t) = \arg\max_{i=1\ldots k} \left(\hat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}} \right). \tag{12}$$

The UCB algorithms give guarantees of bounds on the regret and are going to be revisited in future work.

4 Prototype Implementation

Along with the theoretical discussions of the previous chapters, a prototype was implemented to investigate more on the feasibility and actual performance of a system like it is outlined in Section 1. As the concept of network tunneling is fundamental for this implementation, a short introduction is given in Section 4.1.1. We then describe the setup of the used testbed and actual implementation details.

4.1 Background

4.1.1 Network Tunneling

While packet tunneling is commonly used to set up Virtual Private Networks (VPN), we use tunneling to provide a logical connection between two nodes concealing the underlying implementation. As the tunnel endpoints can be used like customary network interfaces, the proposed system is transparently usable for any kind of application.

In general, a tunnel is splitting location from identification in networks by shipping a protocol through a network layer which the protocol usually does not belong in, meaning that the regular encapsulation order of the ISO/OSI[3] stack is violated. For example, in VPNs, entire Ethernet frames may be encapsulated as a layer-4 payload and sent through the Internet. Upon decapsulation, it appears like the distant machine is located in the same layer-2 network segment as the machine where the tunnel itself ends. However, tunnels can be deployed on different layers of the ISO/OSI model depending on actual requirements. Basic IP-in-IP tunneling like it is provided using GRE[4] is one of the most trivial forms of network tunnels providing plain datagram service for the delivery of encapsulated data. More complex approaches operating on higher layers include L2TP, SSH tunneling or PPTP.

In this implementation we use virtual network interfaces in Linux to write a custom tunnel implementation for the specific needs of this work. So called tun-devices[**?**] act as normal network interfaces. That means traffic can be routed through these interfaces with standard kernel routing table entries. Internally, tun-devices work like any Unix I/O device where data can be read from and written to using standard file descriptors and the POSIX I/O API. This makes it possible to read arbitrary layer-3[5] data the operating system is routing through the tunnel device and then perform custom action, such as encapsulation or filtering. On the other hand, packets destined for the local machine can be somehow obtained and then written to the tun-interface where a client application is connected making the transport completely transparent for applications.

4.1.2 Linux Kernel Routing

The Linux kernel supports simple IP forwarding functionality based on the kernel's routing table. When activated, a Linux machine with multiple interfaces can operate like a normal

[3]Open Systems Interconnection Model [23]
[4]Generic Routing Encapsulation [22]
[5]Access to layer-2 data is available by using the similar tap-interfaces[**?**]

network router. By default, this functionality is deactivated but can be activated by running

```
$ sysctl -w net.ipv4.ip_forward=1.
```

For the proposed setup, kernel routing is immanent as the Gateway machine transfers packets from the local tunnel endpoint to public interfaces sending the data out to the Internet.

4.2 Testbed Setup

In order to reduce complexity in the evaluation process which is primarly aiming at early and general results and further directions, we decided to deploy the prototype on an indoor testbed. While a real-world application for our scenario would make use of wide-area networks (possibly of different cell phone providers), we use simple wireless (WiFi) links to get a sense for the behavior of our system in a relatively similar setup.

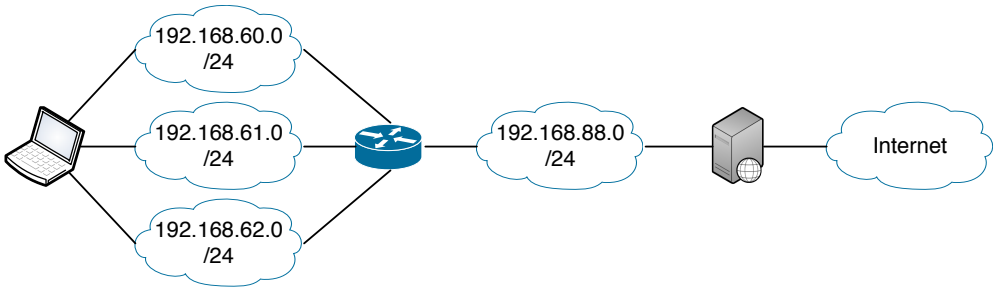


Figure 10: Testbed network layout

The testbed consists of a multi-homed client with three wireless interfaces, a gateway with two ethernet interfaces, as well as three different IP networks connecting those. The gateway (which would normally be some machine in the Internet) is connected to the wireless network via one interface and to the Internet via a normal gateway on the other interface. The client's only direct connection to the Internet is via the wireless networks where traffic must be routed through the gateway machine. Table 1 shows the routing table at the intermediate router.

The three wireless interfaces at the client side are connected to three distinct IP /24-networks. These networks are connected to a normal router operating on layer-3. Each of the three networks also operates in an own layer-2 domain reducing correlations in behavior between the wireless links. The router is also connected to the network where one interface of the gateway machine is in and routes traffic between these four networks. The layout of this network is depicted in Figure 10.

4.3 Implementation Details

4.3.1 System Architecture

As the aim of our system is to improve reliability and quality of service of the first part of a mobile network connection (i.e. the part which is actually using wireless technology),

destination	interface	via
192.168.60.0/24	eth3	local
192.168.61.0/24	eth4	local
192.168.62.0/24	eth5	local
192.168.88.0/24	eth2	local

Table 1: Routing table at the intermediate router

we need access to the packet flow before sending packets onto the wirless link and when receiving packets from the wireless links to route them to their destination via the more reliable core Internet. Thus, the system consists of two integral parts, that is the client and a second entity we call gateway. The client is deployed at the wireless device while the gateway is some publicly addressable computer which should be located somewhere closer to the network's core. The general setup of this is shown in figure 1.

In particular, the job of the multi-homed client is to distribute packets in an efficient manner over the available wireless connections to the gateway. The gateway then forwards the packets to the destination and from the destination back to the client. As we split up the data communication at layer-3 onto multiple networks (thus resulting in multiple client IP addresses), we need a mechanism to map one logical network connection between a client and a server to one IP-address pair which then consists of the IP address of the gateway and the IP address of the server. Otherwise, socket operations would not work and packets arriving at the server from different source IP addresses but the same physical client machine would not be considered one single flow. In other words, for any server receiving a request from a client using our system, it looks like the client would be the gateway machine. The gateway thereby conceals the multiple IP endpoints at the client which might have been involved in a single flow.

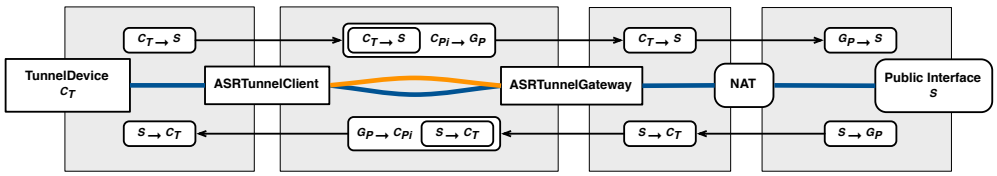


Figure 11: Packet Encapsulation

To achieve this, the connection between the client and the gateway is tunneled. Packets leaving the client get encapsulated and sent via a connectionless UDP socket to the gateway. The client maintains one socket per available outgoing interface. These sockets are bound to the IP address of the corresponding wireless interface. On the server side only one socket is opened and accepting packets from the client. Upon arrival of the packet at the gateway, the packet gets unpacked. The resulting packet on the IP-layer now has a private source IP address (the internal address of the tunnel-interface on client side). As these private endpoints are only addressable from within the client or gateway, the packets source IP

address must be translated to the publicly reachable IP address of the gateway. Network Address Translation (NAT) is used for this purpose. The NAT-system keeps track of IP addresses and corresponding ports and translates addresses from our internal tunnel network to the public Internet. When a packet returns from a server, NAT also translates this packet back to its original address and port. Packets then get again encapsulated and sent over the tunnel back to the client, making the entire process transparent for arbitrary client applications. Figure 11 shows the encapsulation and addressing system, where C_T stands for the private tunnel address at the client, S for the server address, C_{P_i} for the public address of interface i at the client, and G_P for the public address at the gateway.

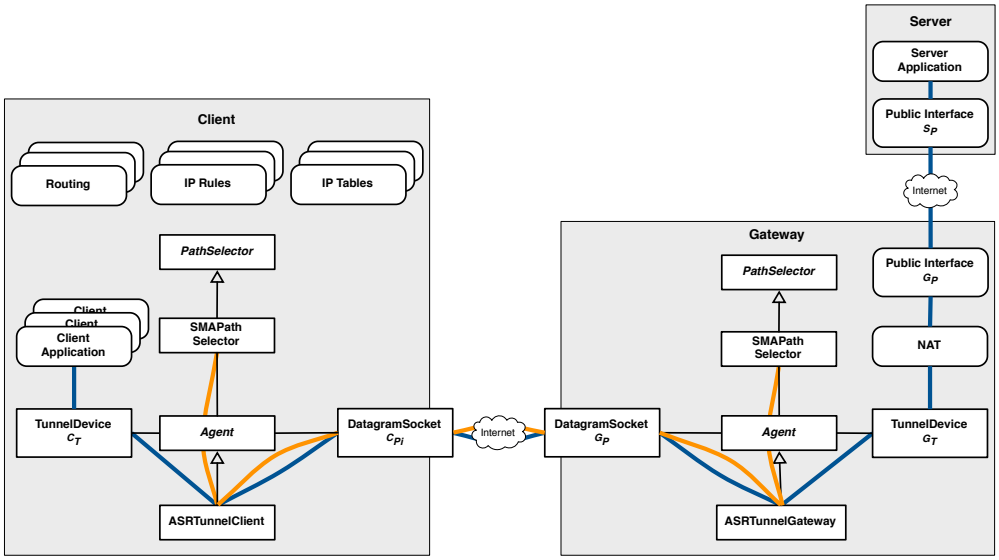


Figure 12: System Components and Classes

Continous active probing between the client and the gateway is used to get information about the status of the different paths in order to take further decisions which path to use for actual data transmission. Figure 12 shows the overall, more detailed architecture of the system. The blue line depicts the flow of data packets; the orange line depicts the flow of probe packets, respectively.

The application is written in approximately 2500 lines of C++ code using the Berkeley C socket API avoiding any overhead through wrappers or third-party frameworks.

4.3.2 Client Application

As previously described, the client application distributes outgoing packets over the available wireless interfaces while keeping track of the performance of the different paths using a reasonable path quality estimation technique. Packets get sent over the speculatively best path. Additionally, for increased performance and resilience, otherwise idling resources (i.e. the not primarily selected interfaces) can also be used to transmit exact copies of a packet, later using only the packet which arrives first at the destination. While this approach produces overhead on the network in general, depending on how distinct the wireless paths really are,

no single wireless link should be negatively affected by this strategy. However, of course also the client and gateway would experience increased load which might influence performance.

In order that data is getting sent through our application at all, the application is reading from a tunnel device, which internally acts like a normal network interface. The tunnel device is then configured as the default outgoing interface for all packets except for packets that go to the gateway. Thus, we need to set up a custom static route for the IP address of the gateway machine. The routing table at the client used in our testbed can be seen in table 2.

table	destination	interface	via
main	192.168.60.0/24	wlan0	local
main	192.168.61.0/24	wlan1	local
main	192.168.62.0/24	wlan2	local
asr-tunnel-1	192.168.88.0/24	wlan0	192.168.60.1
asr-tunnel-2	192.168.88.0/24	wlan1	192.168.61.1
asr-tunnel-3	192.168.88.0/24	wlan2	192.168.62.1
main	10.0.0.0/24	tun0	local
main	0.0.0.0/0	tun0	

Table 2: Routing table at client

Looking at this routing table, it becomes apparent that the gateway can be reached via any of the wireless interfaces. If the system would not be set up this way, the sockets in the client application (although binding to the particular IP addresses of the outgoing interface) would not find a route to the gateway. However, as the standard kernel does not support multiple routes for a single destination, we have to introduce additional packet handling. This constraint of one route per destination does not apply to the entire system but to any single routing table. Linux may however maintain multiple routing tables. Therefore, if we are able to split packets over multiple routing tables depending on the outgoing interface or any additional attributes, multiple equal routes can easily be implemented. In fact, the Linux iptables [1] support custom tagging of packets depending on the packet header; furthermore packets can get assigned to routing tables based on assigned tags. As a consequence, we can just mark each outgoing packet with a tag unique to its outgoing interface and the packets then get routed using different tables sharing the same destinations but different outgoing interfaces and gateways. A packet can be tagged using commands similar to this:

```
$ iptables -A OUTPUT -t mangle -s 192.168.60.2 -d 192.168.88.2
  -p udp --sport 42742 --dport 42742 -j MARK --set-mark 1
```

All outgoing UDP port 42742 packets destined for 192.168.88.2 from 192.168.60.2 get tagged with the integer 1. Using ip rules, we can then assign all packets tagged with 1 to the routing table "asr-tunnel-1":

```
$ ip rule add from all fwmark 1 table asr-tunnel-1
```

Finally, this table then contains a route letting the packets leave on one of the particular wireless interfaces:

```
$ ip route add table asr-tunnel-1 192.168.88.0/24 dev wlan0
  via 192.168.60.1 src 192.168.60.2
```

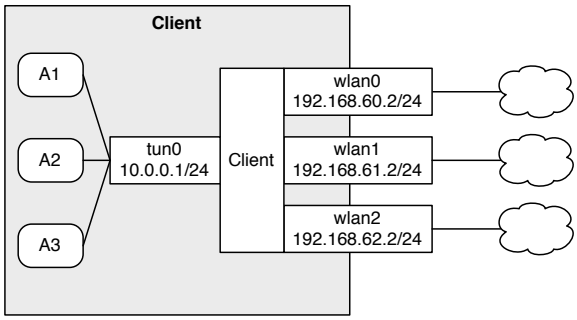


Figure 13: Client Components and Interfaces

Having solved this issue, the remaining implementation of the client software is relatively straightforward. Based on the currently best path, the packets, that are getting read from the tunnel interface get encapsulated, assigned a sequence number and sent out on the corresponding interface. Arriving packets get unpacked and written to the tunnel interface for the actual client application without any modification. The basic architecture of the client system is shown in figure 13. A call for starting the client application with three different wireless interfaces and a gateway IP-address looks like this:

```
$ ./client -g 192.168.88.2 wlan0 192.168.60.2 wlan1 192.168.61.2
  wlan2 192.168.62.2
```

Figure 14: Wifi Interfaces conntected to client in testbed setup

4.3.3 Gateway Application

The gateway application is performing a similar task as the client application. The main difference is that it is using only one socket and one tunnel device. Packet encapsulation, as

well as active probing and the selection of the return path work analogous to the client. The basic components of the gateway are shown in figure 16.

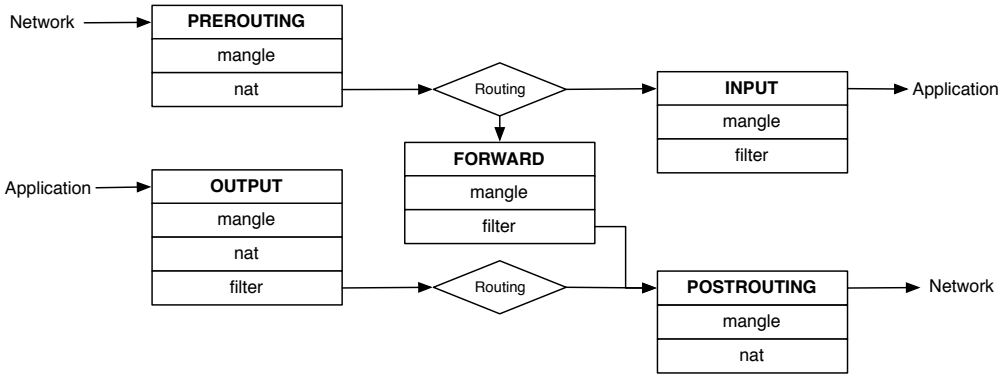


Figure 15: iptables Chains and Tables

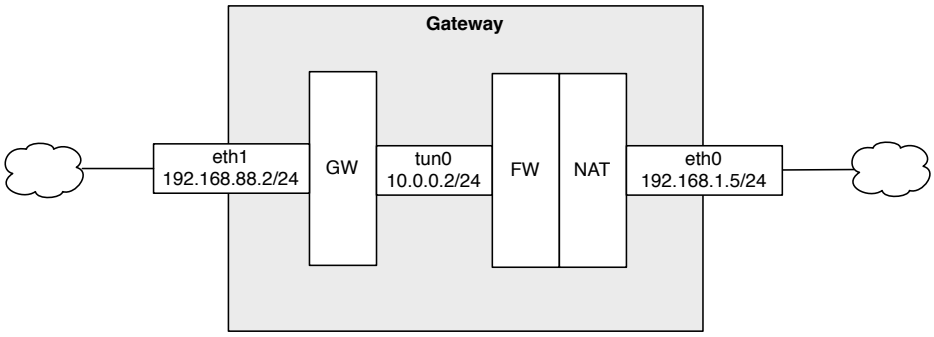


Figure 16: Gateway Components and Interfaces

Beside the application code though, there is forwarding and packet modification in form of network address translation needed to transparently transport data to the server. Network address translation is implemented through the standard Linux iptables. The corresponding rule, to automatically translate packets, use random source port numbers and keep track of all translated flows, can be set up within the OUTPUT chain. The iptable's output chain is that chain that packets traverse when being sent from a user-space application before any routing decisions are made. All three possible tables (nat, mangle and filter) are available in this chain. An overview of the different chains and tables of iptables is shown in figure 15. The command to enable address translation at the gateway is

```
$ iptables -A POSTROUTING -t nat -s 10.0.0.1 -o eth0
  -j MASQUERADE,
```

where 10.0.0.1 is the IP address of the client-side tunnel-device and eth0 is the public interface at the gateway which is used to send the packets to its final destination. The major drawback of this setup is, that iptables is not able to perform network address translation on fragmented packets. This is due to the reason, that IP fragments may not contain the packets

transport header and therefore can not be matched with the NAT-table which is requiring this information. While we know, when fragmented packets are about to leave the gateway application through inspecting the IP header's flag and offset fields, this implementation does not yet take any actions to reassemble packets before passing it to iptables and the kernel's forwarding capability.

Additionally, in order to send packets back to the client, the gateway application also needs the set of client IP addresses. In our testbed, the gateway application was started with the following command:

```
./gateway 192.168.60.2 192.168.61.2 192.168.62.2
```

destination	interface	via
192.168.60.0/24	eth1	192.168.88.1
192.168.61.0/24	eth1	192.168.88.1
192.168.62.0/24	eth1	192.168.88.1
192.168.88.0/24	eth1	local
192.168.1.0/24	eth0	local
10.0.0.0	tun0	local
0.0.0.0/0	eth0	192.168.1.1

Table 3: Routing table at gateway

5 Results

5.1 Path Selection

In this section, we present results from both trace collection from the Internet as well as measured data from our prototype wirless setup. After exploring different possibilities for path selection in section 3, we found a simple moving average heuristic with a window size of $\omega = 4$ as a reasonable approach to identify well performing paths while reducing the frequency of path changes.

The datasets collected on PlanetLab did not send any real data traffic but just used probes which were then analyzed in retrospect. We can see, that the actual path latency, the algorithm would have selected with this simple technique is fairly close to the optimal for the majority of cases. In fact, on the first tested topology, we get optimal latencies in about 5% of the cases. Even in the 99th percentile the algorithm would yield latencies that are only about 10ms higher than the best achievable (cp. fig. 17). In the second (random) topology, the algorithm is showing worse but still very good behavior. In the mean, again, we get close to optimal results. The tail, however shows significantly worse than optimal results beginning at about the 80th percentile (cp. fig. 18). However, the results are surprisingly good and it seems like this simple proposed heuristic is a reasonable strategy. Figures 19 and 20 show the same data after applying the EWMA-algorithm. The results, clearly are inferior

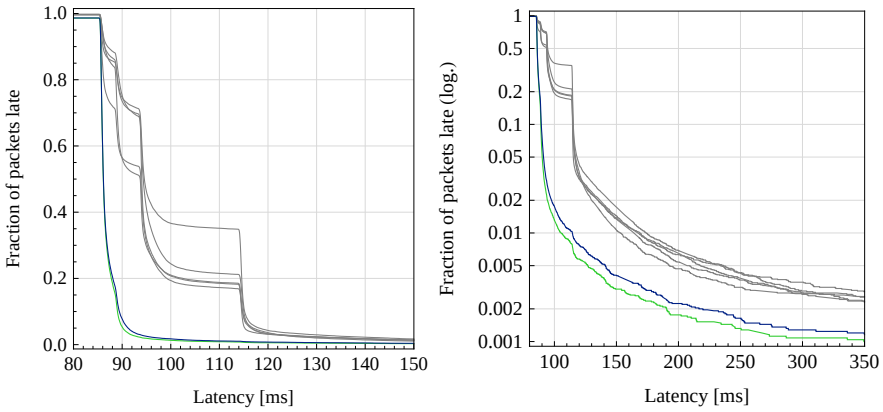


Figure 17: Dataset 1: CCDF of latencies seen in experiment in linear and log. scale after applying SMA4-heuristic (blue = achieved latency, green = best latency in retrospect, and gray = measured path latencies)

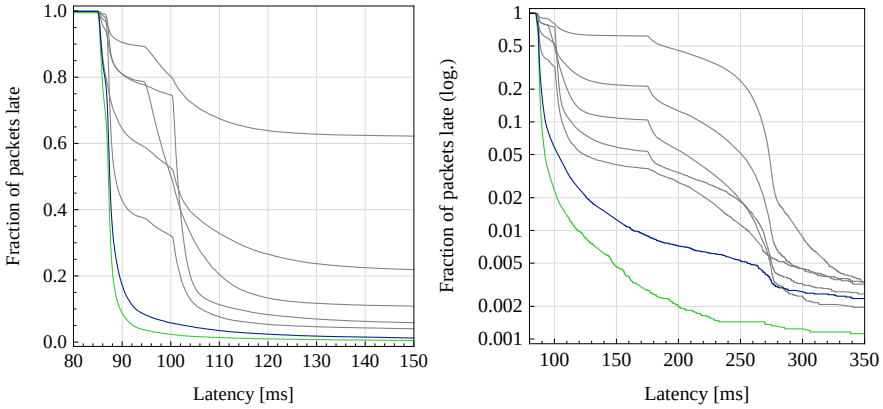


Figure 18: Dataset 2: CCDF of latencies seen in experiment in linear and log. scale after applying SMA4-heuristic (blue = achieved latency, green = best latency in retrospect, and gray = measured path latencies)

to the SMA results even though the EWMA-strategy tends to react to abrupt performance changes more quickly.

For the prototype application, path selection was evaluated sending a real data stream with a rate of between 64 and 128 kbit/s over the best interface. The blue line in the plots depicts the latency seen by the actual data packets being sent back to back between the client and the gateway. For this experiment, the data packets were not forwarded to any server and were randomly generated in the client application itself. Probing of the three paths was continously performed also while sending data. Figure 21 shows the seen latencies of the three paths (gray), and the data latencies in blue. We see, that even in this case we can do surprisingly well. Especially, the tail behavior (beginning at the 98th percentile), which is imperative for numerous applications and especially for our focus of medical care, is very satisfactory.

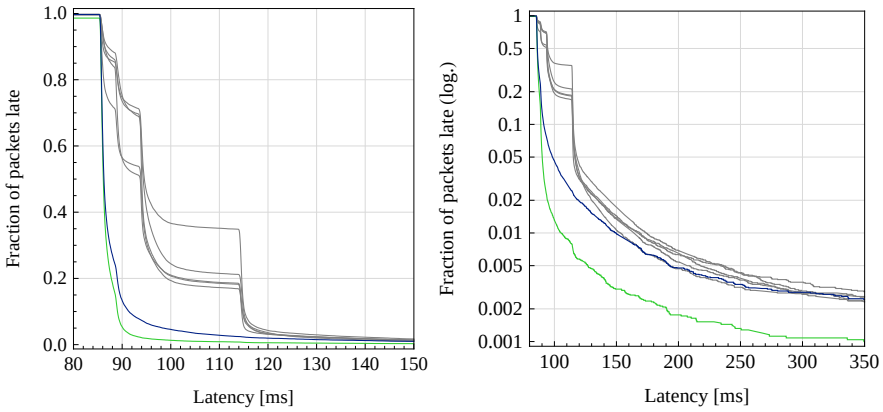


Figure 19: Dataset 1: CCDF of latencies seen in experiment in linear and log. scale after applying EWMA4-heuristic (blue = achieved latency, green = best latency in retrospect, and gray = measured path latencies)

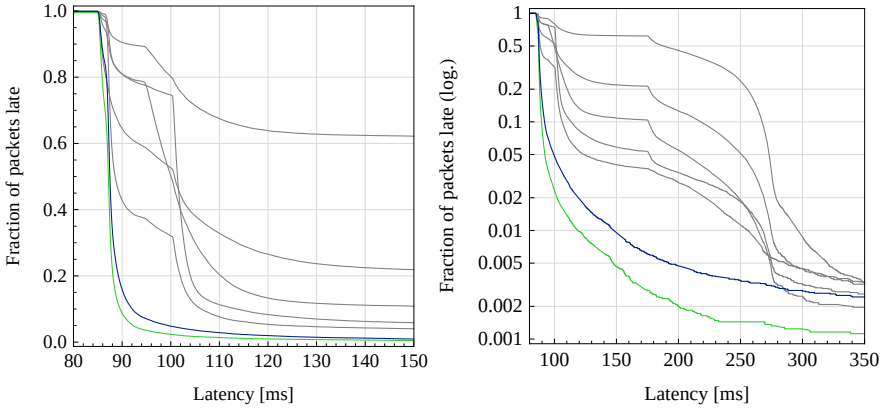


Figure 20: Dataset 1: CCDF of latencies seen in experiment in linear and log. scale after applying EWMA4-heuristic (blue = achieved latency, green = best latency in retrospect, and gray = measured path latencies)

5.2 Fast Adaption and Handover

As the title of this thesis contains the term "adaptive routing", meaning we adaptively and quickly react to variance in the network's behavior, the ability to respond to failing links or significantly decreased performance quickly, is very important. Looking into our traces in more detail shows that usually only a couple of packets get completely lost when the currently active path is failing completely. However, this measure relies highly on the path's roundtrip time, the set timeout for considering a packet lost and the current throughput on the wire. Clearly, when sending at a very high rate, more packets may get lost. The same applies for high timeouts and long inter-probe delays. When sending data at rates around 100 kbit/s, using a timeout of 500ms and an inter-probe delay of 0.5s, we achieve good adaptability to complete outages. On the other hand, the more often probes are sent, it might happen, that the algorithm changes the path when there is only packet loss on the network for one or two seconds where it does make more sense to stay on this path than bearing consequences and

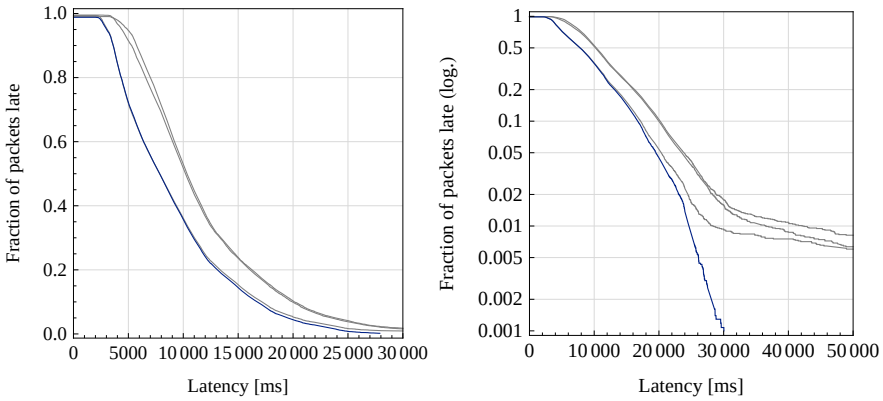


Figure 21: Achieved Packet Throughput Latency and probed path-latencies in Wireless Tunnel Setup (blue = achieved latenc, gray = measured path latencies)

undergoing uncertainty that come with switching to a different path.

Figure 22 shows a throughput graph obtained using Wireshark while the client was sending a continous stream of data over one best path. The different colors in the plot show the three different interfaces. During the transmission, the primary path is being changed five times. The traffic visible at the plot's baseline belongs to the continous probing traffic on all three paths. The plot shows, that the throughput remains fairly constant when the client switches between different data paths. Subsequently, the unused capacity at the time of switching seems to be negligible.

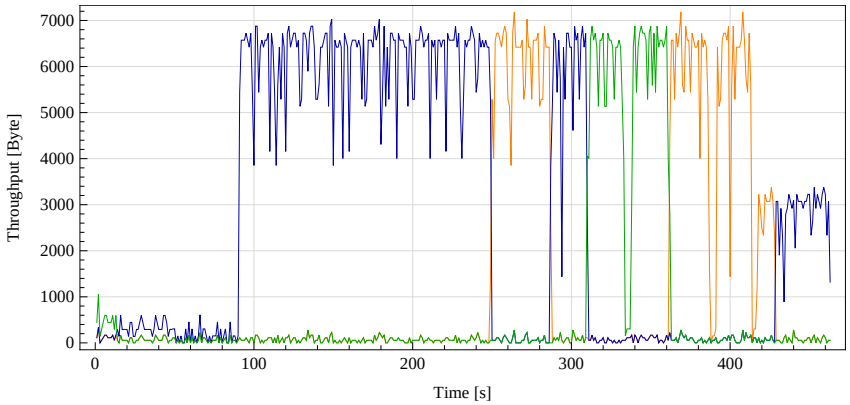


Figure 22: Throughput and adaption to path changes

The result of a second conducted experiment is depicted in figure 23. In this experiment, netem was used to artificially add latency to a path. In this experiment, no actual payload data was sent. Again, out of the three different paths, one path was selected as the active path (green line). The gray lines show the actual measured latencies on the paths, while the blue lines show the 10-period simple moving averages of the path latencies to make the change of performance better visible. Using a script, there were 20ms of latency added to the first path, 40ms to the second paths, and 60ms to the third path. After 20s the configuration

was changed, such that every path in the end had each additional delay configuration. The graph shows the three phases clearly. The plot shows, that even during the switching of the performance and the resulting change of the active path, the achieved latency steadily remains on a low level. Unfortunately though, it is noticeable that one packet (packet number 105) on the active path is getting lost (yielding a latency of zero in this statistic). The other packet-losses (latency = 0) in the given time frame do not affect the selected path.

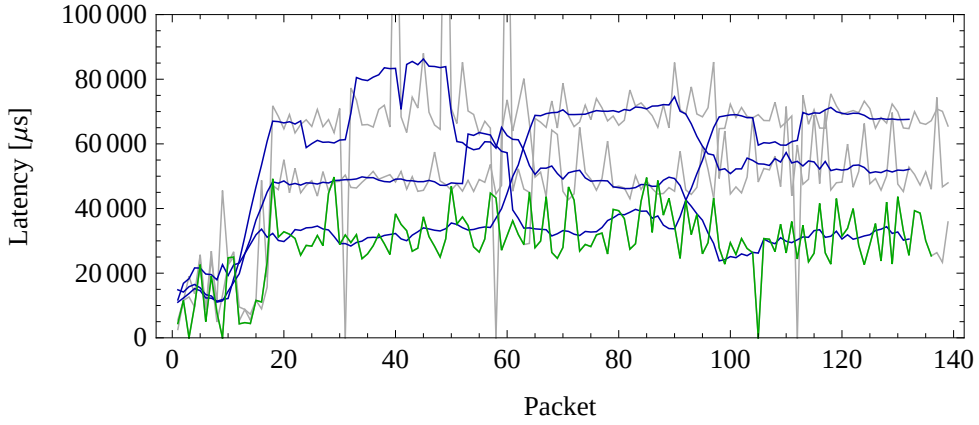


Figure 23: Path latency during switching

5.3 Packet Replication

As previously pointed out, having a choice of multiple paths, allows us to replicate traffic over multiple paths, using only the first packet that arrives at the destination. This concept, can be used to convert otherwise idling resources into additional resilience and fault tolerance as a packet loss on one path does not affect the successful transmission of a packet. In previous work [52], we proposed this strategy for any system that offers multiple resource choices. In a similar setup using overlay routing this technique was evaluated, showing drastically reduced latency in the distribution's tail when sending over more than one path compared to only one (already SMA-selected) path. A sample latency distribution comparison between sending over one, two and three paths simultaneously is shown in figure 24.

This plot represents data collected on a PlanetLab overlay and is therefore not comparable with the setup we used in this thesis. However, since wireless links involve even more uncertainty, we believe this technique to yield even better results than in the investigated wide-area, low-latency and low-variance experiment. We leave this open for future reseach.

6 Application in Preclinical Medical Care

With an oldening population and an increasing occurrence of life-threatening diseases such as strokes or heart-attacks, there is an increased need for continuously improving our emergency medicine infrastructure and especially the seamless cooperation and communication between the different involved parties such as paramedics, emergency physicians in the ambulance and physicians and nurses in the hospital, all caring for a single patient in a highly distributed

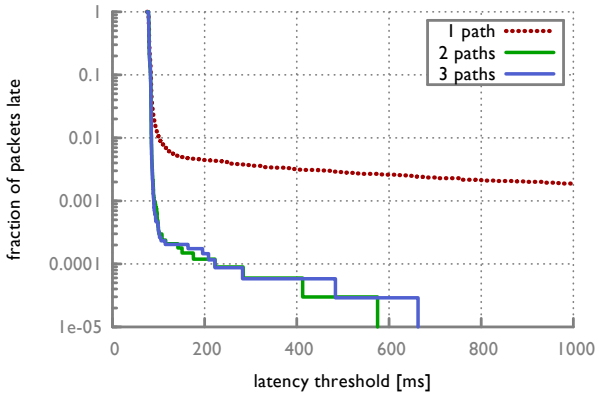


Figure 24: Latencies seen in redundant multipath experiment (taken from [52])

manner. Considering the fact that e.g. Coronary Heart Disease is the most common cause of death in developed countries and that approximately a third of the patients die before getting clinical treatment [14], we see that the optimization of pre-hospital medical services may save a considerable fraction of lives.

Today, decent emergency medical services systems are deployed throughout the world. However, even in developed countries with an overall high quality in their health-care system, there is often a lack of efficiency in the care transition between EMS crews and hospital staff [28]. Also, due to very constrained time, information about the patient (such as medication or allergies) is likely to get lost in this phase. Moreover, the problems often arise even earlier in the decision which hospital is the right one for a particular patient or which procedures are reasonable to conduct in the ambulance to provide best preparatory work for emergency room personnel.

The occurrence of these problems is due to a variety of reasons. Usually, the only means of communication between paramedics and the hospital are the cell phone and radio. Subsequently, EMS crews call hospitals in order to determine which hospital has available beds for the patient. This becomes even more complicated with critically ill patients that require treatment in an intensive care unit or special medical disciplines such as neurosurgery or pediatrics. As hospitals are more and more specialized and the status of (especially old) patients in EMS is becoming more complex mainly due to multimorbidity [34], determining the best place to bring a patient to within minutes of time is becoming a non-trivial matter. Furthermore, while the demand for emergency medicine is increasing over the past years, the lack of personnel (in particular of physicians [29]) in this field is becoming more severe and actually abandoning quality of care. As a consequence EMS-staff is obtaining more responsibility and duties since there are simply no more physicians available outside of the hospital. Critical decisions therefore must often be made in short time and complex procedures must be conducted by staff that is lacking practice and experience to save a patients life eventually.

These problems call for a substantially deeper integration of telemedical systems, decision support and clinical information systems in preclinical emergency medicine. While this

demand seems to be evident, there is comparatively little progress made in this area. This is mainly due to constrained and insufficient connectivity between these entities particularly in rural areas. Furthermore, deployment, testing and evaluation of systems solving these problems, seems to be a complex issue in this critical field due to high requirements in availibility and security and the shortage of time to perform actions twice as a backup in extremely constrained time.

One project [8, 47] which is trying to solve this problem is originating from the RWTH Aachen University medical center in Germany, where together with local authorities and the fire department select ambulances were equipped with cameras and patient monitoring devices that transmit the current situation to an emergency physician that is supporting the EMS-crew from the hospital through telemedicine. In this project, the physician sitting in his office possibly miles away from the scene, is able to talk to the paramedics wearing headsets, and to see the patient through a video stream. Additionally, she or he has the patient's vital parameters such as blood pressure, peripheral oxygenation, pulse and ECG on a screen directly transmitted from the patient monitor. This goes as far as that special stethoscopes are used which communicate to the monitor via Bluetooth, such that the consulting physician can even hear the sounds from the stethoscope. The so-called "tele-physician" is able to get an impression of the patients situation, order the paramedics to place intravenous lines, give certain medication or to take other actions. He may also consult other colleagues to discuss the case and subsequently make a decision how critical the situation is, what the diagnosis could be and which hospital is adequate for further therapy and admission.

While the EMS crew is engaged in providing care to the patient and conducting the transport to the hospital, the "tele-physician" has enough time to call the responsible emergency room team at the destination-hospital, explain the situation, submit results and reports several minutes before the patient is even arriving. This gives the local team sufficient time to plan and prepare the required upcoming steps such as CT scans, certain procedures or even surgery if the diagnosis is fixed.

In a randomized, controlled simulation study [46], the emergency care between physician staffed EMS teams (control group) and paramedic teams that were supported telemedically by an EMS physician (telemedicine group) was compared. The scoring was based on the granularity of information obtained from the patient. As a result, patient details were inquired comparatively between the two groups while allergies and medication were inquired more frequently in the telemedicine group. No significant differences were found regarding the case-specific items and in both groups no potentially dangerous mistreatments were observed. Taking these results into account, the given approach seems to be a reasonable and feasible response to limited staffing, time and resources in emergency medical care allowing one physician to care about more patients in shorter time while improving the level of care eventually.

When looking at this project, it becomes apparent that for such extensive data communication via wireless wide-area technologies an extremely high level of fault-tolerance, availability, performance and resilience is essential. The system proposed in this thesis aims at mitigating these exact challenges, we experience in today's wireless systems, through

(a) Physician's station (b) Ambulance with camera (top right corner)

Figure 25: The two entities involved in the "tele-physician"-system (taken from [8])

multi-homing, path-selection and multi-path transmission.

Although, in this thesis, we focus on the field of emergency medical services, comparable systems may be deployable in a variety of application fields such as disaster medicine [39], field medicine or generally speaking in all settings with limited availability of specialist for certain complex illnesses or injuries.

7 Conclusions and future directions

In this thesis, we showed how to effectively mitigate unreliable communication links in wireless networks keeping an eye on special requirements and the demand in the field of pre-hospital emergency medical care. We proposed a framework which makes it possible to control the distribution of packets over different available paths and showed how to monitor path performance and how to react accordingly.

The proposed system could be deployed in a real world setting without tremendous additional modification. Furthermore, due to the modularity of the system, custom algorithms for probing and path selection can easily be integrated. Although, this system is a prototype, we showed that for basic scenarios the system works as expected, conceals internal complexity, is transparent for any kind of application, and most importantly improves overall quality of service with even very simple heuristical methods like those presented in section 3.

Still, questions and problems remain unanswered and unsolved in this thesis which we like to point out as future research directions.

First, although TCP connections go through our system without any problems in the normal case, things get complex, packets get lost, latency increases quickly or sometimes the connection gets even stuck as soon as either paths switch too often or retransmissions occur for other reasons which is more likely to happen the longer the connection is running. This is mainly happening due to out-of-order transmissions. Therefore, it needs to be investigated how well these problems can be solved introducing additional custom reordering buffers in the client or gateway application.

Another severe problem is IP fragmentation. As previously pointed out, it is not possible

to perform conventional network address translation as soon as raw IP fragments traverse the system. It needs to be explored how this issue can be solved by defragmenting packets at the gateway before writing them back to the tunnel device for further processing or forwarding.

While, we investigated reasonable parameters for the different path selection techniques, it remains unclear if the way probes are sent out to the network and especially the frequency for sending probes is appropriate. Problems that might occur when sending too many probes although yielding higher measurement resolution, are pointed out in section 5.2.

Lastly, while the proposed heuristics seem to perform reasonably well, there are several more strategies including MAB-algorithms that need to be evaluated in order to find more reasonable approximations to the Internet path selection problem. Future strategies might also include other metrics, such as bandwidth.

As this system is working well for most cases and many problems in the design of such a system have been solved or are identified in this work, there is room for improvement we will investigate after this thesis

In conclusion, we see the challenge of improving Internet- and especially wireless connection resilience and performance e.g. through intelligent path selection as an important direction for future research.

Also, especially in the field of medical care requiring high resilience and fault-tolerance, more mobile networked applications are going to be deployed increasing the need for systems like the proposed one. This work is though not limited to application in medical care. We state that intelligent selection out of a variety of resources will gain interest as we may have more diverse sources to select from in the Future Internet (esp. in the context of network virtualization and fedaration). In general, we expect to see higher diversity in all kinds of usable resources in the systems community such that selection and efficient usage of otherwise idling resources will become inevitable in the upcoming decades.

List of Figures

List of Tables

References

[1] http://www.netfilter.org, last visits 11/24/12.

[2] *NIST/SEMATECH e-Handbook of Statistical Methods*. National Institute of Standards and Technology, 2003.

[3] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 131–145, New York, NY, USA, 2001. ACM.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, FOCS '95, pages 322–, Washington, DC, USA, 1995. IEEE Computer Society.

[5] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61:55–65, 2010.

[6] Francçois Baccelli, Sridhar Machiraju, Darryl Veitch, and Jean C. Bolot. The role of pasta in network measurement. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 231–242, New York, NY, USA, 2006. ACM.

[7] Francois Baccelli, Sridhar Machiraju, Darryl Veitch, and Jean C. Bolot. On optimal probing for delay and loss measurement. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 291–302, New York, NY, USA, 2007. ACM.

[8] S Bergrath, A Reich, R Rossaint, D Rörtgen, J Gerber, H Fischermann, SK Beckers, JC Brokmann, C Schulz, JB andLeber, C Fitzner, and M Skorning. Feasibility of prehospital teleconsultation in acute stroke–a pilot study in clinical routine. *PLOS ONE*, 7(5), May 2012.

[9] Jean-Chrysotome Bolot. End-to-end packet delay and loss behavior in the internet. In *Conference proceedings on Communications architectures, protocols and applications*, SIGCOMM '93, pages 289–298, New York, NY, USA, 1993. ACM.

[10] Baek-Young Choi, Sue Moon, Rene Cruz, Zhi-Li Zhang, and Christophe Diot. Practical delay monitoring for isps. In *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, CoNEXT '05, pages 83–92, New York, NY, USA, 2005. ACM.

[11] Jonathon Duerig, Robert Ricci, Leigh Stoller, Matt Strum, Gary Wong, Charles Carpenter, Zongming Fei, James Griffioen, Hussamuddin Nasir, Jeremy Reed, and Xiongqi Wu. Getting started with geni: a user tutorial. *SIGCOMM Comput. Commun. Rev.*, 42(1):72–77, January 2012.

[12] Chip Elliott and Aaron Falk. An update on the geni project. *SIGCOMM Comput. Commun. Rev.*, 39(3):28–34, June 2009.

[13] Eugenio Epprecht, Francisco Aparisi, Marco A. De luna, and Andrés Carrión. A computer optimization of a set of ewma quality control charts. In *Proceedings of the 9th WSEAS international conference on Applications of computer engineering*, ACE'10, pages 93–98, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS).

[14] Robert Beaglehole et al. *The World Health Report 2004 – Changing History*. WHO, 2004.

[15] D. J. Farber and J. J. Vittal. Extendability considerations in the design of the distributed computer system (dcs). In *Proceedings of the National Telecommunications Conference, Atlanta, Georgia*, Nov. 1973.

[16] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/id separation protocol (lisp). IETF Draft, March 2009.

[17] Igor Ganichev, Bin Dai, P. Brighten Godfrey, and Scott Shenker. Yamr: yet another multipath routing protocol. *SIGCOMM Comput. Commun. Rev.*, 40(5):13–19, October 2010.

[18] P. Brighten Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. Pathlet routing. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 111–122, New York, NY, USA, 2009. ACM.

[19] Cesar D. Guerrero and Miguel A. Labrador. On the applicability of available bandwidth estimation techniques and tools. *Comput. Commun.*, 33(1):11–22, January 2010.

[20] H. Han, S. Shakkottai, CV Hollot, R. Srikant, and D. Towsley. Overlay tcp for multipath routing and congestion control. In *IMA Workshop on Measurements and Modeling of the Internet*, 2004.

[21] H. Han, S. Shakkottai, CV Hollot, R. Srikant, and D. Towsley. Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet. *IEEE/ACM Transactions on Networking (TON)*, 14(6):1260–1271, 2006.

[22] S. Hanks, T. Li, D. Farinacci, and P. Traina. Rfc 1701: Generic routing encapsulation (gre), 1994.

[23] ISO. Iso/iec standard 7498-1, 1994.

[24] J.R. Iyengar, P.D. Amer, and R. Stewart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *Networking, IEEE/ACM Transactions on*, 14(5):951–964, 2006.

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

[26] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 295–308, New York, NY, USA, 2002. ACM.

[27] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *In Proceedings of Passive and Active Measurements (PAM) Workshop*, pages 14–25, 2002.

[28] JK Johnson, JM Farnan, P Barach, G Hesselink, H Wollersheim, L Pijnenborg, C Kalkman, and VM Arora. Searching for the missing pieces between the hospital and primary care: mapping the patient process during care transitions. *BMJ Quality and Safety*, November 2011.

[29] Frederik Jötten. Notarztmangel auf dem land: Hubschrauber ans bett. *Die Zeit*, (10), March 2011.

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

[31] Robert Kleinberg. Anytime algorithms for multi-armed bandit problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 928–936, New York, NY, USA, 2006. ACM.

[32] Volodymyr Kuleshov and Doina Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning Research 1*, 2000.

[33] B. Melander, M. Bjorkman, and P. Gunningberg. Regression-based available bandwidth measurements. In *International Symposium on Performance Evaluation of Computer and Telecommunications Systems*, pages 14–19, 2002.

[34] SW Mercer, SM Smith, S Wyke, T O'Dowd, and GC Watt. Multimorbidity in primary care: developing the research agenda. *Fam Pract*, (26):79–80, April 2009.

[35] Oliver Michel, Ashish Vulimiri, and P. Brighten Godfrey. Adaptive Source Routing. Poster at the 13th GENI Engineering Conference, Los Angeles, CA, 2012.

[36] Douglas C. Montgomery. *Introduction to statistical quality control.* Hoboken, N.J. : John Wiley, 2005.

[37] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. Path splicing. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 27–38, New York, NY, USA, 2008. ACM.

[38] P. Natarajan, J.R. Iyengar, P.D. Amer, and R. Stewart. Sctp: An innovative transport layer protocol for the web. In *Proceedings of the 15th international conference on World Wide Web*, pages 615–624. ACM, 2006.

[39] M. Oldenburg, X. Baur, and C. Schlaich. Assessment of three conventional automated external defibrillators in seafaring telemedicine. *Occupational Medicine*, 62(2):117–122, 2012.

[40] Vern Paxson. End-to-end routing behavior in the internet. In *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '96, pages 25–38, New York, NY, USA, 1996. ACM.

[41] Vern Paxson. End-to-end internet packet dynamics. In *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '97, pages 139–152, New York, NY, USA, 1997. ACM.

[42] Larry Peterson and Timothy Roscoe. The design principles of planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1):11–16, January 2006.

[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] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths, 2003.

[45] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society 58*, 1952.

[46] D Rörtgen, S Bergrath, R Rossaint, SK Beckers, H Fischermann, IS Na, D Peters, C Fitzner, and M Skorning. Comparison of physician staffed emergency teams with paramedic teams assisted by telemedicine - a randomized, controlled simulation study. *Resucitation*, June 2012.

[47] MT Schneiders, F Hirsch, D Büscher, C andSchilberg, and S Jeschke. Patient-oriented emergency care - a telemedical rescue assistance system. *Biomed Tech (Berl)*, September 2012.

[48] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Oper. Syst. Rev.*, 40(1):17–24, January 2006.

[49] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 39–44, New York, NY, USA, 2003. ACM.

[50] Carl A. Sunshine. Source routing in computer networks. *SIGCOMM Comput. Commun. Rev.*, 7(1):29–33, January 1977.

[51] Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European conference on Machine Learning*, ECML'05, pages 437–448, Berlin, Heidelberg, 2005. Springer-Verlag.

[52] Ashish Vulimiri, Oliver Michel, P. Brighten Godfrey, and Scott Shenker. More is less: Reducing latency via redundancy. In *Proceedings of the Eleventh ACM Workshop on Hot Topics in Networks (HotNets-XI)*, October 2012.

[53] Ronald W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2), 1982.

[54] Wen Xu and Jennifer Rexford. Miro: multi-path interdomain routing. In *SIGCOMM*, pages 171–182, 2006.

[55] Xiaowei Yang, David Clark, and Arthur W. Berger. Nira: a new inter-domain routing architecture. *IEEE/ACM Trans. Netw.*, 15(4):775–788, August 2007.

[56] Rambabu Yerajana and A. K. Sarje. An adaptive multipath source routing protocol for congestion control and load balancing in manet. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*, ICAC3 '09, pages 456–459, New York, NY, USA, 2009. ACM.

[57] Thomas Zinner, Dominik Klein, Kurt Tutschku, Tanja Zseby, Phuoc Tran-Gia, and Yuval Shavitt. Performance of Concurrent Multipath Transmissions - Measurements and Model Validation. In *Proceedings of the 7th Conference on Next Generation Internet Networks (NGI)*, Kaiserslautern, Germany, June 2011.

[58] Thomas Zinner, Kurt Tutschku, Akihiro Nakao, and Phuoc Tran-Gia. Using Concurrent Multipath Transmission for Transport Virtualization: Analyzing Path Selection. In *Proceedings of the 22nd International Teletraffic Congress (ITC)*, Amsterdam, Netherlands, September 2010.