Modelling users in networks with path choice: four studies in telecommunications and transit

Christopher David Pluntke

A dissertation submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Electronic & Electrical Engineering

University College London

2014

I, Christopher Pluntke, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Department of Electronic & Electrical Engineering

University College London

Abstract

Networks of interacting users arise in many important modelling applications. Commuters interact with each other and form traffic jams during peak-time. Network protocols are users in a communication network that control sending rate and server choice. When protocols send with too high rates, network links get overloaded resulting in lost data and high delays. Although these two example users seem very different, they are similar on a conceptual modelling level. Accurate user models are essential to study complex interactions in networks.

The behaviour of a user with access to different paths in a network can be modelled as an optimisation problem. Users who choose paths with the highest utility are common in many different application areas, for example road traffic, Internet protocol modelling, and general societal networks, i.e. networks of humans in everyday life. Optimisation-based user models are also attractive from the perspective of a modeller since they often allow the derivation of insights about the behaviour of the entire system by only describing a user model.

The aim of this thesis is to show, in four practical studies from telecommunications and transit networks, where optimisation-based models have limitations when modelling users with path choice. We study users who have access to a limited number of paths in large scale data centers and investigate how many paths per user are realistically needed in order to get high throughput in the network. In multimedia streaming, we study a protocol that streams data on multiple paths and path properties matter. We also investigate complex energy models for data interfaces on mobile phones and evaluate how to switch interfaces to save energy. Finally, we analyse a long-term data set from 20,000 transit commuters and give insights on how they change their travel behaviour in response to incentives and targeted offers.

We use tools from optimisation, simulation, and statistics to evaluate the four studies and point out problems we faced when modelling and implementing the system. The findings of this thesis indicate where user models need to be extended in order to be of practical use. The results can serve as a guide towards better user models for future modelling applications.

Acknowledgements

I would like to thank several people without whom this thesis would not have been possible. First of all, my advisors Damon Wischik and Miguel Rio who provided me with the two most valuable things for a student: exciting problems in unexplored territory and strong feedback that made me improve the methods I used.

Some of the chapters in this thesis were the result of work I spent at Nokia as an intern and at Stanford University as a visitor. At Nokia, I would like to thank Lars Eggert and Niko Kiukkonen. I am grateful to them for giving me the opportunity to work as an intern on energy efficient scheduling and experience Finland for six months. I am very grateful that I was given the opportunity to visit Prof Balaji Prabhakar's Societal Networks group at Stanford University. I will always cherish the inspirational atmosphere at Stanford. I also thank Wenkang Wong and William Wong at LTA for their support.

I owe special thanks to Costin Raiciu and Mark Handley for a cooperation on multi-path routing research in data centers that ended up as a central part of this thesis. I would also like to thank them for their wonderful support when I was changing research groups.

Although I was at the NSRL office at UCL for only a couple of months, I am grateful for the friendly atmosphere that allowed cooperations and interactions of ideas that would not have been possible otherwise. At this point, I would like to especially name Joao Araujo, Richard Clegg, David Griffin, Raul Landa, Eleni Mykoniati, Prof George Pavlou, Ioannis Psaras, Lorenzo Saino and Daphne Tuncer.

I would like to especially thank Frauke Uhlenbruch for her support especially in the last weeks of writing this thesis and in the months that I was abroad on internships and visiting Stanford. Finally, my special thanks go to my parents for giving me the opportunity to end up where I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Users are the basic particles of models for networks. Understanding user models is essential to be able to control the system performance and to avoid congestion. Congestion is a common phenomenon in constrained networks that arises when too many people try to access a limited resource at the same time. Congestion leads to more crowdedness, but also to increased delay, higher error rates in communications, and worse quality of experience. In its yearly urban mobility report from 2012 [70], the Texas Transport Institute showed that congestion caused the average auto commuter an additional yearly delay of 38 hours. The monetary impact of congestion on the US economy in that year was estimated to be $121 billion. Similarly, in data networks, congestion causes throughput to deteriorate, high error rates, and long delays. In the early Internet for example, network congestion caused the backbone throughput to drop by an order of magnitude from 32kbit/s to 40 bit/s [36] which forced researchers to rethink the protocols applied on a per-user level. Recent research on data center networks focuses again on congestion control and resource allocation in order to minimise delay [10].

1.1 Resource allocation problems in networks

There are two conceptually different ways to relax congestion: increasing the network capacity at congested resources or improving the match between network capacity and demand. Increasing network capacity is usually expensive and time-consuming. In addition, it is only possible if maximum capacity has not been reached yet. However, often communication networks and transit networks have enough capacity to serve the user demand on average. Spikes in utilisation during peak hours or on highly popular, but short stretches of the network, require overprovisioning of big parts of the network. Changing the allocation of user demand to available resources by diffusing peak-time utilisation spikes or changing the response of an Internet protocol to loss and delay is usually a cheaper and more flexible option to combat congestion. For example, many cities use congestion charging in core areas or on busy motorways in order

to relax congestion and lower delay and crowdedness. Similarly, the Internet congestion collapse in the 80s was the initial spark for congestion control algorithms that still regulate Internet flows on a per-user basis today.

Resource allocation algorithms generate the best match of user demands to available resources. They can be classified into centralised and decentralised solutions. In Internet routing, if all user demands were known, a centralised scheduler or planner could work out the best assignment of user flows to paths within the network so that no transmission link is overloaded. However, due to the size of the Internet and the uncertainty about how users respond to a change in path conditions, centralised resource allocation is often impractical except in small networks owned by a single entity.

Decentralised solutions treat the system as a collection of interacting users that respond to feedback. They rest on two main building blocks: Firstly, it is essential to know how users behave when they are exposed to feedback information. For example, how much charge should be asked for on a motorway in order to halve the utilisation? What is the response of an Internet protocol when the transmission is very lossy and error-prone? These user models are the heart of modelling complex systems in order to control system behaviour and to avoid congestion. Secondly, user feedback must be designed in a way that the overall system objective is met when users respond to the feedback. Overly aggressive feedback can lower the congested stretches by too much and too little feedback does not change it by enough.

In 1947, George Dantzig invented an algorithmic way to solve centralised resource allocation problems with linear programming. It was still an intricate process, involving hundreds of person hours to solve small-sized problems with fewer than 100 variables [23, 24]. In addition, in a real world system, it is unlikely that one can get a list of all users, resources, constraints, and utility functions which makes a centralised solution difficult to generate. Further research in the following decades explored ways to decentralise the solution process of centralised problems. General methods were identified, for example primal and dual decomposition methods [86]. These methods lower the amount of communication needed between users in order to obtain the global optimum, but they still require users who use the same resources to communicate directly. In 1998, the first connection between one of the Internet's most prominent congestion control protocols, the Transmission Control Protocol (TCP), and decentralised resource allocation problems was made [38]. Essentially, it was shown that TCP solves an important class of global resource allocation problems, maximum utility problems while only using feedback information such as delay and loss rate that is readily available for each user. This insight prepared the way for the reverse direction. Researchers were now able to formulate interest-

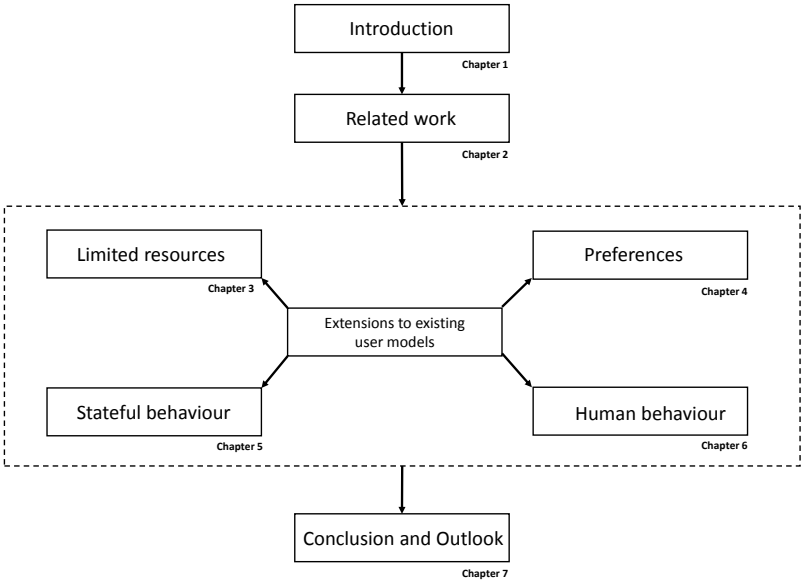


Figure 1.1: Structure of the thesis

ing global problems and derive TCP versions that solve them in a distributed way. However, for about a decade, this was predominantly a theoretical exercise. Only recently, some of the theoretical concepts have been put into practice.

1.2 Contributions of this thesis

This thesis is about modelling users in constrained networks. We argue that commonly used user models do not cover some of the particularities that can be found in many application areas and we explore ways to enrich them and make them more realistic. Each chapter addresses one user model, both in terms of theory and with an application or feasibility study to confirm its use in practice. Figure 1.1 shows the graphical structure of the thesis.

Chapter 2 gives an overview of optimisation-based user models for networks. This chapter introduces the problem and notation used and shows the advantages of optimisation-based user models in sample transport and data network applications. Furthermore, it describes the problems approached in this thesis in more detail and surveys related work that is common to all following chapters. Note that each of the following chapters contains an additional section on related work if it is not common to any of the other chapters.

The main chapters of this thesis are chapters 3 to 6, each treating one extension to existing user models. All of the main chapters depend on chapter 2, but they can be read independently of each other. Chapter 3 studies the case of resource-limited users. The optimal solution for a decentralised user algorithm can be overwhelming for users to process. For example, in a large

network, the optimal solution might require the user to keep track of changing feedback information on millions of paths. Every path that is managed by a protocol incurs some setup cost in terms of memory which can reach a limit if too many paths have to be observed. Secondly, data flows are not infinitely divisible, the smallest unit is a packet with standard size 1500 bytes. A typical data connection has a maximum speed of hundreds of packets per second. Keeping track of feedback on a path can only be achieved by sending data on the path. The maximum connection speed effectively limits the feedback on individual paths as the number of paths that has to be considered is large.

Some user models have to include preference in choices and we explore how to handle them in chapter 4. Standard TCP models use throughput as the only path differentiating metric, but there are many applications where throughput is not the main factor. For example, in real-time live-streaming with deadline constraints on delay, high delay variance on some paths can cause many packets to time out. Sending on a high throughput path where most of the packets time out, can lead to less goodput than sending on an alternative path with less throughput and lower delay.

Chapter 5 introduces users that do not always respond similarly to the same feedback – their behaviour is stateful. Mobile data interfaces for example switch between different power states depending on load and by using timers. An interface with a high power state while transmitting that returns into a low power state after 30 seconds of inactivity might be trapped in a high power state if some application keeps sending a keep-alive packet before the timer ends. In this case, another interface that has higher energy consumption during transmission, but shorter timeouts can result in lower overall energy consumption. We derive an energy conserving user by modelling the problem as an optimisation problem over time, a Markov decision process.

Finally, chapter 6 points out the main influences on user models in societal networks, networks of people in everyday life. For example, a commuter makes a choice every morning about which mode of transport to use, at which time to depart and which route to take. Trains and buses have limited capacity. On Singapore's Mass Rapid Transit (MRT) network, the biggest trains made up of six carriages can carry up to 1920 people. Transport networks give natural feedback information back to the commuter in the form of congestion and crowdedness. Still, the network is in overload for some periods of the day and the negative feedback experienced by the commuter is not enough to relax it. This leads to the question how human behaviour influences decision making in constrained networks. We study user models with human behaviour in societal networks by analysing the outcome data of a six month long, large-scale deployment of

a commuter reward program, Insinc (Incentives for Singapore's Commuters), part of Stanford University's societal network program. Insinc had over 20,000 active users at the end of the first phase of the trial, which makes it possible to get a very detailed picture of user responses to system feedback. We show how user models for human users in networks exhibit similarities to models used in data networks, but also point out major differences. While human users are also stateful and have path preference, we explore further influences on the user model such as dependencies of user behaviour on the behaviour of their social connections, and effects of repetition and user adaptation. Chapter 7 summarises the thesis and points out future work.

Parts of this thesis are published as follows:

- Data Center Networking with Multipath TCP, Costin Raiciu, Christopher Pluntke, Sebastien Barre, Adam Greenhalgh, Damon Wischik, and Mark Handley, Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets 2010
- Saving Mobile Device Energy with Multipath TCP, Christopher Pluntke, Lars Eggert, and Niko Kiukkonen, Proceedings of the Sixth ACM International Workshop on Mobility in the Evolving Internet Architecture, MobiArch 2011
- Improving Datacenter Performance and Robustness with Multipath TCP, Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley, Proceedings of ACM SIGCOMM 2011
- Exploring TCP-friendly Real-time Streaming on Multiple Paths, Christopher Pluntke and Miguel Rio, London Communications Symposium, 2011
- TCP-friendly Rate Control for non-TCP Multipath Flows (best poster award), Christopher Pluntke and Miguel Rio, poster and mini paper, student workshop, ACM CoNext 2011
- Method for Multipath Scheduling Based on a Lookup Table, Christopher Pluntke, Niko Kiukkonen, Lars Eggert, patent application, publication date 24/10/2013
- INSINC: A Platform for Managing Peak Demand in Public Transit, Christopher Pluntke and Balaji Prabhakar, p. 31 - 39, Special Issue of JOURNEYS, September 2013

The following paper is currently under submission:

- Multipath Resource Allocation with per-path Utility Functions, Christopher Pluntke, Raul Landa, and Miguel Rio, under submission.

Chapter 2

Modelling users in networks

In resource allocation problems users are entities who take actions on choices. Different users' choices interact with each other through a shared network of constrained resources. For example, the choice of when to commute to work in the morning affects the load on the road which influences other commuters on the road in terms of crowdedness and delay.

User behaviour in networks is commonly modelled as an optimisation problem that each user solves over different choices exposed to them. For example, a driver on the road might aim for minimum delay, or a protocol on the Internet tries to maximise throughput. This has several advantages: Firstly, methods from distributed optimisation make it often possible to aggregate user problems and describe them in form of a global system problem. Vice versa, the technique can be reversed and one can derive user problems from global system problems. This makes it possible to predict and control global system behaviour through decentralised users. Secondly, optimisation problems are an elegant way to describe intelligence of a user without specifying algorithmic details. For example, the assumption of a utility-maximising user fully specifies the outcome of the user choice. Modelling can then concentrate on estimating or specifying the utilities perceived for the choices that each user faces. Thirdly, optimisation-based models are broadly applicable and consistent in different areas: Users in transportation problems can be described in a similar way as protocols in Internet flow control or behaviour of humans that maximise their own utility when taking choices.

This chapter gives an overview of previous work on optimisation-based user models for networks in section 2.1. In section 2.2, we show where these user models fall short in practical applications and how they should be extended to better reflect how users work.

2.1 Describing user behaviour as optimisation problems

This section describes different approaches to optimisation-based user models. We start with a transport example that shows how user modelling works and how individual user problems can be aggregated into a system-wide optimisation problem. Subsequently, we review a user model for Internet congestion control, the Transport Control Protocol (TCP). We describe literature about optimisation-based modelling for TCP that shows that TCP users cooperatively solve a network-wide maximum-utility optimisation problem. This makes TCP an important user model that serves as a foundation for decentralised, cooperative users. Finally, we show how users in societal networks, i.e. networks of humans in everyday life, can also be modelled as optimisation problems to illustrate the broad applicability of the optimisation-based approach.

2.1.1 Network model and notation

Consider a network of nodes $\mathcal{N}$ and links $\mathcal{L}$ between nodes. A path p in the network is defined as a set of adjacent links from source node s to a destination node d. A user $s \in \mathcal{S}$ is a source-destination pair that is associated with a set of paths, $\mathcal{P}_s$. The set of all paths is called $\mathcal{P} := \bigcup_{s\in\mathcal{S}} \mathcal{P}_s$. We also assume that a path can be uniquely associated with a user and $s(p)$ denotes the user associated with path p. $\mathcal{P}(l)$ gives the set of all paths that use link $l \in \mathcal{L}$.

If the number of paths per user cannot be greater than one, we call the user a singlepath user, otherwise it is a multipath user. In that sense, a system that cannot use more than one path is a singlepath system, but a system that chooses to use only one path although it could use more, is a multipath system. The amount of traffic or throughput on path p is denoted by $x_p \geq 0$. y_s captures the total throughput for user s and it can be calculated by summing up the throughput on all their paths:

$$y_s = \sum_{p\in\mathcal{P}_s} x_p. \tag{2.1}$$

The amount of throughput on a link $l \in \mathcal{L}$ is captured by z_l and it can be calculated by summing up all route throughputs that use that link l, i.e.

$$z_l = \sum_{s\in S} \sum_{p\in\mathcal{P}_s : l\in p} x_p. \tag{2.2}$$

We distinguish between networks with link capacity constraints and networks with additive link costs. In networks with link capacity constraints, we require that the throughput of each link l be smaller than its capacity $C_l \geq 0$

$$z_l \leq C_l \tag{2.3}$$

In a network with additive link costs, each link l has an associated cost function depending on the link throughput $c_l(z_l)$. For example, the more commuters travel on a stretch of the road,

the longer it takes to cross it, so delays can be described by the edge costs. c_p denotes the total cost of path p and it is defined as sum of delays along the route:

$$c_p = \sum_{l \in p} c_l(z_l). \tag{2.4}$$

2.1.2 Users in road networks

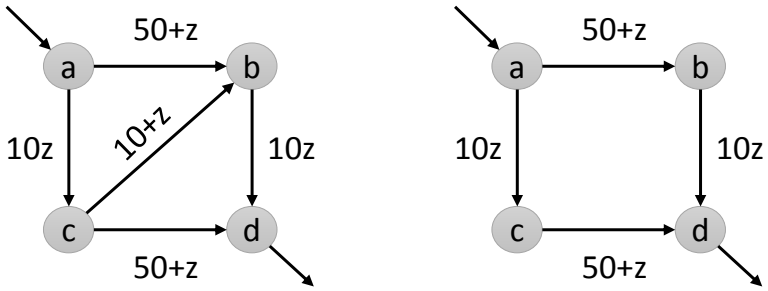
One of the earliest examples of an optimisation-based user model in a network problem with interaction between users was published in 1952 by John Glen Wardrop [81] in the context of road network modelling. Wardrop's paper, although being an early publication in the area, is described here in detail because it illustrates the benefits of optimisation-based user modelling and it thereby sets the ground for the problems addressed by this thesis. Wardrop contrasts the bottom-up view of a system consisting of greedy individual drivers who minimise their own journey time by choosing the fastest path available, against the alternative view of a top-down system designer who has a global system problem in mind and cares about the performance of the entire system, for example minimising the average delay in the network. A typical question for a system designer is: Do the users solve the system problem? If not, how should I change the perception or utility of their choices such that they do?

Wardrop studies a network with additive link costs (cf. section 2.1.1) where the cost function, $c_l(z_l)$, reflects delay on a road. He points out that user and system view can lead to different equilibria: "(1) The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route. (2) The average journey time is a minimum." [81, p. 345]. He further explains that the "first criterion is quite a likely one in practice, since it might be assumed that traffic will tend to settle down into an equilibrium situation in which no driver can reduce his journey time by choosing a new route" [81, p. 345]. The "second criterion is the most efficient in the sense that it minimizes the vehicle-hours spent on the journey" [81, p. 345]. Wardrop's first and second criterion are also called first and second Wardrop equilibria.

Wardrop's second equilibrium is formulated as a system problem: minimise average cost over all users in the network. Wardrop's first equilibrium is a user problem that describes a state where if a path has non-zero traffic flow, then it is no more congested than the other paths that a user has access to. Intuitively, this is a description of a greedy user minimising their own travel time. Mathematically, it is described by a vector $(x_p : p \in \mathcal{P})$ that satisfies for all paths $p \in P$:

$$x_p > 0 \implies c_p \leq c_{p'}, \forall p' \in \mathcal{P}_{s(p)} \tag{2.5}$$

We noted that one of the benefits of modelling users as optimisation problems is that we



(a) Average cost: 92 (b) Average cost: 83

Figure 2.1: Braess' paradox

can use the user model to make conclusions about the global system behaviour. In the case of Wardrop's network, the first question to ask is whether the first and the second equilibrium give the same outcome. Wardrop found that the two formulations describe different optimisation problems and it is possible to construct a simple case that illustrates the difference. Braess' paradox was first described by Dietrich Braess in 1969 [18]: Consider the network shown in figure 2.1a. Commuters travel from node a to node d and incur link delays indicated by the link cost functions next to the links. There are three paths available to travel from a to d: $p_1 = \{(a,b),(b,d)\}, p_2 = \{(a,c),(c,d)\}, p_3 = \{(a,c),(c,b),(b,d)\}$. The total amount of traffic between a and d is six units and everyone wants to travel, i.e. the six units have to be split on the three paths. For example, if all traffic is sent on p_1, the delay experienced would be $c_{p_1} = (50+6) + 10 \cdot 6 = 116$. A greedy driver would in this case shift traffic from p_1 to p_2 and p_3 since their delay is 0 at the moment and drivers would continue to shift flows until delay on all three paths is equal. The equilibrium for greedy drivers solves as two units of traffic on each of the three routes giving a per-route cost of 92. In this allocation, no local change in flow can improve the experienced delay.

Now we remove the cross link and the three hop path p_3 that uses it as shown in figure 2.1b. The equilibrium rates are now achieved when three units of traffic is sent each on path p_1 and path p_2. Surprisingly, this causes the average delay to go down to 83 per route although capacity is subtracted from the network. Obviously, this could have been achieved in the first case as well, by forcing the drivers on the two-link routes, but the greedy nature of the users prevented this solution.

Braess paradox illustrates two points: Firstly, Wardrop's first and second equilibrium do not always match, i.e. greedy users do not always solve the right system problem. Secondly, the system outcome resulting from greedy users is not monotone when capacity is added to or subtracted from the network. In other words, an additional motorway that was supposed to relax

congestion might even increase average congestion when users are greedy. In game theory, the quotient of the system performance from greedy users and the globally optimal system performance is called *price of anarchy*, first introduced by Koutsoupias and Papadimitriou [48] and later extended to computer networks [68].

Kelly [37, p.866] attributes the following result to Beckmann et al. [13]. It characterises the existence of a Wardrop equilibrium as the solution to a particular system problem.

Theorem 2.1. *For strictly increasing cost functions $c_l(.)$, the solution of the following optimisation problem describe a system in a Wardrop equilibrium and conversely, a system in Wardrop equilibrium is described by the following optimisation problem:*

$$\begin{aligned} \text{minimise} \quad & \sum_{l\in\mathcal{L}} \int_0^{z_l} c_l\left(z'\right) dz' \\ \text{over} \quad & x_p \geq 0, z_l \in \mathbb{R} \\ \text{subject to} \quad & y_s = \sum_{p\in\mathcal{P}_s} x_p, \quad \forall s \in \mathcal{S} \\ & z_l = \sum_{s\in S} \sum_{r\in\mathcal{P}_s : l\in p} x_p, \quad \forall l \in \mathcal{L}. \end{aligned} \tag{2.6}$$

From this theorem it is clear why greedy users do not minimise average delay: they solve a system problem that is not connected to average delay. The ability to make this connection between individual users and global system behaviour illustrates the power of an optimisation-based user model. Next, we go one step further and show how decentralised Internet congestion control leads to a different decentralised user model; a user model that solves a more useful global system problem.

2.1.3 Users in Internet congestion control

In Internet congestion control, users are source-destination pairs running a protocol that selects a sending rate. The most prominent congestion control protocol is called TCP (Transmission Control Protocol) [61]. The reason why TCP is important in this context is that it is possible to describe TCP users as an optimisation problem on user-level. Similar to Wardrop's model, it is possible to derive a system optimisation problem that TCP solves if all users in the network use it. While the system problem for Wardrop's first equilibrium did not correspond to a sensible system problem, in TCP it does: we will see that TCP solves a utility-maximisation problem on system scale while being completely decentralised on user level. Conversely, this property makes TCP very attractive as an algorithm to solve global optimisation problems on system scale. We will now review the literature that leads to these results starting from singlepath TCP without path-choice to multipath TCP with path-choice.

Users without path choice

TCP is a transport layer protocol that was established in order to provide reliable end-to-end connections between two hosts and that regulates the sending rate, i.e. flow control. TCP ensures reliability of transmission by requiring acknowledgments packets (ACK packets) from the receiver for all successfully received packets. Missing packets can be requested again by the receiver by monitoring the sequence numbers of arriving packets. Flow control is provided by TCP's congestion controller originally proposed by Van Jacobson in 1988 [36]. ACK packets serve both as feedback for lost data and measurement tool for round-trip time delay (RTT) by keeping track of the time from sending the original packet until the acknowledgement arrives. TCP controls the sending rate at the sending user by maintaining a congestion window, w, which is the maximum number of unacknowledged packets that can be in flight between sender and receiver. The window size can be used to control throughput. The larger the window size, the more unacknowledged packets can be in flight, leading to more throughput. Throughput x [packet/s] is related to the window size by the formula $x = w/\text{RTT}$. Van Jacobson suggested to adjust the window size as follows:

TCP:

- Each ACK increase w by $1/w$.
- Each loss event, decrease w by $w/2$.

TCP's window update rule increases the window size linearly by one packet each RTT as long as ACK packets are coming in, in the case of a loss event, the window size is cut in half, this is also called linear increase/multiplicative decrease behaviour. It leads to TCP's characteristic saw-tooth like throughput over time.

Adjusting the window size in this way, the equilibrium window size given loss rate d evaluates as [85]:

$$w = \sqrt{\frac{2}{d}} \text{ [packets]} \tag{2.7}$$

In terms of throughput, this equation changes to

$$x = \frac{\sqrt{2}}{\text{RTT}\sqrt{d}} \text{ [packets]} \tag{2.8}$$

for a static RTT.

TCP and its congestion controller have evolved over time and several refinements have been suggested to both the protocol and Van Jacobson's original congestion controller. Popular

versions that are used at the moment are Cubic [33] in Linux and Compound TCP [75] in Windows.

In a paper from 1998, Kelly et al. [38] identified a utility maximisation problem as the system problem that is solved by singlepath users applying TCP for regulating the sending rate of Internet flows. TCP is fully decentralised and only uses packet drops and delay as feedback – properties that can be estimated by a user without any additional communication. So, the feedback needed for the decomposition to work is readily available to each user. This makes TCP a good starting point to find similar, fully decentralised user problems with corresponding utility maximisation problems as system problems.

Later research turned the problem around: starting from a utility-maximisation system problem, it is possible to derive a TCP-like algorithm that solves the system problem in a decentralised way [73, 72]. To do so, consider the following system problem in a network with constrained link capacities and singlepath users. Jointly maximise the sum of all user utilities while ensuring that no link is overloaded:

$$\begin{aligned} \text{Maximise} \quad & \textstyle\sum_{s\in\mathcal{S}} U_s\left(x_s\right) \\ \text{over} \quad & x_s \geq 0 \\ \text{subject to} \quad & z_l \leq C_l \quad \forall l \in L \end{aligned} \tag{2.9}$$

where x_s is the flow rate on the path p associated with user s. The utility function $U_s(.)$ relates the flow rate experienced by user s and the utility that the user associates with that rate. In the context of utility maximisation, a general family of suitable functions is given by the notion of α-fairness introduced by Mo and Walrand [53]:

$$U_s(x) = \begin{cases} w_s \frac{x^{1-\alpha}}{1-\alpha} & \text{if } \alpha \neq 1 \\ w_s \log x & \text{otherwise} \end{cases} \tag{2.10}$$

where w_s is a weight parameter for user s and $\alpha \geq 0$ controls fairness properties of the flow allocations. Note that for $\alpha = 0$ and unit weights, the optimisation problem is the same as maximum throughput. For $\alpha \to \infty$ it tends to max-min-fair flow allocations. As we will see later, TCP uses $\alpha = 2$.

The system problem (2.9) is straightforward to formulate and solvable by standard convex programming methods (see [17] for details on convex programming). It is a centralised problem that assumes global knowledge about the entire network. This means that to obtain a solution, the entire network has to communicate with a central control node which recomputes the rates for each flow and feeds the optimal flow rates back to the sending nodes. Since flows are not

static, this computation has to be done in regular time intervals. Running this computation on Internet scale would be prohibitively time-consuming.

To decompose the system problem into user problems, the standard way is using a barrier approach [38], that is incorporating all hard constraints as additive soft constraints into the objective function. The following derivation contains only the main results without going into details. We give a more detailed derivation for a more general system problem in section 4.2.

In the case of system problem (2.9), we need to add a barrier function for every link. The penalty function $B_j(.) > 0$ for link j is chosen in a way that it is close to 0 if a link is underutilised and sharply increases if the link capacity is violated. A soft-constrained version of system problem (2.9) is:

$$V(x) = \sum_{s \in S} U_s\left(x_s\right) - \sum_{l \in \mathcal{L}} B_l\left(z_l\right) \tag{2.11}$$

where the objective is to maximise $V(x)$ over $x_s \geq 0$. With the right choice of the penalty function $B_j(.)$, the optimality conditions for (2.11) are [38]

$$\forall s : x_s \left(U_s'(x_s) - d_s\right) = 0 \tag{2.12}$$

where d_s is the loss rate experienced by user s. The loss rate is high when links are overloaded and it is small otherwise which makes it a good indicator of the path conditions. This leads to the following system of differential equations whose equilibrium solution correspond to the optimality conditions (2.12).

$$\forall s : \dot{x_s} = k_s(x_s)\left(U_s'(x_s) - d_s\right) \tag{2.13}$$

where $k_s(.)$ is non-negative, increasing, and continuous. These differential equations describe an algorithm that drives the change in user utility, $U_s'(x_s)$, and path loss rate, d_s, towards balance. Kelly et al. [38] show global stability for (2.13), i.e. in a network where all users adjust their sending rates according to (2.13), the algorithm converges to the equilibrium solution from any starting point of initial rates.

It is possible to interpret algorithm (2.13) as TCP. [72] explains the appropriate choice of parameters which we reproduce here: Using $\alpha = 2$ and weights $w_s = \frac{2}{\mathrm{RTT}_s^2}$ for the α-fair utility function and exchanging the rate based by a window size based equation, the differential equation becomes

$$\dot{w_s} = w_s\left(\frac{1}{w_s^2} - \frac{1}{2}d_s\right) \tag{2.14}$$

To get TCP's window dynamics, the window size needs to increase by 1 unit every RTT if there are no drops and decrease by $w_s/2$ after a drop where drops have the rate $d_s w_s/\mathrm{RTT}_s$.

By multiplying the right-hand side of equation (2.14) by w_s/RTT_s, a differential equation or 'fluid' version of TCP emerges:

$$\dot{w_s} = \frac{1}{\text{RTT}_s} - \frac{1}{2} d_s w_s^2 / \text{RTT}_s \tag{2.15}$$

Note that multiplying the right hand side does not change the equilibrium point of the system.

In summary, for TCP, user problems with the right feedback match up to a system problem that makes sense from a system performance view since it is a global maximum utility system problem. This is useful in two ways: firstly, it helps us to design feedback. Secondly, it reassures us that there is a unique optimum that the system is converging to. This provides the basis for a stable system with a coherent objective.

Users with path-choice

Multipath TCP (MPTCP) is an extension to TCP where users have path-choice. A simple way to extend singlepath TCP to multipath operation is by applying a so-called 'uncoupled' controller. In the uncoupled case, a multipath user runs singlepath TCP on each subflow. Uncoupled is unable to load balance between paths and does not shift congestion, but it serves as a reference algorithm that is straightforward to implement.

The main MPTCP proposals by Kelly et al. [39] and Han et al. [34] are optimisation-based. They approximately solve a global system problem of the form:

$$\begin{aligned} \text{Maximise} \quad & \textstyle\sum_{s\in\mathcal{S}} U_s \left(\sum_{p\in\mathcal{P}_s} x_p \right) \\ \text{over} \quad & x_p \geq 0 \\ \text{subject to} \quad & z_l \leq C_l \quad \forall l \in L \end{aligned} \tag{2.16}$$

where x_p are subflow rates associated with user s. Similarly as before, we require positive flow and link capacity constraints. Optimisation problem (2.16) looks very similar to the optimisation problem solved by single-path TCP (2.9), however there is a main difference: Instead of a single path per user as in (2.9), in this version, user s has acces to multiple paths $\mathcal{P}_s$. Since the data stream on a single path is usually called a 'flow', we refer to each of the data streams for a multipath user as 'subflows' of a user. The option to assign multiple subflows per user enables load balancing and routing within the network. Users whose subflows use overloaded links can now shift their traffic away to less loaded paths on different subflows.

Kelly et al. [39] propose a multipath extension to scalable TCP [40] that can achieve the load balancing in a distributed way. The resulting fluid model is of the form

$$\frac{d}{dt} w_p(t) = \frac{w_p(t)}{\text{RTT}_p} \left[a - b w(t) d_p \right] \tag{2.17}$$

where w_p is the window size on path p, d_p is the drop rate on path p, $w(t) = \sum_{r \in \mathcal{P}_{s(p)}} w_r$ is the sum of window sizes on all paths for that particular user, and a, b are parameters that control the increase and decrease behaviour of the algorithm. The paper proves stability for a linearised controller using the right choice for a and b. We will refer to the multipath congestion controller for TCP introduced by Kelly et al., equation (2.17), as the 'coupled' controller.

Han et al. use a similar fluid model where the right-hand side is multiplied by $\frac{w_p(t)}{w(t)}$:

$$\frac{d}{dt} w_p(t) = \frac{w_p(t)}{\text{RTT}_p} \left[a \frac{w_p(t)}{w(t)} - b w_p(t) d_p \right] \tag{2.18}$$

In [84], the authors identify three problems with these fluid models when implemented as a packet-based protocol.

1. *TCP-friendliness on shared bottleneck links*: MPTCP's increase-decrease behaviour is different from standard TCP. When multiple subflows of the same MPTCP connection share an intermediate link, they can push harder than a single TCP flow. For public acceptance, the authors aim to ensure fairness against legacy TCP.

2. *Flappiness*: The algorithm randomly flips paths when multiple paths have the same drop rate. The problem was identified as a measurement problem. The rate controller always selects the minimum drop path as preferred path. Since drop rates are determined by measurements and measurements are not perfect, it is unlikely that all paths are found to have the same loss rate even in the case that they do. The algorithm will then select the minimum loss rate path for transmission. As the utilisation grows, the drop rate on this path is likely to increase until another path has lower drop rate. At this point, the traffic flaps from the previous path to the new path. As pointed out by the authors, the rate controllers from [39] and [34] try to equalise congestion, so the case where multiple paths have the same drop rate is fundamental.

3. *Multipath deployment incentive*: By updating a flow from standard TCP to multipath TCP, the user ought not suffer. MPTCP should try to ensure that its throughput on multiple paths is at least as high as the best single-path flow on the same paths.

The proposed multipath rate controller is called the 'Linked Increases Algorithm' (LIA), later refined in [85] as:

- On receiving an ACK packet on path p at user s, increase w_p by

$$\min_{\mathcal{A} \subseteq \mathcal{P}_s : p \in \mathcal{A}} \frac{\max_{r \in A} w_r / \text{RTT}_r^2}{(\sum_{r \in \mathcal{A}} w_r / \text{RTT}_r)^2} \tag{2.19}$$

- On a loss event on subflow p, decrease the window w_p by $w_p/2$.

The algorithm is designed to work at an operating point between coupled congestion control which is flappy, but shifts congestion, and uncoupled congestion control which is not flappy and does not shift congestion. LIA shifts traffic but it is not overly flappy, hence solving problem 1. The equilibrium window sizes for LIA on path p at user s evaluate as

$$w_p = \sqrt{2a}\frac{1/d_p}{\sqrt{\sum_{r\in\mathcal{P}_s} 1/d_r}} \tag{2.20}$$

where a is a parameter that controls aggressiveness. a is adjusted dynamically:

$$a = w\frac{\max_{r\in\mathcal{P}_s}\sqrt{w_r}/\mathrm{RTT}_r}{\sum_{r\in\mathcal{P}_s} w_r/(\mathrm{RTT}_r)}. \tag{2.21}$$

where $\sum_{r\in\mathcal{P}_s} w_r$.

The authors argue that this choice of a ensures TCP-friendliness on shared bottleneck links (problem 2) and provides the multipath incentive (problem 3).

A follow-up paper showed that LIA is not Pareto-optimal [45]. The authors highlight two main problems: Firstly, when TCP users switch to MPTCP with LIA, it might negatively impact the throughput of the remaining TCP users while not increasing the throughput of the users who switched. Secondly, they show fairness problems of MPTCP with LIA when competing with standard, singlepath TCP. Their solution is called 'Opportunistic Linked Increases Algorithm' (OLIA). It is also based on Kelly et al.'s coupled controller and the authors show that it also provides TCP-friendliness on shared bottleneck links and gives a multipath deployment incentive like LIA while not showing any flappy behaviour. Packet-level simulations and testbed implementations support the claims.

Internet standards are mainly developed by the Internet Engineering Task Force (IETF). It has started a multipath TCP working group to explore how the multipath TCP protocol should be designed to work in practice. Publications by the IETF are called Request For Comments (RFC). Currently the multipath TCP working group has published several RFCs: RFC 6182 [28] summarises high-level design decisions for Multipath TCP and RFC 6824 [29] outlines the changes to legacy TCP in order to enable multipath operation. RFC 6356 [64] describes the MPTCP congestion controller with LIA in detail. RFC 6897 [69] specifies how the application layer handles MPTCP flows. Also, a threat analysis on how to attack MPTCP has been made in RFC 6181 [12]. The revised multipath controller with OLIA has also been submitted as an Internet draft [44].

Note that we described multiple algorithms that are all published under the name 'MPTCP'. When this thesis refers to MPTCP, we mean a multipath rate controller running the LIA congestion control algorithm.

2.1.4 Users in general societal networks

In daily life, individual people make distributed decisions while interacting with each other. This network of interacting people is called a societal network. Chapter 6 of this thesis studies user models with human behaviour in societal networks. Just like for users as protocols, human users can be described by optimisation problems: Behavioural models for choice in human decision making is typically explored in discrete choice theory [76]. Discrete choice theory builds on the assumption that users who have a finite number of choices between multiple options associate different utility with each of their options and then deterministically choose the one with the highest utility. As a researcher, it is impossible to know exactly what the utility functions look like which introduces a random component. Research in the area of societal networks is focussed on estimation of user utilities and unexpected influences on the user utilities.

In a discrete choice model, user s has a number of choices $\mathcal{C}_s$ and selects the choice with the highest utility value: $\text{argmax}_k\ \mathcal{U}_{s,k} = \text{argmax}_k\ (\mathcal{V}_{s,k} + \epsilon_{s,k})$ where $\mathcal{V}_{s,k}$ is the utility of choice $k \in \mathcal{C}_s$ as it is perceived by user s and $\epsilon_{s,k}$ is a random error term for that particular user and choice option. The probability of a particular choice $k \in \mathcal{C}_s$ being taken by user s can be computed as the probability that its utility function evaluates higher than all other possible choices:

$$\mathbb{P}(\text{choice } k \in \mathcal{C}_s \text{ is used}) = \mathbb{P}(\mathcal{U}_{s,k} \geq \mathcal{U}_{s,i} \quad \forall i \in \mathcal{C}_s) \tag{2.22}$$

The most common techniques used to estimate discrete choice models are multinomial logit and probit models. Multinomial logit models make these three assumptions: Firstly, they assume that the utilities are given by independent and identically distributed Gumbel distributions. Secondly, they assume homogeneity of responses for all individuals and thirdly, homogeneity of error covariance for all individuals. These assumptions make it difficult to account for dependencies that arise when modelling choice behaviour. For example in [47], a study about departure time change using logit-type choice models, time is discretised into 15 minute intervals to quantify by how much commuters are willing to change their schedules. Since neighbouring time slices are not independent, justifying the use of a logit type model in this context is difficult. It is mainly used due to its computational simplicity.

A model that does not require the assumption of utilities being independently and identi-

cally distributed is the multinomial probit model. The dependencies are captured by the error terms which are assumed to be multivariate normal distributions. This model is more difficult to fit since the number of parameters to be estimated grows like the square of the choice set. It can only be applied to small scale data.

We mention discrete choice models here as a motivation for optimisation-based user models. We will also use a choice model for evaluating the effect of Insinc, albeit a model that is simpler in construction having only two levels: desired behaviour and undesired behaviour. Choice models of this form can be estimated with logistic regression and we will review the tools needed in more detail in section 6.3.

2.2 Limitations of user problems

It is intuitively clear that a mismatch between theoretical user model and user behaviour in practice will result in a system problem that does not mirror the actual system behaviour. For that reason, this thesis investigates several weak spots of optimisation-based user models commonly used in research and points out ways how to mitigate the problems. This section describes the main areas of interest.

2.2.1 Users have limited resources.

User models derived from system optimisation problems do not make assumptions about individual users' resources. In practice, however, having too many path choices is problematic. Every path can constantly change its state which means that a minimum amount of traffic has to be sent on it in order to probe it; however, too many paths means too much probing traffic. Also on implementation level, every path has to be managed, data structures have to be maintained, and memory has to be reserved which makes it difficult for a user to manage too many choices. For these reasons, the study of path selection on the user side is central for a working system. The better the path selection algorithm works and the less paths are needed to achieve performance close to what theory predicts with a high number of paths, the easier it is for adoption in practice.

One way that is often used to analyse the system behaviour for multiple users transmitting a static amount of traffic is the well-known balls and bins model. In a balls and bins problem, one studies the system behaviour of n balls being placed into m bins. Questions about the placement algorithm tie back to resource allocation: e.g. how the balls should be placed into bins with minimal information about the entire system in order to minimise the maximum load in any bin. This model is closely connected to flows in a network by interpreting a bin as a route choice. A balls and bins model where each user has access to k bins out of m and selects one to

place the ball into can be interpreted as a user who has access to k routes out of m and selects one to transmit on. An interesting question to ask is how the minimum throughput behaves under random assignment of balls to bins.

Key et al. in [42, 43] compare coupled and uncoupled congestion control in a balls and bins model. Each user chooses paths (bins) with unit capacity uniformly at random from a given candidate set. They study a system where the number of users is $a \cdot n$, the number of resources is n and each user selects k paths uniformly at random. As n grows, the worst-case allocation for uncoupled congestion control scales like $k^2 log(log(n))/log(n)$. For $k = 1$ they remark that this is equal to the worst-case allocation in classical balls-and-bins where n balls are placed into n bins. For coupled congestion control in the same scaling, they prove that the worst case allocation is bounded away from zero as n goes to infinity. A similar question has also been studied by Mitzenmacher in [52]. He shows that in a balls and bins model where each user selects k bins at random and puts a ball into the least loaded bin, the worst-case allocation scales as $1/log(log(n))$ for $k > 1$ which also goes to zero as n tends to infinity. As pointed out by Key [43], the main difference between Mitzenmacher's and Key's model is that in the latter, the user has to decide which path to take on arrival and the decision cannot be changed.

This suggests the following conclusion about load balancing with independent paths:

- Coupled rate control outperforms uncoupled rate control.
- The performance of all load balancing schemes greatly improves if there is path choice (at least two paths per user).
- The performance of greedy least-loaded and coupled with at least two paths ($k > 1$) does not depend on the number of paths used as the system becomes large.

The main problem with these results are their independence assumptions between paths and the ideal network layouts as balls-and-bins. We will study the question about the performance of a resource-limited multipath controller that can only access a limited number of paths in networks with heavily dependent paths. Data center networks need decentralised control in order to respond quickly to changing demands and to maximise their performance. In chapter 3 we tested the MPTCP controller with LIA in different data centers of various sizes and show how dependencies between paths change the prediction that theory provides.

2.2.2 Users have preferences

In general, users are not indifferent about their paths. Even in an Internet setting, some paths might be more attractive for example due to lower delay, higher throughput, less energy consumption, or less money to be paid for their usage. To illustrate the problems in such a system,

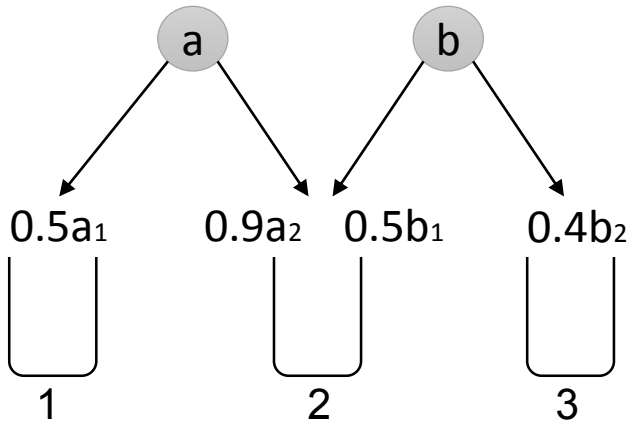


Figure 2.2: An example where users have to communicate to determine the best allocation of rates in order to maximise the average system utility.

consider the following example of balls and bins with simple user preferences: two balls are to be placed into three bins and every ball can choose from two bins to be placed into. This is illustrated in Figure 2.2. The bin represents a resource with unit capacity. If only a single ball lands in a bin, it will get throughput $x = 1$, and multiple balls share the capacity equally. In addition, users have preferences and they weight their choices differently. The resulting share of resource i for user a is called a_i and similarly for user b as b_i. In difference to traditional balls and bins, user a values bin 1 as 0.5 and bin 2 as 0.9, while user b values bin 2 as 0.5 and bin 3 as 0.4. Bin 2 is the most desirable for both users. We call the resulting allocation of throughput times user preference the utility for that user and that bin. Utility for user a is $u_a = 0.5a_1 + 0.9a_2$ and utility for user b is $u_b = 0.5b_1 + 0.4b_2$. The average system utility is $\hat{u} = (u_a + u_b)/2$. The system problem is maximise average utility. An obvious user problem is maximise user utility.

In the single-path case, each user places one ball into one bin. If both users greedily choose bin 2, each of them will only receive half the share of the resource times what they value it for. In this case, the total utility of the system is $\hat{u} = (0.5 \cdot 0.9 + 0.5 \cdot 0.5)/2 = 0.35$. In the multipath case, users are allowed to split the balls as long as the sum of assigned pieces equals 1. Assuming there is a system in place that lets both users share resources fairly, they will end up placing half a ball each into the middle bin since it is the best choice for each of them. Placing more does not change the utility since each bin has unit capacity. The left-over half ball does not have any other choice than being placed into bins 1 and 3. This allocation gives a total system utility of $\hat{u} = (0.5 \cdot 0.5 + 0.5 \cdot 0.9 + 0.5 \cdot 0.5 + 0.5 \cdot 0.4) = 0.575$. However, had both users cooperated, they could have achieved mean utility of $\hat{u} = 0.65$, by letting user a access bin 2 exclusively, while user b only uses bin 3 although bin 2 is more attractive. We see that

enriching the user model with per-path (or per-bin) utilities makes sense in a model-world. We will derive a decentralised load balancer with per-path utility functions in chapter 4 and give practical applications of the enriched model for Internet streaming traffic.

2.2.3 User behaviour is stateful.

Optimisation models for TCP use the assumption that users only have a limited sense of state and that the optimisation problem governing TCP's global system is a 'well-behaved', concave optimisation problem. As we have showed in the first part of this chapter, these assumptions make it possible to apply methods from convex optimisation to derive the solution of the problem and even to decompose the global system problem in order to derive user problems that solve the global system problem. However, this model of smooth response with limited state does not hold in all application areas.

The introduction of MPTCP makes it possible to shift TCP sessions between different interfaces without breaking. This raises the question of how to shift data flows on mobile, battery-powered devices dynamically between different radio interfaces in order to improve the user experience, for example to save energy. Different radio interfaces differ significantly in terms of energy consumption, transmission capacity, delay and range, as well as usage cost and reliability. The overall mobile user experience could be greatly enhanced if it was possible to seamlessly leverage the flexibility afforded by these different radio technologies for Internet communication. This is not straightforward: a device can naively try to always use the network path through the "best" radio interface based on its current situation, but especially for longer-lived sessions, that path is unlikely to stay "the best" as time passes. Over time, a device will end up with active sessions across multiple radio interfaces, which is highly undesirable – it wastes energy to keep a radio ready for transmission or reception especially under light communication loads, even more so for multiple radios. A naive policy that always prefers to use a path via a radio interface that is already in use by other sessions is also problematic, because a session may be established over an inferior path, e.g. one with less capacity.

In order to maximise battery life time, it is important to realise that the per-bit energy cost for radio interfaces varies with load [51]. In some devices, Wifi is very energy-efficient for the low throughput range while 3G is more efficient if the throughput is high. However, even on a multi-homed device that is able to seamlessly shift connections, multipath scheduling is not straightforward. Consider for example a data transmission with varying throughput. Data interfaces often have stateful energy models. Let the energy model of the data interfaces consist of three states: transmission energy if data is being actively transmitted, high stand-by energy after transmission has ended, and low idle energy after the stand-by phase is over. In general,

the stand-by phase is very short for Wifi and considerably longer for 3G, whereas the energy consumption for the idle phases of 3G and Wifi are similar. If we greedily shift the data stream to the radio interface that would be most energy-efficient for a given load, we do not take the timeout behaviour of radio interfaces into account. Assuming that throughput continuously oscillates between high and low, this policy alternates between 3G and Wifi to exploit the interface with the best energy to bit ratio for transmission. However, every time it switches from 3G to Wifi, the stand-by energy for 3G adds to the transmission energy for Wifi and vice versa. If the changes in throughput oscillate very quickly, this policy can easily waste more energy for stand-by, compared to using even the least energy-efficient interface for the entire transfer. It is therefore not possible to simply shift all sessions to the radio interface that would be most energy-efficient if fully utilised, without paying attention to the actual throughput rates of those sessions and detailed energy models for the interfaces. In other words, we know that it is sometimes more energy-efficient to use one interface, sometimes it is better to use another interface, but how can we generate a load balancer that allows flexible handover of flows between interfaces to maximise those savings? In chapter 5, we will use Markov decision process models as a way to model stateful user behaviour in this multipath setting with stateful energy models.

2.2.4 User models with human behaviour are complex.

Societal networks usually arise in common life situations where constrained resources are involved, for example road or energy networks. The aim of societal network research is to find ways to shift demand or change habits in order to control the network. This aim is tackled by three means: fine grained measurements making use of current technology such as mobile computing devices, providing fast feedback to the user using an online platform, research in feedback methods such as positive random rewards, and social effects such as peer pressure. The Stanford Center for Societal networks has so far conducted four field trials of reward based societal networks with the aim to understand the main factors of a user model for human users in constrained networks.

In general societal networks, a change in policy is motivated by a system objective to be achieved by the change. To ensure that the intended and the actual system objective match up, the human behaviour has to be modelled as a central part of the user problem. Incorrect models lead to unintended system outcomes. For societal networks, research has only just begun and we are not at a point yet where we can write down a final version of a user model. Chapter 6 gives an analysis of data from the Insinc project, a societal networks project, aiming to increase off-peak usage of the Singaporean train-based, public transport service with the help of incentives. The project was run with about 20,000 participants over six months in the first half of 2012.

The size of the data set collected allows us to give a very detailed look at user utility functions and at factors that influence them. We will see that human behaviour is complex to model by showing for example that users' behaviour is influenced by their social connections and that they respond in a stateful way to stimuli.

2.3 Summary

The main point to take away from this chapter for the remainder of this thesis is: Modelling users as optimisation problems is a general tool that has advantages in terms of model simplicity, applicability to different areas, and the possibility to draw conclusions about the system as a whole by looking at the individual user without the need of large-scale simulation. We gave examples from three areas: Public transport, communication networks, and societal networks. Wardrop's transport model showed how to derive system behaviour from individual user behaviour. It also illustrated that greedy users who try to maximise their own benefit can drive the whole system in a suboptimal state. Subsequently, we introduced TCP's system and user problem: A TCP user maximises a global system problem in a decentralised way. Finally, we saw that human decision making can also be modelled as an optimisation problem: humans take choices with the highest utility.

In the second part, we pointed out four weaknesses of commonly used optimisation based user models. Users have limited resources and cannot consider too many choices – especially in a large scale system such as a data center where paths are not independent, prohibitively many choices might be needed for high utilisation of the system. In addition, users have preferences and not every path is alike for them. Users can also be stateful which commonly arises when energy-models come into play. Incorporating human behaviour into user models requires to reverse engineer how the human mind works when exposed to incentives in a real world setting. Previous research showed that user decisions in societal network depend on unexpected factors, for example the number of their social connections. Each of these weaknesses will be addressed in form of an extended model in one of the following chapters.

Chapter 3

Users have limited resources

Intuitively, having choice is good – more routing choice increases flexibility. In this chapter, we study the application of MPTCP as multipath user protocol in data center networks. Data center networks are server farms consisting of thousands of hosts and providing a large number of paths between any two communicating hosts. MPTCP is highly attractive to use in data centers since MPTCP drives the system towards a maximum utility solution of global system problem (2.16), promising high throughput and fair flow allocations without the need of a centralised controller. Too much path choice also has negative sides: in a data center, a user is a protocol with access to limited memory and limited computational power. Managing a large number of paths at an end-user means setting up data structures for management of the paths which consumes computational and memory resources. In addition, probing traffic has to be sent on each path to sense its drop and delay properties. If a high number of paths has to be probed, the user wastes too much throughput without performing any load balancing.

We show that user resource limitations have to be taken into account when modelling the system and point out how current theoretical work on resource limited users does not fully predict the effects that arise when users communicate in realistic network topologies. The main question we ask in this chapter is: How many paths are needed to get high utilisation in big data centers where the network topology is highly structured and paths often overlap and introduce dependencies?

The work presented in this chapter is based on a collaborative project on MPTCP in data centers with Costin Raiciu, Sebastien Barre, Adam Greenhalgh, Damon Wischik, and Mark Handley [65, 63]. The project consisted of a technical performance evaluation of MPTCP in the setting of data centers as originally proposed by Costin Raiciu and Mark Handley. The evaluations for the project were split into three parts: large-scale evaluations with fixed-point models, packet-based simulation, and implementation. The results shown in this thesis focus on the first part, evaluations with fixed-point models, since they are the own work of the author of

this thesis (with exception of figure 3.2 which contains packet-level results marked as collaborative work). The results were obtained by a large scale fluid solver implemented by the author of this thesis that allows to analyse the behaviour of data center-scale topologies and flows. The architecture of this fixed-point solver is described in detail in section 3.3. Further tools and evaluations from the collaborative project that involved packet-level results were mainly the work of Mark Handley and Costin Raiciu. They are not a part of this chapter and they are only mentioned for the sake of completeness. Parts of this chapter have been published in [65, 63] where a deeper coverage of packet-level results and a small-scale implementation can be found. In addition, the idea of extending the study of MPTCP to users with constrained resources which is the central topic of this thesis chapter was not part of the original collaborative project.

The remainder of this chapter is organised as follows. We review the technical background in section 3.1 and present our model in section 3.2. Section 3.3 describes the simulation setup and methodology used to generate the results for the evaluation presented in section 3.4. A summary of our findings in section 3.5 concludes this chapter.

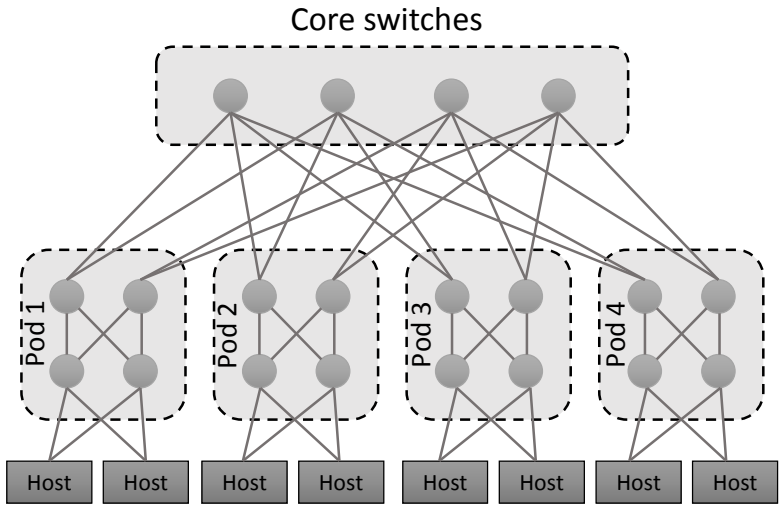
3.1 Technical background

This section gives an overview of techniques for routing and resource allocation currently in use or proposed for use in data centers. This material is very technical, but also essential when building a good model of the system. The reader may consider to skim it on first reading and come back to it as a reference.

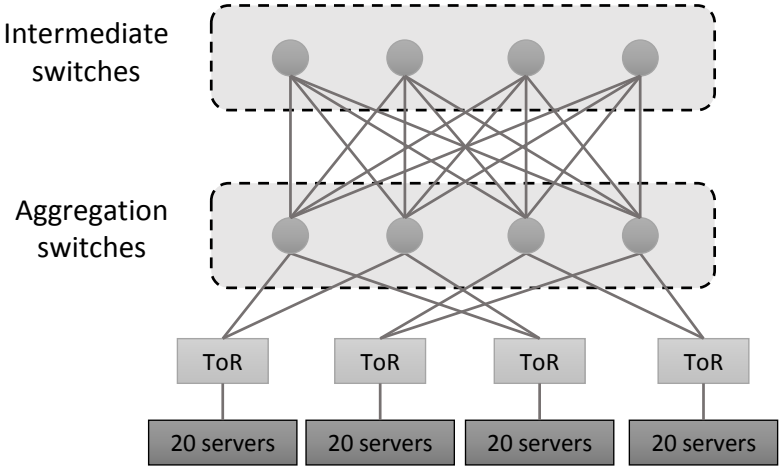
3.1.1 Physical topology

Data center networks are usually built and maintained by commercial operators. This makes it difficult to get insight into the network layout used in real-world deployments. In [8], the authors describe best practices for traditional data center networks layouts according to a survey they conducted: Traditional data center topologies have a hierarchical design with a core tier, an aggregation tier and an edge tier (three tier design) or core and edge tier (two tier design). The two tier design can service 5000 - 8000 hosts, while the three tier design supports up to 25,000 hosts. The network is supported by switches with 1 GigE (Gigabit Ethernet) ports and 10 GigE ports. The edge switches typically being connected to the hosts with GigE ports (48-288 GigE ports per switch) and to the aggregation or core switches with 10 GigE ports. The aggregation and core switches operate in the 10 GigE regime (32-128 10 GigE ports per switch). All switches allow full speed, simultaneous communication between all connected hosts. The cost for building such a network to support 20,000 hosts was around $37 million in 2008.

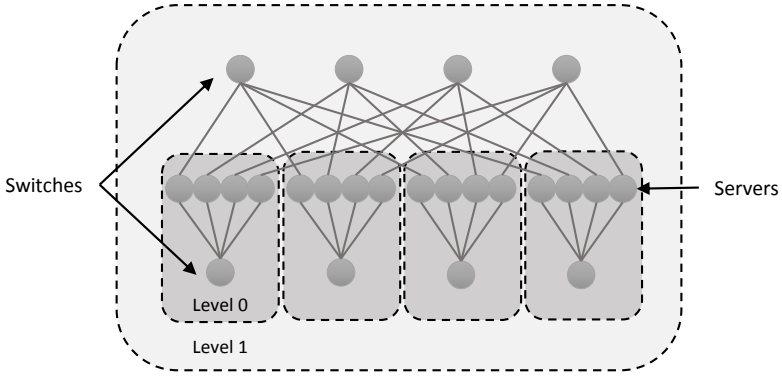
In recent years, many alternative data center topologies have been suggested that scale



(a) FatTree (k=4)



(b) VL2 ($n = m = 4$)



(c) BCube$_1$ ($n = 4$)

Figure 3.1: Data center topologies

beyond the traditional design and are often cheaper to build. We describe three of the major topologies: FatTree, VL2, and BCube. FatTree and VL2 are Clos [21] topologies which are provisioned such that there is enough interconnection capacity for each host to connect to any other host with full interface line speed. BCube on the other hand is a recursively defined, hypercube-like topology.

FatTree topologies have been proposed as data center topologies in [8, 56]. In the design of [8], FatTrees are made up of many commodity switches (1 GigE ports) instead of big core switches in the core network as in the traditional design. A k-ary FatTree consists of two layers: an aggregation layer and a core layer. The aggregation layer consists of k pods, i.e. two layers of $k/2$ switches. A k-port switch in the lower layer is connected to $k/2$ hosts and to the $k/2$ switches in the upper layer. The upper layer is also connected to the $k/2$ core switches each. The core layer consists of $(k/2)^2$ k-port core switches. A k-ary FatTree supports up to $k^3/4$ hosts and offers $(k/2)^2$ equal length paths between any two hosts in different pods.

VL2 [31] relies on the concept of resource pooling, i.e. making the individual resources in the network appear as a single pooled resource. Its design aims to make VL2 behave like a single big switch and to isolate the performance of the individual ports. This is achieved by using a two-tier Clos architecture with n intermediate switches and m aggregation switches. It supports $n \cdot m/4$ Top of Rack (ToR) switches with 20 servers per ToR switch. ToR switches are double-homed with 10 GigE to two aggregation switches, and each aggregation switch is connected to all intermediate switches with 10 GigE links. 1 GigE links are only present from the end hosts to the ToR servers. The topology gives $2 \cdot m \cdot n$ equal length paths between all source destination pairs that are in different ToRs.

BCube [32] uses a recursively defined hypercube-like topology where the end hosts themselves are connected directly to intermediate switches. A BCube$_k$ consists of n BCube$_{k-1}$ and n^k n-port switches. In total, BCube$_k$ makes use of n^{k+1} servers, $k+1$ levels of switches, and n^k n-port switches per level. Hosts are addressed by k bit labels and switches are addressed by a k bit switch id and the level of the switch. A switch in level i connects two servers whose ith bit is different. The routing algorithm then relays packets by correcting one bit per hop, alternating between switches and intermediate hosts along the path until all digits equal the destination host.

3.1.2 Routing

Data center topologies typically provide more than one path between hosts. The aim of the routing algorithm is to find and expose these paths to the network. There are several standard ways to enable multipath routing in a network. The Open Shortest Path First (OSPF) [22, 55] routing

protocol has the option to spread incoming traffic on multiple paths if the paths from the current router to the destination are of equal cost. This is called equal-cost multipath or ECMP [55, 16]. In OSPF, path costs are calculated by summing up link costs along the path. The link cost unit can be set at the router adjacent to the link and by default it uses unit cost per link. The main problem of ECMP is that it only splits flows between equal minimum cost paths and paths with higher costs are not accessible for load balancing. ECMP load balancing is used for example by VL2. ECMP can also be combined with Multi-Protocol Label Switching (MPLS) [67]. MPLS enables the router to set up so-called label-switched paths (LSPs) through the network, tunnels from the source router to destination on a predefined path. The cost for each tunnel can be set freely. Setting the cost of different length tunnels to an equal value at a router forces OSPF with ECMP to spread the load equally on all different length tunnels. This enables flexible load balancing on any paths between two routers. VLAN (Virtual Local Area Network) switching [35] is an alternative way to load balance. VLANs are used as a way to isolate a channel in a network consisting of switches. Ethernet frames tagged with a specific VLAN id are forwarded to their destination along the shortest path in a shortest path tree anchored at a specific switch in the network and stretching to every node (IEEE 802.1D). By using a shortest path tree as topology, each path in the network is unique which eliminates routing loops. The Multiple Spanning Tree Protocol (MSTP, IEEE 802.1s) enables load balancing by setting up multiple VLAN spanning trees anchored at different root switches. More recently, alternatives to VLAN routing with the spanning tree protocol have been proposed for data centers where more flexible routing is needed. In addition, many switches allow fixed forwarding rules between different ports and interfaces (e.g. [3]). This allows the setup of arbitrary topologies by adding all switches along a route to the same VLAN and installing the appropriate forwarding rules on these switches[1]. In Ethernet backbone architectures (Metro Ethernet), Generalized Multi-Protocol Label Switching (GMPLS) [49, 15], has been introduced to enable explicit path control similar to MPLS on the routing layer. In Metro Ethernet architectures, labels such as wavelengths or time-slots are used to specify the VLAN.

3.1.3 Path selection

Data centers do not expose all possible paths to the end users. They typically make a random selection of paths and keep it static (as used for example in VL2 [31]). Static randomised assignment has only little control overhead since no adaptive rerouting of flows is required. It can be implemented at router level by telling OSPF how many equal-cost paths to expose to the

[1]This way of achieving fixed routing with VLAN tagging was pointed out to me by Richard Clegg during the defense of this PhD.

user. However, it has been shown that static path assignment with ECMP and singlepath flows has serious shortcomings. In FatTree networks, static, randomised path assignment neither performs well in terms of throughput nor in terms of fairness [9]: The paper shows that using ECMP routing the total network throughput is less than the theoretical optimum. This effect is significant, in many cases giving throughput reductions to less than 50% of the optimum for different sized FatTree ($k \in \{4, 16, 32\}$) networks with various traffic matrices. The authors suggest a simulated annealing heuristic that recomputes the optimal flow assignment and then reassigns flows in periodic intervals. Other designs also favour the usage of a centralised flow scheduler [8] due to the shortcomings of static routing.

3.1.4 Congestion control

We have already reviewed literature about singlepath and multipath TCP in section 2.1.3. This section provides additional material that is related to TCP modifications for data centers.

TCP is the Internet's main congestion algorithm and as such many applications still use standard singlepath TCP. TCP ensures that the load per link is matched to the available capacity and links are shared in a fair way. Recently, several modifications to TCP for data centers have been proposed to address queue build-up [10], incast congestion (many flows connecting to the same destination host) [10, 78], and deadline constraints [77].

DCTCP [10] is a version of TCP refined for data centers. It uses singlepath flows and no ultipath extensions have been proposed yet. While traditional TCP halves the sending rate when a loss is detected, DCTCP responds to marked packets with Explicit Congestion Notification (ECN) marks [66]. Packets get marked by ECN-aware routers to signal congestion back to the senders before the queue at the router builds up and packets would have to be dropped. DCTCP uses feedback from ECN marked packets to determine a more fine-grained amount by which the window size has to be decreased. The sender keeps an exponentially moving average, α, of the proportion of marked packets per window. If an ACK packet with set ECN mark is received and a fraction α of packets were ECN marked, the window size is updated as follows:

$$cwnd \leftarrow cwnd \times (1 - \alpha/2) \tag{3.1}$$

That means that if the queue sizes in the network are long and a high fraction of packets are ECN marked, the window size is cut in half like regular TCP. If the queue sizes are small and only a small fraction of packets are ECN marked, the window size is only slightly reduced. Deadline-aware data center TCP [77] builds on DCTCP, but changes the α value such that flows with far away deadlines halve the window size like regular TCP on a loss event while flows with imminent deadlines only slightly reduce it, to give them a higher chance to meet

the deadline. [78] suggests to complement standard TCP with fine-grained timers to reduce the minimum retransmit timeout in order to reduce TCP incast congestion.

In summary, all proposals focus on modifying standard singlepath TCP and no solution uses multipath congestion control to load balance flows from the end host and solve the problem of low utilisation without a centralised scheduler. For this reason, we evaluate the use of MPTCP as transport protocol in this chapter in order to gain multipath routing capability on a per-user basis.

3.1.5 Traffic

The applicability of load balancing on multiple paths depends on the type of traffic that is present in data centers. If all flows are short, then there is not enough time to balance load since the network demand changes rapidly. In [8], the authors argue that Internet traffic is long-tailed and flows separate into elephants and mice, i.e. there are a small number of flows that are responsible for most of the traffic. In [31], measurement results from a 1,500 node cluster are presented. They were collected over two month in a live data center and confirm the elephant and mice theory. In this particular data center, only 1% of the flows are large (more than 100 MB long), but 90% of bytes are accounted for by flows between 100 MB and 1 GB. The authors also show that traffic matrices are very unstable, making it difficult for centralised flow rerouting schemes to react quickly. After clustering the traffic matrices in 100sec intervals into 40 representative traffic matrices, the authors analyse the holding time for a particular traffic matrix. They show that up to the 60th percentile, the holding time is 1, i.e. most of the time the traffic matrix changes every 100sec. Alizadeh et al. [10] present a measurement study of 6000 servers from a data center over the course of a month. They report about traffic characteristics: query traffic is characterised as short flows between 1.6 to 2 KB in size that are sensitive to latency. These flows are control flows that start processes on remote servers or collect results. Background traffic consists of short message flows and long update flows, the long flows between 1MB and 50MB in size. Similar to the previous measurement study, they also show that most bytes are contributed by the long flows and that statistical multiplexing of the long flows is low.

These results suggests that it is necessary to carefully route long flows in the network because they contribute the most bytes. Since long flows last in the order of seconds, that makes them long enough to be affected by load balancing (currently proposed flow schedulers run up to every 100ms [9]).

3.2 System model

We reviewed theoretical results about cooperative rate control with a limited number of independent paths in balls-and-bins models in section 2.2.1. Theory predicts that there is a significant gap between having path choice and no path choice, but the gap between having two paths available per user and more than two is small. This so-called 'power of two choices' was introduced by Mitzenmacher in [52]. In real network topologies paths can overlap with each other, introducing dependencies into the path selection process. These dependencies are not random, they are highly structured and governed by the underlying network topology.

Simply avoiding path overlap is not a solution and it can overly limit the capabilities for load balancing. For example, a host might be connected to a top of rack server that only has one link to an aggregation switch. That limits the number of collision free paths to one and effectively prevents this host from load balancing its flow in the remaining network. In addition, it is usually not the case that edge hosts know the path that a packet is going to take.

To study the problem of resource limited users in a realistic data center model, we have to fix four components: The network topology that governs dependencies between paths, the traffic matrix, the protocol that balances traffic on multiple paths for each host, and an algorithm that returns a small number of candidate paths from the set of all available paths for multipath routing. As topology, we chose FatTree, VL2, and BCube data center topologies with up to 11,520 hosts. When two hosts from different racks in a data center want to exchange data, the connection goes through a hierarchical network of switches where collisions can happen at any layer of the hierarchy.

Section 3.1.5 showed that elephant flows dominate today's data center traffic. Elephant flows last for seconds and give a host-based protocol enough time to balance traffic. We study two different traffic matrices with long flows for the evaluation: a random permutation traffic matrix where every host connects to exactly one other host. Hence, every host can send with maximum rate if the data center network has enough capacity and if there is no congestion. This traffic matrix avoids incast collisions at hosts where multiple hosts try to connect at the same server. It is used here as a worst-case scenario that only reflects internal network congestion. We also studied a random connection traffix matrix where k hosts connect to a randomly chosen host each. This traffic matrix permits incast collisions and it lets us study lighter loads ($k <$ #hosts) and overloaded networks ($k >$ #hosts with reselection of hosts).

Traditional approaches to load balancing in data centers make use of standard TCP without path choice, often in combination with a centralised controller that maps the singlepath flows to underutilised paths in the network in order to give high throughput in the entire network [9].

The centralised controller operates under very tight constraints. It has to update its internal view of the network very quickly since even long flows only last in the order of seconds. It has to be very efficient in recalculating optimal flow assignment. If the calculation takes too long, the flows to be reassigned will have finished transmitting already. In addition, it is a single point of failure and has to be protected from outages and failures which could potentially affect the entire network.

Instead of centralised control, giving the choice between different paths directly to the end hosts is very attractive since it increases flexibility at the side of each user. Unfortunately, neither of the two predominant Internet transport protocols (TCP and UDP) support multiple paths well: once a communication session using these protocols has been established over one path, *i.e.*, between one local IP address and one remote IP address, that session is "stuck" to that path until it finishes. It is not possible to "shift" an active communication session to a different path or enable it to use multiple paths simultaneously. In addition, if the local or remote IP address becomes unavailable, *e.g.*, because an interface loses connectivity, all sessions over that network path are terminated. It is therefore difficult to exploit the flexibility offered by multiple paths with existing protocols. The root cause of these issues is the lack of support for multipath support in the traditional Internet transport protocols. MPTCP (cf. section 2.1.3) enables the concurrent use of multiple network paths between two end systems for a single TCP connection and it can also be used to shift active connections from one path to another. We use MPTCP with LIA congestion control for the evaluations in this chapter.

Finally, the path selection protocol we apply is: select k random paths between sender and receiver. The reason for this choice is its simplicity which enables straightforward implementation for example using VLAN switching or ECMP-based routing on MPLS tunnels outlined in the section 3.1.2. We study if this choice of topology, load balancing algorithm, and path selection protocol results in a system that achieves high throughput and gives fair throughput allocations and how these metrics depend on path-limited users. If resource limitation of users in the presence of highly dependent paths in data center topologies has only a small effect on the number of paths needed to get high throughput that is fairly allocated between users, we have identified a decentralised way to achieve high throughput in a data center network without the need of a centralised scheduler.

3.3 Methodology

In order to evaluate the system, it is common practice to use a tool that simulates the transmission of the individual hosts on a per-packet level. That means, every packet is simulated as it

is sent out by the sender, it is queued at intermediate links on the way to its destination, and it can be lost if a queue overflows. To study data centers in realistic scale, we cannot rely on packet-level simulation since even the fastest available packet-level simulators cannot simulate thousands of hosts in reasonable time. For this reason, the author implemented a fixed-point solver for large-scale data centers. The fixed-point solver does not simulate single packets, instead it predicts the behaviour of the system governed by MPTCP's throughput equation and lossy links modelled as large queues when the system has reached equilibrium.

The fixed-point solver works as follows:

1. For every path p in the system, initialise flow rates x_p to an arbitrary positive value and loss rates $d_p \leftarrow \epsilon_p$ where $0 < \epsilon_p < 1$. For every link l in the network, initialise the initial drop rate $d_l \leftarrow \epsilon_l$ where $0 < \epsilon_l < 1$.

2. For every path p, calculate per-path loss rates from per-link loss rates: For a flow traversing a series of links along path p and of per-link losses, $d_1, ..., d_k$, the approximate per-path loss rate, d_p, is given by $d_p \leftarrow 1 - \prod_{i=1}^{k}(1 - d_i)$ assuming independence between links.

3. Solve the throughput equations for the LIA version of MPTCP (equation 2.20) given per-path loss rates to get candidate throughputs x_p^{cand} for every path. Update the current flow rates with an exponentially weighted moving average: $x_p \leftarrow (1 - \alpha)x_p + \alpha(x_p^{\text{cand}})$.

4. Calculate the loss rates at link l, d_l, given the sum of rates for flows using the link, x_l, and the transmission capacity of the link, c_l. We assume that all queues are large and calculate the approximate per-queue loss rate as $d_l \leftarrow \max\left(0, \frac{x_l - c_l}{x_l}\right)$.

5. Calculate the deviation in throughput as compared to the last assignment. If the maximum difference is small, stop and output the equilibrium rates x_p for every path. Otherwise, go to 2.

In [85] the authors show that the throughput of MPTCP in a packet-based simulation is predicted well by its throughput formula. In order to confirm the validity of the output of our fixed-point solver in large networks, we compare the packet level simulator and fixed-point solver on equally sized topologies. Figure 3.2 shows a comparison of sorted per-connection throughputs on a FatTree topology with 128 hosts. The fixed-point approximation usually closely matches the packet-level simulation. The small differences in throughput we notice are mostly due to missing timeouts and retransmissions that are not modelled by the fixed-point equations. Also, the fixed-point solver uses a static RTT of 1ms for every path whereas the RTT in the per-packet

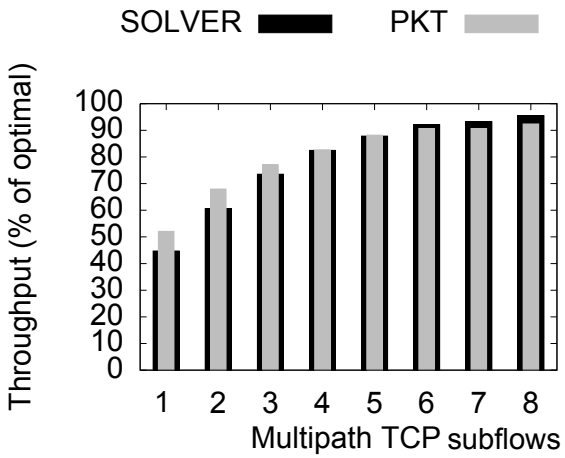


Figure 3.2: Comparison between packet and fluid level simulations.

simulation can vary due to queue build-up at links. This is the only result in this chapter that relies on per-packet modelling, all other results were generated with the fixed-point solver.

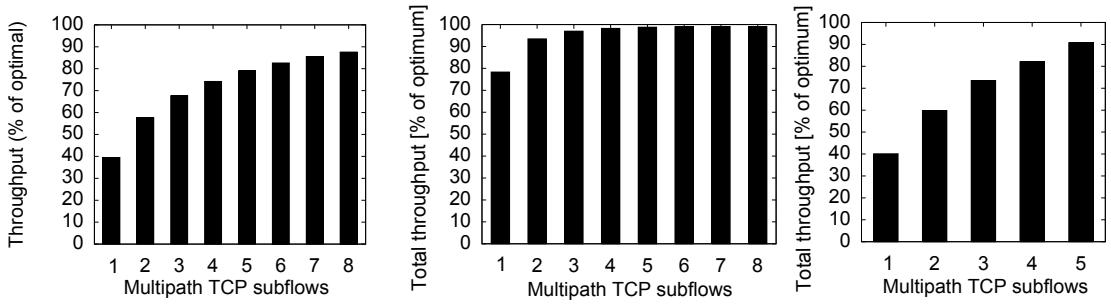
We found that in order to make the fixed-point solver scale up to large data centers with over 10,000 nodes, we needed to change the rate update parameter α in step three during the execution of the algorithm. We chose a high value of $\alpha = 0.1$ at the beginning and decreased it to $\alpha = 0.01$ after 1000 iterations. This modification let the rates quickly converge into a neighbourhood of the final values for the first 1000 iterations and use the subsequent steps to refine the result. For a small data center network of about 100 nodes, the fixed-point solver converges in seconds, for larger data centers with 10,000 nodes, it takes up to an hour to converge. This is still much faster than packet-based simulation and lets us study data centers at a realistic scale.

3.4 Evaluation

The evaluation is structured into three parts. First, we test if two random paths per user are sufficient to get close to optimal throughput as predicted by the theory described in section 2.2.1 (which assumes all paths are independent). Secondly, we evaluate scaling effects. This is important since a small number of subflows per user might be sufficient for a fixed size topology. However, if the number of subflows grows with the size of the network, data centers above a certain size might be out of reach of resource limited users. Finally, we test MPTCP's fairness properties and their dependence on the number of paths per user.

3.4.1 Number of paths

We compared the performance of fixed FatTree with 8192 hosts, VL2 with 11,520 hosts and BCube with 1024 hosts with random permutation traffic using the fixed-point solver described in section 3.3. Figure 3.3 shows that as expected there is a gap between using singlepath and



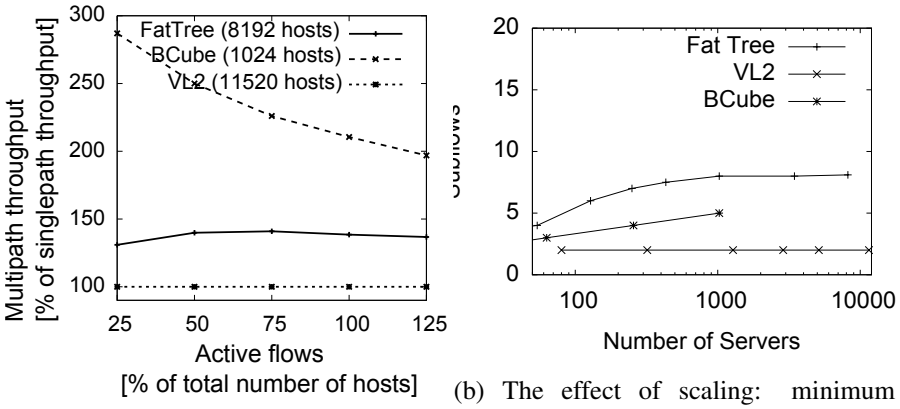
(a) FatTree (8,192 hosts) (b) VL2 (11,520 hosts) (c) BCube (1024 hosts)

Figure 3.3: Throughput for FatTree, VL2 and BCube data centers with a permutation traffic matrix using different number of subflows.

multipath congestion control. However, the total throughput depends on the topology used. In VL2, singlepath TCP achieves 77% throughput while FatTree and BCube fare worse. In addition, our experiments showed that two paths are not enough to get high total throughput in general. We found that FatTree needs eight subflows to reach close to 90% utilisation of the network, VL2 needs only two subflows and BCube needs five.

The reason for the dependence on the topology lies in the design of the core network in the different examples. To achieve 100% throughput with one subflow, the network would have to make sure that no link in the core network is overloaded. The intuition behind VL2's good performance is the fact that it uses links in the core that have higher capacity than the host's maximum sending rate. When multiple flows collide on these core links, they can share it without reducing their rate if there is some slack and only slightly reduce their rate if the link is oversubscribed. To reduce the throughput for a flow by 50% in FatTree, it only has to collide with another flow on a core link. Since they could both send with 1 Gbps and the maximum link speed on core links is 1 Gbps, both of them share the link fairly and get 0.5 Gbps each. If 11 flows share a core link in VL2, all of them only lose $1/11$th capacity. The more a topology is provisioned like a single switch with high capacity links in the core, the better singlepath randomised load balancing works since load balancing imbalances are not punished as severely as in topologies with small core links. However, since big core links are expensive and their failure would have a severe impact on the network, it is attractive to use topologies with a higher number of small capacity links in the network while load balancing with a managable number of subflows.

We also tested networks in under- and overload by using the random connection traffic matrix with the highest number of subflows we could simulate (5 subflows for BCube, 16 subflows for FatTree and VL2). We varied its parameter k to generate offered loads between 25%



(a) The effect of load: Multipath advantage under different network loads

(b) The effect of scaling: minimum number of subflows needed to get 90% throughput for permutation traffic

Figure 3.4: Utilisation experiments: Load and scaling.

and 125% and evaluated the throughput difference between singlepath and multipath operation in each case. The results are compiled in figure 3.4a. We found that for BCube the performance difference between MPTCP throughput and singlepath TCP is almost 300% for 25% load. This is because BCube hosts are multi-homed, and the interface capacity is higher than the host's fair share in a fully loaded network. In a lightly loaded network, each host can get more throughput than a 1 GigE interface would allow in the other topologies. As the network gets more loaded, the throughput advantage vanishes as expected due to capacity limits. VL2's throughput is already high and only increases moderately while FatTree resides in between the two extremes. In all networks, multi-homing can be highly beneficial when the network load is not close to capacity. This observation was later used in [63] to propose a multi-homed version of FatTree.

3.4.2 Scaling effects

Assume that we grow the network in size and scale the number of users such that the network is always at maximum load under permutation traffic. If in this scaling the number of subflows increases without bound, it is infeasible to implement MPTCP based load balancing in very large networks since too many subflows are required. Figure 3.4b shows how the size of the data center network influences the minimum number of subflows needed to get at least 90% utilisation with a permutation traffic matrix. For BCube, we could not find any saturation within the limits of topology sizes that we could simulate. For topologies with up to 1000 hosts, BCube needs up to 5 subflows per user. On the other hand, FatTree and BCube topologies saturate at 8 and 2 subflows respectively. FatTree seems to be a good trade-off: It does not need high capacity core links that VL2 needs which makes it cheap to build. In addition, even large FatTree networks do not need more than eight subflows in order to reach high system utilisation.

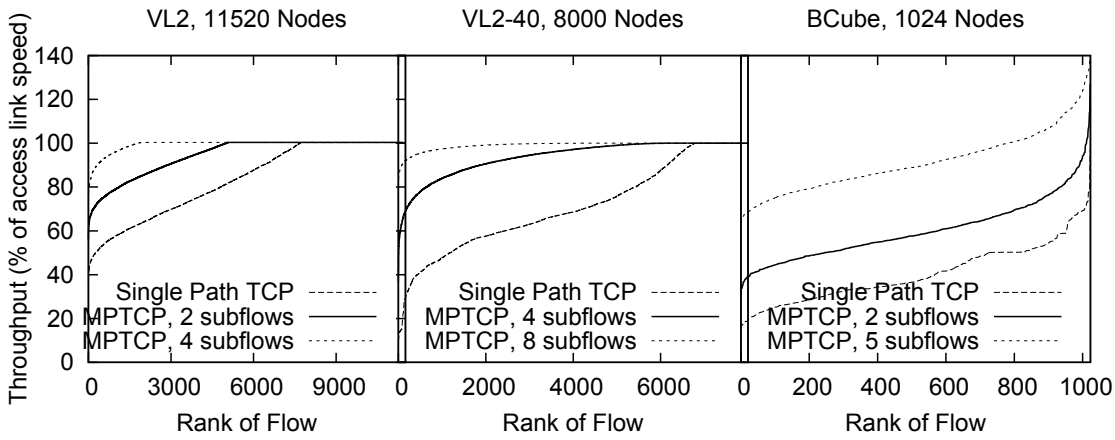


Figure 3.5: Throughput distributions for a sample run in VL2 and BCube.

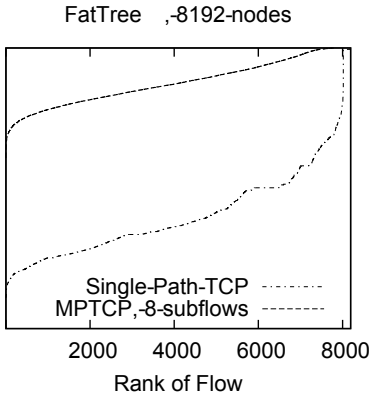


Figure 3.6: Throughput distributions for a sample run in FatTree.

3.4.3 Fairness

Fairness between connections is especially important for data center applications. Popular applications that run on large clusters of hosts in data centers are highly distributed by using for example a MapReduce [25] framework. MapReduce is a programming paradigm that allows decentralisation of data processing by using a central master node that splits a job and sends it to several worker nodes. When the worker nodes have completed their task, they send the result back to the master node which has to combine their results. Unequal flow throughputs can lead to long delays until the last worker node has received the data or finished sending the result which makes fair flow allocations with little variance in throughput a highly desirable goal.

We explore fairness with two measures: Jain's fairness index and the average worst-case throughput. Jain's fairness index is a number between 0 and 1 where 1 indicates high fairness

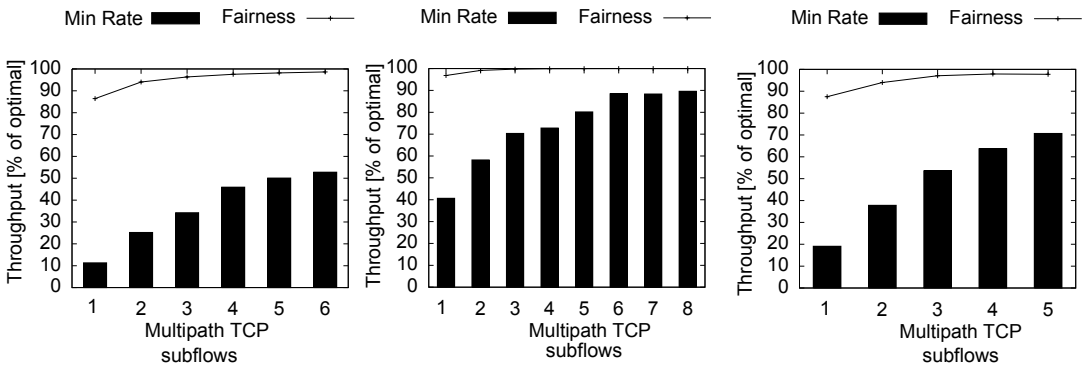


Figure 3.7: Average minimum throughput for FatTree, VL2 and BCube data centers with a permutation traffic matrix using different number of subflows.

between flows, i.e. even connection rate allocations. It is defined as

$$\mathcal{J}(x_1, ..., x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{3.2}$$

where x_i are the per-host throughputs for n hosts. For a fixed total throughput, it is maximal when all x_i are equal and it is smaller otherwise. By increasing the minimum throughput of any connection in the network, a data center operator is able to offer better quality of service guarantees and improve the performance of highly distributed data center applications.

To get an impression of the per-host throughput distributions and how they change with increasing number of subflows, figures 3.5 and 3.6 show flow throughputs for a sample run of singlepath TCP and MPTCP with different numbers of subflows in large FatTree, VL2 and BCube networks. For FatTree with singlepath TCP, the distribution of per connection throughputs is widely spread with some flows getting less than 10% of the line rate. MPTCP with 8 subflows gives most connections more than 90% of the line rate while also helping the worst performing flows.

Figure 3.7 shows the average minimum throughput per topology over five simulation runs and Jain's fairness index. The minimum throughput consistently improves with increasing number of subflows in all topologies. Jain's fairness index also indicates better fairness when the number of subflows is increased. This illustrates that multipath routing benefits not only in throughput, but especially in fairness in comparison to singlepath routing. This is especially interesting for VL2 where multiple subflows make a difference in minimum throughput, even though the increase in total throughput is not as high (cf. section 3.4.1).

3.5 Summary

The application of MPTCP to data center networks revealed three main points about resource-limited users in networks with dependent paths. Theory correctly predicted that users only need a small number of paths for load balancing. Secondly, the underlying topology of the network proved to be a major factor that influenced the utilisation of the system. Some topologies that are closer to a big switch with high backbone capacities such as VL2 show only little difference when increasing the number of paths. Other topologies with less backbone capacity where collisions have a higher impact on every flow, such as FatTree, have low singlepath throughput and benefit more from multipath enabled protocols. Finally the effect of increasing subflows on throughput is richer than the "path choice versus no path choice models" studied in theory before. We showed that having two random paths to choose from per user is often not enough. However, in the topologies studied, the number of paths needed did not grow prohibitively large. FatTree topologies were the worst case with eight paths needed under a permutation traffic matrix in order to achieve high system throughput.

Chapter 4

Users have preferences

This chapter considers users who have preferences amongst their paths. Recall that TCP's utility maximisation framework uses α-fair throughput as the main measure (cf. section 2.1.3). However, throughput is not always the most important property for some applications. Consider for example real-time streaming: In real-time streaming, timely delivery of packets is very important. Due to randomness in delay, some packets will time out as they miss the delivery deadline. Hence, a late packet is as good as lost. If most of the packets time out on a low loss, high throughput path, a low delay, but slightly higher loss path might be more attractive. However, TCP and MPTCP do not know about the utility that a user or an application puts on a specific path.

While the concept of user preference is natural for some user models such as human users, we argue that user preference is a more general concept with many potential applications in areas where there is no human concept of preference. We derive a decentralised rate controller that takes preferences for each user into account and study its feasibility as a rate controller for streaming traffic. It is configurable by each user's per-path utility functions which makes it easily extensible for different applications.

The rate controller solves a global optimisation problem and we evaluate its performance in large scale data center topologies using fixed-point equation modelling. In order to show its usefulness in practice, we apply the rate controller to TCP-friendly rate control (TFRC) [26], a rate control protocol used for streaming traffic. In addition, we address performance and scalability questions.

4.1 An example system with path preference

In section 2.2.2, we gave a simple balls-and-bins example that showed that path preference makes sense as a concept and that different users will need to cooperate when choosing rates, in order to optimise global system performance. The example used artificial weights per path

to express the concept of path-preference. We will go one step further now and describe a more realistic example of a system with path-preference.

Consider a delay-sensitive streaming application that sends high definition video to a receiver by splitting the data stream onto three paths. Packets that take longer to send from sender to receiver than a deadline are rendered useless, e.g. because the video frame they contained has already been played out. We choose a deadline of 300 ms for packet delivery. Let x_1, x_2, and x_3 be the throughputs on the three paths. The rate of packets that are delivered before the deadline are called goodput. We can now attach constants $0 \leq c_i \leq 1$ to each interface that represent the percentage of goodput per unit of transmitted data. The utility function of a streaming user is its overall goodput, i.e. $U(x_1, x_2, x_3) = c_1x_1 + c_2x_2 + c_3x_3$.

Let RTTs on each path be be exponentially distributed with given mean for each subflow and losses are independent. Let the packet drop rates and the mean RTTs for the three paths be as follows: $(d_1 = 10\%, RTT_1 = 20ms), (d_2 = 2\%, RTT_2 = 100ms), (d_3 = 1\%, RTT_3 = 300ms)$. There are no queues in this system. We solve the fixed-point equations for MPTCP's LIA controller given drop rates and RTTs. The rates it achieves are: $x_1 = 83.8$ packets/s, $x_2 = 83.8$ packets/s, $x_3 = 55.9$ packets/s. With these rates about 15% of all packets are lost or time out. On the other hand, sending only on path 1 the overall throughput stays the same, while only 10% of all packets are lost. So, this simple system shows that there is a case for a controller that can handle multiple paths and takes per-path utility functions into account.

Similarly, the constants $c_1, ..., c_3$ can refer to bits transmitted per unit of energy with the aim to maximise energy efficiency, money made per packet transmitted with the aim to maximise revenue, or a quality of service measure in order to maximise the user experience.

4.2 Deriving a decentralised, utility-observing load balancer

We suggest to use the following modified version of MPTCP's global system problem, problem (2.16), to capture users' per-path preference:

Maximise:

$$\sum_{s \in S} U_s \left(\sum_{p \in P_s} V_p(x_p) \right) \tag{4.1}$$

over $x_p \geq 0$

subject to:

$$z_l \leq C_l \quad \forall l \in L$$

where $V_p(x_p) = c_p x_p$ is a utility function which indicates the utility a user gets by sending on path p with rate x_p. c_p are non-negative weights. This utility function can be controlled by

an application. In the streaming example from section 4.1, the weights $0 \leq c_p \leq 1$ represent goodput on the path. More general concave $V_p(.)$ could be used to describe utility functions with more structure, but we will leave it for future work to evaluate their possible applications. This chapter makes only use of per-path utility functions $V_p(.)$ that are weighted throughputs.

The second utility function, $U_s(.)$, describes the value of aggregate utility for all subflows of user s. We imitate TCP's behaviour in terms of resource sharing of bottleneck links by applying the α-fair utility function with $\alpha = 2$ for our purposes, equation (2.10). Note that the α-fair utility function also uses weights w_s. These weights are different to the weights c_p since they are set on aggregate rates of a user. The weights c_p are more flexible since they apply to each subflow.

We are now ready to derive a decentralised multipath controller with per-path utility functions. This derivation is based on techniques from [73] and [72]. We start by integrating the hard capacity constraints from system problem (4.1) into the objective function as a soft constraint or a penalty function. The soft-constrained version of (4.4) is:

Maximise:

$$\mathcal{B}(x) = \sum_{s \in S} U_s \left(\sum_{p \in P_s} V_p(x_p) \right) - \sum_{l \in L} B_l \left(z_l \right) \tag{4.2}$$

over all subflow rates $x_p, p \in \mathcal{P}$.

The key to decentralising this problem is to choose a well-behaved penalty function. As in [39], we choose this penalty or link utilisation function to be $B_l(x) = \int_0^x f_l(y)\,\mathrm{d}y$ where $f_l(y)$ is a continuous, increasing function.

Taking partial derivatives $\frac{\partial \mathcal{B}}{\partial x_p}$ and setting the derivatives to 0, we get

$$\forall s, \forall p \in \mathcal{P}_s : V_p'(x_p) U_s' \left(\sum_{r \in \mathcal{P}_s} V_r(x_r) \right) - \sum_{l : l \in p} f_l \left(\sum_{r : l \in r} x_r \right) = 0. \tag{4.3}$$

The definition of the penalty function $B_l(.)$ helps in this step: firstly, x_p is only present in $B_l(.)$ on links l along its path. When taking partial derivatives, penalty functions for all other links vanish. Secondly, $B_l(.)$ has a simple derivative, $f_l(.)$. Hence, in order to find optimal subflow rates for path p at user s, it is only necessary to know the other subflow rates of that user (to compute $\sum_{r \in \mathcal{P}_s} V_r(x_r)$) and also the sum of $f_l(.)$ functions along path p.

A common choice for $f_l(.)$ is the loss or drop rate at link l. Loss rate is a continuous increasing function of load and fulfils the requirements on $f_l(.)$. Consider a path p consisting of k links with link loss rates $d_1, ..., d_k$. We can calculate the path loss rate as $d_p = 1 - \prod_{i=1}^{k} (1 -$

$d_i) \approx \sum_{i=1}^{k} d_i$ for small values of d_i. Thus, we can set $\sum_{l:l\in p} f_l\left(\sum_{r:l\in P(s)} x_r\right) \approx d_p$ and use drop rate as a heuristic to estimate the sum of $f_l(.)$ along path p. End-users can measure per-path loss rate without having to know exactly who shares links with their flows at which link in the network.

This leads to the final conditions on the maximiser of (4.2):

$$\forall s, \forall p \in \mathcal{P}_s : V_p'(x_p)U_s'\left(\sum_{r\in P_s} V_r(x_r)\right) - d_p = 0 \tag{4.4}$$

This implies that we have to find an allocation of rates where conditions (4.4) are met by all users. The next section describes a heuristic method to find this allocation in a decentralised way. The main idea is that each user manages the equations from 4.4 that relate to their own paths. For these equations, they solve for x_p and update their current sending rate smoothly with the calculated rate. This is repeated until the system settles down. When convergence is achieved in the entire network and all subflow rates solve (4.4), we are guaranteed to have found an optimal solution of (4.2) which approximates the global system problem (4.1).

4.3 TCP-friendly rate control: the multipath case (MP-TFRC)

In this section, we describe a rate controller that acts as a heuristic to drive the system towards the solution of equations (4.4) in order to approximately solve the global system problem (4.1). We use it as a multipath extension of TCP-friendly rate control (TFRC), a rate control protocol for single-path, non-TCP flows that was first proposed in [26].

We decided to apply the rate controller to TFRC rather then to TCP because TFRC was designed for non-TCP flows such as streaming flows and the work in this chapter was motivated by real-time live-streaming. Streaming traffic has to share the network in a fair way with legacy TCP flows in order to be considered safe for deployment. Using TCP directly results in unstable throughput due to its additive increase-multiplicative decrease congestion controller. For streaming traffic, a sudden reduction in throughput is difficult to compensate while a spike in throughput can often not be used efficiently, for example if there is no more data to send. TFRC uses a smooth way to update the rates while keeping the controller responsive to changes in path properties and guaranteeing that it is not more aggressive than standard, single-path TCP. This provides more stable throughput for non-TCP protocols. In this section, we describe the legacy elements of TFRC and especially the modifications that we found necessary to extend it to multipath operation.

4.3.1 The TFRC architecture

Originally, TFRC was proposed for singlepath flows [26], later work extended it to multicast operation [83]. In summary, the TFRC algorithm is a rate control algorithm between a sender and a receiver. It works in three parts:

1. The receiver measures loss rate by keeping track of loss intervals, i.e. the time between the last n loss events. If multiple packets are lost in one RTT, they count as one loss event. Information about delivered packets is sent from the receiver in form of control packets.

   TFRC then uses preset weights to calculate a smoothed average loss interval from the last i loss intervals. The authors in [83] recommend to use between 8 and 32 loss intervals for the calculation of the smoothed average and give a heuristic set of weights for each loss interval that trades off stability of the estimate and reaction time to changes in loss rate. The estimated loss rate is also periodically sent back to the sender in a control packet.

2. The sender measures the roundtrip time (RTT), i.e. the time between sending a packet and receiving an acknowledgment from the receiver. RTT is estimated with an exponentially weighted moving average. When a new RTT sample, t_p^{sample}, is received on path p, the receiver updates its RTT estimate t_p by applying the following formula:

   $$t_p \leftarrow \beta \cdot t_p^{\text{sample}} + (1 - \beta) \cdot t_p \tag{4.5}$$

   In [83], the authors recommend the update constant β to be set to 0.95 while [27] uses 0.9 and states that the algorithm does not depend on the exact value of β. We used $\beta = 0.99$ in our simulations since high throughput flows were not smooth enough for lower values of β. For more details about the loss interval method and RTT estimation in TFRC, see [26, 83].

3. RTT and loss rate are provided to a rate equation for TCP in order to calculate the permissible rate that the sender should use. [26] proposes to use an extended version of equation (2.8). The sender then sets this rate as its sending rate.

The main extension to make TFRC work as a multipath protocol is the use of multiple subflows and a modified way to calculate the flow rates. We use TFRC's measurement techniques for loss rate and RTT (points 1 and 2) without modification. These measurements are run on every subflow of a user. We next describe how to change the rate controller (point 3).

We reported about three well-known problems that had to be address in order to make MPTCP fit for deployment (cf. section 2.1.3): TCP-friendliness on shared bottleneck links

(fairness against legacy traffic), providing a multipath deployment incentive (the utility of the simultaneous use of multiple paths should be at least the utility of any singlepath TCP flow), and avoiding flappiness of the controller (balance of the controller). These problems are general and they extend beyond MPTCP to multipath rate controllers for Internet traffic. We will address them next.

4.3.2 Making the utility-observing, multipath load balancer ready for deployment

In summary, the load balancer works by calculating rates that are close to the solutions of (4.4) for each user subject to being TCP-friendly and to providing a multipath deployment incentive.

The first step to modify the rate controller is to formulate additional constraints that guarantee TCP-friendliness and the multipath deployment incentive. We will describe the constraints first and end this section with the full description of the rate controller for MP-TFRC.

TCP-friendliness in the multipath case means that any combination of subflows of the same user that share a common bottleneck link should not take a greater share of the link than a legacy singlepath TCP flow would. So, if two or more subflows of a multipath user share a link with a singlepath TCP flow, they should share the link equally assuming that no other bottlenecks along their paths limit the rates.

Formally, TCP-friendliness on shared bottleneck links is defined as follows [85]: let x_p^{TCP} be the rate that single-path TCP would get on path p given drop rate d_p and RTT_p:

$$x_p^{\text{TCP}} = \frac{\sqrt{2}}{\sqrt{d_p}\text{RTT}_p} \tag{4.6}$$

Consider a multipath flow from user s using a set of paths $\mathcal{P}_s$. TCP-friendliness on shared bottleneck links requires that these flows cannot transmit with higher rate than standard TCP would get on the best path of any subset $\mathcal{A}$ of $\mathcal{P}_s$:

$$\text{for all } \mathcal{A} \subseteq \mathcal{P}_s : \sum_{p \in \mathcal{A}} x_p \leq \max_{p \in \mathcal{A}} x_p^{\text{TCP}} \tag{4.7}$$

Secondly, user s should have an incentive for using the controller: So far, we only require that the subflow rates are TCP-friendly, we do not put any other constraints on the rates of a user. In theory, there might be the case that the user switches from single-path to the multipath controller and the observed utility for that particular user goes down. To avoid this case, we require that the resulting utilities from using the multipath controller match at least the best utility that any singlepath flow would give:

$$\sum_{p \in P_s} V_p(x_p) \geq \max_{p \in P_s} V_p\left(x_p^{\text{TCP}}\right) \tag{4.8}$$

To solve for subflow rates that obey the constraints outlined above while being close to the unconstrained rates from (4.4), we introduce a smaller sized optimisation problem that each user solves to update the rates. This is called the "rate update" problem at user s:

Maximise

$$U_s\left(\sum_{p\in P_s} V_p(x_p)\right) - \sum_{p\in P_s} d_p x_p \tag{4.9}$$

over $(x_p)_{p\in\mathcal{P}_s}, x_p \geq 0$ for all p

subject to:

$$\begin{aligned} \text{for all } \mathcal{A}\subseteq\mathcal{P}_s : \sum_{p\in\mathcal{A}} x_p &\leq \max_{p\in\mathcal{A}} x_p^{\text{TCP}} \\ \sum_{p\in P_s} V_p(x_p) &\geq \max_{p\in P_s} V_p\left(x_p^{\text{TCP}}\right) \end{aligned}$$

Without constraints, the objective function of (4.9) corresponds to the optimality conditions (4.4). With constraints, the optimisation problem guarantees TCP-friendliness (constraint 1) and the multipath incentive property (constraint 2). Note that the feasible set of this optimisation problem cannot be empty. A trivial solution is to check all solutions that only use a single subflow for their maximum utility within TCP-friendly rate bounds. This allocation is TCP-friendly by definition and it also has at least as much utility as the best single subflow solution (although there might be a better allocation that uses more than one subflow).

This leads us to the final rate controller for MP-TFRC. User s updates the subflow rates with the following procedure.

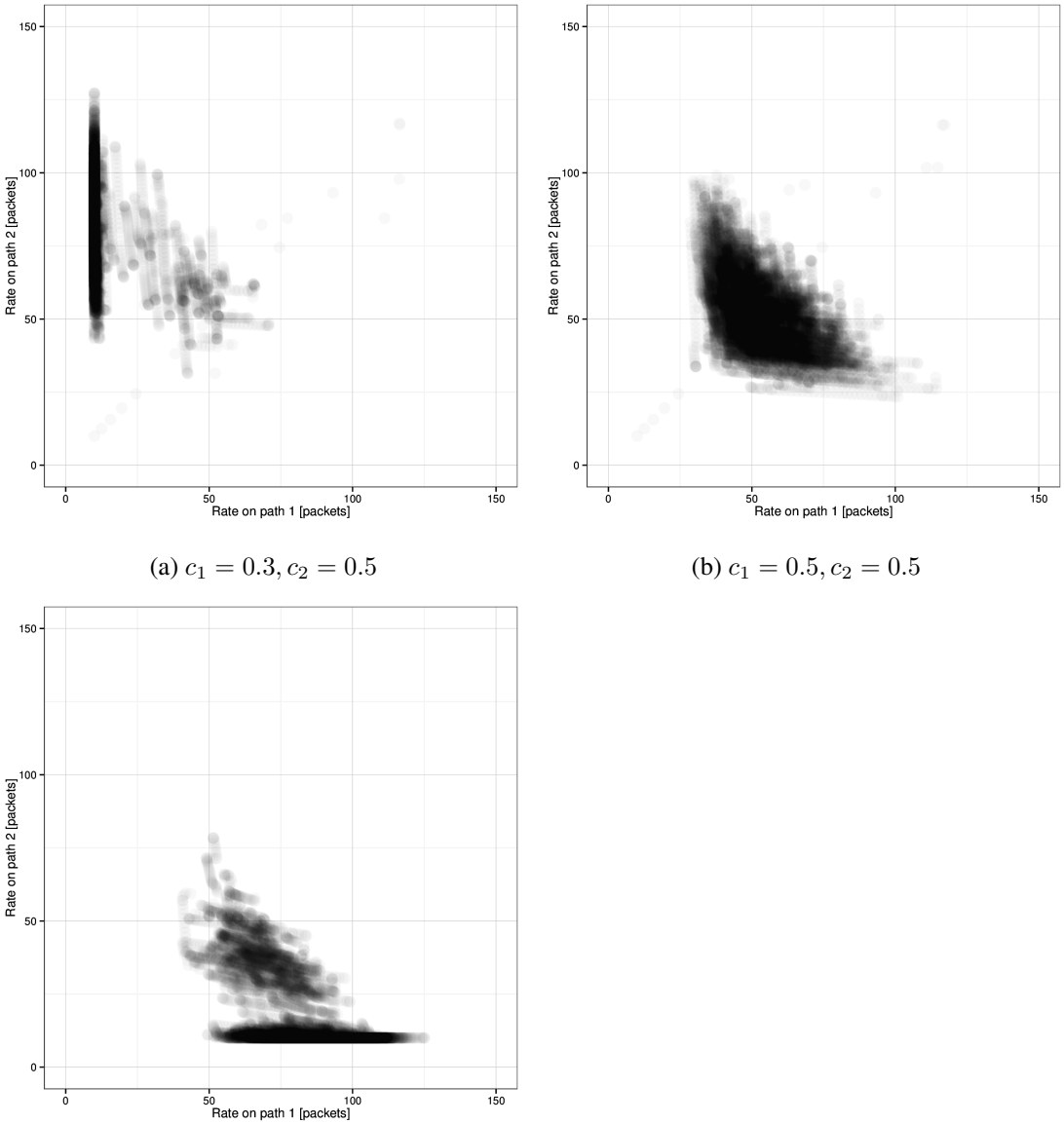
Every t_0 seconds:

1. Solve the rate update problem (4.9) to obtain the optimal subflow rates for subflows $\mathcal{P}_s$, and call them x_p^{opt}.

2. Choose a filter constant for every subflow p: if $x_p^{\text{opt}} > C$[packets/s] then let $\alpha = \alpha_1/x_p$, otherwise $\alpha = \alpha_2/x_p$.

3. Calculate the target rates for each subflow with an EWMA to counter-act flappiness: $x_p \leftarrow (1-\alpha)x_p + \alpha x_p^{\text{opt}}$.

4. Since we update the rates using an EWMA with a high filter constant, it is possible that a sudden increase in congestion cannot be handled quickly enough. For this reason, we cap the rates on every subflow to be at most what singlepath TCP would get on the subflow with the same loss rate, d_p, and RTT, t_p: $x_p^{\text{final}} = \min\left(x_p, \frac{\sqrt{2}}{t_p\sqrt{d_p}}\right)$



(a) $c_1 = 0.3, c_2 = 0.5$

(b) $c_1 = 0.5, c_2 = 0.5$

(c) $c_1 = 0.7, c_2 = 0.5$

Figure 4.1: Load balancing on two paths with equal loss rate $d_{1/2} = 0.1\%$ and varying utility $c_{1/2}$: Three simulation traces of length 30 minutes each.

The sending rate of subflow p is then adjusted to get exactly x_p^{final}. We tested different parameters C, α_1 and α_2 and found that the following choices prevented flappiness while staying responsive to changes in all scenarios that we evaluated: $C = 50$, $\alpha_1 = 0.8$ and $\alpha_2 = \alpha_1/10$.

4.3.3 Some notes about the implementation

Flappiness

It might seem counterintuitive to smooth the updated rates before applying them instead of using the calculated rates directly. The reason for this is that without smoothing MPTFRC's load balancer, the equilibrium rates are not attracted by a single fixed point which makes it possible for the rates to switch between multiple equilibria. For MPTCP, the term "flappy" is used to describe this behaviour (cf. section 2.1.3). The direct way of measuring flappiness is counting the number of route flaps where the load balancer changes the flow allocation from one route to another route although both paths have the same properties. We measure flappiness indirectly by calculating Jain's fairness index (equation 3.2) at each second in a time-series of subflow rate allocations. Jain's fairness index tends towards 1 when all flow rates are equally allocated to all subflows and minimal when all flow is allocated to a single subflow. This directly relates to flappiness: We ensure that Jain's fairness index is high when all paths have equal properties and small when one path is a clear favourite. Flappy load balancers spend more time at the extremes, allocating high flow rates to one of the subflows in the case that all paths have equal properties.

Figure 4.1 shows the behaviour of the system in a simulation with two paths going through the same bottleneck queue. Both paths have equal drop rate of 0.1%. The figure shows that the EWMA rate updates (step 4 of the final rate control algorithm) keep the rates on both paths in balance when they have equal utility (4.1b, Jain's fairness index: 0.96) and exploits the better path when one of them has higher utility (4.1a – Jain's fairness index: 0.63 – and 4.1c – Jain's fairness index: 0.79).

Furthermore, we tested MP-TFRC's ability to adapt to changing conditions. In the same simulation setup with one flow transmitting on two subflows, we set up the following initial base configuration of loss rates and per-path utilities: utility constants $c_p = 0.5$ for both subflows, independent drops on each flow with probability 0.1%, and RTT set to 20 ms. Both flows go through a shared bottleneck queue with service rate 200 packets/s and queue size 10 packets.

We ran the following experiment:

1. Start in equilibrium with drop rates, utility constants in the base configuration (all parameters are equal on both subflows)

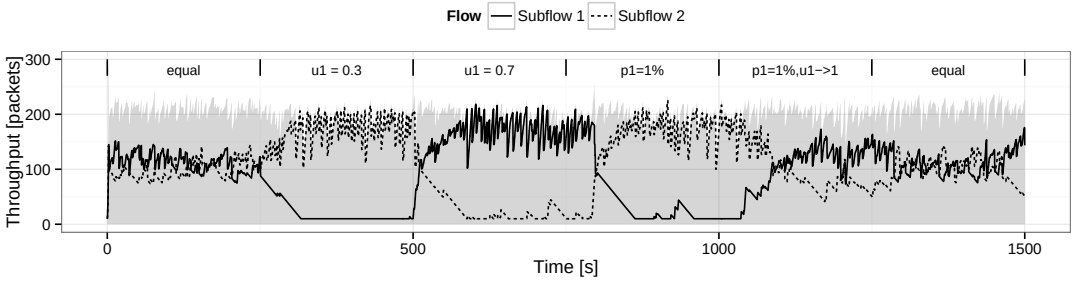


Figure 4.2: MP-TFRC load balancing adapts quickly to changing conditions.

2. Set subflow 1 to have worse utility than subflow 2
3. Set subflow 1 to have better utility than subflow 1
4. Set subflow 1's drop rate to be higher than subflow 1
5. Keeping the high drop rate on subflow 1, the utility value is linearly increased up to 1
6. Revert back to base configuration

Figure 4.2 shows a time series of the rate allocations to both subflows. The text on top of the plot indicates the changed parameters in deviation from the base configuration. The shaded area shows the total throughput on both subflows while the solid and dotted lines indicate the rates assigned to each subflow by the rate controller. It illustrates that MP-TFRC's rate controller manages to adapt to changing conditions in terms of per-path utility and loss while staying in balance without being overly flappy.

Efficiency

Since every t_0 seconds the subflow rates have to be adjusted by solving a convex program, the question of efficiency comes up. How long does it take to solve a convex program of that size? We found that some standard solvers were too slow for the task, so we implemented Photon, a light-weight convex program solver that directly interfaces with the rate controller. Photon is a Java-based interior point solver that has been optimised to efficiently solve MP-TFRC's rate update problem. Interior point solvers work by solving convex optimisation problems with Newton's method [17, p. 561 - 621]. Hard constraints are integrated into the objective function as soft constraints with a parameter that adjusts the rate of increase when the constraint is violated. Interior point solvers increase the sharpness of the soft constraint in every iteration until a cut-off criterion is met. The cut-off criterion can be parameterised by the permissible deviation of the solution from the optimum.

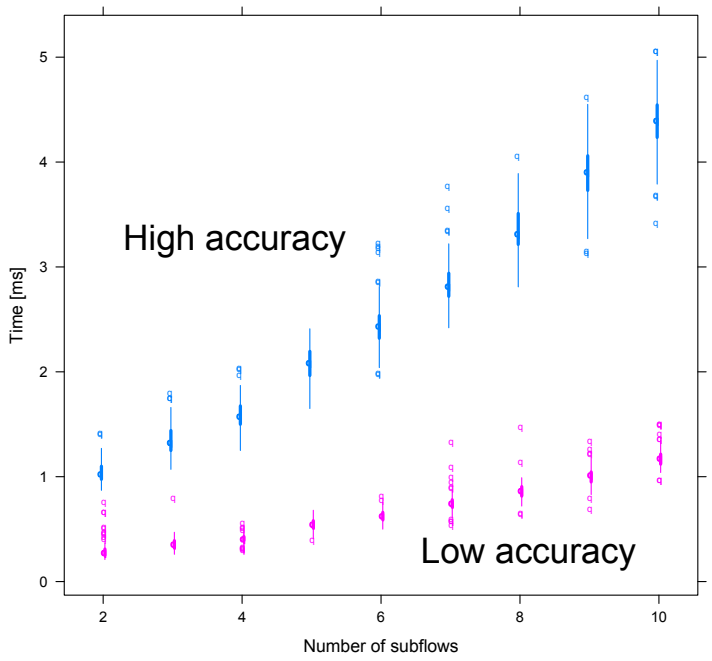


Figure 4.3: Photon efficiently solves the rate computation problem for low (bottom) and high (top) accuracy

We benchmark Photon by solving the rate update problem for up to 10 subflows where the per-subflow utility constants c_p are uniformly distributed between 0 and 1 and the loss rates per subflow are uniformly distributed between 0 and 0.1. For each subflow, we test 100 realisations of loss rates and utility functions and for each realisation, we note the average execution time of 100 consecutive runs. The processor used was a low-voltage Intel CPU U4100 1.30GHz with two cores, but only one core was used for the calculation.

Table 4.1 shows the mean execution time of the rate update problem for low accuracy (error $< 10^{-1}$) and high accuracy (error $< 10^{-5}$). In general, we found that the low accuracy solution was sufficiently accurate in practice, leading to execution times of less than 1ms for up to eight subflows. This is encouraging since we observed in chapter 3 that MPTCP users with access to a limited number of paths in large data center networks do not need more than eight subflows to get close than 90% throughput even for worst-case permutation traffic. Large benefits were already observed by as few as two subflows. The run times observed are in millisecond range even for high accuracy which enables us to compute the rates for all subflows multiple times per second without putting too much stress on the processor. Figure 4.3 shows that the distribution of execution times is close to the mean and it is hence unlikely that rate computation is stalled by Photon taking excessively long to compute the subflow rates. If run

#subflows	low accuracy	high accuracy
2	0.30 ms	1.05 ms
3	0.36 ms	1.35 ms
4	0.41 ms	1.58 ms
5	0.55 ms	2.08 ms
6	0.63 ms	2.46 ms
7	0.76 ms	2.84 ms
8	0.86 ms	3.36 ms
9	1.00 ms	3.90 ms
10	1.19 ms	4.38 ms

Table 4.1: Mean time to solve the rate update problem with Photon for low and high accuracy.

time becomes an issue nonetheless (possibly when running the algorithm on a mobile device with very low computational power), either the accuracy of the computation or the frequency of the update can be lowered. While not implemented in our solution, we recommend accuracy and update frequency to be dynamically controlled by the transport protocol in response to system load.

4.4 Sample application: data centers

In this section we evaluate the performance of MP-TFRC and the utility-compliant load balancer proposed in the previous sections. We focus on two main points:

- With subflow specific utility functions, does MP-TFRC do well in shifting traffic to maximise utility or is a simple greedy load balancer enough to get most of the benefits?
- Does MP-TFRC provide fairness among its flows?

4.4.1 Simulation setup

To study the behaviour of the MP-TFRC load balancer in terms of fairness and goodput in a practical environment, we use the FatTree 8 data center topology [8] with 128 hosts for all of our following simulations. This topology was used in chapter 3 to evaluate MPTCP performance, so we measure MP-TFRC on it as a benchmark. In this topology, every host has an access capacity of 1Gb/s. The traffic matrix is a permutation matrix, i.e. each host is connected to exactly one other host and no collisions can occur in the access link. In addition, FatTree topologies have enough interconnection capacity to provide 100% throughput. However, flow

collisions within the network and subflow-specific utility functions can reduce throughput and goodput respectively and make load balancing more challenging. As subflow specific utility constants we use $c_p \sim \mathcal{U}[0,1]$, i.e. the utilities are chosen uniformly at random between 0 and 1. In a data center setting, this choice could for example be interpreted as the price per packet with the global aim to maximise revenue, as some kind of quality of service measure or as a measure of goodput. Since large scale simulations are very time consuming, we modified the fixed-point solver for multipath rates in data center networks described in section 3.3 by implementing the rate update problem (4.9) to derive target subflow rates. We also used the same random path selection algorithm as before.

4.4.2 Reference algorithms

MP-TFRC's load balancing algorithm is governed by two main principles:

1. P1 – Each path has its own utility function.

2. P2 – Flows of different users solve a global system problem.

We compare MP-TFRC to two other algorithms that incorporate either P1 or P2: MPTCP and GREEDY. MPTCP and GREEDY can be described as optimisation problems that are based on (4.9) with modified objective functions:

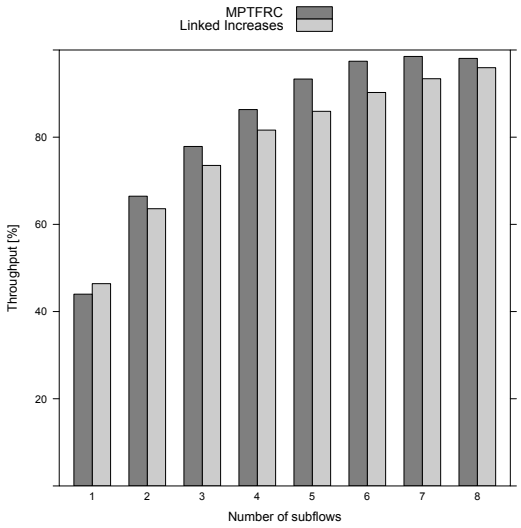
- MPTCP implements only P2. It solves optimisation problem (4.9) with the identity as per-path utility function: $V_p(x_p) = x_p$.

- GREEDY implements only P1. It solves optimisation problem (4.9) with modified objective function $U_s\left(\sum_{p\in\mathcal{P}_s} V_p(x_p)\right)$, i.e. it only tries to maximise its own utility without cooperation. Only the TCP-friendliness constraints keep it from attempting to send with infinite rate.

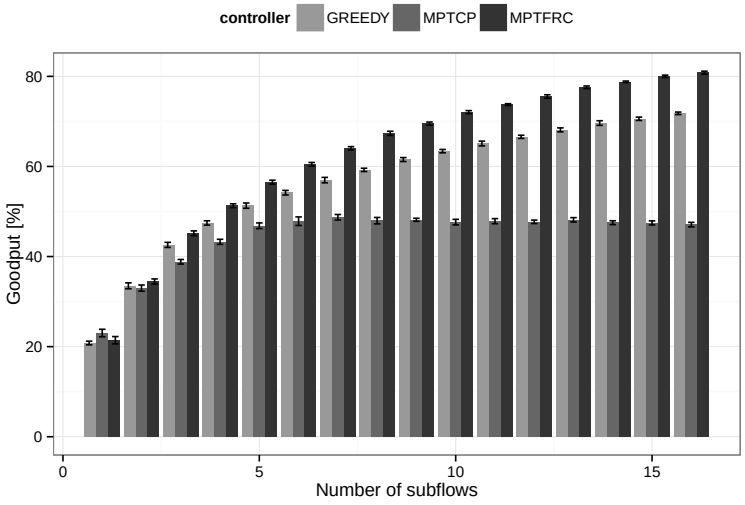
Teasing apart the principles by applying them to different users lets us quantify the impact of each principle on performance and trade off the extra performance against the extra complexity in implementation.

4.4.3 MP-TFRC shifts traffic to maximise utility.

Firstly, we tested MP-TFRC's load balancing behaviour in the absence of subflow specific utility functions. In this case, we compete directly with MPTCP's LIA algorithm (cf. section 2.1.3). Figure 4.4a shows that MP-TFRC gives comparable throughput to LIA and even slightly outperforms LIA for more than two subflows. This can be explained by the fact that LIA was

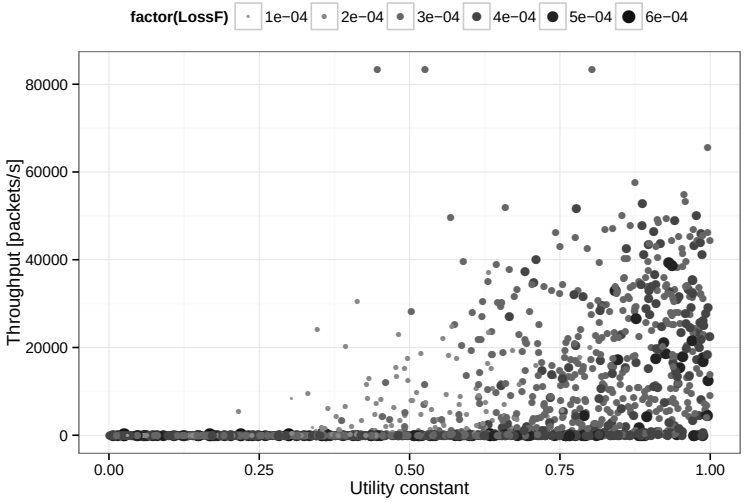


(a) Optimising for throughput: MP-TFRC can compete with MPTCP's LIA algorithm.

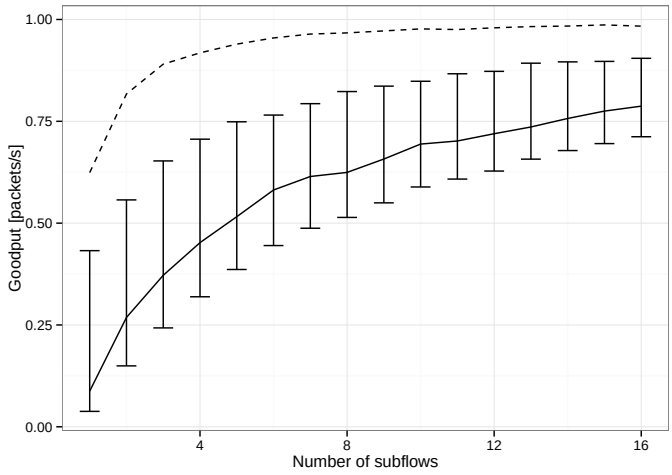


(b) Optimising for utility: MP-TFRC outperforms GREEDY and MPTCP in terms of utility.

Figure 4.4: Performance of MP-TFRC in a FatTree8 topology with 128 hosts and permutation traffic. Error bars indicate 95% confidence intervals.



(a) Plot of all subflows in FatTree8 with 16 subflows: MP-TFRC allocates more traffic to high utility flows, but for fairness reasons there can be high utility flows with no throughput.



(b) Fairness comparison: Jain's fairness index (dotted line), 10% and 90 % quantiles as bars and their quotient as solid line.

Figure 4.5: Flow allocations and fairness of MP-TFRC in an FT8 topology.

designed to work at an operating point between coupled congestion control and uncoupled congestion control. This forces LIA to send more traffic on high loss paths than an optimal load balancer would.

In order to quantify the awareness of the load balancer to utility functions, we compare MP-TFRC to GREEDY, i.e. each user tries to maximise their own utility without cooperation between users (figure 4.4b). Both GREEDY and MP-TFRC manage to increase their goodput beyond MPTCP which is stuck at 50% as the number of subflows increase. The latter is not surprising since the utility constants per sub-flow evaluate as 0.5 on average and MPTCP does not take them into account when selecting sub-flow rates. It is also worth noting that our evaluations indicated that total throughput is not very important when goodput matters – in some cases we have seen that MP-TFRC has the lowest throughput of all three algorithms, but outperforms them in terms of goodput.

4.4.4 MP-TFRC is fair.

In figure 4.5a, we show the throughput allocation to all subflows in a sample run with 16 subflows in the FatTree 8 topology. Note that MP-TFRC generally allocates more throughput to high utility flows. To obtain higher throughput, MP-TFRC allocates in some cases no throughput to subflows with high c_p while the throughput of GREEDY's subflows generally increases as the utility constant increases – almost no subflows with high utility constant have low throughput when using GREEDY.

To test MP-TFRC's fairness properties, we used Jain's fairness index on the utility-weighted per-user goodputs. Figure 4.5b shows the result. In terms of aggregate goodput per connection, MP-TFRC scores highly on Jain's fairness index (dotted line). We observe that similar to the MPTCP evaluation in data centers, two subflows significantly increases this metric compared to singlepath TCP. More than two subflows give a moderate increase. More expressive is the range of allocations (solid line), i.e. the spread between the 10% and the 90% quantile of goodput. In a perfectly fair world, we would expect the low utility flows and the high utility flows to match. MP-TFRC with 5 subflows brings the low and high performing flows within 50% of each other, while 16 subflows increase this value to over 75%. This means most of the flows have very small spread in user-perceived goodput, indicating good fairness properties of the protocol.

4.5 Sample application: a mobile multipath live-streaming system

To show the benefits of the rate controller with per-path utility functions, we describe a simulated live-streaming, multi-path system and use it to test the utility-based rate controller under

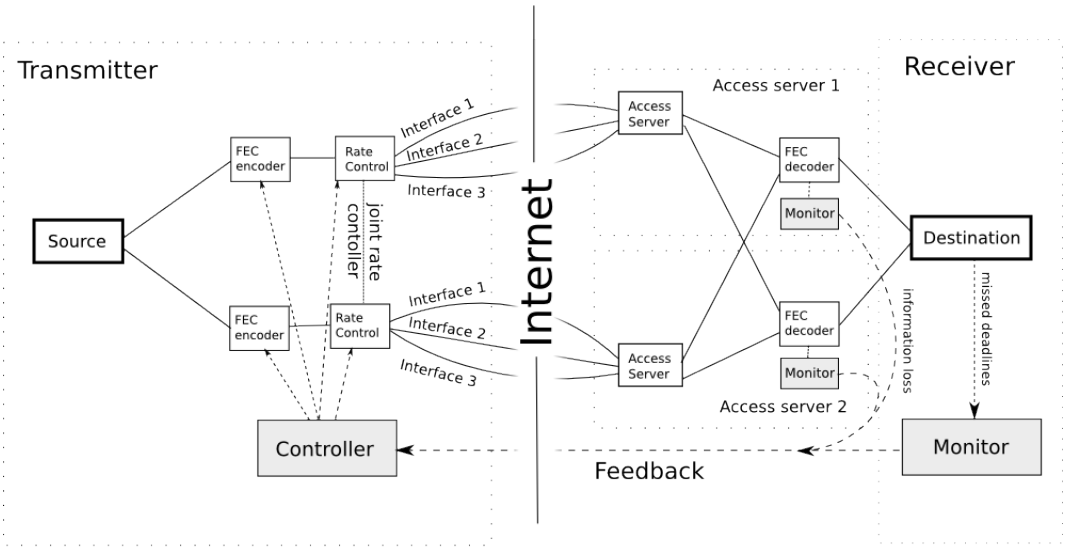


Figure 4.6: Architecture of a mobile streaming system using multiple paths for load balancing

more realistic circumstances. Live-streaming HD traffic usually involves a satellite truck that has to be out in the field close to the source of the transmission. Recently, mobile transmitters in a backpack that use multiple mobile data connections to deliver HD traffic have been introduced and promise to be competitive in terms of streaming quality while enabling transmission in highly mobile scenarios and hard-to-access locations [5].

Figure 4.6 shows the mobile streaming architecture we consider in this section. A streaming source sends data packets with a pre-set delivery deadline to a selection of access servers using multiple outgoing interfaces. Each server can be contacted by multiple interfaces at the same time. By sending redundant information, lost data packets can be recovered from the extra data. This is called forward error correction (FEC). We decided to use a FEC encoder and decoder between the sender and each access-server, so each access server can reconstruct the source packets individually and give feedback to the sender. In this design, an access server can be a dedicated content delivery network (CDN) server or a user in a peer-to-peer live-streaming network which then takes care of the final dissemination of the data to the end users. The access servers give feedback to the sender about the slack time, i.e. the time difference between deadline constraint and actual packet delay. We report the details of the system setup and parameter choice in appendix A.

In order to see the advantages of the load balancer with per-path utilities in a practical scenario, we collected RTT samples from a server over two mobile broadband connections while travelling on the train and replayed them on two subflows in our simulator. We used the greedy version of MP-TFRC since cooperation between flows was not needed and compared its per-

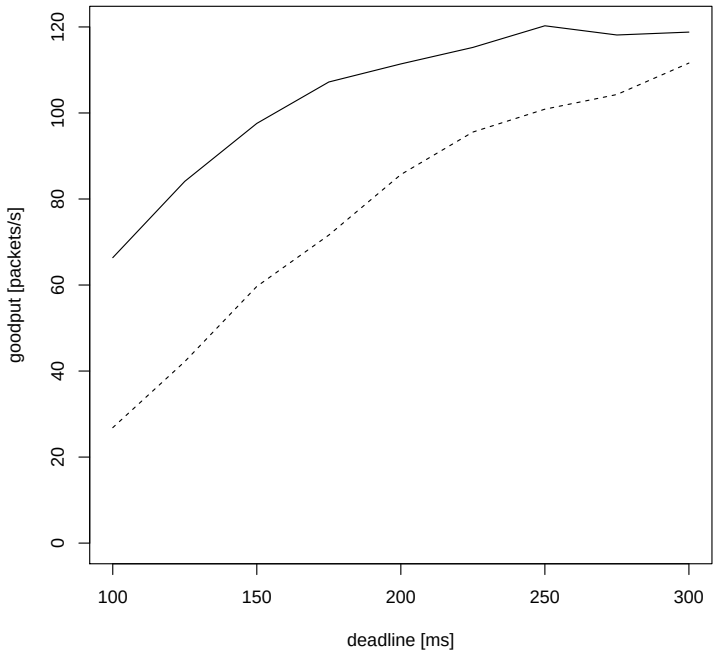


Figure 4.7: Goodput at the receiver after FEC depending on the deadline constraint. Utility-based load balancer (solid line), MPTCP (dotted line)

formance to MPTCP. Figure 4.7 shows the results. The load balancer managed to outperform MPTCP increasing the receiver goodput by up to 2.5 times (in the case of a low deadline constraint of 100ms). When increasing the deadline constraint, the difference between MPTCP and utility based load balancer has to become smaller. As the deadline constraint increases, less and less packets time out, and the utility constants do not give any additional information that helps the MP-TFRC load balancer. For practically reasonable deadline constraints of up to 300ms, we found measurable benefits of MP-TFRC in comparison to MPTCP.

4.6 Conclusion

The main point of this chapter is that users with choice preferences have to be considered as a general concept in modelling. We developed a decentralised rate controller for users with preferences that maximises subflow-specific utility in a cooperative way. We showed how to integrate the load balancer into a multipath version of TCP-friendly rate control and pointed out possible applications. There is more systems work needed to make the load balancer ready for practical deployment, for example protocol specifications and fine-grained parameter studies to confirm its applicability to different kinds of traffic. This is left for future work.

Chapter 5

Users are stateful

This chapter analyses a user model whose response to feedback is stateful. This means that a user maintains an internal state that governs its response to feedback. The same feedback information can cause different responses depending on the internal state. The aim of this chapter is to show how complex stateful behaviour arises in common energy models for mobile phones and why it is important to integrate statefulness into the user model.

The problem we are solving using stateful user models is energy efficient interface switching on mobile phones. Mobile devices are typically equipped with several radio interfaces, such as Wifi and 3G. When initiating a communication session to another Internet host, a multihomed device can consequently choose between several paths to that destination IP address. Protocol extensions such as MobileIP [57] or HIP [54] try to address this problem at the network layer, but have failed to see adoption. In this context, switching interfaces is a simple way to reduce energy consumption of mobile devices by choosing interfaces with low energy consumption for data transmission. This is where statefulness of data interfaces is important: data interfaces are governed by different energy states depending on how much data is sent at the moment and also depending on timeouts in the order of seconds to minutes.

We propose a method for generating multipath schedulers by formalising them as Markov decision processes (MDPs). One set of necessary inputs to this process are energy models for the different radio interface technologies which are described as finite state machines and parameterised with on-device measurements. A second set of inputs are application models which are parameterised by continuously monitoring a user's Internet communication sessions. Generating these schedulers is computationally expensive. Therefore, the required sets of input data are uploaded from a user's device to a cloud service, where the scheduler is being derived. Whenever a personalised scheduler has been generated, it is downloaded to the user's device and takes over the scheduling.

The chapter is structured as follows. We review related work on energy efficient schedul-

ing in section 5.1. Section 5.2 describes our proposed scheduler architecture and presents an optimisation problem that generates an optimal scheduler for interface switching. The scheduler is then evaluated for randomly generated application models and a selection of realistic applications in section 5.4. We point out future work in section 5.5 and summarise the work in section 5.6.

5.1 Related work and technical background

Research about energy efficient scheduling on mobile devices can be classified in two groups. Many results exist about how to optimise energy performance of a single interface without load balancing on multiple interfaces [71, 74, 19]. In this line of work, Markov decision process (MDP)-based algorithms to optimise energy consumption of single interface mobile devices taking time variability into account are widely used. We will describe MDPs in more detail in the next section. To give an example of a model used in the literature, [19] solves a energy optimisation problem for a simplified Wifi interface consisting of only of startup and transmission energy.

The second line of research deals with load-balancing data over multiple interfaces and is closer to our line of work. Although MDP-based modelling is well-suited in this area as well, we are not aware of any solution using an MDP-based load-balancing scheme with a transport layer architecture to minimise energy consumption with complex energy and application models. One of the major papers in this area is [62] where the authors propose to use context information for deciding which interface to use for a single-flow fixed size data transmission on a mobile device. Their proposed algorithm uses non-stateful user models, i.e. it always chooses the best interface at the moment without taking time-varying traffic and complex energy models into account. These simplifications and assumptions are hard to justify for complex energy models and bursty Internet traffic.

We also propose a scheduling architecture using MPTCP. MPTCP for energy aware scheduling has first been proposed in [60]. The author uses simple heuristics to shift traffic, but does not take complex energy models or time varying traffic into account. Using a proxy server to enable MPTCP for a non-MPTCP service was examined in [82].

Our contribution is bringing all three areas together: we propose stateful modelling of users' transmission interfaces and their application models in order to derive a scheduler that decides which interface should be transmitted on. In addition, we propose an architecture using MPTCP to handle the interface switching that goes beyond previous work in its methods by using stateful energy and application models in a multipath context.

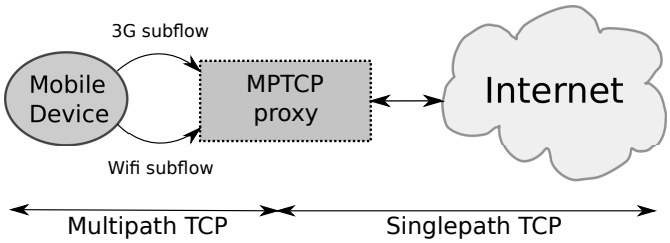


Figure 5.1: An MPTCP-enabled proxy server can be used to deploy the scheduling architecture when end-hosts are not MPTCP-capable.

5.2 Technical architecture

This section describes the technical architecture used to implement interface switching in detail. This is a pre-modelling step that sets up the architectural framework that we are working with. The architecture consists of three components: Firstly, an MPTCP-capable mobile device, so that multipath transfers can be used with MPTCP-capable servers. Secondly, an MPTCP-capable proxy that enables the mobile device to use MPTCP for connections to standard TCP servers on the Internet. And thirdly, a scheduler on the mobile device that is responsible for controlling interface switching.

Our energy-efficient scheduler dynamically decides which path, *i.e.*, which radio interface, to use for the current set of active connections. This decision is made every Δt time units. It is based on the device's current state – in our case, this is the measured throughput in the last Δt time units and energy state of all interfaces. The scheduler simply looks up the best interface in a table of scheduling rules of the form (state) $\rightarrow$ (action). The table of scheduling rules is pre-computed and loaded onto the mobile device which then monitors the state of its interfaces.

Based on this scheduling decision, it initiates the necessary MPTCP protocol actions to shift the affected subflows. Interfaces are identified by IP addresses and paths are identified by IP address pairs. For all active connections, MPTCP will consequently establish new subflows via the new preferred interface and then tear down all subflows over all other interfaces. New subflows are established using the standard SYN, SYN/ACK, ACK three-way handshake. The MPTCP "join" option specifies which connection it will be part of. To remove a subflow, the "remove address" MPTCP option is used, and the subflow is closed after a standard FIN, FIN/ACK exchange. This procedure is transparent to the application, *i.e.*, the connection remains active, although throughput and RTT can change.

In order to use MPTCP, both end systems involved in a communication session need to support it. However, very few MPTCP-enabled servers exist on the Internet at the moment. This can pose a deployment problem, because it could be more difficult and less beneficial to

deploy MPTCP in data centers or other content servers. When MPTCP is not available end-to-end, ISPs or other third parties that have an interest in improving the Internet connectivity of mobile devices can deploy proxies that translate MPTCP communication from and to the mobile device to standard TCP which is supported in the rest of the network. The multipath scheduler on the mobile device thus moves flows between interfaces by adding and deleting subflows to the proxy server. Figure 5.1 illustrates this architecture.

Due to the necessary MPTCP signalling, it takes two RTTs to shift a subflow to a new path, and it takes one RTT to tear down a subflow over a path that is no longer preferred. Because proxies are envisioned to be deployed by mobile ISPs, they are likely to be topologically close to the network edge. This means that the RTTs needed for signalling will be low, resulting in only brief shifting delays.

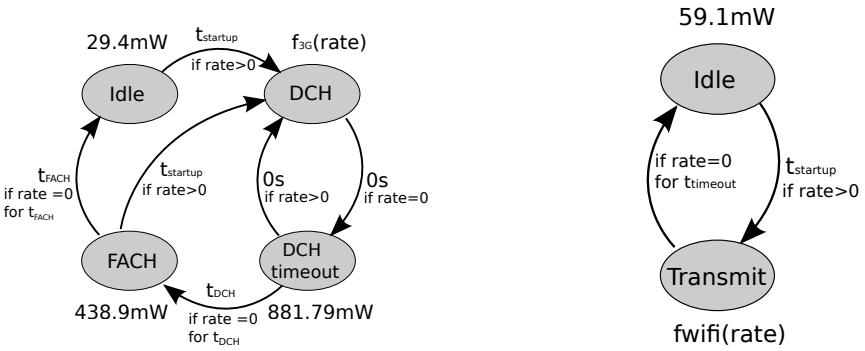
5.3 Energy-optimal scheduling: an optimisation formulation

Next, we show how to design an online scheduler based on MDP modelling. This scheduler takes the stateful nature of interface energy models described as finite state machines (FSMs) and application throughput models for each interface into account when periodically choosing a single interface for transmission. FSMs make it possible to describe stateful user behaviour with simple state diagrams.

In order to be able to compare its efficiency, we also give a description of an omniscient oracle scheduler that has perfect knowledge over the entire future of the process and serves as a lower bound. To make it clear that we refer to a scheduling algorithm, we use capital letters, *e.g.*, 3G, WIFI, MDP, or ORACLE.

5.3.1 Problem specification: proposed user description

The first task before deriving an MDP scheduler is to find FSM representations of the energy consumption for all interfaces. Data interfaces usually operate with algorithms that can be easily cast as finite state machines. Figure 5.2 shows the example FSMs for 3G and Wifi interfaces that were used for the evaluation. In general, the states of the FSMs can have different energy consumption that itself might depend on the throughput achieved in the state. Timeouts between states – if throughput has remained low for long enough – are common in interface FSMs. The FSMs for each interface are then translated individually into *energy models* the MDP optimiser can work with. They are also FSMs, but have special structure: In the energy model, state transitions may only be made every Δt time units (in our practical evaluations, we used $\Delta t = 1s$). Hence, energy consumption in each state has to be averaged over a time period of Δt. If Δt is chosen small enough, the resulting approximation is close to the actual energy consumption.



(a) 3G (b) Wifi

Figure 5.2: 3G and Wifi state machines used for the evaluation.

Timeouts in Δt granularity can be taken into account by adding extra states.

In order to get a realistic interface model, we measured energy consumption for different energy states for Wifi and 3G transmissions on the Nokia N900 mobile phone. The 3G interface can send data with high speed in dedicated channel mode (DCH) and with low speed in forward access channel mode (FACH). In our experiments, the interface always used DCH even for low-rate data transmission in our measurements and we model this by restricting data transmission to DCH. Wifi only has two states: it can transmit or be idle. For the energy model, the functions relating throughput x [kB/s] and energy consumption [mW] in transmission have been measured as:

$$f_{\text{3G}}(x) = 915.096 + 0.352x \tag{5.1}$$

$$f_{\text{wifi}}(x) = 873.438 + 0.532x \tag{5.2}$$

This implies that 3G energy consumption is lower than Wifi for high data rates. However, 3G incurs a DCH timeout, t_{DCH}, leading to a energy consumption of 881.79mW for 5 seconds and a FACH timeout, t_{FACH}, consuming 438.9mW for 2 seconds after transmission has stopped, which prevents agile switching. $t_{startup}$ is short in general. We used $t_{startup} = 0s$. Note that DCH and FACH timeouts are set by the 3G operator and can vary. In practice, different energy models with a common set of timeouts have to be used. Wifi startup and timeout times were less than 100ms which we approximated by using no startup and timeout delay for Wifi.

The second step for the scheduler derivation is to generate an *application model* for a traffic trace from a specific application or user activity. We assume that in each interval, there is an amount of data that is required to be sent. To make the problem tractable for modelling, we use two assumptions: First, we model throughput in Δt granularity up to a maximum rate. Second, we model throughput as a random variable and apply a Markov approximation, *i.e.*, we

assume that the future of the throughput process only depends on the last observed throughput value. We assume that the stochastic process governing the throughput for all interfaces does not depend on the transmitting interface. This makes the optimisation objective equivalent to *minimising energy per bit*. The application model depends on the transmitting interface and even on different states in the FSM description of the interface energy model (since different states might have different capacity limits). In general, the energy and application models can be conditioned on major factors influencing the interface energy consumption and throughput behaviour, such as signal strength or channel quality.

5.3.2 Online scheduling with incomplete information – the MDP scheduler

The online scheduler running on the mobile device has to decide every Δt time units which interface to choose for transmission, in order to minimise the average energy consumption for an application and energy model. MDP theory provides us with tools to efficiently solve for the optimal scheduling rules, assuming the FSM representation of the interface energy model predicts the actual energy consumption well and the Markov approximation for the application model holds.

We will now introduce the MDP model that can capture these stateful user models. Formally, an MDP is a 4-tuple (S, A, P, C). S is a set of states. We model a state s for the MPTCP scheduler with k interfaces as a $k+1$-tuple: the first k state variables represent the energy state of its k interfaces. Their domain is equal to the set of states of the FSM energy model description for the corresponding interface. The $(k+1)$th state variable is the observed throughput during the last Δt time units on the active interface. Its domain is determined by the chosen granularity of transmission speed and the maximum transmission speed set for the application model.

A is a set of actions. An action or, in our context, a scheduling rule specifies which interface to use exclusively for the next Δt time units. For a device with k interfaces, it is described as $a \in \{1, ..., k\}$. The interface identified by a is chosen as the unique transmitting interface for the next Δt time units.

$P(s_t|s_{t-1}, a)$ are transition probabilities from state $s_{t-1} \in S$ at time $t-1$ to state $s_t \in S$ at time t when taking action $a \in A$ at time $t-1$. In our case, the transition probabilities between different states under a given action are derived from the application and energy models. Therefore, we shall discard all state transition for which one of the energy models for an interface has an invalid state transition according to its energy model. For all remaining transitions, use the throughput transition probabilities from the application model as the overall transition probabilities for the MDP.

$C(s_{t-1}, s_t, a)$ with $s_t \in S$, $s_{t-1} \in S$ and $a \in A$ is a cost function specifying the cost incurred when going from state s_{t-1} to state s_t taking action a. In our case, the cost function is the sum of the energy consumptions for all interfaces, which can be read off the energy model for all interfaces given the state of their energy model and the incurred throughput.

The global optimisation objective is: minimise the expected sum of discounted costs over an infinite time horizon, *i.e.*,

$$\text{minimise } E_\pi \left[\sum_{t=0}^{\infty} \gamma^t C(s_{t-1}, s_t, \pi(s_{t-1})) \right]$$

over all scheduling policies $\pi : S \to A$ where $0 < \gamma < 1$ is a so-called discount factor regulating the time horizon taken into account by the scheduler – the time horizon goes to ∞ as $\gamma \to 1$. For the optimisation of average cost, γ should be chosen close to 1. This corresponds to minimising the average energy consumption. The solution to this MDP can be derived by running, *e.g.*, the well-known value iteration algorithm [14].

In the evaluation, we set the time Δt between scheduling decisions to 1 second and restrict the application model of the MDP scheduler to throughput during the last second. The optimisation objective is to minimise energy consumption without side constraints over an infinite time horizon. The parameter γ was set to 0.9, which provided a good trade-off between computation time and quality of the scheduler. Generating the scheduler on a laptop computer with an Intel Core 2 Duo 1.3GHz processor using one CPU for the computation task takes less than 5 minutes using the value iteration algorithm. The time to generate the scheduler does not depend on the amount of captured data that is used to generate the application model. It mainly depends on the discount factor γ and on the size of the overall state space, i.e. the size of the state space of the energy models and the granularity of the application model. More powerful algorithms for solving MDPs than value iteration are available that can further reduce the computation time.

5.4 Evaluation of the model

In this section, we first introduce a benchmark algorithm, an omnicient oracle scheduler that gives a lower bound on energy consumption. We use the oracle scheduler to compare the performance of the MDP-based interface scheduler against its lower bound on application models derived from measurements and on generalised random application models.

5.4.1 Benchmark

In this section, we introduce an omniscient offline scheduler that serves as a lower bound and a benchmark for comparison. The offline scheduler knows the entire future of the process – this is why an omniscient scheduler is traditionally called an *oracle*. Assuming that there is only

one throughput process governing the throughput of all interfaces, we can describe the incurred throughput at all times independently of the transmitting interface as a sequence of numbers. For this sequence, we can work out what the best scheduling is.

We now show how to compute the oracle scheduler for a fixed finite sequence of throughputs. States in the description of ORACLE are tuples of the form (energy state of interface 1, ..., energy state of interface k, throughput, scheduling time slot). An edge between states of neighbouring scheduling time slots exist if and only if at time t state (state of interface 1, ..., state of interface k, throughput, $t + 1$) can be reached by any state at time t with an action from the action set given the throughput seen in time instant t. We assign weights to all edges according to the energy consumption the action corresponding to the action the edge stands for causes given the throughput of the successor state the edge points to. Let s_0 be the initial state of the interfaces and S_f the set of all possible states for the last time instant. We add an additional final state s_f with edges of weight 0 to all states in S_f. The problem of finding the optimal scheduler with perfect information now reduces to finding the shortest path between s_0 and s_f. The scheduling decisions are given by the actions corresponding to the edges along the shortest path.

5.4.2 MDP closely matches ORACLE for realistic application models

So far, we have introduced two scheduling algorithms, MDP and ORACLE. Since we would like to compare them to manual scheduling (because that is what is in use today), we add single interface policies to the comparison that are not allowed to change the interface. In the evaluation, we will specifically use 3G and WIFI. Let E_a be the average energy consumption of scheduling algorithm a over an infinite time horizon for a fixed application model. Furthermore, let the MDP discount factor be $\gamma \to 1$ and assume the Markov approximation of the application model holds. Then, optimisation theory gives us the following guarantee:

$$E_{\text{ORACLE}} \leq E_{\text{MDP}} \leq \min(E_{\text{3G}}, E_{\text{WIFI}}) \tag{5.3}$$

If the first inequality did not hold, the rules derived by the MDP solver could be used to find a shorter path in the state space of ORACLE. Since the path between s_0 and s_f is already a shortest path, this produces a contradiction. The second inequality holds because a single interface policy can be described with action rules of the form (any state) $\to$ (fixed interface). Since MDP optimises energy consumption over all policies that can be described as (state) $\to$ (interface) rules, single interface policies are contained in the feasible set of policies MDP is optimising over.

We first explore the performance of MDP for traffic models derived from measurements.

Application	3G	WIFI	MDP	ORACLE
HQ Stream	708.7mW	604.0mW	545.6mW	533.9mW
LQ Stream	753.0mW	367.7mW	367.7mW	367.7mW
Download	204.5mW	163.9mW	163.9mW	161.4mW
Browsing	327.5mW	139.5mW	139.5mW	139.5mW

Table 5.1: Average interface energy consumption for running the schedulers for four example application models generated from measured traffic traces.

Therefore, we collected traffic traces on a Nokia N900 for four characteristic applications: Low quality (LQ) and high quality (HQ) video streaming, application download and mostly text-based browsing. All traffic traces were collected on Wifi. For high quality video, the lack of processing power for rendering the content on the mobile device was a major bottleneck. To be able to evaluate this high-throughput case without secondary effects, the traffic traces were taken on a laptop computer on the same Wifi network and the energy model of the mobile device was applied to the traces. All other traces were directly collected on the mobile device.

Next, we calculated the application model for each application and generated the MDP schedulers. Figure 5.3 shows a sample run of the MDP scheduler for HQ video streaming. The scheduler combines the energy efficient areas of 3G throughput and Wifi throughput most of the time. Notice the brief spikes in energy consumption that are a result of the online nature of the MDP scheduler. High throughput for a short time is used as an indication that video streaming has begun. ORACLE does not show these spikes, since it knows that it is more efficient to switch to 3G in advance.

We now make the performance gap between 3G, WIFI, MDP, and ORACLE concrete. Table 5.1 shows the average energy consumption for 3G, WIFI, MDP, and ORACLE for each application model. In comparison to 3G and Wifi, MDP either matches or improves over the best performing interface in terms of energy consumption. The improvement for HQ streaming in comparison to the best-performing interface is 9.7%. This is aligned with our expectation that long-running high-throughput flows enable the scheduler to shift traffic to the more energy efficient 3G interface for that throughput area. However, the main point is that MDP always keeps the gap to the omniscient ORACLE very small. MDP guarantees performance benefits, if they are achievable by any scheduling policy at all, and makes it possible to give performance guarantees for automatically-derived scheduling policies based, *e.g.*, on user-uploaded traces. Savings mainly depend on the hardware. That makes more diversity in terms of throughput

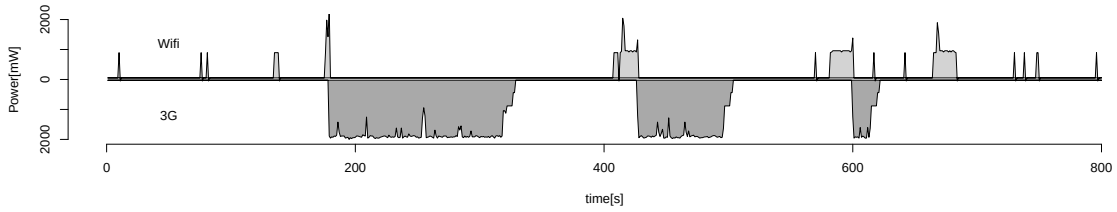


Figure 5.3: Energy consumption plot of WLAN (top) and 3G (bottom) subflows with the MDP scheduler for one MPTCP connection.

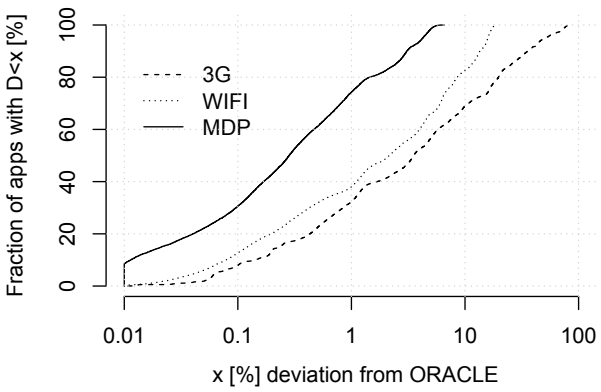


Figure 5.4: ECDF illustrating the deviation, D, in energy consumption from ORACLE for random application models using 3G, WIFI, and MDP.

dependent efficiency and timeout behaviour for different interfaces attractive.

5.4.3 MDP is robust for random application models

In order to study more than selected application models generated from measurements, we now explore the robustness of the 3G, WIFI, and MDP schedulers for a large number of random application models as a benchmark. When we collected the realistic application traces, we noticed that given the throughput in the last second, the throughput in the next second could be closely approximated by an appropriately parameterised, discretised Beta distribution, if we excluded the case of dropping back to $0kB/s$. This led us to a random application model that imitates this behaviour. Our random application model uses $2n + 1$ randomly-chosen parameters, where n is the number of throughput levels in the application model: the first parameter is p_0, which controls the burstiness of the application and it is chosen uniformly at random. p_0 close to 1 will result in very bursty applications, whereas p_0 close to 0 will result in an application that is unlikely to stop transmission. The transition probabilities given the throughput of the last Δt time units, P(throughput in the last Δt time units, throughput in the next Δt time units, action), is formed by combining p_0 with a discretised Beta distribution for the non-zero throughputs. $P(\cdot, 0, \cdot) \in [0, p_o]$ is chosen uniformly at random. $P(x, y, \cdot)$ where

for $y > 0$ and fixed x is a discretised Beta distribution with parameters α and β scaled down by factor $1 - P(x, 0, \cdot)$. α and β are distributed uniformly at random in $[0, 10]$ and chosen independently for every x.

We generated a large number of random application models and compared the performance of 3G, WIFI, and MDP to the ORACLE policy. Every application was evaluated by measuring the long-term average energy consumption for all algorithms. All schedulers transmit the same number of bytes in total. The deviation, D, of a scheduling algorithm for a given application model is defined as the fraction of additional energy use of the scheduler in percent, in comparison to ORACLE. The results are illustrated in Figure 5.4. On average, MDP's energy consumption was within 0.9% of ORACLE's energy consumption, whereas WIFI deviated by 4.5% and 3G by 11.4%. The maximum deviation of MDP for all applications was 6.5%, whereas WIFI was up to 18.5% and 3G up to 84.1% off. In addition, for 74% of all random applications, MDP was not more than 1% off from ORACLE's energy consumption.

5.5 Future work

Several questions remain for future work. We have discussed a simple state space model that only takes throughput into account. Some other alternatives of context information for scheduling have been explored in [62] and have shown to be effective for multipath scheduling. Future work includes research on how to enrich the state space with additional variables, *e.g.*, the channel state of an interface, the average delay during a measurement period, access point or base station information for all interfaces, time and day as well as active application processes. In general, the more state variables the scheduler takes into consideration, the more adaptive it will be regarding its context. However, more state variables also cause a state space explosion, which makes solutions with many state variables difficult to obtain. This leaves the question what a good trade-off between efficiency of the scheduler generation and the scheduler performance is.

In addition, energy consumption alone is not all that matters. Soft constraints can be included in the modelling by adding penalty terms for each soft constraint into the cost function. Hard constraints are satisfied by preselecting the interfaces for transmission that are able to meet them. Minimising energy per bit instead of total energy consumption can be covered by adding a battery level state where empty battery is a terminal state. Each state transition reduces the battery level according to the throughput. The optimisation objective becomes maximise throughput.

In our model, we assume a stationary application model for the entire simulation run. In

general, many factors influence the application model in a dynamic way. Similarly, energy models vary depending on, *e.g.*, signal strength and modulation formats, which results in different MDP schedulers for each combination of application model and energy models, depending on all factors that influence them majorly. This means that we could face a large number of different schedulers that have to be deployed to cover all different circumstances. To address this problem, good application models could be generated by clustering application throughputs for long-term measurements. Schedulers for each cluster are derived such that the majority of all cases is covered. In operation, the mobile device selects the scheduler closest to its current state determined by context information. If there is no scheduler available close to its state, it will use a default scheduler that has been optimised for the entire set of measurements. The schedulers for our evaluation are 375 bytes in size without compression (3000 states with a binary decision in each state), so even a large number of them will fit comfortably onto a modern mobile device. We suspect that the actual number of schedulers needed to get high energy efficiency in practice is not very large. A large-scale measurement study is needed to measure average user application throughput behaviour on mobile devices. This study would allow us to determine how many schedulers a mobile device needs for good performance.

Another open question is how to generate realistic application throughput models. In the evaluation, we used traces from Wifi interfaces and assumed that the throughput process does not depend on the chosen interface, which allowed us to measure throughput on the higher bandwidth interface. In general, throughput distributions depend on many external factors, such as interface bandwidth caps or signal strength. We are currently investigating how to map flow arrival and departure data with flow size information to interface throughputs given context information such as signal strength, modulation format, SSID of the base station or ID of the 3G towers the device is connected to. This procedure allows us to measure application flows independently of the throughput and then map them into expected 3G and Wifi throughputs, which makes the application model more expressive while simplifying its generation at the same time.

5.6 Summary

We have studied energy efficient interface scheduling using multipath TCP as an example problem where stateful user behaviour matters. Stateful energy and application models complicate the modelling process and the derivation of a scheduler. Simple switching between interfaces depending on the measured throughput is not optimal since some interfaces use long timeouts or complex finite state machines to control throughput and energy consumption. We described the

scheduling problem as a Markov decision process which enabled us to algorithmically derive a scheduler given application and energy models. Our evaluation shows that our scheduler is robust for random application models and approaches a lower bound for realistic traffic. Because future-generation mobile devices will feature more and more diverse interfaces, fine-grained flow switching using the MDP scheduler can effectively exploit the best properties of all interfaces, to make scheduling more resilient and more energy-efficient.

Chapter 6

User models for human behaviour are complex

In a communication network, users are protocols interacting with each other through a network of bandwidth limited links. In a societal network, users are humans interacting in daily life. The aim of this chapter is to study how feedback in the form of positive incentives works on users and which other factors influence them. We study data from the first six months or phase 1 of the Insinc project (Incentives for Singapore's commuters), a societal networks project with data from about 20,000 active users that was performed as a collaboration between Stanford University, the National University of Singapore and Singapore's Land Transport Authority (LTA) with the aim to incentivise peak-time commuters to shift their trips to off-peak time. The Insinc experiment was designed and conducted by the societal networks group at Stanford University led by Prof Balaji Prabhakar. We show that user models in societal networks exhibit similarities to user models studied in the previous chapters of this thesis, namely user preference and statefulness of users. However, there are more factors connected to human behaviour which go beyond user models for computer protocols. This chapter does not intend to give a user model with a claim to completeness. For that aim, the Insinc data is not suitable since the programme was not run as a randomised trial which makes reasoning about causality difficult. The data set nonetheless provides a great level of detail which we use to investigate the main influences on users' travel behaviour.

This chapter is structured as follows. First, we give a summary of the Insinc project in Section 6.1 and describe related work in Section 6.2. Section 6.3 introduces the statistical models used. The evaluation in section 6.4 describes long-term and short-term effects of Insinc's reward mechanism. Finally, we summarise the lessons learnt from the Insinc study in section 6.5.

6.1 Insinc is a reward programme that incentivises Singapore's commuters to travel off-peak.

Before Insinc launched in January 2012, Stanford University received a data set with all MRT (Mass Rapid Transit), LRT (Light Rail Transit), and bus trips taken in Singapore from 11-17 April 2011. This data set was analysed by the societal network group at Stanford University to identify the problem zones of the transport system. The results were used to design the Insinc programme with the intent to reduce the system utilisation during the morning peak time. The author of this thesis was not involved at this stage of the project. For that reason, this thesis will not focus on Insinc's design process. Instead, we present the programme as it was run from January 10th to July 10th 2012 and give details about the participants.

6.1.1 The programme

Insinc works similar to an airline miles reward programme: Commuters in Singapore travel on the public subway MRT (Mass Rapid Transit) and LRT (Light Rail Transit) networks either by buying a ticket or by paying for the ride with a travel card, the so-called EZlink card. If they use a travel card, they can sign up on the Insinc website [4] by entering their personal information and registering their card. As soon as their card is approved which usually takes two to three days, their account is activated. In simple terms, Insinc awards credits for commuting by MRT and LRT on weekdays. Active users can access an online account that displays previous trips and current credits, and gives them the option to exchange credits for monetary rewards either with a fixed exchange rate or by playing a game where the prizes paid out can be up to SG$100. Figure 6.1 shows the home screen (redacted to remove all personal data) of a sample account.

Insinc's reward mechanism worked as follows. Users were rewarded by distance. Every weekday trip scored x credits per km. In general, $x = 1$ for every weekday trip including peak trips, but if a trip was started in decongesting hours (weekdays 6.30-7.30 am and 8.30-9.30 am), $x = 3$, thus tripling the credits awarded. By incentivising trips during decongesting hours, Insinc's aim is to incentivise peak commuters to start their commutes outside of peak time (weekdays between 7.30 and 8.30 am). Weekend trips are not scored. For example, a 20km trip taken on Monday morning at 7.25am scores 60 credits while a 20km trip at 8.15am only scores 20 credits. When a user signed up, they also authorised Insinc to receive a limited amount of previous trip records. The previous trip records enabled us to do a before/after analysis of travel behaviour change. In the absence of a randomised control group, before/after data provides a valuable tool to quantify Insinc's effect. Insinc received data about new trips for active users once per day. Note that there was a time delay of up to a week until a trip that was made

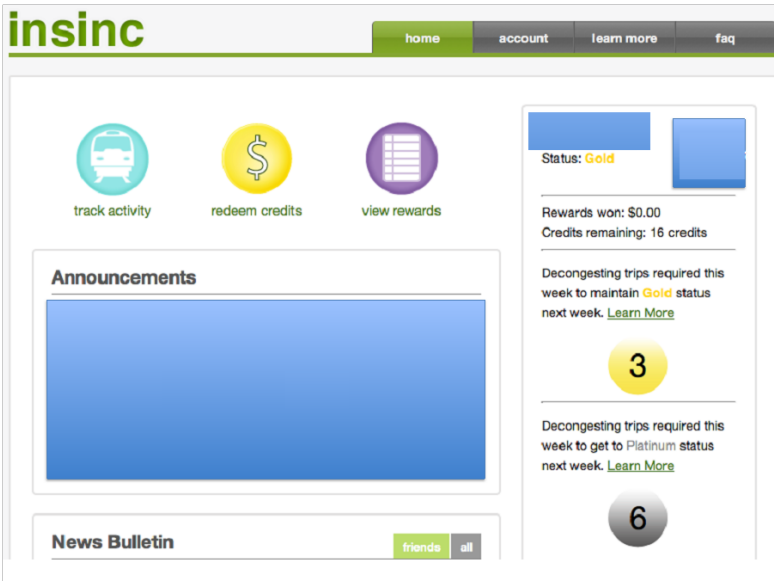


Figure 6.1: The homepage for a user on Insinc

reached Insinc's database. Trip data provided to Insinc did not contain location information, only card number, start time, end time, and distance of a trip. Users could have one of four status levels: member, silver, gold and platinum. The status level was influenced by the average number of decongesting trips per week that a user took (0.5 decongesting trips $\sim$ Member, 1.5 decongesting trips $\sim$ Silver, 2.5 decongesting trips $\sim$ Gold, 3.5 decongesting trips $\sim$ Platinum).

The total prize money paid in phase 1 was SG$137,639 or SG$0.30 per commuter per week. Users had two options to convert Insinc credits into money: deterministic or random rewards. If users chose the deterministic option, their payout was fixed to an exchange rate of 1000 credits to SG$1. The random option let users play their credits on a chutes and ladders-style game. Rolling a die subtracted a fixed number of credits from the user's account and moved the game piece forwards according to the number rolled. Some of the tiles gave out cash rewards straight away, some gave out cash rewards with a probability unknown to the user (double-valued tiles with two possible outcomes) and some gave them credits. The others tiles were either blank, chutes or ladders. There was one board for each status level, giving higher reward tiles to high status users; the option of winning SG$100 was only present on the platinum board, for example. Cash rewards ranged from SG$1 to SG$100. If users did not want to play their credits manually, they had the option to let the system play their credits every week. The so-called 'autoplay' took place every Sunday and informed users about the outcome by email. All rewards won were transferred to the registered travel card at the end of the month. Users had to go to an AXS top-up machine and claim their winnings. AXS machines are well-known and present all over Singapore. In summary, shifting from peak to off-peak earned commuters

more credits per kilometre, increased their status level and, hence, led to a higher payout rate when playing the game. Rewards won were transferred back to the travel card and could be used for future trips.

On the website, users were able to send friend requests to other users. If the other user accepted the request, both users could see each other's status, rewards won, and outstanding credits in a friend list. The friend list had been a successful tool in Steptacular [30], a previous societal networks project, where it was used to explore social interactions.

On April 13th 2012, three months after launch, Insinc introduced the 'magic box' feature, a method to send targeted incentives to users. New magic boxes were allocated every Friday to all active users and an email informed users that they got a new magic box and they had to log onto their account in order to open it. Magic boxes were a mixture of condition-free rewards, rewards with condition, and informational texts. The main types of magic boxes given out in phase 1 of Insinc were:

- Credits: Condition-free credits accompanied by a short informational text.
- Conditional credits: Credits that were awarded if the user fulfilled a target number of decongesting trips in the next week. This offer had two versions: 100 credits for two decongesting trips or 150 credits for three decongesting trips.
- Off-peak multiplier: In the following week, the multiplier for decongesting trips was increased from 3 to 6.
- Friend bonus: For every user who was invited to join Insinc by the recipient of the magic box while the magic box was valid and who was finally activated, the recipient of the magic box received 250 credits.
- Social bonus: Bonus credits for a user, awarded for every friend of the user who opens their magic box.

In addition, there were one-off magic boxes that were given as a trial, such as evening decongesting hours with 3x multipliers for credits during decongesting hours or a facts magic box, an informational text that awarded credits when users had finished reading it. We will restrict this analysis to the five main magic box types that were repeated multiple times over the course of phase 1.

6.1.2 Participants

Figure 6.2 shows the number of registered and activated users on Insinc during phase 1. Insinc launched on January 10th, 2012 and about one quarter of the total number of commuters in

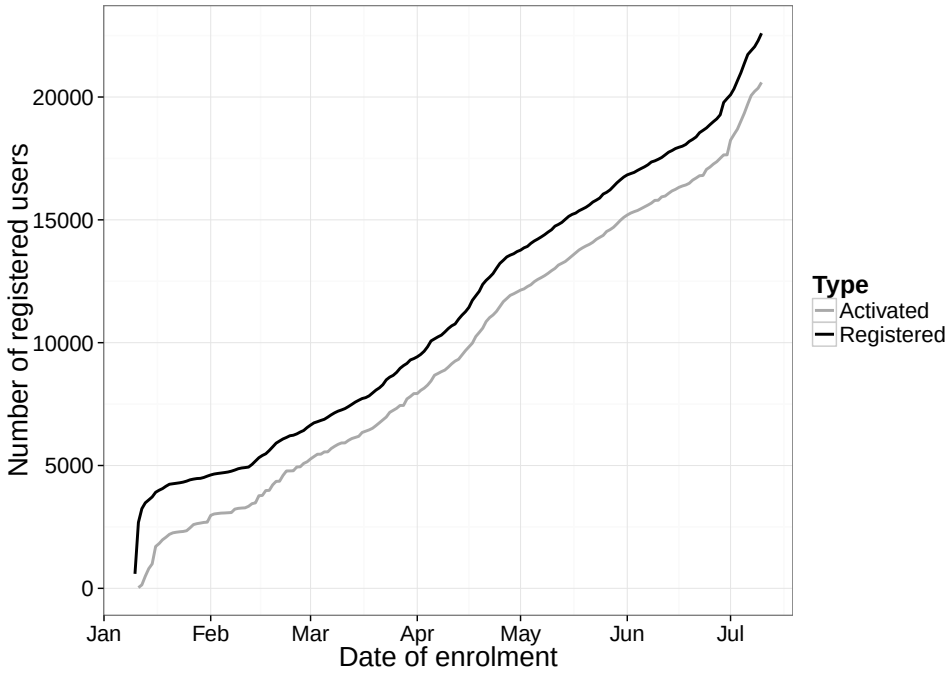


Figure 6.2: Insinc's enrolment was increased by the introduction of friend recommendations on February 14th 2012 and personalised offers (magic box) on April 13th 2012.

phase 1 joined in January. In the following week, especially over Chinese New Year (after January 23rd 2012), the signup rates slowed down. For that reason, a new 'recommend-a-friend' feature was introduced on February 14th 2012. The feature allowed users to recommend friends for Insinc by sending them an email through the Insinc website and if their friend was successfully activated, the recommender was awarded 250 extra credits. This new feature changed the slope in enrolments over time. The third feature that influenced enrolment was the introduction of Magic Box on April 13th 2012. One of the magic boxes offers up to 500 credits if a user recommends a friend and the friend joins successfully. In total, 98,834 emails were sent out from Insinc users to invite other users to participate. The total number of registered users at the end of phase 1 was 22,867 with 20,319 admitted. The difference between registered and admitted users was due to two main reasons: firstly, if a travel card number, with which a user signed up, was found invalid, the account could not be activated. Secondly, in the first month account activation required users to confirm their email address with no option to change the email address if it was entered incorrectly. After two months, the signup policy was changed to a one-step method without the requirement to confirm email addresses. Due to this, about 1,000 applicants in the first two months were eligible for activation, but did not confirm their email address and could not be activated.

In terms of morning travel behaviour, Insinc's final commuter population consisted of two

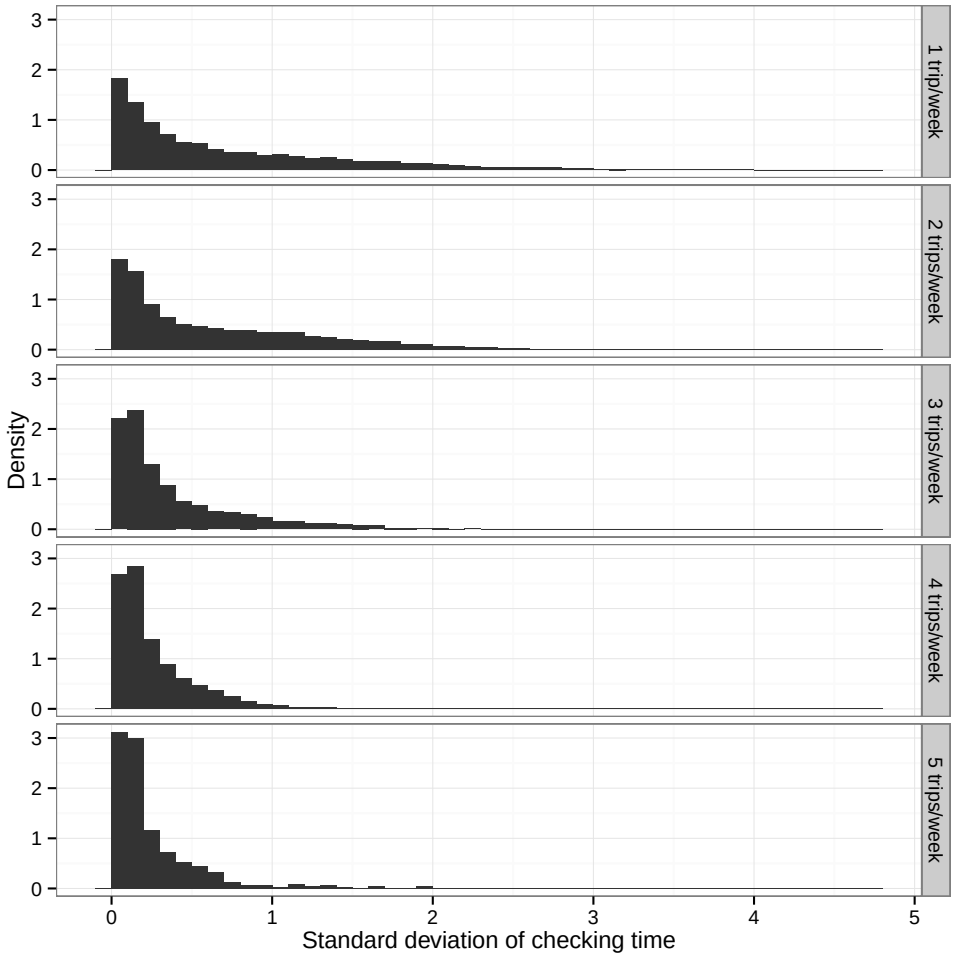


Figure 6.3: Distribution of standard deviation of morning check-in time per commuter grouped by the average number of trips taken per week by the commuter. Regular commuters with higher number of trips per week have less flexibility in their check-in time than irregular commuters. However, even amongst regular commuters taking five trips per week, 48% have a check-in time standard deviation of over 10 minutes, indicating that they are flexible enough to shift for a few minutes.

extremes according to their pre-joining trip records (see Figure 6.3): frequent commuters (more than four morning trips per week on average) and infrequent commuters (less than one morning trip per week on average). Within these groups there are two subgroups each: commuters with flexible travel times and commuters with very consistent schedules. It is worth noting that frequent commuters are less flexible in general. However, even frequent commuters with very regular schedules take trips with a standard deviation in check-in time in the order of 10 to 15 minutes and, thus, show potential to shift their commute schedules in order to lower peak time congestion.

6.2 Related work

Previous research related to this chapter has been conducted in two main areas: congestion pricing as an alternative tool to relax congestion, and previous projects that used positive rewards as a tool for demand change in transport networks.

6.2.1 Congestion pricing

A well-known alternative to incentivising travel time change in public transport is congestion pricing. Fundamental work in the area of congestion pricing has been published by Vickrey [79, 80] following in the footsteps of Pigou [58]. On a conceptual level, congestion pricing for the undesirable choices and paying out rewards for the desired choice are similar tools. In both cases, achieving the desired outcome is a question of finding the right price to charge or incentive to pay. Congestion pricing schemes have been successfully implemented in many major cities, for example London [6] and Singapore [41]. The main difference between congestion pricing and giving positive rewards as Insinc does, lies in public perception: from the perspective of a user, paying money is less attractive than receiving money. Since the aim of this chapter is to explore user models in societal networks and not to evaluate Insinc as a tool for transport problems, we point to [46] for a more comprehensive collection of related work for congestion pricing programmes in transportation.

6.2.2 Previous incentive projects in societal networks

Several projects have been conducted in the past that used positive rewards in order to change the behaviour of people interacting in a network. Stanford's Center for Societal Networks [1] conducted two projects prior to Insinc, INSTANT and Steptacular, while a third project, Capri [2], was launched shortly after Insinc. INSTANT [50] was the first experiment held at the company Infosys Technologies in Bangalore. Over 27 weeks about 14,000 workers were incentivised with monetary rewards to come to work during off-peak time in order to avoid rush hour delays. The main goal of the INSTANT experiment was to evaluate the general

feasibility of randomised reward-based incentive schemes for demand shift. It showed that the average morning commute time over all bus commuters was reduced by 17 minutes during the operation of the scheme and that the system was feasible to build. After INSTANT was finished, the commuting behaviour of the participants returned back to what it was before the trial, indicating the power of monetary incentives.

Steptacular [30], the second experiment, was an incentive programme for Accenture employees to promote walking. Participants used pedometers to monitor their daily steps walked and uploaded the readings to their account on the Steptacular website. There were two main novelties about Steptacular: firstly, it used a new online platform developed for quick feedback to users including a chutes and ladders-style game that could be played on the website to redeem the points earned for walking against money. Secondly, it gave users the choice to add friends in the system and view their progress. The main result was the discovery that a user's behaviour change depended on the number of friends they had in the system. The more friends a user had, the more steps they took and the longer they stayed active in the system.

Another series of experiments that used incentives as a traffic demand shaping tool was performed in the Netherlands. The Spitsmijden (English: peak avoidance) project explored the use of deterministic rewards to reduce peak time congestion on road and train networks [47, 46]. It consisted of a series of trials. The first trial was performed over a time period of 10 weeks on the A12 motorway from Zoetermeer to The Hague with 341 participants. The participants were equipped with GPS enabled devices that monitored their behaviour starting two weeks before the trial to collect data for reference. In addition, road side equipment with license plate recognition monitored the travel behaviour of participants, who were also required to keep log books of their travel activity. The goal of the reward programme was defined as not travelling during the morning peak hour (7.30 - 9.30 am) and the reward was initially set to 5 Euros and later changed to 3 - 7 Euros depending on when exactly during the peak the trip was made. The study reported 50% fewer peak trips by the participants in comparison to trips recorded before the trial. The second Spitsmijden experiment was conducted from April to June 2008 on the A12 motorway between Gouda and The Hague with 705 participating drivers. The project divided the motorway into two zones with different peak times. Participants were rewarded with 4 euros per zone for avoiding a peak time trip in the relevant zone. The second experiment also resulted in 50 % less peak trips for the participants. A third Spitsmijden experiment was performed on the train line from Utrecht to The Hague. 125 participants were able to purchase an off-peak pass that lowered the ticket price during off-peak by 20%. The main differences between Insinc and Spitsmeijden is the type of reward. While the Spitsmijden experiment used

high deterministic rewards for every avoided peak trip, Insinc rewards taking decongesting trips with a significantly lower payout rate (about US$0.30 per commuter per week). In addition, Insinc's targeted offers constitute a flexible way to increase the payout rate to subgroups of commuters when needed.

6.2.3 Contribution

There are three features of Insinc data that allow us to go beyond earlier work:

- Targeted incentives: Insinc's magic box feature made it possible to target subgroups of commuters and test the effect of offers on the subgroup. This makes it possible to get a more detailed look at how incentives work, how they influence the behaviour of the user in short- and long-term, and which other effects are triggered by particular offers.

- Base level comparison: Insinc was provided with pre-joining trip data for each participant which makes it possible to quantify the behaviour change after joining. Previous projects either had no previous data (INSTANT, Steptacular, Capri) or a limited amount (two weeks in the case of Spitsmijden).

- Scale: Insinc's user population at the end of phase 1 was over 20,000 which gives a rich view of user behaviour and responses to incentives. None of the previous projects had access to this amount of data.

6.3 Methodology

The methodology used in this chapter is based on statistics unlike the previous chapters in this thesis that are based on optimisation. While protocols made it easy to change user models, research on user models with human behaviour make it necessary to reverse engineer how the human mind is influenced by incentives. On a higher level, the different methodologies play together. In the case of human users, we do not know which optimisation problem users are solving, which utilities they are using to make their decisions. So the aim of statistics is to discover the main influences on the hidden utilities of users. The statistical methods used for this evaluation are briefly reviewed in this section. We follow Agresti [7, p.113-206] for statistical theory.

6.3.1 Linear regression

In a linear regression model, data is assumed to consist of m independent rows with $n+1$ columns for the measured effect $y_i \in \mathbb{R}$ and n covariates $x_i \in \mathbb{R}^n, i \in \{1, ..., m\}$. The effect y_i is assumed to be generated from a linear combination of covariates. Every row is assumed to

be distorted by i.i.d. noise, ϵ_i with density d. This results in a model of the form

$$y_i = a^T x_i + \epsilon_i. \tag{6.1}$$

The parameters $a \in \mathbb{R}^n$ are estimated such that the model closely resembles the measured data. The similarity of the data to the model is measured by a likelihood function:

$$l(a) = \prod_{i=1}^{m} p(y_i - a^T x_i) \tag{6.2}$$

where p on $\mathbb{R}$ is the density of the error term ϵ_i. Instead of maximising the likelihood, it is often easier to maximise the log-likelihood function instead:

$$\text{maximise} \log l(a) = \sum_{i=1}^{m} \log p(y_i - a^T x_i) \text{ over } a \in \mathbb{R}^n. \tag{6.3}$$

The most frequently used type of noise for linear modelling is Gaussian. With Gaussian noise

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2} e^{-z^2/2\sigma^2}} \tag{6.4}$$

the log likelihood maximisation reduces to a least squares problem that can be solved by means of linear algebra [17, p.352].

The test for significance of the estimated parameters is done with a t-test. The hypothesis that there was a non-zero effect in coefficients $H_1 : \hat{a_j} \neq 0$ is tested against the null-hypothesis that the estimated effect is likely to be from randomness $H_0 : \hat{a_j} = 0$. The t-score is a function of the parameters estimated by maximum likelihood and their standard error:

$$t_{\text{score}} = \frac{\hat{a_j}}{\text{se}_{a_j}}. \tag{6.5}$$

The significance of an estimated parameter is reported with a p-value, the probability of the estimated effect given that the null-hypothesis is true. The p-value can be directly computed from the t-value and is usually accessed by table-lookup or numerical integration. If the p-value is low, the null-hypothesis is rejected and it is unlikely that there was no effect. For the Insinc evaluation, we fixed a p-value of 5% as threshold to reject the null-hypothesis.

6.3.2 Logistic regression

Logistic regression estimates the probabilities in a Bernoulli distributed response variable $Y \sim$ Bernoulli(p) with $P(Y = 1) = p$ and $P(Y = 0) = 1 - p$. p is assumed to depend on a linear combination of covariates which is captured by using a logistic function to link the linear model to the probability estimate:

$$p(x) = \frac{e^{-a^T x}}{1 + e^{-a^T x}} \tag{6.6}$$

where x is a vector of covariates and a are parameter estimates. Equivalently, the logit function is given by

$$\log \frac{p(x)}{1-p(x)} = a^T x. \tag{6.7}$$

This formulation gives a simple interpretation of the estimated coefficients a_i: the $\frac{p}{1-p}$ is called odds. Hence, exponentiating a parameter estimate e^{a_i} can be interpreted as a change in odds: increasing x_i by one unit increases the odds by a factor of e^{a_i}. Alternatively, it is possible to calculate the probabilities p from the estimates and report them directly. We will use both interpretations of the estimates and silently convert between estimated coefficients, odds, and probabilities when appropriate.

The parameter estimate is generated with maximum likelihood estimation as before in the case of linear models. The likelihood function is

$$l(p_i) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \tag{6.8}$$

and the log-likelihood function is

$$\begin{aligned}
\log l(p_i) &= \sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i) \\
&= \sum_{i=1}^{n} y_i \left(\log p_i - \log(1-p_i)\right) + \log(1-p_i) \\
&= \sum_{i=1}^{n} y_i \log \frac{p_i}{1-p_i} + \log\left(1 - \frac{e^{a^T x_i}}{1+e^{a^T x_i}}\right) \\
&= \sum_{i=1}^{n} y_i (a^T x_i) - \log\left(1 + e^{a^T x_i}\right)
\end{aligned} \tag{6.9}$$

Hence, $\log l(p_i)$ is a concave function of a and it is possible to solve for optimal coefficients a with convex optimisation tools.

The test for significance of parameters is performed with Wald's test. For coefficients $\hat{a_j}$ generated with maximum likelihood estimates from logistic regression, we can test the null hypothesis $H_0 : a_j = 0$ against $H_1 : a_j \neq 0$ by calculating the z-score:

$$z = \frac{\hat{a_j}}{\text{se}_{\hat{a_j}}} \tag{6.10}$$

where $\text{se}_{\hat{a_j}}$ is the estimated standard error for a_j. z is approximately normal for large sample sizes and p-values can be derived from the standard normal distribution.

6.3.3 Generalized linear mixed models (GLMMs)

One of the assumptions of the models seen so far is independence of rows in the data. Especially for user centric measurements with repeated entries for the same users, this assumption

does not hold. To remedy this, mixed models assume that there are different random effects, measurement noise per row ϵ_{ij} and a random effects term z_j that models correlations between subsets of the data. Consider for example the following model:

$$y_{ij} = a_0 + \sum_{k=1}^{n} a_k x_{kj} + z_j + \epsilon_{ij} \tag{6.11}$$

where j denotes user j and i is the ith sample of that user. In this example, z_j sets a base level for every user j that addresses the correlation between users. We use GLMMs in this thesis only with Bernoulli response as in logistic regression. In general, GLMMs are more general. They fix a probability distribution from the the natural exponential family that the responses y_{ij} are assumed to stem from, and a linear prediction model. Then they use a link function to connect the estimates from the model to the parameters of the probability function of the response. Depending on the link function and the distribution of y_{ij}, they can handle various different models.

Maximum likelihood estimation for GLMMs is not as simple as in the previous cases due to the necessity to integrate over the random effects. There are different approximate solutions including numerical integration, expectation maximisation, and penalised quasi-likelihood (PQL) methods. All linear mixed models in this chapter have been fitted with R's glmmPQL function using penalised quasi-likelihood methods. glmmPQL is included in the MASS package. We leave out the technicalities of GLMMs here, since it would go beyond the scope of this thesis. For details see Pinheiro and Bates [59].

6.3.4 Notation

Consider a table with three columns: x, y, and z. Assume that z is the effect to be predicted by a linear combination of x, y and an intercept term:

$$z_i = a_0 + a_1 x_i + a_2 y_i \tag{6.12}$$

where a_0, a_1, and a_2 are the coefficients to be estimated. Instead of the explicit form (6.12), we use the following notation for better clarity of presentation:

$$z \sim 1 + x + y. \tag{6.13}$$

This notation is similar to how model formulae are described in R. The first terms asks for the estimate of a base level of y when x and y are 0 or false. The second and third term for the estimate of the effect of covariates x and y on z. The model assumes that there is no interaction between x and y. There are some notational abbreviations to be aware of.

1. A constant intercept term is included as 1.

2. $x : y$ is called an interaction term. Interaction terms measure the change in the effect of x on z when y is present. Important interactions are for example the additional trips taken after joining Insinc or additional friends invited after getting a magic box.

3. Another abbreviation that is commonly used in model formulae in this thesis is $x * y$ which expands to $x + y + x : y$.

The table that is used for most estimations in this chapter consists of trips with user id, trip start time, trip end time, trip start date, trip end date, and trip distance. In addition, we always add a number of auxiliary rows:

- decon: a boolean row that is true if a morning trip is a decongesting trip.
- peak: a boolean row that is true if a morning trip is a peak trip.
- hasjoined: a boolean row that is true if a trip was taken after joining Insinc.

Other tables that we use are the friends table that captures when two users were made friends on Insinc. We also frequently use the magic box table that captures when a user opens a magic box and what the content of the magic box was. None of the tables contain any personally identifiable information. User ids were randomised and do not correspond with user ids used currently on Insinc.

Most models in this thesis use decon and peak as effect measuring the change in trip behaviour using logistic regression or using a GLMM with Bernoulli response and random per-user intercept to account for correlations caused by multiple trips from the same user. When we use a mixed-model with per-user intercept, the model formula includes the words "with per-user intercept". We only report fixed effects in the text. If the distribution of the random effect is of interest, they are included in the full model summaries in the appendix. Note that an alternative way to handle per-user correlations is to add user id as a covariate. However, since there are close to 20,000 users, this usually results in estimation problems that are too big to handle. GLMMs with per-user intercept are more difficult to fit, but easier to handle in that case since only one distribution has to be estimated for the random effect and not 20,000 per-user intercepts.

Note also that we report coefficients, standard error, and p-values for every model. R's glmmPQL function sometimes reports p-values as 0 which we replaced with ϵ to indicate a small number.

6.4 Evaluation

This section explores the user level effects of Insinc. For incentive-based systems such as Insinc, user level behaviour is essential to understand in order to improve the design of incentives, the effectiveness of targeting, and personalised offers. Ampt [11] lists motivation, acceptance and appeal of the programme as the main reasons for understanding user motivation to change in travel behaviour programmes. In her words: "We will be in a better position to develop travel behaviour change tools that appeal if we try to understand individuals"[11, p.5].

Insinc's user-level effects are grouped into long-term change due to the overall program (sections 6.4.1 to 6.4.5) and short-time change mainly due to targeted offers (sections 6.4.6 to 6.4.8). All models fitted in this section are summarised in the text. For more details, R's summary output is reproduced in the appendix.

6.4.1 Overall system effect

In this section, we present a before-after analysis of Insinc's ability to shift commuters from peak to off-peak. This section focuses on the system view. Insinc as a system is concerned about the success of shifting peak trips to off-peak. For example, on system level, it is not important who shifts nor how they shift. Consider a group of ten frequent peak commuters taking five peak trips per week each or 100 peak trips in two weeks. If everyone in the group changes one peak trip to off-peak trip every two weeks, they take 90 peak trips and 10 off-peak trips over two weeks. This is the same aggregate number as if nine of the peak commuters do not change at all and one commuter takes all their trips in off-peak. The system-level question that we ask here is: Does Insinc work and how much less likely are commuters to check in during peak time?

We define a peak shift metric that captures the percentage change in the fraction of peak trips 90 days before the launch of Insinc phase 1 versus the fraction of peak trips during Insinc phase 1. Consider a set of commuters $\mathcal{C}$. Denote the total number of morning trips (between 5am and 12pm) during 90 days before they registered for Insinc as $N_{\mathcal{C}}^{\text{pre}}$ and the number of peak trips in this time period as $P_{\mathcal{C}}^{\text{pre}}$. For the post-joining trips, define similarly $N_{\mathcal{C}}^{\text{post}}$ as all morning trips taken after joining Insinc and $P_{\mathcal{C}}^{\text{post}}$ as the peak trips during that time. Note that users joined during the entire period of six months, not only at the launch date. The fraction of peak trips in morning trips before joining is given by $B = |P_{\mathcal{C}}^{\text{pre}}|/|N_{\mathcal{C}}^{\text{pre}}|$ and after joining by $A = |P_{\mathcal{C}}^{\text{post}}|/|N_{\mathcal{C}}^{\text{post}}|$. The peak shift is then defined as the change in the fraction of peak trips before and after:

$$S = (B - A)/B \tag{6.14}$$

	intercept	hasjoined
coef.	-0.482	-0.123
std. err.	0.003	0.004
p-value	$< 2e-16$	$< 2e-16$

Table 6.1: Insinc reduces the fraction of peak trips after users join (appendix B.1).

Figure 6.4 shows Insinc's system wide shift effect as a before-after plot of check-in densities. Peak and decongesting times are indicated with dotted, vertical lines in the diagrams. The fraction of peak trips for the entire population before Insinc is $B = 38.2\%$ and after $A = 35.3\%$. Applying shift formula (6.14) gives an overall shift of $S = 7.5\%$. These numbers are confirmed by a logistic regression model, estimating the change in probability of taking a peak trip (ispeak) after joining Insinc (hasjoined): ispeak $\sim$ hasjoined. The estimates are shown in table 6.1.

The other three diagrams contain subsets of users conditioned on the number of peak trips they had in their pre-Insinc history (more than 5, 10 or 20 peak trips). Note that the more peak trips users had in their history, the bigger is the peak shift effect on them ($10.1\%, 10.7\%$, and 11.3% for more than 5, 10 and 20 historical peak trips. This is important because the percentage of users in the group with more than 20 peak trips is only 19.9%, yet the number of peak trips they contribute is close to 80%. This means, Insinc can mainly concentrate on a small subset of users while keeping its shift performance.

To compensate for the correlations due to repeated measurements for the same commuters, we used a GLMM model of the same form with user id as random effect. The model estimates the probability of taking a peak trip before Insinc to be 24.3% and after 21.2%, resulting in a shift of 12.8%. The mixed-effects model estimates the odds for taking a peak trip after joining as 0.84, while the odds for the fixed-effects model only gives 0.88. This means after compensating for repeated measurements, the effect estimated by the mixed effects model is larger than the fixed effects model.

It is important to note that the shift metric reflects percentages and not the total number of trips. A commuter who after joining Insinc increased their number of trips on the MRT network in terms of total trips per week, but kept the ratio of peak to off-peak trips after joining, has zero shift. However, since they increased the number of decongesting and off-peak trips per week, this influences the load in the system. In this way, the shift metric is probably not useful for commuters who changed the number of morning trips before and after joining. On the other

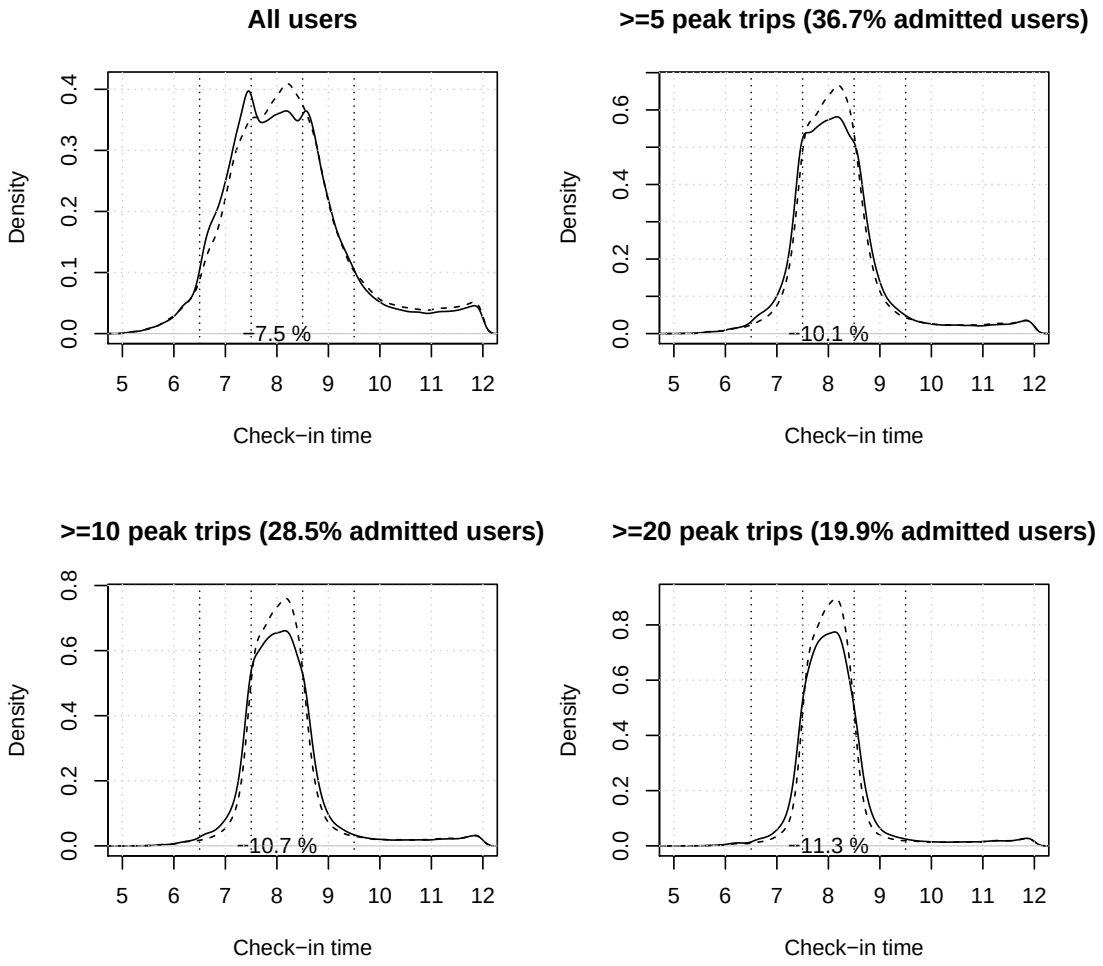


Figure 6.4: Insinc before-after joining check-in densities. Users shift their travel behaviour and the more so the more peak trips they had in their history.

hand, for a user who took the same number of morning trips on average per day before and after joining Insinc, the shift metric tells us that users converted peak trips into off-peak trips because the sum of their pre and post joining trips per day are the same. In order to find out if the shift metric reflects system load, we have to check that only a small percentage of users changed the number of trips significantly after joining.

We define a subgroup of users characterised by two conditions: firstly, they have to have enough pre and post joining trip data for morning trips. We require at least ten morning trips before and ten trips after joining. Secondly, they changed the number of morning trips per day by less than 0.2 when comparing before and after joining morning trip rates. Assuming that a commuter mainly travels during weekdays, the second condition amounts to a difference of about one trip per week. The first condition leaves 9968 users, the second condition only eliminates a further 1541 users. 84.6% of users with enough data (condition 1) changed the number of trips per week before and after joining by one per week or less (condition 2); 10.1% increased the number of trips, while the remaining users decreased them.

We can calculate the shift metric for users who did not change the number of trips per day before and after joining significantly and who have enough data (condition 1). For this group, the shift metric makes sense and it evaluates as 10.0% which is in line with the results from the linear model and the linear mixed effects model for all trips.

6.4.2 The life of an Insinc user: a time-profile analysis of user responses

An incentive programme relies on permanently binding and engaging its customers in order to keep its influence. This property is commonly referred to as "stickiness" of the programme. This section explores the development of user responses to Insinc over time to give a big picture overview of how users are affected by Insinc. Time relative to joining lets us study the lifetime effects of Insinc on a user. For this reason, we set the timestamp of trips for each user relative to the user's join date; week 0 is the week started on the day when the user joined the programme. We fitted the following mixed-effects model:

$$\text{decon} \sim 1 + \text{hasjoined} + \text{twoweeks} + \text{fourweeks} \quad \text{with per-user intercept} \tag{6.15}$$

where the decon is an indicator of whether the trip is decongesting, hasjoined is a boolean factor indicating if the user has joined Insinc, twoweeks and fourweeks are boolean factor indicating if the user was at least in the second or fourth week respectively. The covariates twoweeks and fourweeks measure the performance difference of a user when they were at least two or four weeks in the system.

The estimation results of this model are shown in table 6.2. It suggests that Insinc improve

	intercept	hasjoined	twoweeks	fourweeks
coef.	-0.535	0.167	0.024	0.047
std. err.	0.016	0.009	0.012	0.010
p-value	ϵ	ϵ	0.0449	ϵ

Table 6.2: Time-profile of an Insinc user (appendix B.2)

over time while they stay in the system:

1. Users significantly increase their probability of taking decongesting trips after joining compared to the time before joining (18.1% increase in odds).
2. After two weeks, we can measure an additional 2.5% increase in odds of taking a decongesting trip.
3. After four weeks, the model predicts another increase of 4.8%.

This model suggests that Insinc users need time to change their travel behaviour. This could be due to travel arrangements, commuters new to Insinc try new times to arrive at work. Another explanation might be that users are progressing in the system, they get the first credits and the first magic boxes that work as positive feedback incentivising them to change more.

6.4.3 Early adopters change more.

We found a significant difference between two particular user groups that does depend on calendar time which makes it worth mentioning at this point. Users who joined within the first 30 days of Insinc's launch are significantly different from users who joined later. Figure 6.5 gives a visual comparison of the development of peak and offpeak trip ratios between both user groups before and after. The oscillations in the top graph are due to synchronisation effects: Most early commuters joined in the first two days. Any significant differences in trip scoring that happen mostly to this group, can be seen in the top graph. Since weekends do not count for trip crediting, oscillations with period one week are clearly visible. The join dates of late joiners are better distributed over weekdays, giving less oscillations in the bottom graph. The following mixed-effects model with Bernoulli response describes the differences between early and late adopters:

$$\text{decon} \sim 1 + \text{week} + \text{hasjoined} * \text{islateadopter} + \text{isfirstweek} \quad \text{with per-user intercept} \tag{6.16}$$

In this model, the additional boolean covariate islateadopter is true if someone joined after the first 30 days of Insinc's launch. As the estimates in table 6.3 show, early adopters change

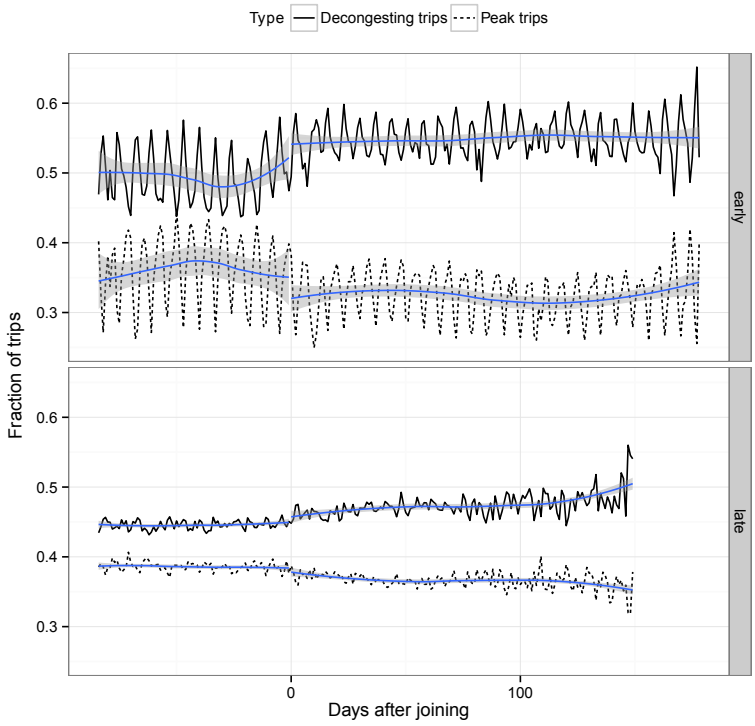


Figure 6.5: Fraction of peak and decongesting trips depending on the number of days since joining conditioned by early (join date within 30 days of Insinc's launch) and late adopters.

	intercept	hasjoined	lateadopter	hasjoined:islateadopter
coef.	-0.270	0.345	-0.318	-0.181
std. err.	0.038	0.009	0.042	0.012
p-value	ϵ	ϵ	ϵ	ϵ

Table 6.3: Early adopters change more than late adopters (appendix B.3).

	intercept	distance [km]	hasjoined	distance:hasjoined
coef.	-0.909	0.265	0.156	0.004
std. err.	0.019	0.001	0.011	0.001
p-value	ϵ	ϵ	ϵ	ϵ

Table 6.4: Long-distance commuters change more than short-distance commuters (appendix B.4).

more than late adopters. This suggests that early adopters can possibly expect high payoffs due to the programme. The users that Insinc tries to target are heavy peaktime users who need more time to be convinced to join.

In general, we learnt that the effect of early adopters has to be carefully taken into account when building a final model. Effects that depend on time after joining for example are more and more dominated by early adopters as time progresses since they have spent more time in the system. This makes it necessary to compensate for their different baselines either with random effects models and per-user intercepts or by taking the date of joining as a covariate into account.

6.4.4 Long-distance commuters change more than short-distance commuters.

We found that commute distance is a major influence on travel behaviour change. Long-distance commuters are more likely to travel during decongesting hours and they are more likely to increase their decongesting trips after joining Insinc. The estimates are highly significant in a general linear mixed-effects model with Bernoulli response:

$$\text{decon} \sim 1 + \text{distance} * \text{hasjoined with per-user intercept} \tag{6.17}$$

Table 6.4 shows the estimates. Per km travelled, the odds of taking a decongesting trip increases by 2.7% per km travelled independent of Insinc (most likely due to timing constraints that require long distance commuters to leave for work early before peak-time) and an additional 0.4% per km after joining Insinc. Since Insinc's credits are awarded by distance, this effect is most likely linked to the increased payout rate for longer distance commutes. In general, this correlation is just an indication that credits indeed matter. Further studies are needed in order to associate credits with shift.

6.4.5 Users are influenced by friends.

Insinc asked all users in a system wide survey at the end of phase 1 whether they thought if their Insinc performance was influenced by their friends. Interestingly, the majority said no.

	intercept	hasjoined	hasfriends	hasjoined:hasfriends
coef.	-0.567	0.179	0.139	0.156
std. err.	0.018	0.006	0.038	0.012
p-value	ϵ	ϵ	3e-04	ϵ

Table 6.5: Users with friends change more than users without friends (appendix B.8).

Previous work in societal networks (Steptacular [30]) suggests that user performance (number of steps taken per day on Steptacular) does indeed depend on the number of friends that users made in the system. On Insinc, before/after joining information can be used to go one step further and distinguish between the influence of friends on performance in general (the more friends, the better a user's behaviour independent of joining) and influence of a user's friends on performance *change*. These two effects are important to differentiate: Since the Insinc population is self-selected, it is likely that users who could expect high payoffs are told by their friends to join. Users who expect high payoffs are those who already travel in decongesting hours, so this just captures a predisposition. The second effect that describes the influence of friends after joining is more interesting. It could suggest some kind of social pressure or peer motivation that makes users with more friends change more.

To tease both effects apart, we fitted a model that tests the influence of friends on the trip performance. It estimates the probability of a trip being decongesting depending on a boolean covariate for joining (hasjoined) and a second boolean covariate for having friends (hasfriends):

$$\text{decon} \sim 1 + \text{hasjoined} * \text{hasfriends} \quad \text{with random per-user intercept} \tag{6.18}$$

That gives us insight into whether users with friends are predisposed to take more decongesting trips and whether users with friends shift more after joining Insinc. The estimated coefficients in the mixed-effects model with Bernoulli response are shown in table 6.5. The estimated p-values for all estimated parameters are highly significant. These estimates suggest that the friends effect is indeed in two parts: A user with friends is more likely to take decongesting trips on Insinc. However, even after joining, a user with friends is more likely to take change their behaviour than users without friends.

Further evidence for a post-joining friend effect or peer-pressure can be derived from one of Insinc's magic box offers. Recall that the 'social' magic box offer rewarded every participant with a given number of credits for every one of their friends that opened this week's magic box. This offer was active from June 1st 2012 to June 8th 2012 and a total of 1014 participants

	intercept	n
coef.	-1.846	0.048
std. err.	0.022	0.002
p-value	ϵ	ϵ

Table 6.6: Users with more friends who got the 'social' magic box offer were more likely to open their own magic box (appendix B.9).

received the offer. It resulted in a significant increase in friend requests sent out by users hoping to increase their chances of winning. Users sent out 253 friend requests during the week of the offer, more than twice as many as in the previous week (88 requests) or following week (106 requests). However, the more interesting question is if peer-pressure works: are users whose friends got the social offer more likely to open their magic box than users whose friends did not get it? In the following fixed-effects logistic regression model, we estimate the probability of opening the magic box (open) depending on the numerical covariate n describing how many of the user's friends had the social magic box:

$$\text{open} \quad \sim \quad \text{n} \tag{6.19}$$

The estimated parameters are shown in table 6.6.

The results indicate that the more friends have the social offer, the more likely the users are to open the magic box. The exact interactions between users have yet to be understood but the social offer makes it obvious that they exist and that they influence the performance of participants. In fact, the actual number are even more surprising than the estimate. For example, the maximum number of friends with the social offer that anyone had was 75. There are 96 users in that class and *all* of them opened their magic box in this particular week.

A different way to confirm a friend effect after joining is to measure a change in behaviour for a user after one of their friends won something. To test that hypothesis, we measure if the change in decongesting trips depends on a friend having won more than x[\$] in the last t days. Unfortunately, independent of x and t, there is no statistically significant effect to confirm this hypothesis. This could mean that friends' winnings are not clearly enough communicated to the user which should be addressed and retested after user interface changes. v Coming back to the survey and the question why users think that their performance is not linked to their friends. We showed that having friends influences users' travel behaviour. This effect might easily go unnoticed for users since it is partially predisposition, i.e. they do not usually

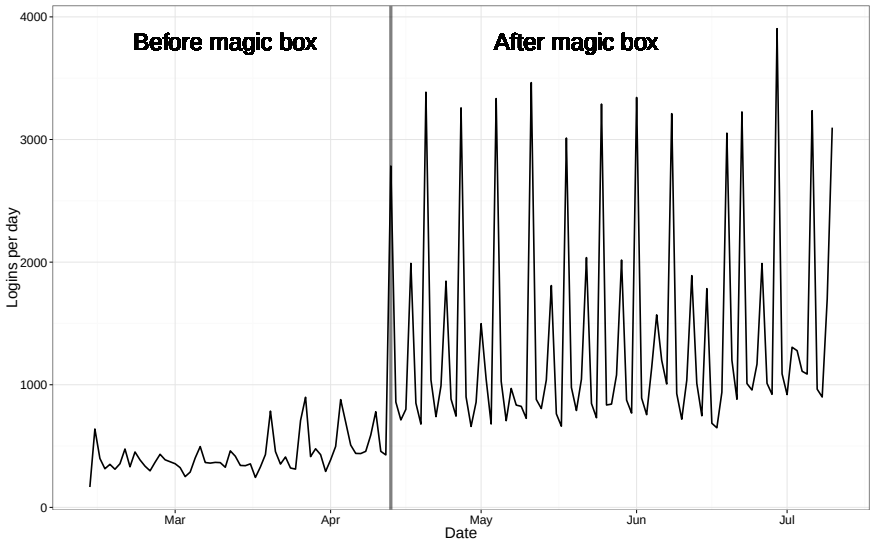


Figure 6.6: Logins to the Insinc website per day before and after launching magic box offers.

experience the difference, and the post-joining change is weak. This could point to a more subconscious influence than a direct one. The social magic box effect on the other hand clearly points out a friend effect, however since it is unrelated to performance change, users might have disregarded it in the survey for that reason. Maybe the answer can be found in the failed attempt to find behaviour change after a friend of the user won something. This would be the most obvious evidence that a friend effect post-joining is taking place. Since the statistical model could not find any significant effects, maybe users think about it in a similar way and for them only significant effects are worth mentioning.

6.4.6 User behaviour can be influenced with targeted offers.

The concept of random weekly rewards seems to be very attractive to users. Figure 6.6 shows the development of daily user logins before and after launching magic box. The spikes coincide with days when magic boxes were given out illustrating the increased engagement of participants after magic box was introduced. Magic boxes were generally given out without randomisation and there was no control group. The only comparison we can make with this data set is the change in user behaviour during the time that the magic box was active as compared to all other times, using users themselves as control. The set of trips is restricted to post-joining trips for every user. In order to make the set of users more homogeneous, we estimated the effectiveness of all magic boxes separately, only looking at users who got that specific magic box at least once. In the resulting trips table, we tagged each trip that was done while a specific magic box was active (hasmagicbox). The reported estimates are generated from a mixed-effects model

	all	off-peak bonus	conditional offer	friend bonus	credits	evening off-peak
coef.	0.236	0.209	0.107	0.129	0.109	0.209
std. err.	0.009	0.017	0.015	0.013	0.019	0.035
p-value	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

Table 6.7: An overview of the effect of targeted offers (appendix B.5).

with Bernoulli response of the form

$$\text{decon} \sim 1 + \text{hasmagicbox} \quad \text{with random per-user intercept} \tag{6.20}$$

This model was run once per magic box type. We show the estimates of the effect in table 6.7. Since the estimates were generated with different models (contained in the appendix), we do not report standard error and p-values. A better way to generate these estimates would be running them in one model. However, in this case, R's glmmPQL did not converge.

Of the directly trip related magic boxes, off-peak bonus increases the odds of taking a decongesting trip during the validity period of the magic box by 23.3% and conditional credits increase the odds by 11.3%. The difference stems most likely from the different amount of additional credits earned for both magic box types. While conditional credits only paid out 100 or 150 credits for taking 2 or 3 decongesting trips respectively in the following week, off-peak bonus doubled the multiplier for decongesting trip credits from 3 to 6. At a mean ride distance of 14.6km and assuming each trip is made as a return trip five days a week, a peak user earns 146 credits as a basis, while the off-peak bonus offers 876, or an additional 730 credits.

Also magic boxes that were not directly trip related changed the decongesting trip likelihood. Friend bonus increased the odds of taking a decongesting trip by 12.9% while even condition-free credits increased the odds by 11.5%. This indicates that magic boxes have more than one function. They mainly give people an offer and promise a reward when the behaviour change is achieved. Secondly, they serve as a general nudge, a reminder about Insinc that already leads to a performance change.

6.4.7 User response to targeted offers is stateful.

We examined the response of users to magic box offers to find out whether users always respond in the same way to an incentive if it is repeated or whether users are stateful and keep track of the offer history. Understanding stateful user responses is essential to building a system that is able to target user responses. Simply giving out the same high-impact offer to the same person might lead to a change in base level of that person, finally leading to a reversed effect where

	intercept	day 1	day 2	day 3	day 4	day 5	>day 5
coef.	-0.767	0.735	0.730	0.790	0.754	1.077	0.575
std. err.	0.100	0.099	0.099	0.099	0.099	0.099	0.294
p-value	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0.0504

Table 6.8: The magic box effect on different days after opening (appendix B.6).

the user does not change their behaviour at all unless the high-impact offer is present. If that particular offer wears off its effectiveness over time, the power of the targeting framework is reduced until its barely operative. On the other hand, if the targeting framework knows the statefulness of user response, it can design the best order of offers for each user in order to maximise user response over time.

Firstly, we study if the effect of a magic box changes over the time it is active. Magic boxes were usually given out on Fridays at around 11am SGT and can be opened at any time until the following week. Hence, the earliest time to observe changes in travel behaviour is Monday since weekends are not counted for Insinc. Most magic boxes are valid for a week after being given out, resulting in between one to five active days. Only survey magic boxes were active for longer than five days.

To evaluate the effect, we estimated the following mixed-effects model with Bernoulli response:

$$\text{decon} \sim 1 + \text{day} \quad \text{with random per-user intercept} \tag{6.21}$$

where day is the nth weekday that a magic box was active. The effect was measured over all magic boxes independent of type.

Table 6.8 shows the results. On days one to five, the effect was different on different days with a peak on day five, the last day most of the magic boxes were open. This suggests that users try harder to achieve the goals of the offer either when the offer is fresh or when they are close to finishing and possibly only need another decongesting trip in order to complete the offer. This is similar to the effect of shifting in general where most users shift just to the least inconvenient time to get decongesting trip incentives instead of shifting uniformly to any time during decongesting hours.

User response to magic boxes also depends on their offer history. We estimated the effect of repetition of the same offer and spacing between offers of the same kind. We fitted the following mixed-effects model with Bernoulli response on a subset of the trip data a magic box

	intercept	repetition	spacing
coef.	0.025	-0.056	0.016
std. err.	0.025	0.022	0.004
p-value	0.317	0.010	ϵ

Table 6.9: Repeated offers lose effect while spacing between offers of the same kind increases their effect (appendix B.7).

was active for a user while they took the trip:

$$\text{decon} \sim 1 + \text{repetition} + \text{spacing} \quad \text{with random per-user intercept} \tag{6.22}$$

Table 6.9 shows the estimates. If an offer is repeated multiple times, it gradually loses its effect. Every repetition decreases the odds of taking a decongesting trip by 5.4%. We found that if there is enough time between the last offer of the same kind, the offer increases its effect, counteracting the decrease by repetition. Every different offer in between a pair of equivalent offers increased the odds of taking a decongesting trip during the repetition of the offer by 1.6%. This implies that there should be at least four offers between two offers of the same kind in order to keep the effect of the offer positive. Too closely spaced offers of the same type can hurt performance!

6.4.8 Unexpected user behaviour

Personalised offers are targeted to special user groups in order to spark certain behaviour patterns. Many offers on Insinc were formulated in a conditional way that they made clear which behaviour was aimed for, for example increasing the likelihood of taking a decongesting trip or inviting friends to join Insinc. In many cases, however, we noticed secondary effects that were not aimed for by the magic box. The offer of 3x credits for evening decongesting trips lead to a 14% increase in the odds of taking a morning decongesting trip in a general linear mixed effects model. This is higher than the effect of conditional credits (13.4%) that were aimed at increasing morning decongesting trips. Furthermore, plain credits resulted in an increase of (12.4%) and even the friends bonus showed an increase of 9.7%. This result suggests that targeted offers activate behaviour patterns that are intrinsically correlated and possibly governed by a common process that is not tied to a particular offer.

6.5 Summary and outlook

In this chapter, we investigated a data set from the Insinc project, a societal networks project with 20,000 active users over six months. This work was started in order to find ways to describe

user behaviour in societal networks

The evaluation presented in the last chapter led to several insights that are useful for future system builders. The main lessons learnt were:

- Users change over time and they need time to get used to the system. The user response on Insinc during the first four weeks on Insinc changes significantly and user take more decongesting trips as they stay in the system longer.

- Incentives have stateful responses. For incentives with a duration, such as the personalised magic box offers, the start time of offers has to be carefully selected on a per user basis to avoid synchronisation effects.

- A social component to the system is very important. Users with friends showed a higher level of change, possibly motivated by seeing their friends winnings or by some form of peer pressure.

- Targeted offers can dynamically change the response of users. Offers have more than one effect, so an offer that is intended to increase enrolment can also influence trip behaviour change.

- Offers should not be repeated too often because users get used to them and they lose effect.

- If offers have to be repeated, the wider the time-spacing between offers of the same kind, the higher the influence. We estimated that on Insinc the same offer should not be repeated if there were less than four different offers in between.

The research done in this thesis on user models for societal networks has already found its way into a working system. Insinc is currently in its second phase and it has grown beyond the 20,000 participants of the first phase. Many of the insights from Insinc's user model have been applied in the design of the second phase of Insinc that ran from July 2012 to December 2013. Motivated by the findings of this thesis, Insinc's website and backend was changed and relaunched in December 2012. The social component has been extended with some targeted offers asking users to cooperate in order to amplify the social effect. The routine that gives out magic boxes ensures that the same magic box has not been given out recently in order to avoid negative effects that we predicted in this thesis for too closely spaced offers of the same kind. In addition, the system now allows better logging of changes in the system for statistical analysis and the ability to perform randomised studies with personalised offers. Randomised

studies will provide the key tool to confirm the findings in this thesis and work in this direction is currently under way.

Chapter 7

Conclusion – in favour of a unified theory for practical user models

We began this thesis by making a case for user models in which behaviour is governed by a user optimisation problem. Modelling users with optimisation problems makes it simple to describe the outcome of user choice, for example maximum utility. It also enables the modeller to derive the behaviour of the global system by only describing the user model. In addition, optimisation-based user models are broadly applicable across fields, e.g. to protocols in communication networks, commuters on road networks, or humans in societal networks. The aim of this work was to apply optimisation-based user models in practical applications and evaluate their current limitations. This work is positioned as a modelling thesis in networking and as such it stands between the world of theory and practical systems building. Theoretical user models for networks are often both powerful in their implications and weak in their potential applications. Systems applications on the other hand excel in execution while falling short in potential benefits that better modelling could have provided.

The main point that this research makes for future work is that we need a better theory of user models; one that is firmly connected to practice. Imagine for example a systems researcher who could use an algebra of user behaviour like a programming language that sits on top of a mathematical framework. Of course it would be possible to simply simulate users in order to predict system behaviour, but in general, theory can go beyond this. It can explain why the system behaves in the way it does. For some models this is already possible today. Consider again Wardrop's greedy users in road traffic (section 2.1.2). Where a simulation could have only told us that greedy users do not always perform as well as they could, the global optimisation problem explains the reasons. As an alternative application, think about a protocol designer for a future Internet protocol who could describe their protocol in the constraints of the user modelling language and see right away if the protocol is stable and if it can be implemented. We

introduced a protocol for multipath streaming in section 4.3 that took first steps into the direction of giving application designers powerful tools to influence the network layer. It extended a well-known model for MPTCP with per-path utility functions, while retaining the optimisation framework that it was built upon. Part of this research goes towards making automatic protocol derivations an automated procedure.

This theory of user models has to stand on a firm mathematical foundation. At the moment, research on user modelling is scattered across many different areas and research publications without a continuous basis. From a mathematical side, I imagine exciting new problems feeding back from practical applications of user models into optimisation, game theory, and many other areas. For example, consider stateful user models that we observed in the world of mobile phones (section 5.3.1) and also in societal networks as a response to targeted offers (section 6.4.7). I imagine applying Markov decision processes not only as a way to solve an optimisation problem, but as a modelling tool. A theory on networks of users who act according to the solution of a Markov decision process, could make it possible to derive system level behaviour for interacting, stateful users. This idea also extends to human behaviour and discrete choice models. While a researcher conducts studies and estimates choice models, a theory of interacting choice models could enable researchers to study system wide effects from measurement data of individual users.

Neither theory nor practice can go there alone. I believe that user modelling is an exciting hybrid that brings fresh ideas into both areas. It will enable practical researchers to use solid mathematical tools and mathematical modellers to integrate their ideas into a unifying framework that feeds directly back into practice.

Appendix A

A multipath streaming architecture

The multipath streaming system we consider uses an error correcting code (forward error correction, FEC) that works on blocks of packets, a so-called *block code*. A FEC encoder for a (n, m)-block code takes n source packets as input and converts them into m coded packets which are transmitted. If any n packets out of m arrive, the code is able to reconstruct the original n data packets. The first question that has to be addressed is how to set the group size m and the redundancy $m - n$ of the block code.

In [20], the authors propose EMS, a real-time streaming system on multiple paths. In EMS, redundancy adaptation is done by monitoring information loss and delay-induced losses due to missed deadlines at the receiver and feeding the information back to the sender using a linear control loop. Group size is adjusted by monitoring slack times, the time difference of packet delay to the deadline, for all packets over a given time window (30s in their simulations) and then linearly changing the group size depending on the amount of slack that the latest packet had in the monitoring window.

In contrast to this approach, we propose group size and FEC overhead adaptation with per-packet feedback and smooth step sizes based on probabilistic estimation. This enables the system to react to rapidly changing conditions as they are expected in wireless systems. Each multipath flow keeps track of mean and variance of information loss, $\bar{I}$ and var(I), at the receiving FEC decoder. Information loss is the rate of packets that could not be reconstructed at the receiver. We make sure that the probability of losing more packets than the amount of redundancy allows is smaller than a preset error probability ϵ, $P(I > r) \leq \epsilon$. Group size adaptation is done on a per-packet basis based on observed slack time. Initially, a slow-start phase increases the group size by 1 packet every time a packet has been successfully received without missed deadlines. This quickly increases the group size to roughly its steady-state value. As soon as one loss is observed, the group size, g, is set back to the last value and is updated from then on with every successfully reconstructed packet that is reported by the

feedback channel. FEC groups that remain incomplete at the receiver due to too much packet loss time out and send feedback after Δ_{TO} seconds. On receiving feedback, the group size is increased linearly if slack time, t_s, is positive and reduced multiplicatively if slack time is negative:

$$g = \begin{cases} g + c_1 t_s & \text{if } t_s \geq 0 \\ g c_2 & \text{otherwise} \end{cases}$$

with $0 \leq c_1 \leq 1$ and $0 \leq c_2 \leq 1$.

In the case of our streaming example with deadlines the objective is to maximise receiver goodput in the presence of variable RTTs and given loss rates. This means the receiver keeps track of mean loss rates as well as mean and variance of RTTs for each subflow. We also estimate the fraction of packets that arrive within the deadline constraint

Appendix B

Model summaries for Insinc data

B.1 The system effect

ispeak $\sim$ hasjoined with Bernoulli response

```
glm(formula = ispeak ~ hasjoined, family = "binomial", data = trips)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9806  -0.9806  -0.9332   1.3879   1.4431

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.482376   0.002855 -168.98   <2e-16 ***
hasjoinedTRUE-0.123449   0.003854  -32.03   <2e-16 ***

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1540789  on 1173237  degrees of freedom
Residual deviance: 1539764  on 1173236  degrees of freedom
AIC: 1539768

Number of Fisher Scoring iterations: 4
```

ispeak $\sim$ hasjoined with random per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.405645 0.9211758

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: ispeak(start) ~ whenF
                  Value    Std.Error   DF       t-value        p-value
(Intercept)     -1.1349150 0.019104541 1149089 -59.40551       0
hasjoinedTRUE   -0.1795486 0.005697834 1149089 -31.51172       0
 Correlation:
          (Intr)
hasjoinedTRUE -0.146

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-10.1076631  -0.4016452  -0.1401404   0.3606058  12.2965342

Number of Observations: 1167172
Number of Groups: 18082
```

B.2 Time-profile of response

decon ∼ hasjoined + twoweeks + fourweeks with per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.060747 0.9509286

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon(start) ~ whenF + I(weekN > 1) + I(weekN > 3)
                      Value  Std.Error      DF   t-value p-value
(Intercept)      -0.5351154 0.01629376 1149087 -32.84175  0.0000
hasjoinedTRUE     0.1667146 0.00920673 1149087  18.10791  0.0000
twoweeksTRUE      0.0244399 0.01218599 1149087   2.00557  0.0449
fourweeksTRUE     0.0477335 0.00975913 1149087   4.89117  0.0000
 Correlation:
                 (Intr) hasjTR twowTR
hasjoinedTRUE    -0.099
twoweeksTRUE      0.002 -0.616
fourweeksTRUE     0.002 -0.007 -0.649

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-9.8493554 -0.5030584 -0.1184002  0.5007662 10.5357586

Number of Observations: 1167172
Number of Groups: 18082
```

B.3 Early adopters change more than late adopters

decon $\sim$ 1 + hasjoined * islateadopter with random per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.054961 0.9508899

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon ~ hasjoined * islateadopter
                               Value  Std.Error      DF   t-value p-value
(Intercept)                 -0.2697847 0.03825219 1149088  -7.05279       0
hasjoinedTRUE                0.3447491 0.00980523 1149088  35.15970       0
islateadopterTRUE           -0.3182348 0.04225773   18080  -7.53081       0
hasjoinedTRUE:islateadopterTRUE -0.1815601 0.01166832 1149088 -15.56008       0
 Correlation:
                                (Intr) hsTRUE isltaT
hasjoinedTRUE                   -0.176
islateadopterTRUE               -0.905  0.159
hasjoinedTRUE:islateadopterTRUE  0.148 -0.840 -0.172

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-10.1812470  -0.5010152  -0.1190403   0.5006831  10.7845937

Number of Observations: 1167172
Number of Groups: 18082
```

B.4 Long-distance commuters change more than short-distance commuters

decon $\sim$ 1 + distance * hasjoined with per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.073287 0.9539737

Fixed effects: isdecon(start) ~ ride_distance * whenF
                              Value   Std.Error      DF   t-value p-value
(Intercept)             -0.9091495 0.018980714 1149087 -47.89859       0
distance                 0.0264745 0.000671218 1149087  39.44241       0
hasjoinedTRUE            0.1561542 0.011198567 1149087  13.94413       0
distance:hasjoinedTRUE   0.0042263 0.000678134 1149087   6.23219       0
 Correlation:
                        (Intr)   dst hasjTR
distance                -0.504
hasjoinedTRUE                 -0.317  0.490
distance:hasjoinedTRUE   0.284 -0.558 -0.879

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-12.2548353  -0.4986704  -0.1176788   0.4974110  12.0951719

Number of Observations: 1167172
Number of Groups: 18082
```

B.5 Targeted offers

All models for targeted offers were of the form isdecon $\sim$ magicbox with random per-user intercept and Bernoulli response.

All offers

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.216452 0.9351298

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon ~ magicbox
                 Value   Std.Error     DF  t-value p-value
(Intercept) -0.2321460 0.022929588 509367 -10.1243       0
offpeakmb    0.2361084 0.008705195 509367  27.1227       0
 Correlation:
          (Intr)
offpeakmb -0.142

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-9.5273059 -0.4705581  0.1305939  0.4947515  9.0740590

Number of Observations: 519810
Number of Groups: 10442
```

Offpeak bonus

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.132785 0.9417122

Variance function:
 Structure: fixed weights
 Formula: ~invwt
```

```
Fixed effects: isdecon ~ magicbox
                          Value  Std.Error     DF   t-value p-value
(Intercept)          -0.3566415 0.03006531 296575 -11.86223       0
OFFPEAK BONUS  0.2091455 0.01661598 296575  12.58701       0
 Correlation:
                     (Intr)
OFFPEAK BONUS -0.074

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-8.8749753 -0.5153174 -0.1133701  0.5254188  8.7009030

Number of Observations: 302047
Number of Groups: 5471
```

Conditional offer

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    1.963963 0.9428031

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon ~ magicbox
                             Value  Std.Error     DF    t-value p-value
(Intercept)               -0.6856449 0.02866150 270069 -23.922155       0
CONDITIONAL CREDIT  0.1065991 0.01503186 270069   7.091541       0
 Correlation:
                          (Intr)
CONDITIONAL CREDIT -0.079

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-7.7371492 -0.5676618 -0.1926093  0.5784528  8.5546937

Number of Observations: 275182
Number of Groups: 5112
```

Friend bonus

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.255645 0.9374207

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon ~ magicbox
                           Value  Std.Error     DF   t-value p-value
(Intercept)         -0.01176362 0.02655160 429497 -0.443048  0.6577
FRIEND BONUS  0.12912144 0.01292929 429497  9.986738  0.0000
 Correlation:
                    (Intr)
FRIEND BONUS -0.07

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-8.9862766 -0.4602246  0.1511391  0.4974440  8.8147183

Number of Observations: 437305
Number of Groups: 7807
```

Credits

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.099989 0.9435518

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon ~ magicbox
```

```
                 Value  Std.Error     DF   t-value p-value
(Intercept)   0.9263741 0.03574163 246029 25.918629       0
CREDIT 0.1091781 0.01908694 246029  5.720039       0
 Correlation:
             (Intr)
CREDIT -0.057

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-9.3160333 -0.3089234  0.2498294  0.5028777  7.9770127

Number of Observations: 249701
Number of Groups: 3671
```

Evening offpeak

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:     1.24272 0.9459248

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon ~ magicbox
                          Value  Std.Error     DF  t-value p-value
(Intercept)            2.0109927 0.03374740 125717 59.58956       0
EVENING OFFPEAK 0.2087128 0.03500627 125717  5.96216       0
 Correlation:
                      (Intr)
EVENING OFFPEAK -0.058

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-8.6413936  0.1582187  0.2906214  0.4730682  1.8803452

Number of Observations: 127231
Number of Groups: 1513
```

B.6 Stateful Magic Box effect, per day effect

decon ∼ 1+day with random per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.234826 0.8789034

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon(start) ~ factor(day)
                      Value  Std.Error     DF   t-value p-value
(Intercept)       -0.7667413 0.10025970 160297 -7.647552  0.0000
factor(day)1       0.7350943 0.09872660 160297  7.445757  0.0000
factor(day)2       0.7299125 0.09884411 160297  7.384482  0.0000
factor(day)3       0.7902921 0.09891136 160297  7.989902  0.0000
factor(day)4       0.7541473 0.09910127 160297  7.609865  0.0000
factor(day)5       1.0767170 0.09953387 160297 10.817594  0.0000
factor(day)6       0.5750512 0.29393608 160297  1.956382  0.0504
 Correlation:
                (Intr)   f(dy1  f(dy2  f(dy3  f(dy4  f(dy5
factor(day)1     -0.965
factor(day)2     -0.964  0.984
factor(day)3     -0.963  0.984  0.983
factor(day)4     -0.961  0.982  0.981  0.981
factor(day)5     -0.958  0.978  0.977  0.977  0.975
factor(day)6     -0.321  0.325  0.325  0.325  0.324  0.323

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-7.9255608 -0.4306612  0.1965034  0.4996725  7.1947519

Number of Observations: 170745
Number of Groups: 10442
```

B.7 Stateful Magic Box effect, repetition and spacing

decon $\sim$ 1 + repetition + spacing with random per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:    2.232665 0.8795584

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: isdecon(start) ~ rep + space
                  Value  Std.Error     DF   t-value p-value
(Intercept)  0.02474906 0.02473396 160301  1.000611  0.3170
rep         -0.05580764 0.02169246 160301 -2.572674  0.0101
space        0.01585329 0.00365900 160301  4.332684  0.0000
 Correlation:
      (Intr) rep
rep   -0.202
space  0.149 -0.780

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-7.0758836 -0.4341071  0.1973765  0.5016335  6.9777457

Number of Observations: 170745
Number of Groups: 10442
```

B.8 Friends effect

decon $\sim$ 1 + hasjoined*hasfriends with random per-user intercept and Bernoulli response

```
Random effects:
 Formula: ~1 | user_id
        (Intercept)  Residual
StdDev:     2.05934 0.9509072

Fixed effects: isdecon ~ hasjoined * hasfriends
                               Value     Std.Error      DF    t-value p-value
(Intercept)                -0.5669382 0.01862062 1149088 -30.446786   0e+00
hasjoinedTRUE               0.1786295 0.00614987 1149088  29.046045   0e+00
hasfriendsTRUE              0.1386160 0.03839429   18080   3.610328   3e-04
hasjoinedTRUE:hasfriendsTRUE  0.1560315 0.01222243 1149088  12.765997   0e+00
 Correlation:
                           (Intr) hsjTRUE hsfTRUE
hasjoinedTRUE              -0.166
hasfriendsTRUE             -0.485  0.080
hasjoinedTRUE:hasfriendsTRUE  0.083 -0.503 -0.161

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-9.8771507 -0.5023967 -0.1185615  0.5012605 10.4678095

Number of Observations: 1167172
Number of Groups: 18082
```

B.9 Social Magic box

open $\sim$ n with Bernoulli response

```
glm(formula = open ~ n, family = binomial, data = actfriends3)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0724  -0.5919  -0.5536  -0.5536   1.9755

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.846370   0.022310  -82.76   <2e-16 ***
n            0.048314   0.001584   30.51   <2e-16 ***

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 20212  on 21211  degrees of freedom
Residual deviance: 19246  on 21210  degrees of freedom
AIC: 19250

Number of Fisher Scoring iterations: 4
```

Bibliography

[1] http://scsn.stanford.edu/index.php, accessed on 01/01/2014.

[2] https://stanfordcapri.org, accessed on 01/01/2014.

[3] http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/cliconfigurationguide/vlans.html, accessed on 10/08/2014.

[4] http://www.insinc.sg, accessed on 10/07/2012.

[5] http://www.liveu.tv, accessed on 15/01/2013.

[6] Central London Congestion charging, Impacts monitoring, Forth Annual Report. Technical report, Transport for London, June 2006.

[7] A. Agresti. *Categorical Data Analysis, Third Edition*. John Wiley & Sons, Wiley Series in Probability and Statistics, 2013.

[8] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM*, 2008.

[9] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *In Proc. of Networked Systems Design and Implementation (NSDI) Symposium*, 2010.

[10] M. Alizadeh, A. G. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). In *ACM SIGCOMM Conference*, pages 63–74, 2010.

[11] E. Ampt. Voluntary Household Travel Behaviour Change - Theory and Practice. In *10th International Conference on Travel Behaviour Research*, 2003.

[12] M. Bagnulo. Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses. In *RFC 6181*, March 2011.

[13] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, New Haven, CT, 1956.

[14] R. Bellman. A Markov Decision Process. *Journal of Mathematics and Mechanics*, 6, 1957.

[15] L. Berger and D. Fedyk. Generalized MPLS (GMPLS) Data Channel Switching Capable (DCSC) and Channel Set Label Extensions. RFC 6002 (Proposed Standard), Oct. 2010.

[16] M. Bhatia, V. Manral, M. Fanto, R. White, M. Barnes, T. Li, and R. Atkinson. OSPFv2 HMAC-SHA Cryptographic Authentication. RFC 5709 (Proposed Standard), Oct. 2009.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[18] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1969.

[19] T. L. Cheung, K. Okamoto, F. Maker, III, X. Liu, and V. Akella. Markov decision process (MDP) framework for optimizing software on mobile phones. In *Proceedings of EMSOFT*, pages 11–20, New York, NY, USA, 2009. ACM.

[20] A. L. Chow, H. Yang, C. H. Xia, M. Kim, Z. Liu, and H. Lei. EMS: Encoded Multipath Streaming for Real-time Live Streaming Applications. In *Proceedings of ICNP*, 2009.

[21] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(5):406–424, 1952.

[22] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008.

[23] G. Dantzig. *Maximization of a linear function subject to linear inequalities, 1947*. New York-London, (Wiley & Chapman-Hall), 1951.

[24] G. Dantzig. The Diet Problem. *Interfaces*, 20/4:43–47, 1990.

[25] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.

[26] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based Congestion Control for Unicast Applications. In *Proceedings of SIGCOMM*, pages 43–56, New York, NY, USA, 2000. ACM.

[27] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. In *RFC 5348*, 2008.

[28] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural Guidelines for Multipath TCP Development. In *RFC 6182*, March 2011.

[29] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. In *RFC 6824*, January 2013.

[30] N. Gomes, D. Merugu, G. O'Brien, C. Mandayam, T. Yue, B. Atikoglu, A. Albert, N. Fukumoto, H. Liu, B. Prabhakar, and D. Wischik. Steptacular: an incentive mechanism for promoting wellness. In *Networked Healthcare Technology (NETHEALTH) workshop, COMSNETS*, 2012.

[31] A. Greenberg, J. R. Hamilton, N. Jain, S. K, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *In SIGCOMM*, 2009.

[32] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In *SIGCOMM*, 2009.

[33] S. Ha, I. Rhee, and L. Xu. Cubic: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42, 2008.

[34] H. Han, S. Shakkottai, C. 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*, 14(6), 2006.

[35] S. HomChaudhuri and M. Foschiano. Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment. RFC 5517 (Informational), Feb. 2010.

[36] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM CCR*, 18/4:157–173, 1988.

[37] F. Kelly. The mathematics of traffic in networks. *The Princeton Companion to Mathematics, Princeton University Press, Timothy Gowers, June Barrow-Green, Imre Leader (ed.)*, pages 862–870, 2008.

[38] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[39] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. In *ACM/SIGCOMM CCR 35(2)*, 2005.

[40] T. Kelly. Scalable TCP: improving performance in highspeed wide area networks. *Computer Communications Review (CCR)*, 32(2):83–91, 2003.

[41] C. K. Keong. Road Pricing: Singapore's experience. In *Essay prepared for the third seminar of the IMPRINT-EUROPE Thematic Network*, 2002.

[42] P. Key, L. Massouli, and D. Towsley. Path Selection and Multipath Congestion Control. In *IEEE Infocom*, 2007.

[43] P. Key, L. Massoulié, and D. Towsley. Path selection and multipath congestion control. *Commun. ACM*, 54(1):109–116, Jan. 2011.

[44] R. Khalili, N. Gast, M. Popovic, and J.-Y. L. Boudec. Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP, draft-khalili-mptcp-congestion-control-01. In *Internet Draft*, July 2013.

[45] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. L. Boudec. MPTCP is not Pareto-Optimal:Performance Issues and a Possible Solution. In *CoNEXT*, 2012.

[46] J. Knockaert, J. Bakens, D. Ettema, and E. Verhoef. Rewarding Peak Avoidance: The Dutch 'Spitsmijden' Projects. In J. A. Nunen, P. Huijbregts, and P. Rietveld, editors, *Transitions Towards Sustainable Mobility*, pages 101–118. Springer Berlin Heidelberg, 2011.

[47] J. Knockaert, Y.-Y. Tseng, E. T. Verhoef, and J. Rouwendal. The Spitsmijden experiment: A reward to battle congestion. *Transport Policy*, 24(C):260–272, 2012.

[48] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. *STACS 99, Lecture Notes in Computer Science*, 1563:404–413, 1999.

[49] E. Mannie. Generalized Multi-Protocol Label Switching (GMPLS) Architecture. RFC 3945 (Proposed Standard), Oct. 2004. Updated by RFC 6002.

[50] D. Merugu, B. Prabhakar, and N. Rama. An Incentive Mechanism for Decongesting the Roads: A Pilot Program in Bangalore. In *NetEcon, ACM Workshop on the Economics of Networked Systems*, 2009.

[51] A. P. Miettinen and J. K. Nurminen. Energy efficiency of mobile clients in cloud computing. In *Proceedings of HotCloud*, 2010.

[52] M. D. Mitzenmacher. The Power of Two Choices in Randomized Load Balancing. In *IEEE Transactions on Parallel and Distributed Systems*, 1996.

[53] J. Mo and J. C. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8:556–567, 2000.

[54] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational), May 2006.

[55] J. Moy. OSPF Version 2. RFC 2328 (Standard), Apr. 1998. Updated by RFC 5709.

[56] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *ACM Sigcomm 2009*, pages 39–50, New York, NY, USA, 2009.

[57] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (Proposed Standard), Nov. 2010.

[58] A. C. Pigou. *The economics of welfare, 1st edition*. MacMillan, London, 1920.

[59] J. C. Pinheiro and D. M. Bates. *Mixed-Effects Models in S and S-PLUS*. Statistics and Computing Series, Springer-Verlag, New York, NY, 2000.

[60] O. Popa. Interface scheduling with Multipath TCP. Master's thesis, University of Cambridge, 2010.

[61] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168, 6093.

[62] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In *Proceedings of MobiSys*, pages 165–178, New York, NY, USA, 2007. ACM.

[63] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *ACM SIGCOMM*, 2011.

[64] C. Raiciu, M. Handley, and D. Wischik. Coupled Congestion Control for Multipath Transport Protocols. In *RFC 5348*, 2011.

[65] C. Raiciu, C. Pluntke, S. Barre, A. Greenhalgh, D. Wischik, and M. Handley. Data Center Networking with Multipath TCP. In *ACM HotNets*, 2010.

[66] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001. Updated by RFCs 4301, 6040.

[67] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), Jan. 2001.

[68] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. The MIT Press, 2005.

[69] M. Scharf and A. Ford. Multipath TCP (MPTCP) Application Interface Considerations. In *RFC 6897*, March 2013.

[70] D. Schrank, B. Eisele, and T. Lomax. TTI's 2012 URBAN MOBILITY REPORT. http://tti.tamu.edu/documents/mobility-report-2012.pdf, accessed on 20/12/2013, December 2012.

[71] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, V. N. Padmanabhan, and K. Jain. Bartendr: A Practical Approach to Energy-aware Cellular Data Scheduling. In *Proceedings of MobiCom*, 2010.

[72] D. Shah and D. Wischik. Principles of resource allocation in networks. In *SIGCOMM education workshop*, 2011.

[73] S. Shakkottai and R. Srikant. Network Optimization and Control. *Foundations and Trends in Networking, NoW Publishers*, 2007.

[74] R. Srivastava and C. E. Koksal. Energy Optimal Transmission Scheduling in Wireless Sensor Networks. *CoRR*, abs/1003.0054, 2010.

[75] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP approach for high-speed and long distance networks. In *IEEE Infocom*, 2006.

[76] K. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, Second edition, 2009.

[77] B. Vamanan, J. Hasan, and T. N. Vijaykumar. Deadline-Aware Datacenter TCP (D2TCP). In *Proceedings of the ACM SIGCOMM*, 2012.

[78] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 303–314, New York, NY, USA, 2009. ACM.

[79] W. S. Vickrey. Pricing in urban and suburban transport. *American Economic Review*, 53/2:452–465, 1963.

[80] W. S. Vickrey. Congestion theory and transport investment. *American Economic Review*, 59/2:251–260, 1969.

[81] J. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1:325–378, 1952.

[82] H. Warma, T. Levä, L. Eggert, H. Hämmäinen, and J. Manner. Mobile Internet in stereo: an end-to-end scenario. In *Proceedings of the Third international conference on Incentives, overlays, and economic traffic control*, ETM'10, pages 64–75, Berlin, Heidelberg, 2010. Springer-Verlag.

[83] J. Widmer and M. Handley. Extending Equation-Based Congestion Control to Multicast Applications. In *ACM SIGCOMM*, 2001.

[84] D. Wischik, M. Handley, and C. Raiciu. Control of Multipath TCP and Optimization of Multipath Routing in the Internet. In *NetCOOP*, 2009.

[85] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *Usenix NSDI*, 2011.

[86] S. B. L. Xiao, A. Mutapcic, and J. Mattingley. Notes on Decomposition Methods. Stanford University Lecture Notes, February 2007. http://www.stanford.edu/class/ee364b/notes/decomposition_notes.pdf, accessed on 20/12/2013.