On the Bleeding Edge

Debloating Internet Access Networks

Toke Høiland-Jørgensen

Faculty of Health, Science and Technology

Computer Science

LICENTIATE THESIS | Karlstad University Studies | 2016:49
On the Bleeding Edge
Debloating Internet Access Networks

Toke Høiland-Jørgensen
On the Bleeding Edge - Debloating Internet Access Networks

Toke Høiland-Jørgensen

LICENTIATE THESIS

Karlstad University Studies | 2016:49

urn:nbn:se:kau:diva-47001

ISSN 1403-8099


© The author

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2016
On the Bleeding Edge: Debloating Internet Access Networks

Toke Høiland-Jørgensen
Department of Mathematics and Computer Science
Karlstad University

Abstract

As ever more devices are connected to the internet, and applications turn ever more interactive, it becomes more important that the network can be relied upon to respond without unnecessary delay. However, this is far from always the case today, as there are many potential sources of unnecessary delay in the network. In this thesis we focus on one of them: Excess queueing delay in network routers along the path, also known as bufferbloat.

We focus on the home network, and treat the issue in three stages. We examine latency variation and queueing delay on the public internet and show that significant excess delay is often present. Then, we evaluate several modern AQM algorithms and packet schedulers in a residential setting, and show that modern AQMs can almost entirely eliminate bufferbloat and extra queueing latency for wired connections, but that they are not as effective for WiFi links. Finally, we go on to design and implement a solution for bufferbloat at the WiFi link, and also design a workable scheduler-based solution for realising airtime fairness in WiFi.

Also included in this thesis is a description of Flent, a measurement tool used to perform most of the experiments in the other papers, and also used widely in the bufferbloat community.

Keywords: Bufferbloat, WiFi, AQM, queueing, network measurement, performance evaluation, fairness
Acknowledgements

I would quite literally not have been where I am today if it were not for Dave Täht and Jim Gettys, who took me under their wings and introduced me to the Bufferbloat community in the fall of 2012, and who arranged for me to go to the 86th IETF in Orlando, Florida in March of 2013 and present my work. Both Jim and Dave have continued to be my friends and collaborators, and I am extremely grateful for their insights and comments on my work, which have sometimes been blunt, but seldom wrong. And a special thank you to Dave for his unrelenting work as a driving force in the community, for taking on anything from reworking the Linux WiFi stack to doing battle with the FCC over misguided regulations, and for many late nights hacking, talking and having fun.

My thanks also to my advisors, Anna Brunström and Per Hurtig, for their insightful feedback throughout all my work, often at odd hours of the day. Theirs has no doubt been the most competent academic supervision I have ever had the fortune of receiving. And an extra thank you to Anna for convincing me to come to Karlstad in the first place; a decision I have not regretted.

My colleagues and collaborators, both at Karlstad University and in the bufferbloat and Linux wireless communities, deserve a huge thank you for inspiring conversations, good ideas, and excellent feedback. It would have been dreary work indeed had it not been for you.

Finally, a special thank you to my friends and family, both in Karlstad and in Denmark, for supporting me in moving to Karlstad in the first place, and for being there for me through both the good times and when the going got tough. It has been fun, and I am glad that the journey is not over yet.

Karlstad, November 2016

Toke Høiland-Jørgensen
List of appended papers


Comments on my participation

For all papers, the ideas have been developed in cooperation with my co-authors. However, I have done the bulk of all writing, implementation, experimental work and evaluation, with the following exceptions:

- In Paper I, the data analysis and most of the text describing the access network dataset was done by Bengt Ahlgren.

- In Paper III, a large part of the MAC layer queue rework implementation was done by Michal Kazior, and the 30 station tests were performed by Sven Eckelmann.

Other publications

# Contents

**INTRODUCTORY SUMMARY**  
1 Introduction 3

2 Research objective 4

3 Background 5  
3.1 Bufferbloat and its mitigation 5  
3.2 The 802.11 WiFi standards 6

4 Research questions 9

5 Contributions 10

6 Research method 12

7 Summary of appended papers 14

8 Conclusions and future work 16

**PAPER I:**  
**Measuring Latency Variation in the Internet** 23

1 Introduction 23

2 Datasets and methodology 24  
2.1 The M-Lab NDT data 25  
2.2 The access aggregation link data 26  
2.3 Sources of latency variation 28

3 Latency variation over time and geography 28  
3.1 Geographic differences 28  
3.2 Development over time 30  
3.3 Different measures of latency span 30

4 Latency variation in the access network 30

5 Examining queueing latency 32  
5.1 Latency reductions after a drop 32  
5.2 Delay correlated with load 34  
5.3 Discussion 36
<table>
<thead>
<tr>
<th>Paper III: Ending the Anomaly: Achieving Low Latency and Air-time Fairness in WiFi</th>
<th>81</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>81</td>
</tr>
<tr>
<td>2 Background</td>
<td>82</td>
</tr>
<tr>
<td>2.1 Bufferbloat in the context of WiFi</td>
<td>82</td>
</tr>
<tr>
<td>2.2 Airtime fairness</td>
<td>85</td>
</tr>
<tr>
<td>3 Our proposed solution</td>
<td>88</td>
</tr>
<tr>
<td>3.1 Re-structuring the kernel queues</td>
<td>88</td>
</tr>
<tr>
<td>3.2 The FQ-CoDel queue management scheme</td>
<td>90</td>
</tr>
<tr>
<td>3.3 Airtime fairness scheduling</td>
<td>93</td>
</tr>
<tr>
<td>4 Evaluation</td>
<td>94</td>
</tr>
<tr>
<td>4.1 Validation of effects</td>
<td>95</td>
</tr>
<tr>
<td>4.2 Effects on real-world application performance</td>
<td>100</td>
</tr>
<tr>
<td>4.3 Summary</td>
<td>103</td>
</tr>
<tr>
<td>5 Related work</td>
<td>104</td>
</tr>
<tr>
<td>6 Conclusion</td>
<td>105</td>
</tr>
<tr>
<td>7 Acknowledgements</td>
<td>105</td>
</tr>
<tr>
<td>8 Appendix: supplementary figures</td>
<td>108</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Paper IV: Flent: The FLExible Network Tester</th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>115</td>
</tr>
<tr>
<td>2 Difficulties when running experiments</td>
<td>116</td>
</tr>
<tr>
<td>2.1 Coordinating different test tools</td>
<td>116</td>
</tr>
<tr>
<td>2.2 Reproducing experiments</td>
<td>117</td>
</tr>
<tr>
<td>2.3 Managing test-bed configuration</td>
<td>117</td>
</tr>
<tr>
<td>2.4 Storing and analysing measurement data</td>
<td>117</td>
</tr>
<tr>
<td>3 Running tests with Flent</td>
<td>117</td>
</tr>
<tr>
<td>3.1 Coordinating different test tools</td>
<td>118</td>
</tr>
<tr>
<td>3.2 Reproducing tests</td>
<td>118</td>
</tr>
<tr>
<td>3.3 Managing test-bed configuration</td>
<td>118</td>
</tr>
<tr>
<td>3.4 Storing and analysing measurement data</td>
<td>119</td>
</tr>
</tbody>
</table>
4 Examples of effective tests enabled by Flent
   4.1 The RRUL test .................................. 119
   4.2 RTT fairness measurements ....................... 120

5 Future planned development .......................... 121
   5.1 Ease of use and wider applicability ............... 121
   5.2 Expanding test coverage and analysis capability 122

6 Related work ...................................... 122

7 Conclusions ....................................... 122
“It’s the Latency, Stupid”

Stuart Cheshire, May 1996
1 Introduction

Technology is becoming ever more ubiquitous, and many people depend on access to the internet for both their personal and professional lives. This has caused a surge in the number of networked devices, and so the quality of our network connections has become ever more important. One thing that can completely ruin the user experience, is having to wait for a response. When performing an action, the response should be immediate; whether it requires interaction across the network or not. However, this is often not the case today; on the contrary, wait times are often long and frequent. Ask any room full of people who use the internet on a regular basis, and chances are that several of them will be used to frequent delays and interruptions.

There are many reasons for such unwanted delays. In this work, we focus on one of the major causes: The latency of the network itself, i.e., the time it takes a signal to travel across the network. Data that traverses the network is affected by this delay, and yet latency is often an overlooked aspect of internet performance. Internet service providers will advertise high-bandwidth connections as being “high speed”, even though many users will not experience an improvement in the performance of their applications, as latency dominates the response time.

There are several reasons for internet latency [1], including transmission delay, media access delay, error recovery, paths changing and processing delays at end-hosts and intermediate nodes, as well as the queueing delay along the path. Since latency is additive, the experience by the end-user is the sum of all these sources of latency. Once you add latency you cannot get it back, as Stuart Cheshire remarked as long as 20 years ago [2]. Furthermore, while there are techniques to increase the effective bandwidth (such as compression), they cannot be used for removing latency. Therefore, eliminating the sources of delay is the only way to ensure a low latency internet experience.

In this thesis we focus on one source of delay: queueing latency along the path. This is by no means a new phenomenon, although it has seen a resurgence in interest over the last several years. The term bufferbloat has been coined to describe the effect that occurs when a network bottleneck is congested and large buffers fill up and do not drain, thus inducing a persistent queueing delay that can be much larger than the path round-trip time [3, 4]. To raise awareness and develop technical solutions to mitigate bufferbloat, an online community\(^1\) has formed, and the work in this thesis should be seen in the context of this community effort, which I have taken part in over the last several years.

The work in the bufferbloat community has had the last-mile access network as its focus point. There are several reasons for this: First, the most appalling instances of bufferbloat has been seen here, and it is something that affects a lot of people. Second, the availability of cheap home routers that can have their firmware replaced, has provided a development platform with a low entry barrier, which works well for a community effort. Finally, there is a sig-

\(^1\)See https://www.bufferbloat.net.
significant motivational factor in working on something that can both solve your own immediate problems with bufferbloat in the home, but also be applicable to hundreds of millions of devices on the wider internet. This last factor has been a great influence for me personally, and is why this thesis follows the wider community effort in focusing on home networks and on real-world implementability, eschewing both big-iron data centres and simulation-only results, which are otherwise so common in academia.

The bufferbloat project has been instrumental in creating a wider acknowledgement of the need for reliably low-latency internet connections, and the work presented here has been part of the work of that community. The work is presented in a straightforward exposition focused on the home network, represented by the four appended papers.

The rest of this introductory summary is structured as follows: Section 2 (below) outlines the main objective with the work presented here. Section 3 summarises related work to give context. Section 4 outlines the main research questions and Section 5 outlines our main contributions. Section 6 relates the work to the traditions of the field of computer science and Section 7 presents a short summary of the appended papers. Finally, Section 8 concludes and outlines future work.

2 Research objective

The overarching research objective of this work is to debloat internet access networks. That is, to eliminate bufferbloat at the edge of the network, where it is most severe and affects the most people. In order to do this, we formulate three subordinate objectives:

We aim to:

1. Increase understanding of the magnitude of the bufferbloat issue in the internet, by measuring the latency characteristics of real-world network connections.

2. Evaluate the efficacy of existing solutions for bufferbloat, by evaluating a series of queue management algorithms designed to mitigate bufferbloat, in a realistic access network setting.

3. Address bufferbloat in WiFi networks, by developing a new set of solutions for this space.
3 Background

This section provides some background on the subject areas of this thesis. This is split into two parts: Section 3.1 outlines the previous work on bufferbloat assessment and mitigation, while Section 3.2 gives an overview of the operation of 802.11 WiFi networks, and outlines previous work on latency performance in the WiFi space.

3.1 Bufferbloat and its mitigation

While bufferbloat is straightforward to demonstrate in a lab setting, there has been some argument over how widespread it is in the real world. An argument which has been hampered by the lack of data. Due to its distributed nature, no one has a global view of the internet, so one is limited to analysing data from a subset of the network. This can be done either by generating measurement traffic to such a look for specific behaviour (i.e., active measurements), or by capturing traffic at a specific point in the network and trying to infer path characteristics from that (passive measurements).

Many studies based on both active and passive measurements have been performed, but most tend to vary in their focus. Many don’t consider latency at all, or only considers it for unloaded links which do not show the symptoms associated with bufferbloat. However, some examples of studies that look at latency do exist. Examples of large scale measurements are [5], which uses 130,000 tests from the Netalyzr test suite to measure queueing latency; [6], which uses data from the BISMark and SamKnows measurement platforms to measure baseline and under-load latency; and [7], which also uses BISMark data (as well as other sources) to measure broadband performance in South Africa. Smaller scale active measurements are described in [8], where active probing of residential hosts is used to measure network connections; in [9], which is specifically targeted at assessing queueing latency; and in [10, 11], both of which use clients under the experimenters’ control to measure bufferbloat in cellular networks. Fewer passive measurement studies specifically look at latency, but a few examples can be found, e.g. [12, 13], both of which employ captures taken at a network edge for their analysis.

The study described in [13] is particularly interesting, because it is one of the few studies that specifically aims at assessing the magnitude of bufferbloat, and because it is framed as a “myth busting” study that questions how often bufferbloat actually occurs in normal internet traffic. One aspect that this highlights is the matter of perspective. While the paper concludes that bloat is “bounded at . . . 100 msec for more than half the samples”, another perspective on this could be that “almost half the samples suffer from as much as 100 msec of bloat”. And so the discussion also becomes one of how much bloat is too much. However, one thing that is certain (and which is also acknowledged in [13]), is that more data is needed to qualify the discussion.

Turning to bufferbloat mitigation procedures, the problem of unmanaged FIFO queues and their impact on latency has been known for a long time, and
the introduction of Active Queue Management (AQM) has been known to be an effective way of handling it if deployed correctly. The most well-known classical AQM algorithm, RED [14], is more than two decades old, and its successor, ARED [15], just turned fifteen. Package scheduling algorithms, which can also help mitigate queueing latency, have even older well-known examples, such as the Stochastic Fairness Queueing (SFQ) algorithm [16], and other fairness queueing algorithms that schedule individual flows to ensure fairness between them.

Many other algorithms have been proposed over the years [17], however few have seen widespread deployment in the internet. This is mostly due to the difficulty of tuning the algorithms, and the adverse effect they can have on performance if tuned incorrectly [18]. And so, until quite recently, no easily deployable queue management techniques were readily available.

As the bufferbloat issue has received increased attention, however, there has been a resurgence of interest in developing algorithms that can effectively control queues without requiring extensive tuning. And so, a number of new algorithms have appeared: Controlled Delay (CoDel) [19] and Proportional Integral controller Enhanced (PIE) [20] are both new AQM algorithms, while FQ-CoDel [21] combines the CoDel AQM with a fairness queueing scheme and an optimisation for improving the latency of “sparse” flows (i.e., flows that don’t use a lot of bandwidth). These algorithms are the focus of the evaluation presented in Paper II.

3.2 The 802.11 WiFi standards

The IEEE 802.11 standards [22] are a series of documents that describe the physical and MAC layers of the wireless network protocol that is commonly known as WiFi. The protocol acts like a virtual Ethernet link, but because the radio waves are propagating through a shared medium, there are several important differences in how the protocol operates. In this section, we summarise the main operational principles of the 802.11 WiFi MAC. We omit the physical layer and focus on the 802.11n revision of the specification, which is what we have used in Paper III. The newer 802.11ac revision mainly differs at the physical layer, and in some constants (such as the max aggregation size) at the MAC layer. The overall operational principles are almost identical between 802.11n and 802.11ac.

When a node in a WiFi network (or “station”, as non-access point nodes are typically called) has data to send, it will first attempt to determine whether or not the channel is free, by listening for other transmissions. If it determines that no other devices are currently transmitting, it will wait a random amount of time (in the order of several microseconds), and then begin transmitting its data frame. After transmission has completed, it will wait for the receiving device to acknowledge the transmission. If no acknowledgement is heard, the sender will assume that a collision occurred (i.e., that another sender transmitted at the same time and the transmissions interfered with each other and were lost), and so will initiate another, longer, random wait period, before
the procedure is repeated. This mechanism of arbitration for the opportunity to send is known as the 802.11 Distributed Coordination Function (DCF). The standard additionally specifies several possible transmission rates (with corresponding physical encodings of the data), and leaves it up to the devices to pick the best rate for a given transmission opportunity, based on the observed signal-to-noise ratio at the time of transmission.

The main advantage of the DCF is that it does not require a central coordinator in the network. However, this comes at a cost: The random back-off periods can lead to inefficiency in the network (i.e., periods of time in which the channel is entirely idle). To limit the overhead of the DCF, aggregation has been introduced into the MAC protocol, allowing several data packets to be sent at once when a device gets a chance to transmit. The standard specifies a maximum size and a maximum transmission time for aggregates, but not how devices should build aggregates. From a queueing latency perspective, however, it is clear that a minimum amount of queueing is necessary in order to have enough packets ready to build a reasonably sized aggregate when given a chance to transmit. The introduction of aggregation also makes the transmission bursty (there will be periods of no data being transmitted, followed by short bursts of several packets when an aggregate is formed). This introduces its own challenges for a low-latency queueing structure, as we will see in Paper III.

Another fundamental property of the DCF is that it gives each device on the network the same probability of winning the contention for the medium, and thus being able to transmit. However, this is independent of the time each device spends transmitting when it gets a chance. Since the transmission rate can vary over a very large interval in the same network, one station transmitting at a very low rate can take up a large fraction of the total transmission time, and so effectively limit the throughput of all stations in the network, which hurts the aggregate efficiency of the network as a whole. This phenomenon is known as the 802.11 performance anomaly [23].

The final property of WiFi networks worth mentioning here is the QoS standards defined by the 802.11e document in the 802.11 standards series. This specifies that a station can transmit data in different priority tiers depending on their Quality of Service markings. There are four QoS levels, labelled (in decreasing priority order) as Voice, Video, Best Effort and Background. The standard specifies different constants for the DCF for each QoS level, giving the higher priority levels a higher probability of winning the contention for the transmission medium. The standard also specifies different aggregation size limits on the different QoS levels, which in theory prevents abuse by trading higher contention win probability for lower maximum throughput (due to the lower aggregation). From a latency perspective, the QoS mechanism can in theory be a big win, but unfortunately there are no agreed-upon standards for how to assign QoS levels to different traffic flows, and implementations tend to be bug-ridden, significantly lowering the usefulness of this feature of the standard.

While there is nothing fundamental in WiFi that says that queues cannot be managed to avoid excess queueing latency, there is little in the literature that
deals with this aspect of the problem. A few examples of work that do treat queueing latency in WiFi networks are [24], which features a WiFi component in a larger evaluation of bufferbloat mitigation techniques; [25], which looks at buffer sizing issues in WiFi networks; and [26], which touches upon congestion in the WiFi hop and uses different queueing schemes to address it, but in the context of a centralised solution that also seeks to control fairness in the whole network. However, none of these studies provides a solution at the WiFi hop itself.

The performance anomaly, on the other hand, has been studied extensively, and several solutions have been proposed, falling roughly into three categories: Those that modify the back-off parameters of WiFi nodes to achieve fairness [27–30], those that change the transmission size [31–33], and those that employ a scheduler at the access point [34, 35]. However, most of the solutions have never made it to actual implementations, whether because the implementation has never left the simulation stage, or because the approach has other tradeoffs that make them impractical to implement. This means that the performance anomaly is still very much present today, as we show in Paper III.
4 Research questions

In order to achieve the overall research objectives outlined in Section 2, we formulate the following research questions that we seek to answer in this work:

1. How bloated is the internet?

   Before one can address a problem, one must first be aware that it exists. In particular, before spending the considerable resources required to deploy upgrades to the edge of the internet (with its billions of connected devices), one must be convinced of the gain. While the bufferbloat problem can be easily demonstrated in a lab, there has been some debate over how prevalent it is in the real world (as discussed in the previous section). Since no one has a global view of the internet, finding the right answer is no trivial task. However, we can make a dent in the problem by analysing the data we do have available, and by looking at smaller scale representative slices of the network. In this work we set out to do just that, and thereby further the understanding of the problem.

2. Can we solve bufferbloat by smarter queue management?

   Knowing that there is a problem to be solved, one naturally starts looking for solutions. In the previous years, several possible solutions to the bufferbloat issue have been proposed, often in the form of queue management and packet scheduling algorithms that seek to ensure low latency even when a link is heavily loaded. However, such algorithms are often evaluated only by their inventors, in scenarios that reflect the design goals of the algorithms. Thus, comparative studies of several algorithms are needed to fully assess their relative efficacy. We set out to provide such a comparison, thereby answering the question of how well the bufferbloat issue can be resolved today.

3. How can we improve the performance of WiFi?

   As explained in the previous section, two main performance issues affect modern WiFi networks: The bufferbloat issue, and the performance anomaly. We set out to find workable solutions for both these issues, using the Linux kernel as our example platform. Linux is widely deployed as the main operating system powering routers and access points (outside of the enterprise setting), which makes it a good choice for an experimental platform. In addition, since we are interested in solving the issues of real-world utility that previous solutions have not, using Linux as our platform is a good fit.
5 Contributions

The main contributions of this work are the following:

1. **A better understanding of the magnitude of bufferbloat in the internet**

   We contribute to a better understanding of bufferbloat and its magnitude in the internet. We do this by combining a large scale active measurement dataset from the public internet with a smaller scale passive measurement dataset from an internet service provider access network. By analysing these two complementary datasets, we are able to say something meaningful of the latency characteristics of the internet as a whole.

   Such a combined analysis has not been performed before, and thus gives new insights: We show that significant latency variation occurs, with large differences between regions, but negligible development over time, despite consistent increases in bandwidth over the same time span. In addition, we use the observed variation in latency as a way to quantify excess latency, and combine it with a novel approach to identify bufferbloat from TCP RTT samples, which we employ on a subset of the data to show that at least some of the latency variation can be attributed to bloat.

2. **An evaluation of existing solutions for bufferbloat mitigation**

   We contribute an extensive experimental evaluation of several modern queue management algorithms in a Linux testbed setup. The testbed is designed to model a typical access network scenario, and the analysis has larger scope than previous work; both in the number of algorithms and in the number of scenarios evaluated. In addition, the use of an experimental setup designed to model the home network provides important real-world performance data on modern queue management algorithms.

   Our evaluation covers a range of simulated access network connection speeds, as well as a WiFi bottleneck scenario. We show that modern queue management algorithms can significantly reduce bufferbloat at the access link, although the tested AQM algorithms have some issues with transient delay spikes as flows start up, and they exacerbate TCP flow unfairness. None of these issues exist for the tested fairness queueing algorithms, which provide low latency and almost perfect fairness. Our web and VoIP application tests show that the improvements in latency afforded by the algorithms translate to real-world application performance benefits. Finally, we show that the algorithms are less effective on a WiFi bottleneck, due to buffering in lower layers of the operating system network stack.

3. **A solution for improving the performance of WiFi**

   We design and implement workable measures to resolve both the bufferbloat issue and the performance anomaly in WiFi, using Linux as an implementation platform. Our solution reduces latency in the stack by
an order of magnitude and increases the efficiency of the network with up to a factor of five. We have worked with the Linux community to get our solution incorporated into the mainstream Linux kernel, to make sure it is not just a theoretical solution, but rather one that can actually be deployed. Our solution builds on state of the art queue management algorithms, and showcases new ways of adapting these techniques to the WiFi domain.

4. *A tool for automation and re-usability of experiments*

In the course of answering the main research questions and developing the contributions outlined above, we have developed and described a testing tool that helps facilitate future experimental work, by making tests repeatable and results sharable. The tool is called The FLEXible Network Tester (Flent), and contains several new features for easy data exploration and automation of experimental work. Flent has shown real-world utility and is widely used in the bufferbloat community.
6 Research method

The field of computing is a diverse one, and has its roots in three different areas: Mathematics, engineering and science [36]. The intermingling of these different traditions has lead to some tension, and in some respects continues to do so to this very day.

The mathematical origins are tied to the theoretical foundations of the field, most notably the theory of computation and the question of what can be computed. Because the practical realisation of computation (i.e., the building of actual computers) has been an integral part of the development of the field, engineering has almost from the beginning played an important role. Today, the engineering influences is most clearly seen in the term “software engineering”. Finally, “computer science” is commonly used when naming university departments (as is the case at Karlstad University), but determining what exactly it is a science of has been the subject of quite some discussion.

The research presented here concerns networks and their function. It wasn’t long after the first computers were built that it became evident that it was useful to have different computers communicate with each other. This soon led to the development of the Arpanet, which in turn has developed into the modern internet [37]. Today, the internet has become such an integral part of society that many functions cannot exist without it. Like the rest of the field of computing, networking rests on the three legs of mathematical theory (e.g., queuing theory, distributed algorithms, etc.), engineering (the internet as a whole is by far the largest and most complicated machine built by humankind) and science (where the network itself can become the subject of study).

The research presented in this thesis lies primarily in the realm of scientific study of the functioning of the network. In networking, this is usually achieved in one of three ways: By simulation (where a computer program will simulate an entire network and the packets flowing through it), emulation (where the network is simulated, but interacts with, e.g., real operating system network stacks) and experiments (where real networking equipment and software is used in the experiments). This work is very deliberately focused on the latter two kinds of experiments. All too often, new concepts in networking are never developed past the simulation stage, which means they languish unused and do not make it to the stage where they can be deployed on the real internet. In addition, simulation is built on idealisations which get ever further from reality as optimisations are added to networking devices, often leading to very different behaviour than those assumed by the simulation tools.

While the main focus of this thesis is scientific study, some aspects of both engineering and mathematical theory is included as well. An important motivation of the work has been to not only characterise the network and discover areas where it can be improved, but to also design and implement such improvements. This is evident in Papers III and IV, which, respectively, builds on previous work to build a workable solution for some of the performance problems seen in WiFi networks, and describes the design of the testing tool. 
used to perform the experiments in the other papers. In fact, the Paper III in itself illustrates the interplay between the different traditions in computing, in that it represents a synthesis of previous ideas that have been discussed in the literature, but never made it into a real-world implementation. While the paper also adds innovations at the algorithmic level, a significant part of its contribution consists of bringing these ideas to a level of maturity where it can be incorporated into a mainstream operating system (Linux, in this case). Finally, Paper III also illustrates how mathematical theory can enhance a work, by incorporating an analytical model of the behaviour of a WiFi network, and using it to inform the design and implementation of the presented solution.
7 Summary of appended papers

Paper I – Measuring Latency Variation in the Internet

In this paper we examine two complementary datasets to assess to what extent bufferbloat is a real problem in the internet. We do this by analysing latency variation in a large-scale active measurement dataset from the Measurement Lab Network Diagnostic Tool, combined with a passive measurement data set from an access link.

The former dataset allows us to look at large scale trends, but because it consists of data from active measurements performed over the public internet, we can only use it to infer the potential for bufferbloat, not the frequency with which it occurs. The other dataset is much smaller in scale, but is based on passive measurements and comes from a network that has known path characteristics. This means that we can make conclusions about what the data shows with higher certainty. The combination of these two datasets allows us to say something meaningful of the latency characteristics of the internet as a whole.

We find that significant latency variation is present in both datasets. Additionally, a more detailed analysis of a subset of the data shows that at least some of it can be attributed to bufferbloat.

Paper II – The Good, the Bad and the WiFi: Modern AQMs in a Residential Setting

In this paper we evaluate a selection of bottleneck queue management schemes in a test-bed representative of residential internet connections of both symmetrical and asymmetrical bandwidths as well as WiFi. Latency under load and the performance of VoIP and web traffic patterns are evaluated under steady state conditions. Furthermore, the impact of the algorithms on fairness between TCP flows with different RTTs, and also the transient behaviour of the algorithms at flow startup is examined.

We show that the tested AQM algorithms can significantly improve the steady state performance, but that they exacerbate TCP flow unfairness and severely struggle to contain queueing delay in transient conditions, such as when flows start up. The tested fairness queueing algorithms, on the other hand, almost completely eliminate these problems and give consistently low latency and high throughput in the tested scenarios.

Finally, we show that all the tested algorithms perform worse on a WiFi bottleneck because they are limited by significant queueing in lower layers of the stack, and thus outside the control of the algorithms.
Paper III – Ending the Anomaly: Achieving Low Latency and Airtime Fairness in WiFi

In this paper we present workable solutions to both bufferbloat at the WiFi link and the 802.11 performance anomaly. We implement a queueing scheme in Linux that is based on FQ-CoDel and tightly integrated with the MAC layer to solve the bufferbloat issue, and a scheduler-based solution to achieving airtime fairness.

We formulate an analytical model for achievable 802.11n throughput with and without airtime fairness and use that to evaluate our solution in combination with a series of testbed experiments. We achieve an order of magnitude reduction in latency under load, large improvements in multi-station throughput, and nearly perfect airtime fairness for both UDP and TCP traffic. Further experiments with application traffic confirm that our modifications provide a significant performance gain for real-world traffic.

Paper IV – Flent: The FLEXible Network Tester

In this paper we present a tool designed to make experimental evaluations of networks more reliable and easier to perform. This tool, called Flent, works by composing well-known benchmarking tools to, for example, run tests consisting of several bulk data flows combined with simultaneous latency measurements. Tests are specified in source code, and several common tests are included with the tool. In addition, Flent contains features to automate test runs, and to interactively plot and explore data collected from experiments.
8 Conclusions and future work

The work presented in this thesis represents a significant step forward in understanding and tackling bufferbloat at the internet edge, and towards deboating the network and thus ensuring low latency internet connections in the future. We have contributed an assessment of the magnitude of excess latency in the public internet, and performed a thorough evaluation of existing queue management solutions targeted at mitigating bufferbloat. In addition, we have developed and implemented solutions to mitigate both bufferbloat in WiFi links and the WiFi performance anomaly. Our solutions are deployable and will contribute directly to reduced latency and reliability in future wireless networks. Finally, we have presented and implemented a testing tool that is widely used in the community today, and which will help future experimenters gain new insights and come up with new improvements.

While these achievements represent a significant step forward, several more steps are needed before we have realised the full potential of the reliably low-latency internet. Some of these steps involve future research, and others are primarily matters of engineering. Of the latter, the most significant is no doubt the effort needed to get the new technology deployed to the hundreds of millions of existing devices all over the world, many of which have an upgrade cycle measured in years. However, as awareness increases of the importance of low latency, and thus of tackling the issues (such as bufferbloat) that prevent it, one can hope that this will happen at an ever increasing rate.

Turning to future research, there are several aspects that warrant further study. On the measurement side, too few studies focus on latency in general, and latency under load in particular. And outside academia the bulk transfer speed test still reigns supreme in both marketing and internet connection tests.\(^2\)

As far as queue management is concerned, there is ongoing work to tune pure AQMs to different workloads and deployment scenarios, in particular for scenarios where fairness queueing mechanisms are perceived (rightly or wrongly) to be inappropriate. Indeed, the question of exactly which scenarios fairness queueing works well for (and which it doesn’t), warrants more study. Furthermore, while the FQ-CoDel algorithm has become the de-facto standard for fairness queueing in the Linux world, there are other fairness queueing schemes in the literature; and FQ-CoDel has not been subjected to the kind of rigorous analysis of its service time guarantees as other algorithms have.

There are still several areas where improvement to WiFi performance is possible, especially when looking at it from a latency point of view. A rate selection mechanism that factor in latency in its decision making is an obvious contender for future study. In the light of the results we have seen, where we can achieve as good or better latency with best-effort traffic as with 802.11e QoS-marked packets, re-evaluating the whole QoS mechanism in the light of modern queueing mechanisms could be another interesting area of inquiry. While the solutions we have seen so far has been limited to single access point cases, it is quite probable that better results could be achieved by cooperation

\(^2\)With [https://www.dslreports.com/speedtest](https://www.dslreports.com/speedtest) as a notable exception.
between neighbouring access points in a dense deployment scenario (which is becoming increasingly common as WiFi becomes ever more ubiquitous). Finally, as raw transmission rates are increasing and more sophisticated signal processing is required for future WiFi standards, more functionality is offloaded to hardware, which means that different architectural solutions are required when not all functions can be controlled by the operating system.

Put together, it is clear that while we have come a long way in improving internet latency, there is still plenty of interesting areas for future inquiry.

References


On the Bleeding Edge

As ever more devices are connected to the internet, and applications turn ever more interactive, it becomes more important that the network can be counted on to respond reliably and without unnecessary delay. However, this is far from always the case today, as there are many potential sources of unnecessary delay in the network. In this thesis we focus on one of them: Excess queueing delay in network routers along the path, also known as bufferbloat.

We focus on the home network, and treat the issue in three stages. We examine latency variation and queueing delay on the public internet and show that significant excess delay is often present. Then, we evaluate several modern AQM algorithms and packet schedulers in a residential setting, and show that modern AQMs can almost entirely eliminate bufferbloat and extra queueing latency for wired connections, but that they are not as effective for WiFi links. Finally, we go on to design and implement a solution for bufferbloat at the WiFi link, and also design a workable scheduler-based solution for realising airtime fairness in WiFi.

Also included in this thesis is a description of Flent, a measurement tool used to perform most of the experiments in the other papers, and also used widely in the bufferbloat community.