

Bachelor Thesis

On Practical Considerations of the Statistical Network Calculus: Traffic Measurement, Distribution Fitting, and Backlog Bounds

by

Ken Klapp

September 10, 2020

Technische Universität Kaiserslautern
Department of Computer Science
Distributed Computer Systems (DISCO) Lab

Supervisor: Prof. Dr.-Ing. Jens B. Schmitt
Examiner: M. Sc. Paul Nikolaus

Abstract

The aim of the presented thesis is to analyze the backlog bounds that were computed for internet traffic, which was captured during regular day to day usage. These backlog bounds were computed using Stochastic Network Calculus (SNC) and Statistical Network Calculus (StatNC) with the latter one being an extension of first one which promises to remove uncertainties that occur in SNC. We use multiple ways of SNC and StatNC to compute the bounds, with each way assuming a different traffic distribution. With the analysis of the backlog bounds, we hope to find out how the captured traffic influences the backlog bound computation in addition to the difference between the bounds computed by SNC and the ones computed by StatNC, as well as the impact the assumed distribution has independent of it matching the distribution of the captured traffic or not.

Acknowledgement

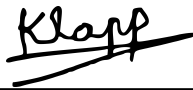
First of all, I would like to thank Prof. Dr.-Ing. Jens B. Schmitt for offering me the opportunity of writing this thesis. Furthermore, I would like to thank my advisor M.Sc. Paul Nikolaus for guiding me so well through this process and always having the time to answer my questions. Also I would like to thank my friend Meris Honsic with whom I share a large part of this topic and who was also always helpful when I struggled. Last but not least, I would like to thank my parents and my brother for always having my back and supporting me emotionally.

Ken Klapp

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Alle wörtlich oder sinngemäß übernommenen Zitate sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Kaiserslautern, den September 10, 2020



Ken Klapp

Contents

Abstract	3
Acknowledgement	5
1 Introduction	1
2 Network Calculus Background	3
2.1 Network Calculus Fundamentals	3
2.1.1 Moment Generating Function (MGF)	3
2.1.2 Arrival Process [JS20]	4
2.1.3 Service Process [JS20]	6
2.2 Traffic Classes	8
2.2.1 Exponential Traffic	8
2.2.2 Bandwidth-Limited i.i.d. Traffic	9
2.2.3 Fractional Brownian Motion	9
2.3 Stochastic Network Calculus (SNC)	10
2.3.1 Exponential Distribution	11
2.3.2 Fractional Brownian Motion Arrival Model	12
2.4 Statistical Network Calculus (StatNC)	12
2.4.1 Framework	13
2.4.2 Exponential Distribution	14
2.4.3 Bandwidth-Limited i.i.d	15
2.4.4 Fraction Brownian Motion Arrival Model	15
3 Traffic Measurements, Backlog Bounds and Distribution Fitting	17
3.1 Purpose	17
3.2 Setup	18
3.3 Execution & Results	19
3.3.1 First Attempts and Obstacles	19
3.3.2 Large Traffic Captures	21
4 Conclusion	29
5 Appendix	31
5.1 Code Snippets	31

List of Figures

2.1	Example of a arrival process in form of a diagram	5
2.2	Arrivals, queue, server, and departures [JS20]	6
2.3	Service of an “always-busy” server. [JS20]	7
2.4	Service of an “sometimes-idle” server. [JS20]	7
2.5	probability density function for various λ parameters	9
3.1	Traffic capture of Grand Theft Auto Online	20
3.2	Traffic capture of one minute of a Family Guy episode	21
3.3	Traffic capture of the movie "The Irishman" on Netflix	22
3.4	Computed bounds the capture of the movie "The Irishman" on Netflix	23
3.5	Traffic capture of the video game "League of Legends" while using the VoIP software "Teamspeak"	24
3.6	Computed bounds of the capture of the video game "League of Leg- ends" with the VoIP service "Teamspeak"	25
3.7	QQ-plots with exponentially assumed distribution plotted against the accumulated amounts of data after each arrival	27

1 Introduction

Network calculus is a theory dealing with queuing type problems encountered in computer networks, with particular focus on quality of service guarantee analysis [Jia09]. One such service guarantee or performance bound is the backlog bound i.e. how large the queue of a server is. There are different branches of network calculus like Stochastic Network Calculus (SNC), which provides stochastic quality of service guarantees, whose research became ever more critical with increasing demand on transmitting multimedia and other real time applications on the Internet [Jia06]. Stochastic Network Calculus has over the last two decades developed as a valuable methodology to compute such performance bounds. As stated in [BHBS14], SNC is very versatile when it comes to the traffic models that can be treated, for which it starts with "clean", a priori and exact probabilistic assumptions. However these assumptions come from observations of the past traffic behaviour which is in the form of measurements and subsequent statistical interference. This interference causes errors and is thus another obstacle for the performance bound computation. Now a new kind of network calculus has been derived, which integrates these statistical interferences into SNC, which moves it towards the so called Statistical Network Calculus(StatNC) [BHBS14]. This now raises the question, how different does StatNC behave compared to SNC when it comes to a performance bound like the backlog bound and also how big the gap is between the bounds of StatNC and SNC and whether StatNC is not too expensive compared to SNC because it integrates these statistical interferences. Also how does the distribution of the traffic and the traffic itself affect this, especially when it comes to Internet traffic that is generated during regular Internet activity, like watching movies or series on an online streaming services or playing video games. This will be the topic of this thesis. It starts of by giving an introduction and explaining the basics of network calculus along with SNC and StatNC. Furthermore the traffic classes that were assumed for the experiments later on are also explained as well as how SNC and StatNC compute the backlog bounds for each traffic class. This is then followed up by row of experiments, in which traffic from the previously mentioned Internet activities is measured and for these measurements, backlog bounds will be computed. These bounds will then be analyzed from which a conclusion can be drawn on whether how big the gap between the SNC and StatNC bound is and what role the distribution of the traffic plays and what the consequences are if the wrong traffic distribution is assumed. This thesis will then be finalized by a conclusion summarizing the discoveries. The last chapter concludes this thesis.

2 Network Calculus Background

This chapter introduces and explains the fundamentals that were employed throughout this thesis. It starts off with the basics of Network Calculus, where alongside an explanation of Network Calculus in general, an explanation of arrival and service processes is provided. Following the basics of Network Calculus, it is now possible to show and explain the traffic classes, that were analyzed throughout this thesis. This chapter is then finalized, by talking about the two kinds of Network Calculus, that are important here, Stochastic Network Calculus(SNC) and Statistical Network Calculus(StatNC), for which the previously explained basics are important as they represent vital fundamentals.

2.1 Network Calculus Fundamentals

As stated in [LBT01], Network Calculus is "a set of mathematical results, often with high complexity, that give insights into man-made systems such as concurrent programs, digital circuits and, of course, communication networks."

With the help of Network Calculus, which uses min-plus algebra, one is able to evaluate the performance bounds of a computer network and analyze the factors that influence it, like the distribution of the arrivals.

In comparison to queuing theory, which gives an average case analyze of a queuing system and where to arrival and service processes are modelled as stochastic processes, network calculus or more specifically SNC and StatNC are used for worst-case analysis and gives stochastic bounds for arrivals and service time.

Such performance bounds could be the backlog bound, which is the performance metric we focus on in this thesis.

2.1.1 Moment Generating Function (MGF)

As written in [JS20] the moment generating function (MGF) $\phi_X(\theta)$ of the random variable X is defined for $\theta \geq 0$ by

$$\phi_X(\theta) := E[e^{\theta X}] = \begin{cases} \sum_{i=1}^{\infty} e^{\theta x_i} P(X = x_i), & \text{if } X \text{ is discrete with } X = x_1, x_2, \dots, \\ \int_{-\infty}^{\infty} e^{\theta x} f(x) dx, & \text{if } X \text{ is continuous.} \end{cases}$$

A MGF of X does not exist at θ if $E[e^{\theta X}] = \infty$. It is called the moment generating function because one can obtain all the moments of a random variable X by "successively differentiating" $\phi_X(\theta)$

In case an MGF exists, then

$$\begin{aligned} \phi'_X(\theta) &= \frac{d}{d\theta} E[e^{\theta X}] \\ &= E\left[\frac{d}{d\theta} e^{\theta X}\right] \\ &= E[X e^{\theta X}] \end{aligned} \tag{2.1}$$

and with $\theta = 0$ we acquire

$$\phi'_X(0) = E[X] \tag{2.2}$$

"In general, the n th derivative of $\phi'_X(\theta)$ evaluated at $\theta = 0$ equals $E[X^n]$, that is,"

$$\frac{d^n}{d\theta^n} \phi_X(0) = E[X^n] \tag{2.3}$$

2.1.2 Arrival Process [JS20]

A arrival process describes a sequence of random variables, also called increments, which are greater than or equal to 0, that arrive between two points in time. More formal, a arrival process is defined by a stochastic process A with time space \mathbb{N} and state space $\mathbb{R}^+ := [0, \infty)$ as

$$A(t) := \sum_{i=1}^t a_i \tag{2.4}$$

The bivariate version is defined as followed

$$A(s, t) := A(t) - A(s) = \sum_{i=s+1}^t a_i \tag{2.5}$$

To illustrate an arrival process, a small and simple example can be seen below in Figure 2.1. Here one can see that the time space is split up into seconds and the arrivals have bit as size. The arrival process $A(3) = 150 + 50 + 100 = 300$ bit that

arrived during the time interval of $(0,3]$. Same goes for $A(2,3) = 100$ bit that arrived between the points in time 2 and 3. This form of an arrival process is also called marked point process, since the arrivals at each point in time are not equal otherwise it would just be called a point process.

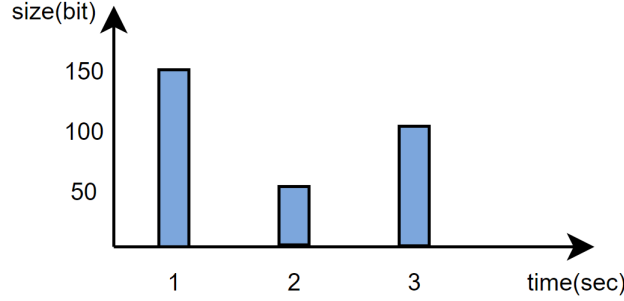


Figure 2.1: Example of a arrival process in form of a diagram

In order to explain certain points later on in the paper, it is necessary to mention the Markov inequality and the Chernoff bound, since these theorems are required for a majority of the proofs later on.

Theorem 2.1. (*Markov Inequality*) Let $X \geq 0$ be a positive random variable and $a > 0$. Then it holds that

$$P(X \geq a) \leq \frac{E[X]}{a}$$

The proof can be found in [Geo15].

Theorem 2.2. (*Chernoff Bound*). Let X be a random variable and $\theta > 0$. Then

$$P(X \geq a) \leq e^{-\theta a} E[e^{\theta X}] = e^{-\theta a} \phi_X(\theta) \quad (2.6)$$

The proof can be found in [JS20].

Theorem 2.3. (*Union Bound / Boole's Inequality*) Let X_1, \dots, X_n be random variables and $x \in \mathbb{R}$. Then holds the Union Bound / Boole's Inequality:

$$P\left(\max_{i=1, \dots, n} X_i > x\right) \leq \sum_{i=1}^n P(X_i > x). \quad (2.7)$$

2.1.3 Service Process [JS20]

A server accumulates incoming arrivals at a point in time t called $A(t)$ and processes them and sends out departures at the point in time t called $D(t)$. It is clear, that at the point in time t , there are less departures than arrivals if there is a constant stream of arrivals. The reason for this is, that for every arrival there cannot be an immediate departure, the arrival has to be processed by the server, which does not happen instantly. Since a server only has a limited capacity of how much it can process at the same time, it cannot process all the arrivals at once and some of the arrivals have to wait until the server has freed up some of its capacity. During this time the unprocessed arrivals are placed into a buffer/queue, the so called backlog.

This process is illustrated inside Figure 2.2 down below.

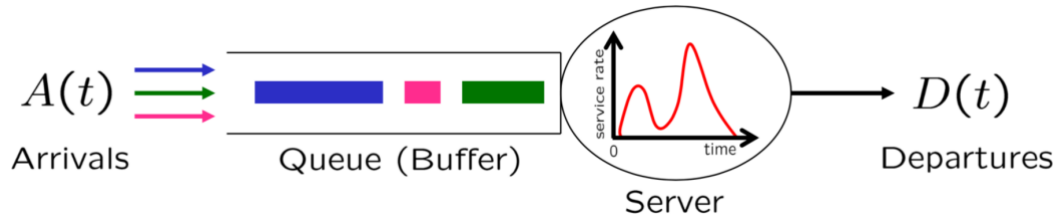


Figure 2.2: Arrivals, queue, server, and departures [JS20]

Backlog

The backlog a server produces at time t depends on the amount of arrivals $A(t)$ and the amount of departures $D(t)$ at that point in time. This backlog can also be seen as the waiting queue length at time t .

$$q(t) := A(t) - D(t) \quad (2.8)$$

There can only be backlog if there are more arrivals than departures, which can only happen if the rate at which a server serves an arrival is higher than the rate at which the arrivals arrive, i.e. when the arrivals exceed the service.

$$A(t) \geq D(t) \quad (2.9)$$

and thus, $q(t) \geq 0$.

"In network calculus, the idea is to use a service process $S(s, t)$ that describes the service between s and t ." This is under the assumption, that the server is always busy between these two points in time. The following equation depicts the service that is done between two points in time during which the server is constantly busy, meaning

there is no idle time. A visualization of such an "always-busy" server is seen down below inside Figure 2.3.

$$S(s, t) = \sum_{i=s+1}^t X_i \quad (2.10)$$

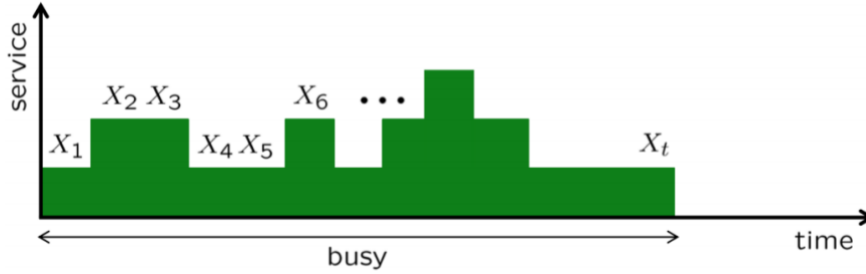


Figure 2.3: Service of an "always-busy" server. [JS20]

However, since a server is not always busy, it is important that relation between arrivals, service and departures is described more generally which includes the idle times. Such a server with idle times could look like the following inside Figure 2.4.

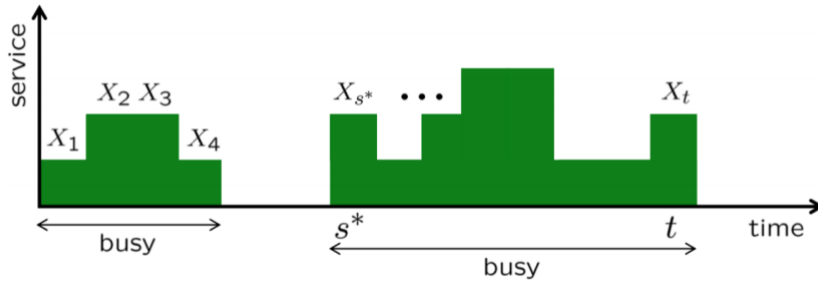


Figure 2.4: Service of an "sometimes-idle" server. [JS20]

The point in time $s^* A(s^*) = D(s^*)$ and thus the backlog at time s^* is also 0. This type of server is a so-called Dynamic S-Server, which will now be defined with one equation.

Definition 2.4. (Dynamic S-Server[BHBS14]) If for all $n \in \mathbb{N}_0$ holds

$$D(0, n) \geq \min_{0 \leq k \leq n} A(0, k) + S(k, n) \quad (2.11)$$

then a service element is called a dynamic S-server. Such a server can also be (σ_s, ρ_s) - bounded.

Definition 2.5. $((\sigma_s, \rho_s)$ - Bounded Service and Arrival[BHBS14]) Let $\theta > 0$, then an arrival is (σ_s, ρ_s) - bounded if

$$\sup_{m \in \mathbb{Z}} \mathbb{E} \left[e^{\theta A(m, m+k)} \right] \leq e^{k\theta \rho_A(\theta) + \theta \sigma_A(\theta)} \quad \forall k \in \mathbb{N} \quad (2.12)$$

A dynamic S-server is (σ_s, ρ_s) - bounded if

$$\sup_{m \geq 0} \mathbb{E} \left[e^{-\theta S(m, m+k)} \right] \leq e^{k\theta \rho_S(\theta) + \theta \sigma_S(\theta)} \quad \forall k \in \mathbb{N} \quad (2.13)$$

Now it is possible to give a stochastic bound on a service element's backlog process which is defined by $q(n) := A(0, n) - D(0, n)$, but before we make use of this, the "Traffic classes" that were assumed for the experiments are going to be explained.

2.2 Traffic Classes

In total, three kinds of traffic classes were analyzed during this experiments. Exponential traffic, bandwidth-limited i.i.d. traffic and fractional Brownian motion. This section will provide a brief overview over the three of them. For the comparison between SNC and StatNC, exponential traffic and fBm were used, while bandwidth-limited i.i.d. is only used with StatNC.

2.2.1 Exponential Traffic

Here exponential traffic means, that the arrivals are exponentially distributed. It has a probability density function which is defined by

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x}, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

With λ being the so called rate parameter and random variable x . The probability density function is depicted inside Figure 2.5.

It has a cumulative distribution function which is given by

$$F(x; \lambda) = \begin{cases} 1 - e^{-\lambda x}, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

The exponential distribution also has the following properties, shown in the list below, which does not show all properties but only the important ones for this paper.

1. Variance: $\text{Var}[X] = \frac{1}{\lambda^2}$
2. Expected Value: $\mathbb{E}[X] = \frac{1}{\lambda}$

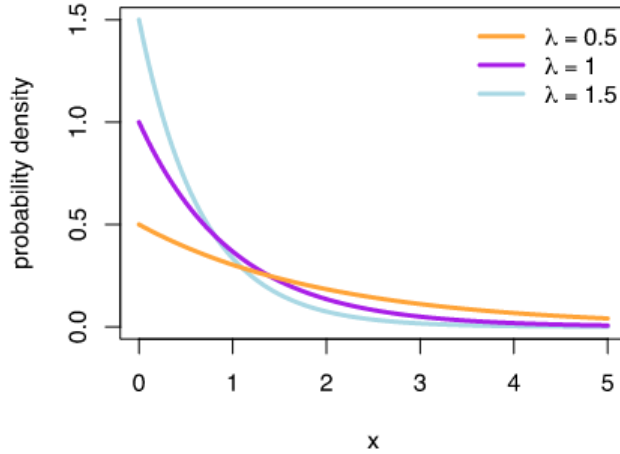


Figure 2.5: probability density function for various λ parameters

2.2.2 Bandwidth-Limited i.i.d. Traffic

This traffic class describes traffic, where the arrivals are i.i.d. meaning independent and identically distributed, but also limited by a bandwidth-limit M . This means, that not more than M data units per time slot are able to arrive, which is a restriction that is present in every network, since no network can process infinite amounts of data at a time.

2.2.3 Fractional Brownian Motion

The fractional Brownian motion [BHØZ08] is a generalization of the Brownian motion. It is defined by a Hurst parameter, which is a constant lying between $(0,1)$.

Definition 2.6. $Z(t)$, which represents a normalized a fractional Brownian motion with Hurst parameter $H \in (0.5,1)$, but only if it upholds the following properties [NHBS18]

1. $Z(t)$ has stationary increments
2. $Z(0) = 0$ and $E[Z(t)] = 0$ for all t
3. $E[Z(t)^2] = |t|^{2H}$
4. $Z(t)$ has continuous paths
5. $Z(t)$ is Gaussian, i.e., all its finite-dimensional marginal distributions are Gaussian.

"A fractional Brownian motion (fBm) $(B^{(H)}(t))_{t \geq 0}$ of Hurst index H is a continuous and centered Gaussian process with covariance function" [BHØZ08]

$$\mathbb{E}[B^{(H)}(t)B^{(H)}(s)] = 1/2(t^{2H} + s^{2H} - |t - s|^{2H})$$

The Hurst parameter H determines to which kind of family the fBm belongs, if $H = 0.5$ then it is a standard Brownian motion. It belongs to one of the other two families, if the Hurst is larger than 0.5, which means that the fBm is persistent, or smaller than 0.5, meaning that the fBm is anti-persistent [BHØZ08].

With the traffic classes explained, we can now look into how the backlog bounds for these are computed using SNC or StatNC.

2.3 Stochastic Network Calculus (SNC)

This section will talk briefly about Stochastic Network Calculus and how it is applied to exponentially distributed traffic and fBm traffic, which were mentioned inside Section 2.2, using the fundamentals that were shown in the previous Section 2.1.

As stated in [FR14]: "The aim of the stochastic network calculus is to comprehend statistical multiplexing and scheduling of non-trivial traffic sources in a framework for end-to-end analysis of multi-node networks."

Stochastic Network Calculus bases itself on the foundations, that were explained in Section 2.1. In SNC, a data flow arrives at a service element and departs again after being processed. Such a flow is represented by a real non-negative stochastic process $(a_k)_k \in \mathbb{Z}$. This arrival flow is defined as shown in Equation 2.5. The service process of the service element can also be described by a doubly indexed stochastic process with the following properties [BHBS14]:

$$\begin{aligned} 0 &\leq S(m, n) & \forall m, n \in \mathbb{N}_0 \\ S(m, n) &\leq S(m, n') & \forall m, n, n' \in \mathbb{N}_0 \text{ and } n \leq n' \end{aligned}$$

This service process along with the arrival and departure process are linked with one another. This is explained through the dynamic S-server explained in the Definition 2.4. Furthermore with the definition of the σ_s, ρ_s -bounded server(2.5), one is able to compute stochastic backlog bounds of a service process.

The theorem used for computing SNC bounds, is the following, which works for any (σ_s, ρ_s) - bounded process and is taken from [BHBS14].

Theorem 2.7. *Let A be an arrival flow served by a dynamic S-server and $\theta > 0$. Assume A is (σ_A, ρ_A) - bounded and S is (σ_s, ρ_s) - bounded. If A is stochastically independent of S , the following probabilistic bound holds:*

$$\mathbb{P}(q(n) > x) \leq e^{-\theta x} e^{\theta(\sigma_A(\theta) + \sigma_s(\theta))} \sum_{k=0}^n e^{k\theta(\rho_A(\theta) + \rho_s(\theta))} \quad (2.14)$$

The proof is demonstrated using the chernoff-bound (2.6) and the union-bound (2.7)

This theorem is the primary key to computing a SNC bound for exponential distributed traffic as we will see in the following section.

2.3.1 Exponential Distribution

When it is assumed that the internet traffic is exponentially distributed, SNC is able to compute a probability P which says how likely the queue size at a point in time t will exceed a given bound B .

Thus we have $P(q(t) > B)$ as indication of how likely it is that the given bound is exceeded.

From the proof of Theorem 2.7 we take

$$P(q(n) > x) \leq e^{-\theta x} \sum_{k=0}^n E[e^{\theta A(k,n)}] E[e^{-\theta S(k,n)}] \quad (2.15)$$

which is our key formula for computing a backlog bound for our internet traffic. It consists of nothing more than $e^{-\theta x}$, the sum of all MGF's of the arrivals between k and n and the MGF of the service process.

We take this formula and solve it for x , because currently $P(q(n) > x)$ gives us a probability based on a given bound but in our case, we require a bound for a given probability, for example we want to have a bound that is not exceeded by 99% of the incoming backlog at a point in time t . The modified formula would look like this:

$$\begin{aligned} P(q(n) > x) &\leq e^{-\theta x} \sum_{k=0}^n E[e^{\theta A(k,n)}] E[e^{-\theta S(k,n)}] \\ \iff \frac{P(q(n) > x)}{\sum_{k=0}^n E[e^{\theta A(k,n)}] E[e^{-\theta S(k,n)}]} &\leq e^{-\theta x} \\ \iff \ln \left[\frac{P(q(n) > x)}{\sum_{k=0}^n E[e^{\theta A(k,n)}] E[e^{-\theta S(k,n)}]} \right] &\leq -\theta x \\ \iff \frac{\ln \left[\frac{P(q(n) > x)}{\sum_{k=0}^n E[e^{\theta A(k,n)}] E[e^{-\theta S(k,n)}]} \right]}{-\theta} &\geq x \end{aligned}$$

Now that we have the formula that computes a bound for a given percentage, we only need define how we compute the MGF's for the arrivals and the service process at the point in time θ .

For the service process, the MGF at θ is computed according to the formula, but for (k, n) , we insert $c * (n - k)$ with c being the bandwidth, with which the traffic was processed.

For the arrival process we use MGF formula mentioned in [BHBS14] which is

$$\phi_{m,n}(\theta) = \left(\frac{\lambda}{\lambda - \theta} \right)^{n-m} \quad (2.16)$$

λ being the arrival rate that results from 1 being divided by the average packet length.

It is also important to mention, that we try to optimize this bound, which means that later during the actual experiments the theta's between 0 and λ are used in order to find the best bound for the chosen percentage.

2.3.2 Fractional Brownian Motion Arrival Model

For fBm traffic we utilize the formula from [NHBS18], where the proof can be found as well, which again makes use of the Chernoff(2.6) and Union(2.7) Bound.

$$P(q(t) > b) \leq \sum_{k=1}^{\lfloor \frac{t}{\tau} \rfloor + 1} e^{-\frac{(b - C\tau + (C - \lambda)\kappa\tau)^2}{2\sigma^2(\kappa\tau)^{2H}}} \quad (2.17)$$

Here λ represents the rate of the arrivals, which is the average packet length and σ^2 being the variance of the arrivals. Furthermore we have C which is the constant rate/bandwidth and which holds a stability condition saying $C > \lambda$. Moreover, we assume that $b > \lambda\tau$ with arbitrary discretization parameter $\tau > 0$. Furthermore H represents the Hurst parameter.

Since this formula again only gives a percentage for a given bound and since we want it the other way around, we would have to rearrange our formula. However this would be far too complicated, since b appears in a binomial term in the exponent. So we solve for b numerically using a bisection method, in which we give a minimal bound and a maximum bound that are high/low enough in order to cover the most solutions and since we try to find the optimal solution again, we optimize between 0 and b divided by the average packet length, which is taken from the assumption that $b > \lambda\tau$ and thus $\lambda > b\tau$.

2.4 Statistical Network Calculus (StatNC)

This section will start with an explanation of the framework on which Statistical Network Calculus operates, which is then followed by explaining how Statistical Network Calculus computes performance bounds for the previously mentioned traffic classes, which also includes bandwidth-limited i.i.d. traffic this time. All these informations are taken from [BHBS14] with the exception of the formula for calculating the performance bound in case of the fBm traffic class, which is taken from [NHBS18].

2.4.1 Framework

Statistical Network Calculus or short StatNC, operates on past observations of the arrival process of a choosen network traffic. With these observations, it is able to calculate statistics on a sample.

"Technically, this can simply be seen as a sufficient condition for the employed statistics, which enables calculations of performance bounds while dealing with uncertainties about the arrival process rising from estimations [BHBS14]. "

First of all, a lemma is defined which proves the monotonic behaviour of the backlog bound in the MGF of A .

Lemma 2.8. *Let $\hat{\phi}_{m,n}(\theta) \geq \phi_{m,n}(\theta)$ for some $\theta > 0$ and all $m, n \in \mathbb{N}_0$ with $m \leq n$, with $\hat{\phi}_{m,n}(\theta)$ being the estimated value of the MGF of A at the point in time θ . Assume A being stochastically independent from S . Then*

$$P(q(n) > x) \leq e^{-\theta x} \sum_{k=0}^n \hat{\phi}_{k,n}(\theta) E[e^{-\theta S(k,n)}] \quad (2.18)$$

for all $n \in \mathbb{N}_0$

Proof can be found inside [BHBS14]

Now we define \mathcal{F} to be the space of functions mapping from $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{R}^+$ to \mathbb{R}_0^+ . In expression, if $\mathbf{f} \in \mathcal{F}$ then:

$$\mathbf{f} : \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{R}^+ \rightarrow \mathbb{R}_0^+ \quad (2.19)$$

An already familiar example for a member of \mathcal{F} is the MGF $\phi_{m,n}(\theta)$ of the arrival flow A .

We now provide a theorem how the uncertainties of using statistics can be combined with the probabilistic bounds derived from SNC.

Theorem 2.9. *Let $\theta^* = \sup\{\theta : \phi_{m,n}(\theta) \leq \infty\}$ and $\Phi : \mathbb{R}^{|n_0|} \rightarrow \mathcal{F}$ be a statistic on $a = (a_{n_0}, \dots, a_{-1})$ such that*

$$\sup_{\theta \in (0, \theta^*)} P\left(\bigcup_{m \leq n} \Phi(a)(m, n, \theta) < \phi_{m,n}(\theta)\right) \leq \alpha \quad (2.20)$$

Then for all $n \in \mathbb{N}_0, \theta < \theta^*$

$$P(q(n) > x) \leq \alpha + e^{-\theta x} \sum_{k=0}^n \Phi(a)(k, n, \theta) E[e^{-\theta S(k,n)}] \quad (2.21)$$

With $\Phi(a)(k, n, \theta)$ being the MGF of $A(k, n)$ at the point in time θ for StatNC.

The proof can again be found in [BHBS14].

From the proof, it is clearer what the nature of the condition in the theorem is. The intersection of the events $\phi(a)(m, n, \theta) \geq \phi_{m,n}(\theta)$ is required in order to leverage the monotonic behaviour of the backlog bound in terms of the MGF of A . This intersection is achieved by partitioning the event $q(n) > x$ and hence have to deal with the corresponding complement, which is the union appearing in the second line of the proof. "This union describes the event, that our statistic delivers a value lying below the real MGF of A at least once. We bound this kind of (estimation) error by a confidence level of α . The confidence level α can be seen as a parameter of optimization." [BHBS14]

2.4.2 Exponential Distribution

The main formula that is used for the computation of a performance bound for network traffic that is the Equation (2.21) from theorem of the previous Section. Just like in Section 2.3.1, where we calculated the bound for exponential traffic with SNC, we have to rearrange the formula because we want to acquire a bound for a given percentage and not a percentage for a given bound.

So we take the formula and rearrange it:

$$\begin{aligned}
 P(q(n) > x) &\leq \alpha + e^{-\theta x} \sum_{k=0}^n \Phi(a)(k, n, \theta) E[e^{-\theta S(k, n)}] \\
 \iff P(q(n) > x) - \alpha &\leq e^{-\theta x} \sum_{k=0}^n \Phi(a)(k, n, \theta) E[e^{-\theta S(k, n)}] \\
 \iff \frac{P(q(n) > x) - \alpha}{\sum_{k=0}^n \Phi(a)(k, n, \theta) E[e^{-\theta S(k, n)}]} &\leq e^{-\theta x} \\
 \iff \ln \left[\frac{P(q(n) > x) - \alpha}{\sum_{k=0}^n \Phi(a)(k, n, \theta) E[e^{-\theta S(k, n)}]} \right] &\leq -\theta x \\
 \iff \frac{\ln \left[\frac{P(q(n) > x) - \alpha}{\sum_{k=0}^n \Phi(a)(k, n, \theta) E[e^{-\theta S(k, n)}]} \right]}{-\theta} &\leq x
 \end{aligned}$$

$E[e^{-\theta S(k, n)}]$ is again calculated with $S(k, n) = c^*(n-k)$, because we assume a constant rate server with rate c .

The only thing that we need to define is how $\Phi(a)(k, n, \theta)$ is computed. For this we again take the formula from [BHBS14], which says:

$$\Phi(a)(k, n, \theta) := \left(\frac{\bar{\lambda}}{\bar{\lambda} - \theta} \right) \quad (2.22)$$

with $\bar{\lambda}$ being:

$$\bar{\lambda} := \frac{\chi_{\alpha}^2(2|n_0|)}{2 \times A(n_0 - 1, -1)} \quad (2.23)$$

$\chi_{\alpha}^2(2|n_0|)$ being the one-sided α -quantile of a Chi-Squared distribution with $2|n_0|$ degrees of freedom.

Through this, we obtain

$$\begin{aligned} 1 - \alpha &= P(\bar{\lambda} \leq \lambda) \\ &\leq \inf_{\theta \in (0, \lambda)} P(\cap_{m \leq n} \Phi(a)(k, n, \theta) \geq \phi(\theta)). \end{aligned}$$

Or, equivalently

$$\sup_{\theta \in (0, \lambda)} P(\cup_{m \leq n} \Phi(a)(k, n, \theta) < \phi(\theta)) \leq \alpha,$$

which is the condition of Theorem 2.9

2.4.3 Bandwidth-Limited i.i.d

We assume that the arrivals are i.i.d. while also clinging to a bandwidth limitation M , which as previously explained means that no more than M data units per time slot can arrive. This is reasonable, since no data link has an infinite capacity. As we see we do not have any knowledge about the distribution of the increments, but as it turns out we still have enough information in order to construct a formula with which we can compute a backlog bound [BHBS14]. In order to compute the backlog bound for bandwidth-limited i.i.d. traffic for StatNC, we utilize the following theorem in order to construct Φ for Theorem 2.9. This theorem is again taken from [BHBS14].

Theorem 2.10. *Let $\epsilon > 0$. The statistic Φ defined by*

$$\Phi(a)(m, n, \theta) := (\bar{A} + \epsilon(e^{\theta M} - 1))^{n-m} \quad (2.24)$$

satisfies the condition in Theorem 2.21. Here

$$\bar{A} := \frac{1}{|n_0|} \sum_{k=n_0}^{-1} e^{\theta a_k} \quad (2.25)$$

2.4.4 Fraction Brownian Motion Arrival Model

For fBm traffic we utilize the formula taken from Corollary 9 from [NHBS18], which is very similar to the SNC formula for fBm, with the difference, that the Hurst parameter H is now unknown. This formula states, that for $b > \lambda \tau$ we have:

$$P(q(t) > b) \leq \alpha + \sum_{k=1}^{\lfloor \frac{t}{\tau} \rfloor + 1} e^{-\frac{(b - C\tau + (C - \lambda)\kappa\tau)^2}{2\sigma^2(\kappa\tau)^{2H_1^{up}} 1 - \alpha}} \quad (2.26)$$

Since here rearranging the formula would be too complicated as well, since we have the same problem as previously for the fBm arrivals in SNC, a bisection method is being employed again, in order to find a bound to a given percentage.

Now that we have defined all the theoretical necessities, we are finally able to move on to the experiments and observe how these different kinds of network calculus behave themselves in practical situations.

3 Traffic Measurements, Backlog Bounds and Distribution Fitting

This chapter will talk about the experiments that were conducted for this thesis. The experiments consist of measuring traffic during the day to day usage of the internet, for which then backlog bounds are computed using the previously mentioned formulas for SNC and StatNC. Also the previously mentioned traffic classes of Section 2.2 are assumed independent of whether the traffic class fits the actual traffic distribution or not. In order to explain the results of a certain distribution, we will use QQ plots later on in order to give an insight, on why these results may have occurred. The resulting bounds will then compared to one another and we will analyze how the assumed distribution and whether SNC or StatNC was used, impacted the resulting bounds and if they fit the desired goal. Also we will try to find out how the traffic itself affects the bound computation. The comparison between SNC and StatNC will be done with exponential distribution and fBm arrivals, while bandwidth-limited i.i.d. is only for StatNC and is observed separately.

The chapter starts off by iterating the goal on why these experiments are done and what we try to achieve by doing them. This is followed up by a description of how these experiments were set up and what tools were used to conduct them, Then this chapter is finalized by a section discussing the results and what these mean.

3.1 Purpose

The purpose of these experiments is to find out how good the performance bounds set by SNC and StatNC under the assumption of certain traffic classes are, regardless whether these traffic classes actually matches the captured traffic or not, which is traffic recorded during day to day usage of the internet. Since SNC and StatNC take different approaches to computing these bounds, it will be interesting to see how large the gap between the SNC and StatNC bound is, since StatNC takes the uncertainty resulting from the estimation errors that appear in SNC into account which results in a larger bound. The question that arises is if this increased cost is tolerable. All the bounds are computed with the goal of the 99%-quantile. The 99%-quantile says which amount of backlogs is higher than the value of this quantile, meaning only 1% of the backlogs are higher than this value. It is also of interest to see which

assumed distribution yields the best result and why it does. In order to explain why a distribution yields a certain result then a different one, the so called QQ plots will be used, which will be explained later during this chapter. The main goal of these backlogs bounds still is to remain cost efficient, while assuring a certain degree of performance, so it is important that it is neither too much overestimated nor underestimated.

Now follows a small example, which tries to clarify the importance of why a backlog bound has to be set correctly.

For example, a person has huge amounts of data which are processed every day. Since his/her bandwidth cannot be infinitely large, it will occur, that large amounts of traffic cannot be processed immediately, but have to be put inside a queue. This unprocessed traffic inside the queue is the so called backlog. Now it is important, that this queue is never filled up to the buffer size, so that no data gets lost. Since this is achievable, but in most cases keeping 100% of the data intact is not required all the time, so in order to save resources a desired goal of 99%-99.99% is often aimed at. Overestimating this goal would result in a waste of resources, which could be allocated otherwise and if this goal is underestimated, a larger amount of data is at risk of being lost, which again is worse than overestimating. This emphasizes the importance of a backlog bound being set correctly as otherwise it would result in negative consequences.

So we see, that a accurate bound is required in order to guarantee that no resources are unnecessarily wasted or no data gets lost.

As final point to this section, it is important to mention again, that the analyzed traffic is traffic that was captured during regular day to day usage of the internet, more specifically traffic that was generated by watching a movie on a streaming platform like "Netflix" or by playing video games. The following section will now talk about the setup of the experiments.

3.2 Setup

First of all, the traffic was captured using the network protocol analyzer "Wireshark" [wir]. The experiments conducted using a wired connection with a 500 Mbit/s bandwidth in a residential area. Since this is so fast, that almost no backlog is generated while streaming a movie or playing a video game, a reduced bandwidth was assumed, which is equal to the average packet length * 1.2, resulting in a utilization of $1/1.2 = 83.3\%$. Furthermore, in order to generate larger and more consistent backlog and to have more increments/arrivals in general, time slots were grouped together. For example the capture file has timestamps for the time 0.01 or 0.2, etc. These will then

be grouped together as time slots of 1 second. The reasons for these steps will become clear later on, when the first attempts at analyzing captured traffic will be discussed.

For the bound computation, the formulas that were described in Chapter 2 were simply implemented using Python code and for calculating the estimated Hurst parameter for SNC fBm and H^{up} for the StatNC fBm bound(2.4.4), R code was used for the implementation taken from [hur]. Furthermore the QQ plots were also generated using Python code. Some of these code snippets can be seen inside the appendix Chapter 5.

3.3 Execution & Results

This section will discuss the ways of how we proceeded through the experiments and the insights we gained by trying out various captured data sets followed by an analysis of the acquired results using plots containing the bounds and the aforementioned QQ plots.

3.3.1 First Attempts and Obstacles

This subsection will talk about the first attempts of capturing internet traffic and analyzing it, during which we encountered a variety of obstacles, which made us rethink our approaches.

We started the experiments of by capturing a video game session of approximately 15 minutes, since we wanted to start small and were unaware of the disadvantages that come with a small amount of data combined with high bandwidth, a variety of problems arose when we tried to analyse the data. The game we played was "Grand Theft Auto Online". In the Figure 3.1 below, we can see, that in the beginning there are two big spikes which occur.

This is the connection phase, during which the game loads up and connect to the multiplayer server. After the big spikes there is constant and low traffic, which consists of UDP packets. At the end there is another spike, which indicates the severing of the connection from the game server. During the establishment and severing of the connection, mostly TCP packets are being transmitted. The x-Axis shows the time in seconds, while the y-Axis shows the packets per second. At the highest peak there are 642 packets being transmitted during 1 second. Assuming the worst case, the largest packet length sent was 14726 bytes. So if all these 642 packets had the same length, which is very unlikely, this would result in a data amount of 9454092 bytes or 9.454092 Mbytes, which for bandwidth of 500 MBit/s is a load that is insignificant when handled by such a large bandwidth and with the following traffic being so

low, this would result in almost none to very little backlog. Furthermore the number of arrivals was too small, since as mentioned before we only captured the traffic for approximately 15 minutes.

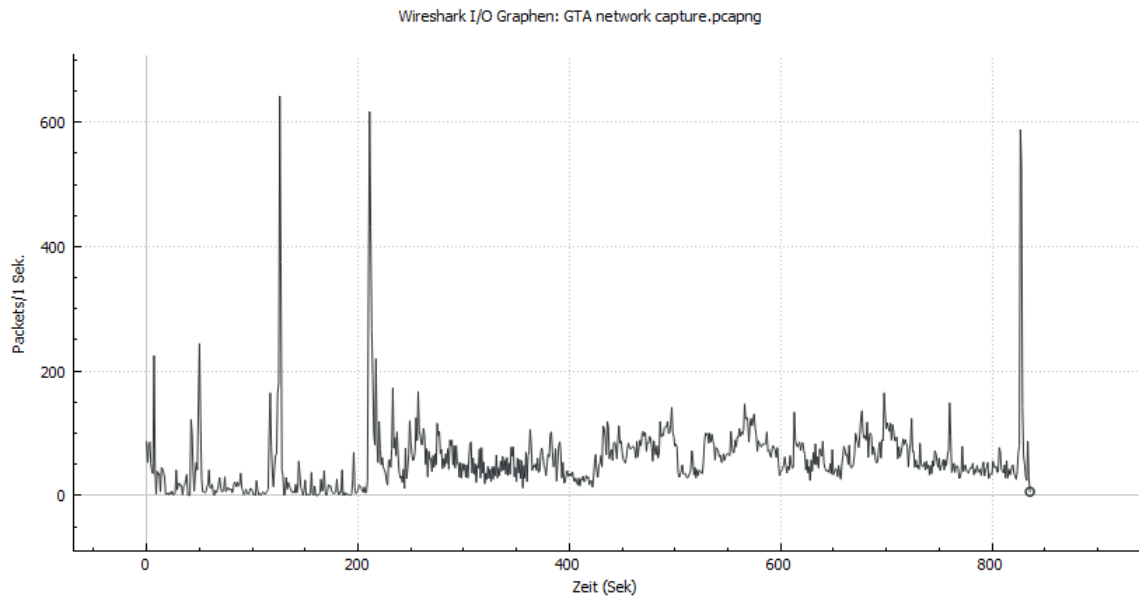


Figure 3.1: Traffic capture of Grand Theft Auto Online

The second attempt was to capture internet traffic while watching a minute of an episode of the series "Family Guy" on the video on demand service Netflix. The series was streamed on the highest possible quality and the codecs that were used are VP9 [vp9] for the video and mp4a for audio. Again in the Figure 3.2 we can see again, that there is not enough data being sent, in order to create a significant backlog with this amount of traffic and the current bandwidth. As stated on their website [net]:

Below are the internet download speed recommendations per stream for playing TV shows and movies through Netflix.

- 0.5 Megabits per second - Required broadband connection speed
- 1.5 Megabits per second - Recommended broadband connection speed
- 3.0 Megabits per second - Recommended for SD quality
- 5.0 Megabits per second - Recommended for HD quality
- 25 Megabits per second - Recommended for Ultra HD quality

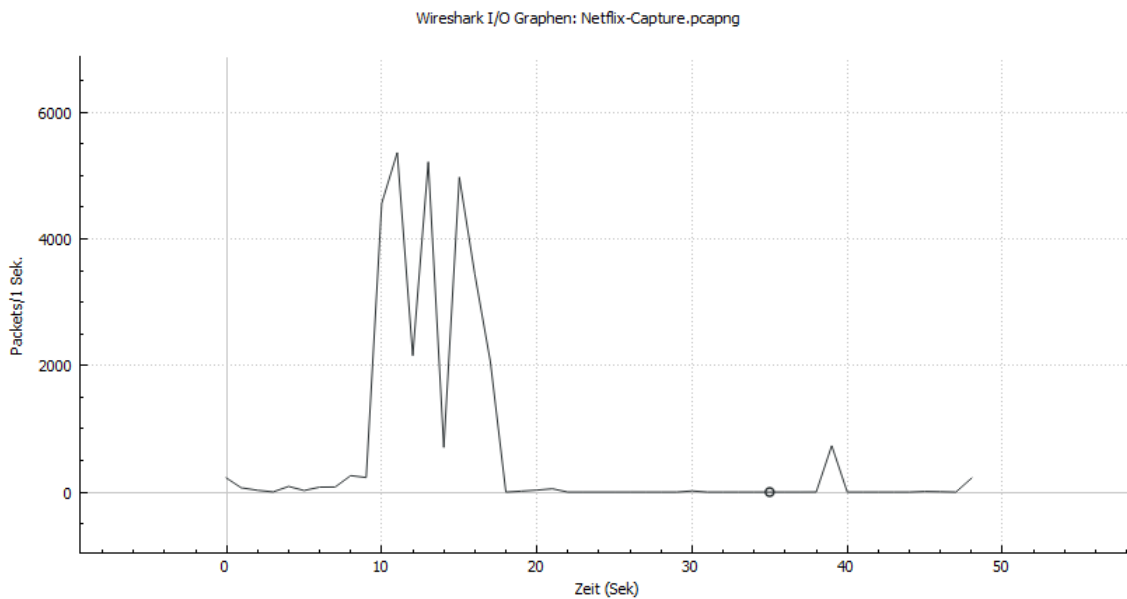


Figure 3.2: Traffic capture of one minute of a Family Guy episode

So after these discoveries we decided to not further analyse the traffic, since using the previously mentioned techniques (grouping timestamps, lowering bandwidth) that were then recommended to us would not have provided the desired effect, since the capture would shrink in size and that would not leave many timestamps and thus increments/arrivals for the analysis. As a next step we decided to capture larger amounts of traffic.

3.3.2 Large Traffic Captures

This subsection will discuss and analyze two larger data traffic captures, which with adjusted bandwidth and grouping of time slots created a backlog, to which a bound from SNC and StatNC can be computed and help us get results that can be analyzed properly. One capture is a capture of the entire movie "The Irishman" from the video on demand service Netflix, which with being 3h30min long provided enough traffic. The second capture was done during an evening of playing a video game with friends with whom we communicated to over a voice over IP service. This session had a duration of almost 4 hours, which again provided enough traffic to conduct a serious analysis, but compared to the movie capture it is less traffic, which we will see influences the results. For each capture the resulting bounds will be shown in form of a graph along with general information about the capture and first observations of the results. The analysis of these two captures will happen in a common paragraph, since as we will see later they show similarities in their results. This will then finalize the subsection and the entire chapter.

These analysis were made with the assumption of a reduced bandwidth. The assumed bandwidth is again the average packet length in Kilobytes * 1.2, since as mentioned before a bandwidth of 500 MBit/s is too high in order to acquire significant backlog. Furthermore the bounds were computed in Kilobytes as well.

"The Irishman"

As mentioned before "The Irishman" is a movie on the streaming platform Netflix and it is 3h30min long. As seen in Figure 3.3 below, it is clear that with the right bandwidth and the right grouping of the time slots a significant backlog can be generated, from which the analysis can be conducted. The graph also depicts that there are constant spikes, while the movie is streamed and that the spikes ease up towards the end, since then the whole movie was done being streamed.

The movie was watched on the highest possible quality and encoded using VP9 [vp9], which is an encoding format developed by Google. The purpose of an encoder is to make the amount of data that is being transmitted smaller, by compressing it. VP9 supports several bit rate modes. It also tries to balance quality and speed, since the quality depends on the time that is given to encode. The more time is given, the better the achieved quality is, but the time required for this may be impractical in certain situations. VP 9 promises that it can reduce the bandwidth as much as 50% compared to other codecs. The audio was encoded using mp4a.

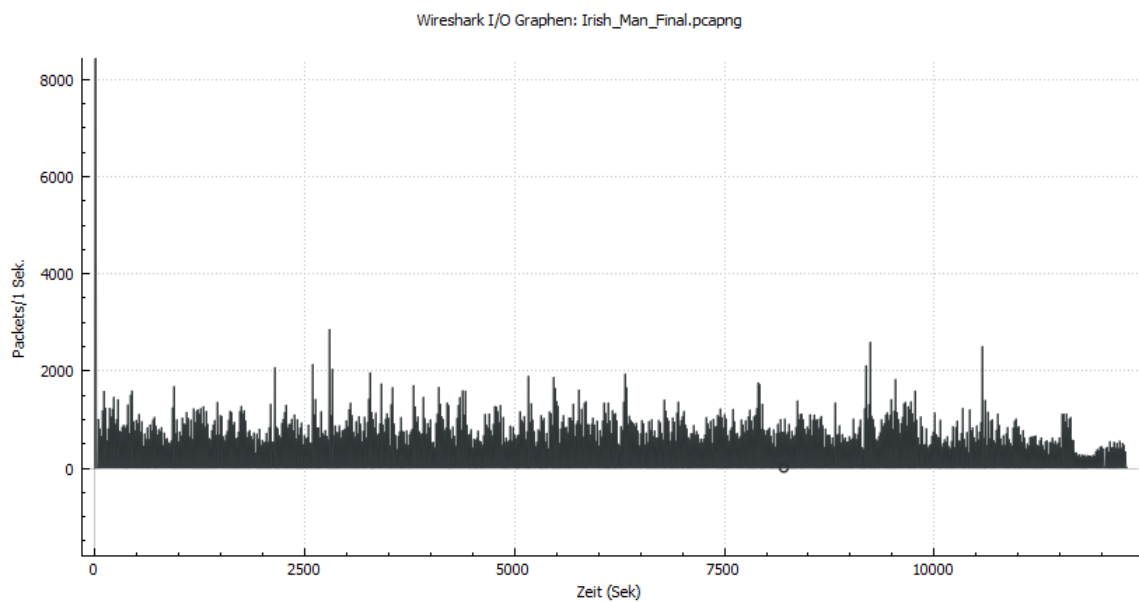


Figure 3.3: Traffic capture of the movie "The Irishman" on Netflix

Time slots of 1 second were used, since as it turns out this duration is fitting for our purposes. The resulting bounds for this capture can be seen below inside Figure 3.4,

where we have the backlog on x-axis and on the y-axis we have the crf(cumulative relative frequency), which says how much of the backlog is below a certain threshold, for example, for crf=0.5 with a corresponding value of 10000 KB on the x-axis says that 50% of the recorded backlogs are lower than 10000 KB. Also it is important to note, that the backlog is only checked every 150 seconds.

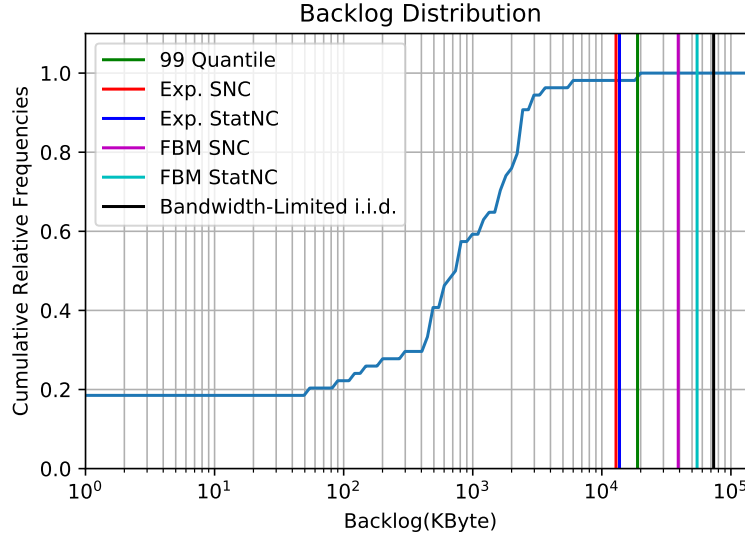


Figure 3.4: Computed bounds the capture of the movie "The Irishman" on Netflix

As we can see the desired backlog bound is the the 99-%-quantile(the green line), the resulting bounds have the following values:

- 99-%-quantile: 18906.193 KB
- Exponential SNC: 12877.871 KB
- Exponential StatNC: 13705.594 KB
- fBm SNC: 39063.461 KB
- fBm StatNC: 54688.445 KB
- Bandwidth-limited i.i.d.: 73498.182 KB

We notice, that the bounds of both fBm SNC and fBm StatNC, as well as bandwidth-limited i.i.d. surpass the 99-%-quantile and cover 100% of the backlogs, meaning they overestimated the bound. This is not necessarily a bad thing as long as the amount of which the bound was overestimated is not too large. On the other hand, the bounds that were computed for exponential distributed traffic, are lower than the 99-%-quantile i.e. they underestimated the bound, with the StatNC bound being obviously slightly higher than the SNC bound in both cases of exponential and fBm. Furthermore approximately 19% of the recorded backlogs are 0.

"Video game with VoIP"

This data was captured on an evening while playing the video game "League of Legends" with friends with whom there was a constant communication using the voice over IP service called "Teamspeak". The game itself does not require a high bandwidth to play it and does not use a lot of data, for 40 minutes of playing the game consumes approximately 40 megabyte [lol]. The same goes for the "Teamspeak" client, which does not require big amounts of bandwidth on the client side with 12.3 Kbit/s used while speaking and $x * 12.3$ Kbit/s while listening to the other users with x being the amount of users currently speaking [tsd]. In our case at most 5 people were speaking at the same time.

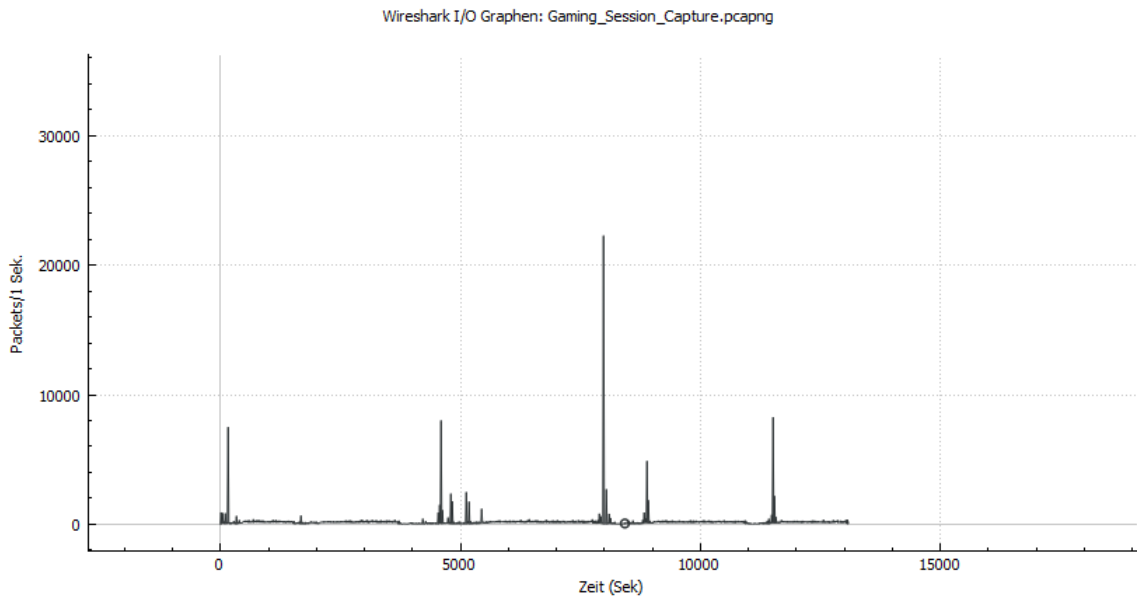


Figure 3.5: Traffic capture of the video game "League of Legends" while using the VoIP software "Teamspeak"

In the graph above in Figure 3.5, we can see that overall there is not much traffic apart from certain high spikes. These spikes occur when a new round of the game was started, when the game client has to load up the game and connect to the other players. One can thus see, that in total 4 rounds were played, with each round being between 20 to 40 minutes. The rest of the graph is very low, because as mentioned before "League of Legends" and "Teamspeak" do not require a high bandwidth.

In the Figure below, we see the resulting bounds that were computed for this capture. Here time slots of 1 are used as well in order to be similar to the previous capture and to emphasise the effect that less captured data has on the results. Also the backlog is written down every 150 seconds again.

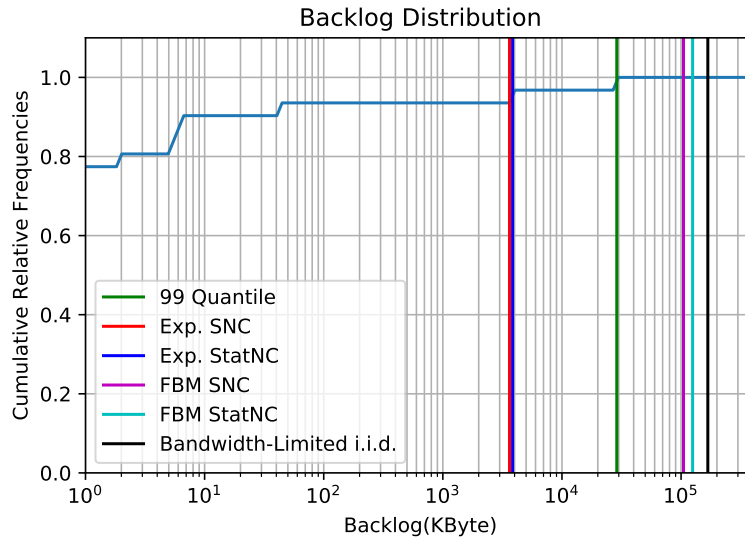


Figure 3.6: Computed bounds of the capture of the video game "League of Legends" with the VoIP service "Teamspeak"

The bounds have the following values:

- 99-%-quantile: 28855.495KB
- Exponential SNC: 3626.913 KB
- Exponential StatNC: 3851.908 KB
- fBm SNC: 104790.815 KB
- fBm StatNC: 125000.875 KB
- Bandwidth-limited i.i.d.: 167749.523 KB

We again notice, that the bounds of both fBm SNC and fBm StatNC, as well as bandwidth-limited i.i.d. surpass the 99-%-quantile and cover 100% of the backlogs. The same goes again for the bounds that were computed with the assumed exponential distribution. They are lower than the quantile and the margin between them and the quantile is larger. Also the majority of the recorded backlogs(almost 80%) are 0, indicating already that there is less traffic when compared to the Netflix capture where only 19% of the backlogs are equal to 0.

Analysis of the Results

As we can see, there are similarities between results that were computed for both data captures.

First of all, what can be said for SNC and StatNC for the exponential bounds as well as the fBm bounds in both captures, is that the margin between the SNC and

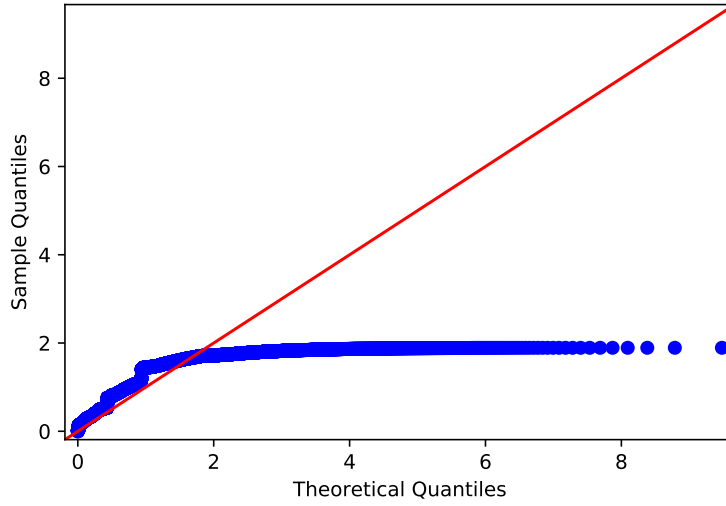
StatNC bound is very insignificant with the largest one being the margin of the fBm bounds from the Netflix movie, which is approximately 15625Kb (15.625Mb), which is a rather small margin and this shows that the price that StatNC pays by taking the uncertainty coming from estimation errors into account, that originate from SNC, is not too high as was already shown in [BHBS14].

Secondly, the bounds that were computed under the assumption of exponentially distributed traffic are lower than the desired quantile, meaning both SNC and StatNC underestimated the bound. In the case of the Netflix capture, both SNC and StatNC cover 98.2% of the traffic, which is very close to the desired 99%. For the video game capture SNC covers 93.5% while StatNC covers 95.3% which indicates, that StatNC delivered a better bound than SNC in this case, but this is a coincidence in this case, since StatNC always has a larger value than SNC.

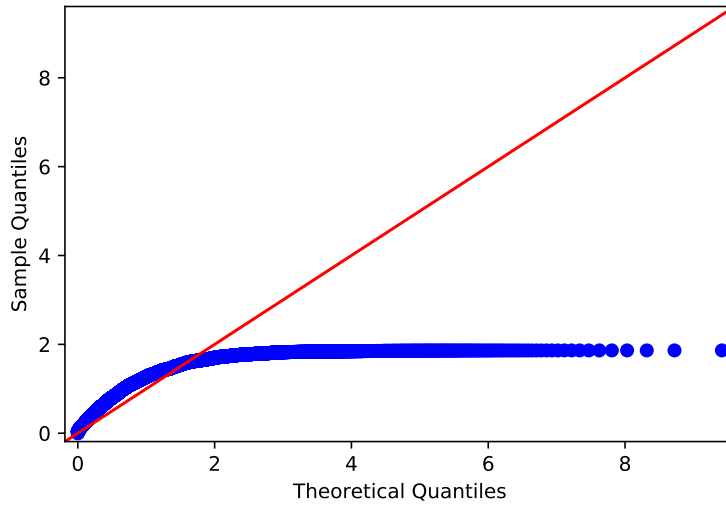
The reason for the underestimating of the bound can be explained with QQ plots, the so called Quantile-Quantile plots, in which quantiles are plotted against each other, with the y-axis containing the input quantiles and the x-axis containing the theoretical quantiles of a given distribution(exponential distribution in this case). The input for the QQ-plot is a list containing the accumulated amounts of data after each arrival. The QQ-plot can be interpreted as the following:

- If the input quantiles match or are identical with the theoretical quantiles of a distribution, the plot should follow along the diagonal 45°line.
- If the QQ-plot is steeper and above the 45°line, then it is generally assumed, that the input quantiles are more dispersed, than the theoretical quantiles.
- If the QQ-plot is flatter and below the 45°line, then it is generally assumed, that the input quantiles are less dispersed, than the theoretical quantiles.

Down below in Figure 3.7 we can see the two QQ plots of the two data traffic captures.



(a) QQ-plot of the video game session capture



(b) QQ-plot of the Netflix capture

Figure 3.7: QQ-plots with exponentially assumed distribution plotted against the accumulated amounts of data after each arrival

As we can see both of the QQ-plots follow the same pattern, they start of by being slightly above the 45° line, meaning the quantiles are more dispersed in the beginning than the exponential ones but later one both of them become less and less dispersed as the theoretical exponential quantiles get larger as we can see that the line gets straighter as it goes on. So we see, that the traffic is not exponentially distributed, hence the wrong distribution is assumed and thus we acquire unsatisfying bounds.

For the bounds computed using bandwidth-limited i.i.d. and fBm, we see that in both data traffic captures, that they cover 100% of the backlogs. As mentioned in the

introduction overestimating the desired goal could lead to resources going to waste, that could be used otherwise. Here these bounds are overestimated by a maximum of approximately 140.000 kilobyte or 140 megabyte, which is a manageable amount of resources.

The fBm bounds are in both cases satisfying, since fBm arrivals allow to capture the characteristics of self-similarity and long-range dependence [NHBS18] which are common characteristics of internet traffic, which seems be the reason, why these bounds are better than the ones set by exponentially assumed distribution. The bound computed by bandwidth limited i.i.d. assumed arrivals is also very satisfying in both cases. The reason for this is that bandwidth limited i.i.d. adheres itself to a bandwidth limit, which as was shown in Section 2.4.3 is enough information in order to compute the bound, without having any knowledge about the distribution of the arrivals and can thus not make any false assumptions about the arrivals. As demonstrated in [BHBS14], StatNC proved to be robust against false distribution assumptions when using the estimator also shown in Section 2.4.3, since unlike SNC, StatNC is not so optimistic and should deliver a bound that is reasonably close to the desired goal, which is the case here for both data captures.

As a last point, one notices, that the amount of backlog influences the gap between the quantile and the bounds, the bounds of the Netflix capture are much tighter and not so far off, than the ones of video game capture, where there is less data traffic in general with some spikes that cause the backlog, which leads to the assumption that the size of the increments/arrivals i.e. the amount of traffic that passed through the network plays a vital role in the backlog bound computation.

4 Conclusion

Concluding we can say, that StatNC certainly has its advantages. We have seen, that the price that StatNC pays is not too high in order to remove the uncertainty that results from estimation errors in SNC. Also we have shown that StatNC is more robust than SNC when using the estimator from Section 2.4.3, which can be used for regular internet traffic, since there is always a bandwidth limitation present. What can be said for certain is, that in case of internet traffic like video games or online streaming platforms, that assuming a distribution, like fBm arrivals, that recognizes characteristics like self-similarity, etc. which are common characteristics of internet traffic, delivers a satisfying result. In other words, assuming the right distribution is a vital part of bound computation, especially when it comes to traffic containing the aforementioned characteristics. Furthermore, the analysis of the captured traffic would not have been possible without the assumption of a reduced bandwidth, since these internet activities do not require a lot of bandwidth, meaning they can be handled by the bandwidth of most people. According to this report [ger] from December 2016, a country like Germany then had an average download speed of 40.38 Mbit/s, which is already more than 1.5 times the required bandwidth to watch Netflix on ultra HD quality, its highest quality. So this were relatively small scale experiments and it would be interesting to see the results that would originate from such an experiment being conducted on a larger scale. This concludes the thesis.

5 Appendix

5.1 Code Snippets

This section will show code snippets which depict some of the implemented formulas, in order to give a picture of how these formulas were implemented. It will cover mainly the StatNC implementation of the exponential distribution and FBM, since concerning the implementation SNC and StatNC do not show any significant difference.

The following code was used for computing a backlog bound under the assumption of exponential distributed traffic using StatNC.

```
#StatNC
from scipy.stats import chi2
print( 'StatNC: ' )
stat_nc_bound = 0

def lambda_bar(alpha):
    wD = 2 * slotwd.shape[0]
    quantile = chi2.ppf(alpha, wD)
    cumulated_arrivals_Byte = slotwd["arrival_process_Byte"].iloc[-1]
    cumulated_arrivals_MByte = slotwd["arrival_process_MByte"].iloc[-1]
    lambda_bar = quantile / (2*cumulated_arrivals_Byte)
    lambda_bar_MByte = quantile / (2*cumulated_arrivals_MByte)
    return lambda_bar_MByte
lambda_bar = lambda_bar(0.001)

print(lambda_bar)

def mgf_stat_nc(theta, s, t, alpha):
    return (lambda_bar / (lambda_bar - theta))**(t - s)

def q_stat_nc_b(theta, t, p, alpha):
    sum_mgf = 0
    temp_bound = 0
    if theta <= 0:
        return math.inf
    for i in range(0, t):
        sum_mgf += mgf_stat_nc(theta, i, t, alpha)
```

```

        * mgf_service(theta, i, t)
    if (sum_mgf == float("inf")):
        temp_bound = math.inf
    else:
        temp_bound = math.log((p-alpha)/sum_mgf)/(-theta)
    return temp_bound

def exp_stat_nc_backlog_bound_b(t, p, alpha):
    #print("lambda:", lambda_bar(alpha))
    rranges = (slice(0, lambda_bar, 0.1),)
    resbrute = optimize.brute(q_stat_nc_b, rranges, args = (t,p,alpha),
    full_output=True, finish=optimize.fmin, disp=True)
    print(resbrute[0])
    print(resbrute[1])
    return (resbrute[1])

exp_stat_nc_bound = exp_stat_nc_backlog_bound_b(number_rows,0.01,0.001)
print(exp_stat_nc_bound)

```

We can see how the first method computes $\bar{\lambda}$ using a given alpha, which is as mentioned previously our confidence level, which is always chosen around 0.001 so that we have a confidence level of 99.9%. The second method simply calculates the service MGF. This is followed by the third method, which is the actual implementation of the formula, it uses the two previous defined methods and only returns the result if it is an actual number and not infinite. Since we try to optimize the bound, the fourth method uses an optimization procedure, with which it tries to find the optimal θ by using all thetas between 0 and $\bar{\lambda}$ and computing the formula with it. The best one is then chosen at the end.

Next up is the formula for FBM in StatNC. The only thing that is computed externally in a R file is the H^{up} parameter. The code used for this was taken from. The rest of the code looks like the following:

```

#StatNC

def fbm_statnc_bound(tau, t, b, alpha):
    k_vec = np.arange(1, math.floor(t/tau)+2)
    numerator = (b-bandwidth_MByte*tau
    + (bandwidth_MByte-avg_packet_length_MByte)*k_vec*tau) ** 2

    denominator = 2*var_packet_length_MByte
    * ((k_vec*tau)**(2*h_up)) * (1-alpha)
    exp_frac = np.exp(-numerator/denominator)
    return alpha + np.sum(exp_frac)

```

```

def fbm_statnc_bound_opt(t,b,alpha):
    rranges = (slice(0.01, (b/avg_packet_length_MByte), 0.01),)
    resbrute = optimize.brute(fbm_statnc_bound, rranges,
    args = (t,b,alpha),
    full_output=True, finish=optimize.fmin, disp=True)
    return resbrute[1]

# https://www.geeksforgeeks.org/program-for-bisection-method/
def bisection_statnc(p,t,alpha,mini,maxi):
    c = mini
    while ((maxi-mini) >= 0.01):
        print(mini,maxi)
        # Find middle point
        c = (mini+maxi)/2
        # Check if middle point is root
        if (round(fbm_statnc_bound_opt(t,c,alpha),2) == p):
            break
        # Decide the side to repeat the steps
        if (round(fbm_statnc_bound_opt(t,c,alpha),2) < p):
            maxi = c
        else:
            mini = c
    return c

fbm_stat_nc_bound = bisection_statnc(0.01,number_rows,alpha,1,1000000)
print(fbm_stat_nc_bound)

```

The first method simply shows the formula from Section 2.4.4. It was implemented using vectors in order to increase the runtime, since this method will be called upon later very often. The second method again serves as optimization, which tries to the optimal τ for the bound. Afterwards in the third and final method, the previously mentioned bisection is finally used, in order to compute the bound by giving a desired percentage. The only thing required is a given starting bound, which is large enough for the bisection to be successful.

Bibliography

- [BHBS14] Michael A Beck, Sebastian A Henningsen, Simon B Birnbach, and Jens B Schmitt. Towards a statistical network calculus—dealing with uncertainty in arrivals. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2382–2390. IEEE, 2014.
- [BHØZ08] Francesca Biagini, Yaozhong Hu, Bernt Øksendal, and Tusheng Zhang. *Stochastic calculus for fractional Brownian motion and applications*. Springer Science & Business Media, 2008.
- [FR14] Markus Fidler and Amr Rizk. A guide to the stochastic network calculus. *IEEE Communications Surveys & Tutorials*, 17(1):92–105, 2014.
- [Geo15] Hans-Otto Georgii. *Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Walter de Gruyter GmbH & Co KG, 2015.
- [ger] Teamspeak data consumption. <https://www.speedtest.net/reports/germany/>. [Online; accessed 01-September-2020].
- [hur] Hurst Parameter. https://github.com/paulnikolaus/StatNC-fBm-Extension/blob/master/estimate_hurst.R. [Online; accessed 19-August-2020].
- [Jia06] Yuming Jiang. A basic stochastic network calculus. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 123–134, 2006.
- [Jia09] Yuming Jiang. Network calculus and queueing theory: Two sides of one coin. In *VALUETOOLS*, page 37. Citeseer, 2009.
- [JS20] Paul Nikolaus Jens Schmitt. Lecture notes in stochastic analysis of distributed systems, 2020.
- [LBT01] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queueing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.
- [lol] League of Legends data consumption. <https://betterplaygame.blogspot.com/2019/03/lol-data-consumption-leage-of-legends.html>. [Online; accessed 01-September-2020].
- [net] Internet Connection Speed Recommendations. <https://help.netflix.com/en/node/306/>. [Online; accessed 3-August-2020].

- [NHBS18] Paul Nikolaus, Sebastian Henningsen, Michael Beck, and Jens Schmitt. Integrating fractional brownian motion arrivals into the statistical network calculus. In *2018 30th International Teletraffic Congress (ITC 30)*, volume 2, pages 37–42. IEEE, 2018.
- [tsd] Teamspeak data consumption. <https://support.teamspeak.com/hc/en-us/articles/360002710657-How-much-bandwidth-does-TeamSpeak-require->. [Online; accessed 01-September-2020].
- [vp9] VP9. <https://developers.google.com/media/vp9>. [Online; accessed 3-August-2020].
- [wir] Wireshark. <https://www.wireshark.org/>. [Online; accessed 01-September-2020].