TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Bachelor Thesis

On Practical Considerations of the Statistical Network Calculus: Data Aggregation, Traffic Analysis, and Backlog Bounds

by

Meris Honsic

September 9, 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

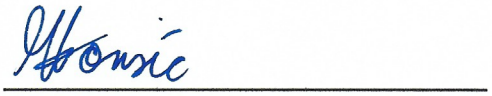
Abstract

Statistical network calculus is a rather new framework which can be used for the case of limited traffic information, since it is based upon already recorded behavior of traffic traces. Due to this property, we can compute guarantees and performance bounds although we do not know all properties of a specific traffic trace. Recently it was proposed in [NHBS18] that StatNC is also a robust framework for this case. In contrast Stochastic network calculus is a versatile framework which is based on, as the name suggests, stochastic. This framework is not considering the traffic behavior but assuming full knowledge about it to calculate performance bounds and delay guarantees. SNC was tested thoroughly but on the contrary StatNC has mostly gone through simulations. This thesis will close the gap and will test both (SNC, StatNC) in a series of chosen distributions, based on captures made by research groups such as Bell Core, to see the behaviour in a real environment instead of a simulation. Afterwards we use Q-Q plots to infer the distribution for our traces. We will check if the claims made from [BHBS14, NHBS18] considering StatNC can hold in real situations, therefore we will compute performance bounds on the traces which will be mapped on the corresponding backlog distribution and deliver a concept of proof for the formulas based on the StatNC framework. Additionally, we will show that we can collect information from distributions that do not match the traffic and therefore result into bad performance bounds. We will explain the difficulties that comes with the testing in a real environment such as limitations and dependencies of parameters. Finally we will analyze our results and show that the fBm model is very robust based on our findings and that with Q-Q plots we can infer the distributions fairly well.

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 9, 2020



Meris Honsic

Contents

List of Figures

1 Introduction

1.1 Motivation

When people first started to analyze traffic on the internet, there was considerably less traffic and minor problems with congestion. This problem grew with the expanding complexity of our traffic which lead to upgrades such as adding congestion control to TCP just to name one. Performance in distributed systems is a timeless problem. The more sophisticated ways were found to fulfill a given capacity the more the demand grew with the increased supply. As a result, the sizes of the messages grow by the time and the complexity how packets are transmitted too. Today, there are loads of different distributions a specific traffic can have and also many properties that can be determined.
For this purpose Network Calculus can be used to derive deterministic or stochastic performance bounds or service guarantees [Fid10]. For the most models the Stochastic Network Calculus (SNC) is used to compute guarantees such as performance bounds or delay guarantees via a violation probability. The Deterministic Network Calculus (DNC) is used for a strict buffer size where we have to give hard guarantees in a network in contrast to SNC where the buffer size is considered infinite. There are also statistical effects which can be utilized by SNC which DNC cannot use for example in multiplexing [Fid10].
On the other hand Network Calculus based on statistics does not assume full knowledge about the traffic, but considers incoming traffic itself. We make less assumptions about the traffic. This new framework is called Statistical Network Calculus (StatNC). Since it is a fairly new model the small amount of theorems about the framework are mathematically proven but not enough researched in a real environment. So far they were only used for simulations but not for real traces.
The growing complexity for network traffic and generally speaking for networks causes us to work out better techniques to handle them. Analysis tools can help developers handle traffic better and chose adequately better buffers for the system for a maximum usage. For that to work, we need a proof of concept that such theorems hold in a real environment.
In this thesis we tackle that lack of real environment tests. We compute performance bounds, based on SNC and StatNC, for the exponential arrivals, fBm arrivals and limited bandwidth arrivals. Since SNC does not take the behavior of the real traces into account, we simply assume the traces have this specific distribution and calculate an SNC based performance bound on it. Afterwards the StatNC based performance

bound is calculated with respect to the traces. This way we have additionally the direct comparison between SNC and StatNC. By providing a real environment we can derive the robustness of the fBm model and especially in StatNC. Furthermore we can showcase the importance of Q-Q plots, since they can check how appropriate the distribution is to a given trace.

The thesis will be structured as follows. First there is a background explanation of Network Calculus and the respective components. Furthermore, the traffic classes are be showcased with their properties and the background to calculate performance bounds in SNC and StatNC. The following chapter explains the experiment in detail with the used methods, implementation and the idea behind it. The last chapter contains our analysis from the results followed by the conclusion.

1.2 Related Work

A great contribution has been done by [BHBS14] which lays the foundation of StatNC and recently by [NHBS18], which added the fBm model to StatNC. They pushed the framework of StatNC to develop it further and achieve better results. They introduced and extended a variety of different performance bounds of different distributions for the StatNC framework, which can be more useful than the stochastic approach, since we make less assumptions. The tests in the work are based on single flows, but the usual traffic is more complex than one flow. They simulated one flow a million times [NHBS18] to derive their Hurst parameter H for the StatNC bound. For the SNC bound they used the true parameter H, since they chose it beforehand for the simulations.

2 Network Calculus Background

Before we start with our work, we provide the necessary background to understand the basic concepts of Network Calculus. Network Calculus is used for example to compute delay and backlog bounds for a given arrival curve (DNC) or distribution (SNC), in deterministic or time sensitive networks. In the following sections we discuss the basics of it and later introduce different approaches on Network Calculus such as Stochastic Network Calculus and Statistical Network Calculus in more detail and how their respective framework function. We discuss the components of Network Calculus such as arrivals and service and in what manner they are connected. Furthermore we showcase further components such as the moment-generating function which is key to determine distributions and further calculate backlog bounds. We categorize different distributions and showcase how they play a role in our thesis.

2.1 Arrivals and Service

2.1.1 Arrivals

Beginning with the arrival process, it is a sequence of random numbers called *increments*, measured over time as seen in Figure 2.1.

Definition 2.1. (Arrival Process [JS20]). Let $(a_i)_{i\in\mathbb{N}} \geq 0$ be a sequence of real random variables, the so-called increments. We define an arrival process by the stochastic process A with time space N and state space $R^+ := [0;\infty)$ as

$$A(t) := \sum_{i=1}^{t} a_i. \tag{2.1}$$

As we see in the figure, we could now calculate the cumulative arrivals of our system for $t1$, $t2$ and $t3$ where we add up our arrivals until the respective time point t.

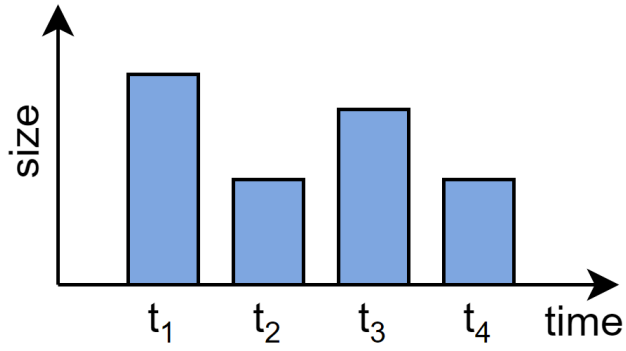


Figure 2.1: A diagram of an arrival process in discrete time.

2.1.2 Service

Now the service needs to be determined. $S(s,t)$ is the service that can be provided between s and t. The Service process can be expressed with the concept of a Dynamic S-server.

Definition 2.2. (Dynamic S-server [NHBS18]) Assume a service element has an arrival flow A as its input and the respective output is denoted by A'. Let $S(s,t)$, $0 \leq s \leq t$, be a stochastic process that is non-negative and increasing in t. The service element is a dynamic S-server iff for all $t \geq 0$ it holds that:

$$A'(0,t) \geq \inf_{0 \leq s \leq t} \{A(0,s) + S(s,t)\}.$$

Additionally, in our thesis we assumed we have a constant rate server which is a special case, to reduce deviation.

Definition 2.3. (Constant Rate Server [JS20]).

$$S(s,t) = c * (t-s), \text{ for all } 0 \leq s \leq t, \text{ with a given capacity } c.$$

This is needed to compute backlog and the respective bound as we see later. If the arrival speed exceeds the speed of our service processing those increments, then we obtain a backlogged system. Now that we have our arrivals and our service, we can start to compute a bound for a given traffic such as illustrated in Figure 2.2. The problem that arises is that, how to find an appropriate distribution for a given traffic exactly and assign an adequate buffer?

The buffer is only filled if we have more arrivals then departures: $A(t) - A'(t) > 0$, or in other words we define the buffer at time t to be the amount of traffic, that entered

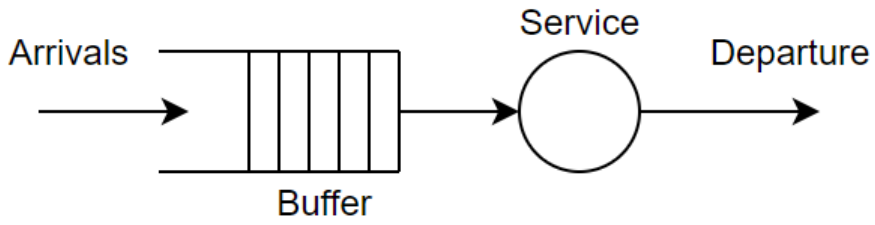


Figure 2.2: An example of a queuing system with arrivals and a buffer if the service cannot handle the arrivals fast enough.

the system, but has not departed yet. The arrivals stored in the buffer are mostly referred as the backlog [JS20]. The idea is we need to assign a fitting buffer size such that the system can handle the backlog at all times without having to drop packets. In our model, we start with assuming at first that we have an infinite buffer, since that is the idea of SNC, otherwise it would be a problem fitting for DNC. We then calculate bounds for our system, with Network Calculus, which declare the maximum backlog that can be accumulated with a given probability. There are multiple factors that need to be considered such as different distributions and their own distributional properties, for example if the traffic has a memory-less or self-similar property. Those factors can be handled with different traffic classes which we will approach in section 2.3.

2.1.3 Backlog

As we mentioned before a system accumulates backlog if we have more arrivals then departures for a given time t. In other words our system cannot process the packets fast enough with the speed they are arriving. A backlog at given time t can be expressed as:

Definition 2.4. (Backlog [NHBS18, JS20]) The backlog of a server at time t is the difference between arrival process $A(t)$ and according departure process $A'(t)$ at time t,

$$q(t) := A(t) - A'(t).$$

For the backlog bound we conclude:

Definition 2.5. (Backlog Bound [NHBS18]) Let $q(t) := A(t) - A'(t)$ be the backlog at time t, $S(s,t)$ be the service provided by a dynamic S-server. Then it holds that

$$q(t) \leq \sup_{0 \leq s \leq t} \{A(s,t) - S(s,t)\}$$

2.2 Moment-Generating Function

Before we start with our traffic classes we need to define a so-called moment-generating function (MGF). The moment-generating function for example helps us computing performance bounds. The distribution of a random variable X can be uniquely determined by its MGF and vice versa a distribution defines a specific MGF [Cur42].
In order to explain the function let us start with what a moment initially is. The first moment of a random variable X is the expected value of $X : \mathbb{E}[X]$.
The Function can calculate the expected value of X^n quickly, because it is equal to the nth moment of a variable, hence the name moment-generating function. For every moment one has to successively differentiate the function [JS20]: $\mathbb{E}[X^n] = \frac{\partial^n}{\partial \theta^n} \phi_X(\theta)$.
More formally: The moment-generating function (MGF) $\phi_x(\theta)$ of the random variable X is defined for $\theta \geq 0$ by

$$\phi_X(\theta) := \mathbb{E}[e^{\theta X}] = \begin{cases} \sum_{i=1}^{\infty} e^{\theta x_i} P(X = x_i), & \text{if X is discrete,} \\ \int_{-\infty}^{\infty} e^{\theta x} f(x) dx, & \text{if X is continuous.} \end{cases} \tag{2.2}$$

For the discrete case, we define $X \in \{x_1, x_2, ..., x_n\}$.
If we assume the MGF exists for a given X we can calculate:

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

Concluding that for $\theta = 0$, $\phi'_X(0) = \mathbb{E}[X]$. In general the nth moment, for $\theta = 0$, can be calculated by successively differentiating our MGF function n times as mentioned before, to obtain $\mathbb{E}[X^n]$ [JS20].

2.3 Traffic classes

On the internet we can have various properties for our traffic we produce, such as different inter-arrival times. These different inter-arrival times are represented by their fitting distribution. To obtain a discrete-time model for our research, we have a conversion where we aggregate them into segments of time.
In this section we explain and showcase now some distributions we analyzed in this thesis.

2.3.1 Exponential Distribution

Beginning with the exponential distribution, exponential distribution means that the increments in our arrival-process are exponentially distributed. This distribution only has one parameter, which is λ. Since there is only one parameter to manipulate, it is a restricted distribution. The functions values are continuous. This distribution is often used in telecommunication traffic due to its simplicity.
First we have the distribution function itself seen in Figure 2.3:

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

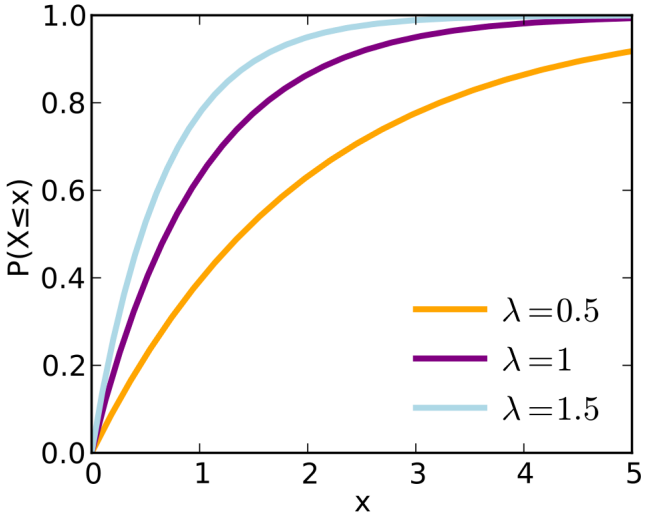


Figure 2.3: The exponential distribution with different arrival rates λ [Skb].

We also have the probability density function as seen in Figure 2.4 which is used to calculate the MGF and defined by :

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

The MGF of the exponential distribution can be written as $\frac{\lambda}{\lambda-\theta}$. By taking the probability density function (PDF) as explained in section 2.2.

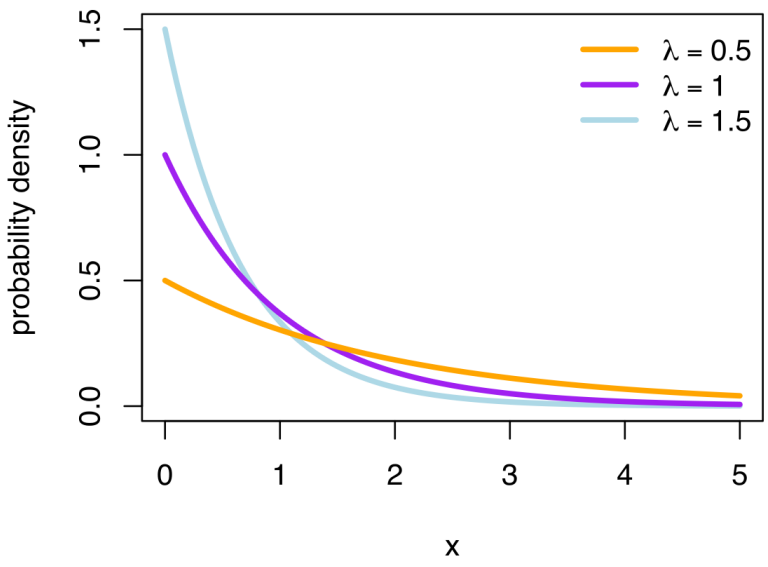


Figure 2.4: PDF of exponential distribution [New].

Proof.

$$
\begin{aligned}
\phi'_X(0) &= \mathbb{E}[X] \\
&= \int_{-\infty}^{\infty} e^{\theta x} f(x) dx \\
&= \int_{-\infty}^{\infty} e^{\theta x} \lambda e^{-\lambda x} dx \\
&= \int_{-\infty}^{\infty} e^{(\theta-\lambda)x} \\
&= \lambda \frac{1}{\theta-\lambda} e^{(\theta-\lambda)x} \Big|_0^{\infty} \\
&\overset{\theta<\lambda}{=} \lambda \left(\lim_{z\to+\infty} \frac{1}{\theta-\lambda} e^{(\theta-\lambda)z} - \frac{1}{\theta-\lambda} e^0 \right) \\
&= \lambda \left(-\frac{1}{\theta-\lambda} \right) \\
&= \frac{\lambda}{\lambda-\theta}
\end{aligned}
$$

If and only if $\theta < \lambda$, then the MGF exists, because our limit will only then lead to 0 instead of infinity [JS20].

2.3.2 Bandwidth Limited Distribution

The bandwidth limited distribution [BHBS14] is a distribution where the increments a_k from a sample arrival A are restricted to the bandwidth limitation $M > 0$. No more then M data units per time slot can arrive. Although we do not make assumptions on distributions as previously we have enough restriction that we can create a statistic ϕ to represent an MGF. Since we do not make assumptions, expect that it is limited, this distribution can be applied to nearly any traffic because every network has a restriction on their bandwidth.
We use the empirical distribution function (EDF) since we do not know the distribution and do not need to know it for the approach of bandwidth limited traffic. Although we use the empirical distribution function [BHBS14], it gives us an estimate about the traffic's cumulative distribution function that generated the points in the sample. The empirical distribution function is constructed as $F_{n_0}(x) := \frac{1}{|n_0|} \sum_{k=n_0}^{-1} 1_{\{a_k \leq x\}}$ and can be expressed as seen in Figure 2.5.

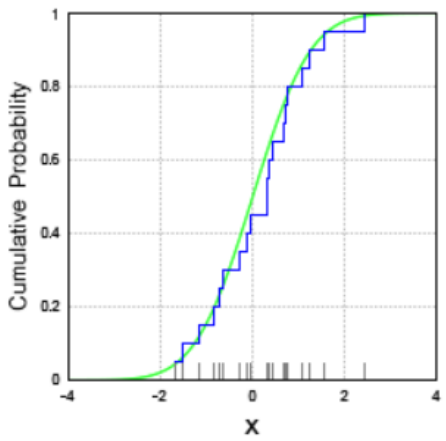


Figure 2.5: Example of the empirical distribution function(blue) compared to the real distribution function(green).

There are various approaches on the traffic for limitation based distributions such as seen in Figure 2.6.
Based on the assumptions for the CDF, the MGF is defined by:

$$\phi(\theta) := \bar{A} + \epsilon(e^{\theta M} - 1)$$

where,

$$\bar{A} = \frac{1}{|n_0|} \sum_{k=n_0}^{-1} e^{\theta a_k}$$

The proof that it works in the StatNC framework can be found in [BHBS14].

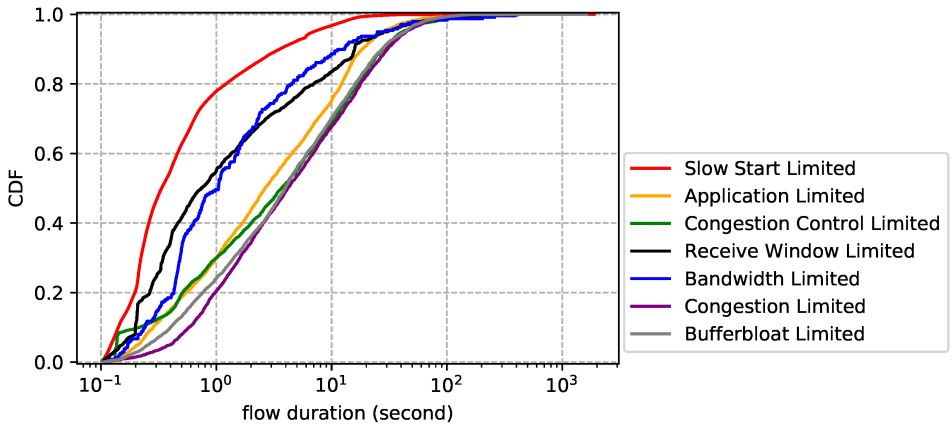


Figure 2.6: Example for various limitations and their assumed CDF's [PGQ18].

2.3.3 Fractional Brownian Motion

Some traffic can also have a self-similar property. The fBm traffic class is a good fit for our traffic with the self-similar property.
The fractional Brownian motion is a generalization of a Brownian motion. It is also a continuous-time Gaussian process.

Definition 2.6. (Fractional Brownian Motion [NHBS18]) A stochastic process $Z(t)$ is called normalized fractional Brownian motion (fBm) with (self-similarity) Hurst parameter $H \in (\frac{1}{2}; 1)$, if it can be characterized by the following properties:

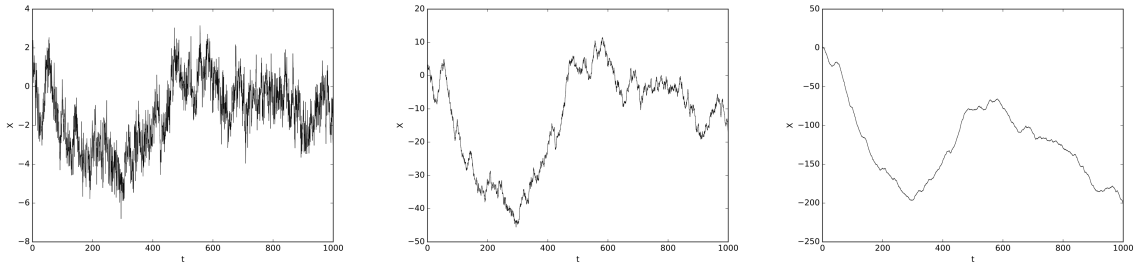
- $Z(t)$ has stationary increments,
- $Z(0) = 0$ and $\mathbb{E}[Z(t)] = 0$ for all t,
- $\mathbb{E}[Z(t)^2] = |t|^{2H}$ for all t,
- $Z(t)$ has continuous paths,
- $Z(t)$ is Gaussian, i.e., all its finite-dimensional marginal distributions are Gaussian.

The increments of this process $Z(t+1) - Z(t)$ are called fractional Gaussian noise (fGn).

Definition 2.7. (Hurst) The Hurst parameter indicates what kind of a process the fBm is [BFH+08].

- If: $H = \frac{1}{2}$ then the process is called an ideal Brownian motion
- If: $\frac{1}{2} < H < 1$ then the increments have a long-range dependence,
- If: $0 < H < \frac{1}{2}$ then the increments have a short-range dependence,

We are not interested for the cases where $H \leq \frac{1}{2}$ as we also did not encounter this case in the experiments of the thesis. An example for a fractional Brownian motion can be seen in Figure 2.7.



Hurst = 0.15 Hurst = 0.55 Hurst = 0.95

Figure 2.7: Discrete sampled points of an fBm process [Shi] with different Hurst parameters.

Notably, the fBm process gets smoother the higher the Hurst parameter.

Lastly there is the fBm arrival model of the fBm process. To construct the model we need to assume the arrivals in the form of:

$$A(t) := \lambda t + \sigma Z(t),$$

where $Z(t)$ is a normalized fractional Brownian motion, λ is our mean arrival rate and σ^2 is the variance of $A(1)$ in our fBm model. The Hurst parameter H controls the MGF $Z(t)$, where as the parameter H is independent from the other parameters [Nor95, NHBS18].
The MGF is then defined as:

$$\mathbb{E}\left[e^{\theta A(t)}\right] = e^{\lambda\theta t + \frac{\theta^2\sigma^2}{2}t^{2H}}.$$

Further proof can be found in [NHBS18].

2.4 Stochastic Network Calculus

Now that we have a basic idea about the components we need for Network Calculus, we can start to combine them and introduce the framework of Stochastic Network Calculus (SNC) [JS20]. To start things, we need to introduce the Chernoff Bound since it allows us to compute bounds on tail probabilities with the help of MGFs.

Theorem 2.8. *(Chernoff Bound [JS20])* Let X be a random variable and $\theta > 0$. Then

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

As of now we can bound the probability of X by using the moment-generating function.

Suppose we now want to estimate our bound for a given backlog in a system consisting of arrivals A and a service S. Let $q(t)$ be our backlog with our Bound B.

$$\begin{aligned}
P(q(t) > B) &\leq P(A \oslash S(t,t) > B) \\
&= P\left(\sup_{0\leq\tau\leq t}\{A(\tau,t) - S(\tau,t)\} > B\right) \\
&\leq \sum_{\tau=0}^{t} P(A(\tau,t) - S(\tau,t) > B) \\
&\leq e^{-\theta B} \sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta(A(\tau,t)-S(\tau,t))}\right] \\
&= e^{-\theta B} \sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right] \mathbb{E}\left[e^{-\theta S(\tau,t)}\right]
\end{aligned}$$

We take our deterministic network calculus bound (first line) and apply the Union Bound (third line). Lastly applying the Chernoff Bound (fourth line), simplifies the calculations for the bound probabilities. We have simplified the calculation of the bound as of now we can incorporate the MGF in this formula [JS20].

In this thesis we rather want to achieve a given percentage of security, our probability of violating the bound, and have a bound calculated to fulfill the requirements. Above it is the opposite, for a given Bound B we obtain a probability. So we solve above formula for B.

$$\begin{aligned}
P(q(t) > B) &\leq e^{-\theta B} \sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right] \mathbb{E}\left[e^{-\theta S(\tau,t)}\right] \\
\frac{P(q(t) > B)}{\sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right] \mathbb{E}\left[e^{-\theta S(\tau,t)}\right]} &\leq e^{-\theta B} \\
\log\left(\frac{P(q(t) > B)}{\sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right] \mathbb{E}\left[e^{-\theta S(\tau,t)}\right]}\right) &\leq -\theta B \\
\frac{\log\left(\frac{P(q(t)>B)}{\sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right]\mathbb{E}\left[e^{-\theta S(\tau,t)}\right]}\right)}{-\theta} &\geq B \\
B &\leq \log\left(\frac{P(q(t) > B)}{\sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right] \mathbb{E}\left[e^{-\theta S(\tau,t)}\right]}\right) * \frac{1}{-\theta}
\end{aligned}$$

Now we can tail our probability for a given bound B. This will be helpful if we have a 99 percent quantile that needs to be fulfilled, as we now do not have to guess until we fulfilled the condition but rather calculate it directly.

2.4.1 Exponential Distribution

For all distributions we calculated the backlog bound as follows in the most basic form:

$$P(q(t) > B) \leq e^{-\theta B} \sum_{\tau=0}^{t} \mathbb{E}\left[e^{\theta A(\tau,t)}\right] \mathbb{E}\left[e^{-\theta S(\tau,t)}\right].$$

Due to the property we discussed in section 2.2 we can insert a bound on the MGF to obtain a bound on the probability. In this case it is the exponential distribution. As a result we get:

$$P(q(t) > B) \leq e^{-\theta B} \sum_{\tau=0}^{t} \phi(\theta)^{t-\tau} \mathbb{E}\left[e^{-\theta S(\tau,t)}\right]$$

$$\implies P(q(t) > B) \leq e^{-\theta B} \sum_{\tau=0}^{t} \left(\frac{\lambda}{\lambda-\theta}\right)^{t-\tau} \mathbb{E}\left[e^{-\theta S(\tau,t)}\right].$$

Since we also assumed a constant rate server for our experiment we can again simplify the formula for the backlog bound into:

$$P(q(t) > B) \leq e^{-\theta B} \sum_{\tau=0}^{t} \left(\frac{\lambda}{\lambda-\theta}\right)^{t-\tau} e^{-\theta * c * (t-\tau)}.$$

As one can see the only aspect changing is the MGF in the formula for every other distribution.

2.4.2 Bandwidth Limited Distribution

For the bandwidth limited distribution we do not have a SNC model, because the idea is that we estimate the distribution with the empirical data only, so it is more an approach for the StatNC model which is why we cannot categorize it here. The only model that comes close to the same approach, based on a SNC model, can be found in [MB98], for further information.

2.4.3 Fractional Brownian Motion

Finally for the fractional Brownian motion process we receive:

$$P(q(t) > B) \leq e^{-\theta B} \sum_{\tau=0}^{t} \left(e^{\lambda\theta t + \frac{\theta^2\sigma^2}{2} t^{2H}}\right)^{t-\tau} e^{-\theta * c * (t-\tau)}.$$

Preparations are complete for the stochastic network calculus based experiment in our thesis, now we move on to a newer, alternative strategy.

2.5 Statistical Network Calculus

An alternative approach to manage the traffic and our arrivals is the statistical based network calculus (StatNC) [BHBS14, NHBS18]. StatNC aims to estimate the parameters in the different traffic classes based on already passed traffic which is a good fit for traffic where we do not have much information about. In contrast to SNC where we assume full knowledge about the traffic, relying solely on the distribution and the traffic is not considered. Often in practical applications, this knowledge is not available so we have to estimate them through measurements. In order to do the calculations properly we have to account for the new uncertainties [NHBS18] that arise from estimating, which where not included in the SNC framework.
In StatNC every estimator has to provide an upper bound on the arrivals MGF [NHBS18], which is again bounded by the variable α. Simply speaking the probability of underestimating the MGF with the estimate is bounded by α. Now let $\mathcal{F}$ be a function from the set of all functions $\{f|f : \mathbb{R}^+ \times \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+\} =: \mathcal{F}$.

We can now introduce as suggested in [NHBS18] a statistic on a sample size t_0 which is treated as our measurement for our estimators in StatNC. Based on this we then can conclude a theorem for the backlog bound.

Theorem 2.9. *[BHBS14, NHBS18]Let $\theta^* = sup\{\theta : \phi_A(\theta, s, t) < \infty\}$ and $\Phi : \mathbb{R}^{|t_0|} \to \mathcal{F}$ be a statistic on $a = (a_{-t_0}, ..., a_{-1})$ such that*

$$\sup_{\theta \in (0,\theta^*)} P\left(\bigcup_{s \leq t} \Phi_a(\theta, s, t) < \phi(\theta, s, t)\right) \leq \alpha.$$

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

$$P(q(t) > B) \leq \alpha + e^{-\theta B} \sum_{s=0}^{t} \Phi_a(\theta, s, t) \mathbb{E}\left[e^{-\theta S(s,t)}\right].$$

As we see in the theorem, the MGF estimated by the StatNC approach is an upper bound of the MGF of the arrivals which is used in SNC. The error is limited by the probability of α, which is also included in the backlog bound calculation. By incorporating α into the formula we also ensure, that if the probability of an estimation error α gets bigger then we also account for it in our backlog bound as sort of a penalty but also as an optimization parameter if we chose it to be smaller.
As proven in [BHBS14] a sub-sample of the main sample a can be sufficient to derive a statistic Φ. In that paper they also showcased a sliding window procedure, since in StatNC, we can continuously get new data to update our estimators in a practical

environment [BHBS14], so we can derive optimized statistics based on the current traffic.

2.5.1 Exponential Distribution

For the exponential distribution we took our statistic from [BHBS14], which is based on the Chi-Squared distribution.

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

Since the MGF of the exponential distribution needs to be smaller then the estimated MGF of StatNC, we conclude:

$$\bar{\lambda} \leq \lambda \implies \frac{\bar{\lambda}}{\bar{\lambda} - \theta} \geq \frac{\lambda}{\lambda - \theta}.$$

Further details can be found in [BHBS14].

2.5.2 Bandwidth Limited Distribution

Secondly in the bandwidth limited distribution, the statistic from [BHBS14] is determined by:

$$P(q(t) > B) \leq e^{-\theta B} \sum_{\tau=0}^{t} \left(\bar{A} + \epsilon(e^{\theta M} - 1)\right)^{t-\tau} e^{-\theta * c * (t-\tau)},$$

where as $\bar{A}$ is defined by $\frac{1}{|n_0|} \sum_{k=n_0}^{-1} e^{\theta a_k}$ as explained in section 2.3.2. This distribution is an idea based solely on Statistical Network Calculus.

2.5.3 Fractional Brownian Motion

Lastly, for the fBm process we have an estimation on the Hurst parameter H based on a sample size n, where one also calculates a confidence interval on the parameter H [NHBS18]. We are only concerned for the upper interval since we are aiming for an upper bound. The estimator $\hat{H}$ is defined by:

$$\hat{H}^{up}_{1-\alpha} := \hat{H} + q_{1-\alpha} \cdot \sqrt{\frac{V_{11}}{n}},$$

where as V is a matrix based on D, $V = 2D^{-1}$. 1/2D is a matrix known as the asymptotic Fisher formation matrix, which guarantees asymptotic efficiency for Gaussian

data [NHBS18]. On V we take the first entry and finally $q_1 - \frac{\alpha}{2}$ is the $(1 - \frac{\alpha}{2})$-quantile of the normal distribution(see [NHBS18] for further details).
For our backlog bound we get:

$$P(q(t) > B) \leq \alpha + e^{-\theta B} \sum_{\tau=0}^{t} \left(e^{\lambda\theta t + \frac{\theta^2\sigma^2}{2} t^{2\hat{H}^{up}_{1-\alpha}}} \right)^{t-\tau} e^{-\theta * c * (t-\tau)}.$$

In [NHBS18] they introduced a formula to calculate the backlog bound based on the previous formula which we incorporated into our evaluation of backlog bounds for comparison purposes. The formula is defined as :

$$P(q(t) > B) \leq \sum_{k=1}^{\lfloor \frac{t}{\tau} \rfloor + 1} e^{-\frac{(B - C\tau + (C-\lambda)k\tau)^2}{2\sigma^2 (k\tau)^{2H}}}, \tag{2.3}$$

where as C denotes the constant server rate. Proof and further explanation can be found in [NHBS18]. The formula is easily transformed to fit the conditions for the StatNC strategy.

$$P(q(t) > B) \leq \alpha + \sum_{k=1}^{\lfloor \frac{t}{\tau} \rfloor + 1} e^{-\frac{(B - C\tau + (C-\lambda)k\tau)^2}{2\sigma^2 (k\tau)^{2\hat{H}^{up}_{1-\alpha}}}}.$$

Now our components are all gathered up, we can now start to introduce and explain our experiment in the following chapter.

3 Experiment

3.1 Experiment

In this section we will show the build-up of the experiment and how we analyzed the given data. We will also show the implementation and explain the ideas behind certain steps we did and what difficulties we encountered during the research. The research was done in Python 3.7 and partially in R, since they are a good fit for our statistical problems with their libraries, especially R.

3.2 Idea

First to clarify the idea of the experiment. Since Stochastic Network Calculus assumes full knowledge about the traffic [BHBS14], it will give good performance bounds if the traffic distribution is estimated correctly. Statistic Network Calculus on the other hand has for the same traffic a slightly worse performance bound because it has the additional parameter α which is added to the calculated backlog bound and because the MFG is estimated. The idea is that StatNC can still give good bounds although we have less knowledge about the traffic, since it is partially based on empirical data, such as the increments. We want to see what results we get for our backlog bounds if we estimate the distribution wrong. On the other hand, we observe the difficulties in a real environment compared to the simulated results. In our environment we get a trace and have to estimate the distribution, based on our backlog bounds and Q-Q plots. In comparison in [BHBS14, NHBS18], the traffic is mostly simulated which means it can be chosen what distribution one wants to simulate, the true parameters are known which is a huge advantage. It is clear that you then get good results with the computed backlog bounds, because the distribution is already known. If we have a capture where the distribution is neither of the three we analyzed or a combination of two, the results can vary. From our findings we try to explain the results or see if we can conclude something new from it. Also parameter estimation and choosing of our time-slots is crucial, we check for dependencies in the real environment. We will observe where problems emerge and how to solve them if we find a solution.

3.3 Captures - Sources

Starting with the source of our traffics and what information could be gathered from the sources about the traffic, such as encoding or data link speed where the traffic was observed. There are three main sources for the traffic. First we had traffic from a University in Italy "Universita' degli Studi di Napoli" in Naples, Italy, which recorded the traffic of the whole university network from 2006 on a 100Mb link [dSdN].
Secondly we had two traffic recordings made by the Wand Group in Waikato, New Zealand from 2011 where they gathered packets for 86 days [Grob]. These captures were cut up into 12 hour segments and some in one hour segments. We analyzed one set of 12 hours and one of one hour duration. The IP addresses were encrypted using Crypto-Pan AES encryption, hence we could not determine different flows, which is why we interpreted it as one big flow of packets from A to B. Again, the capture point of the traffic was the University's network infrastructure and the commodity Internet [Grob].
Lastly, we took three previous captures of the Bell Core group from 1989, since they did further analysis to their captures with the fBm traffic class. That gives us a comparison point to see if we got the same results on certain aspects, such as parameter or bound calculation.
Unfortunately, we did not find any information about the dependencies of the packets and for the most part no IP addresses to distinguish between different traffic flows [Groa, SXJY99].

3.4 Tail Backlog

As we already established, we took 3 different distributions and analyzed them based on our traffic and their performance of the calculated bound. Before we even started to analyze them we first evaluated the quantiles of the traffic compared to pre-defined quantiles. This is done by using Quantile-Quantile (Q-Q) plots, where we can observe the resulting curve. For this we used a Python module **statsmodels**, which has a function called **qqplot**, where the input is your increments of the data, and a given distribution, the function then plots the quantiles as mentioned before.
Now evaluating the plot is simple, since it estimates the parameters to fit the plot as best as it can. We now need to watch our data-points compared to the generated 45° line as seen in Figure 3.1. The points plotted are always in ascending order from left to right [Rap]. If the data-points are following the line then our two distributions are identical. However, this can also vary since one of the distributions can have a heavier tail, then it could slightly deviate from the 45° line but it is still informative to evaluate the distribution. If the general trend of the Q-Q plot is steeper then our line, then our distribution is more dispersed then the one we initially gave as comparison. Vice versa when the general trend is flatter [Rap].

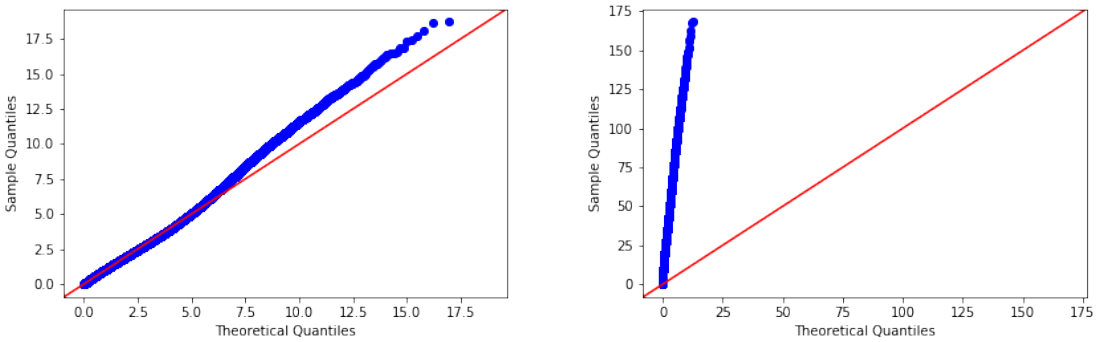


Figure 3.1: Example of a Q-Q Plot where the traffics distribution is identical(Left) or more dispersed (Right) to the compared distribution.

By running Q-Q plots with different distributions, we can therefore assume the distributions beforehand and compare it to our calculated bounds. In theory the result of the Q-Q plot should support the results of the calculated backlog bounds.

3.5 Structure

The numerical experiments are mostly conducted with Python. The goal was to exploit the vast amount of Python libraries for statistics. Since we worked with huge amounts of data, such as 13 million packets in one capture, it was an advantage to aggregate the data once into our custom time-slots and then further on work with it without recalculating the arrivals after every action. The first hurdle was to choose the time slot at which we aggregate our captured packets. Choose the slot too big and you get huge fluctuation in sizes and you do not have enough time slots to perform a proper backlog analysis. If you chose it too small, you get a large number of small arrivals or even idle slots. We mostly chose a time slot of 0.05 with a few exceptions, so that we obtain sufficient amounts of data per time slot, but also enough data-points so we can perform an analysis on them.

A key library we used in our research was **SciPy** since the library has various functions to optimize our calculations, which we required for a better run-time. For our optimization to find the best backlog bound we used the function **optimize()** from **SciPy**, which delivers optimization methods such as **brute**. Other libraries we used such as **Pandas** to read our data in and **NumPy** to aggregate our numbers or do further calculations with our data more efficiently, such as using vectors, which shortened the run-time significantly.

Moving on, we chose the bandwidth ourselves such that the utility would be at 91%, so we calculated the average packet length and multiplied by 1.1 to get our custom bandwidth. This served to achieve a maximum backlog and thus to obtain

information-rich data later on. Since all our bounds depend on θ, we had to optimize based on θ, so we chose to use the optimize method **brute** from **scipy.optimize**. Brute is not as the name suggests just searching per brute force a solution, but more sophisticated it searches along a grid and executes a downhill simplex optimization as a second step. As for example you have to give your range for θ into the method, and a step size for example 0.05. So the method starts with $\theta = 0.05$ and goes on until it is at the boundary of your range. If it finds the best θ for the smallest bound for example at the edge of your range, it will further search for the case that your optimal θ is out of your input range. This is enabled by the in-build **finish** function which can find the minimum with a higher precision exceeding in some cases the range. This feature can also be turned off if the range should not be surpassed.
Down below is an example of the implementation for the exponential distribution, how we calculated performance bounds for the backlog with the MGF, service and the optimize.brute method, whereas lambda_estimate is the statistically estimated parameter $\frac{1}{averagePacketLength}$ and lambda_bar is the statistically estimated parameter according to the formula from 2.5.1 based on the StatNC framework.

```
from math import exp, log, log10
from scipy import optimize
from scipy.stats import chi2

def exp_snc_arrival(theta, s, t):
    return (lambda_estimate / (lambda_estimate - theta))**(t - s)

def exp_statnc_arrival(alpha, theta, s, t):
    quantile = chi2.ppf(alpha, dF)
    cumulated_arrivals = slotdf["arrival_process"].iloc[-1]
    lambda_bar = (quantile / (2*cumulated_arrivals))
    return (lambda_bar / (lambda_bar - theta))**(t - s)

def service(theta, s, t):
    return exp(-theta*server_rate*(t-s))

def exp_snc_bound(theta, probability, t):
    sum_mgf = 0
    for i in range(0,t):
        sum_mgf += exp_snc_arrival(theta, i, t) * service(theta, i, t)
    if(sum_mgf == math.inf):
        return math.inf
    return log(probability/sum_mgf)/(-theta)

def exp_snc_bound_opt(probability, t):
    step = round(log10(lambda_estimate)-1)
    rranges = (slice(10**step, lambda_estimate, 10**step),)
    resbrute = optimize.brute(exp_snc_bound, rranges, args = (
   probability,t), full_output=True, finish=optimize.fmin)
    return resbrute[1]  # function value at global minimum
```

For the fBm model we calculated the Hurst parameter in R but we embedded it into Python using an interface to the R language, **rpy2**. The code for estimating the Hurst

parameter H is from [KG02].

As we implemented the methods to calculate backlog bounds for our distributions, we also had to implement a method to calculate the backlog distribution, so that we can additionally get the empirical 99.8-quantile from our traffic and represent the bounds according to it. Below is the implementation for the backlog distribution. The idea is that we wait for time unit T and then measure the backlog at that point in time. If the backlog is for example 0 at that time, we write it down and wait again for T time units, otherwise if we have backlog > 0 we write the backlog value down and then wait until the backlog is processed. Only then we wait again for time unit T to measure it again.

```
import numpy as np

#Backlog
TIME = 150
x = np.exp(np.arange(0, 9, 0.005))
incidents = [0] * len(x)
backlog_list = []
index = 0
q_t = 0
while(index + TIME < slotdf.shape[0]):
    for i in range(index, index + TIME):
        q_t = max(slotdf["Length"].iloc[i] + q_t - server_rate,0)
    for j in range(0, len(x)):
        if(q_t <= x[j]):
            incidents[j] += 1
    backlog_list.append(q_t)
    index = index + TIME
    while(q_t > 0 and index < slotdf.shape[0]):
        q_t = max(slotdf["Length"].iloc[index] + q_t - server_rate,0)
        index += 1

crf = [x / len(backlog_list) for x in incidents]
```

By this method, we add our backlog values to a list, with which we can create our plot for the backlog distribution. We have our x which is a list of ascending backlog boundaries, now if the backlog is over the value of the current x we increment the number by 1 to our number of incidents on that x. After, you result in a list with a number of incidents that reveals how many times each specific boundary got exceeded. Finally you take every entry from the incidents and divide it by the number of backlog entries to obtain a Cumulative Relative Frequency(CRF).

We also implemented confidence intervals to estimate the fault tolerance for our performance bounds. This was implemented with the help of the python module **scipy.stats.mstats** which contains a large number of statistical functions. Here we used the method **mquantiles_cimj** to calculate our confidence intervals, which needs the data, the probability and an α as input. The confidence interval served as a proof to see if a bound fits or does not fit, no matter if the bound was located above the empirical 99.8-quantile.

Lastly we used the **Matplotlib** library to plot our respective graphs, which is used for creating static, animated and interactive visualizations in Python.

3.6 Pitfalls around real Environment

Here we will discuss problems we encountered throughout the experiment regarding optimization, limits and overall difficulties testing the classes in a real environment.

3.6.1 Optimization

First we will begin talking about the optimization problems. Considering we had to choose a time-slot size on which we aggregated our data and did our calculations, we had to choose the time-slot size precisely. By choosing the time-slot too large, we obtain rarely idle time-slots, but the trade-off was that we had larger numbers to compute into our methods, which resulted often into "**OverflowError: (34, 'Result too large')**". So we had first to consider our optimization method, but ultimately we could not change much in our optimization method since it came from a library. We tried other optimization methods but brute worked best for our research, since for example it is also improving itself as mentioned before in 3.5.
Secondly, we struggled with our methods since they often got optimized to have a large number, for example if we had $\frac{1}{x}$, the goal is to get x as big as possible to obtain a smaller result. After we fixed some problems with the "result too large" error, we had the problem that the result of $\frac{1}{x}$ got so small, that it evaluated it to 0. So we also considered that and after some trial and error we discovered that with a better chosen time-slot the values would not exceed the limit and consequently not evaluate to 0 when they are in a fraction as a denominator.
We also tried libraries such ass **decimal** to handle too large numbers, but then the optimize.brute method from SciPy could not handle the new type of numbers and returned errors, which we could not fix so we abandoned the idea eventually.

3.6.2 Limits

Secondly we had our limits in computing power again. We had our time unit T, where we measured the backlog every time after T passed. The larger our T is the more we can say about the future behaviour in the traffic and ultimately for our backlog. So a bigger T is always favorable. The problem that arises is, if we chose our T larger then we can say more about the traffic in the future, but the computation can result in math errors, such as "result too large". This is due to the property of our methods and the theory. We take the MGF and take it to the power to $t - s$. Here, s is our starting time and t our end time where traffic behaviour should be considered.

Since we only had our time unit T as $t - s$, we chose s always as 0 so we would have one less parameter to consider. This calculation explodes the larger our T is and the results are getting faster bigger then the previous result with every increment in T. If you choose T too high you also get a lot less backlog values in your list, which can conclude a wrong estimation of the backlog distribution.
Choosing the highest T possible was our goal.

Also, since we choose the utility, it was crucial to choose not a too high utility such as 99%. The result would be that we constantly have backlog but with our approach we would skip more than half of the backlog since we wait after we write a backlog value down, until the backlog is fully processed. Unfortunately, the lower the utility the less backlog we have and the chance is higher with a higher T to skip all the backlog values in our analysis. The amount of noted values is again dependent on the time unit T. So there was some kind of a trade-off to choose a good utility, in our case 91%, but not too high so that we would skip all the backlog values at once.

We also did not know if the packets were independent and identically distributed (i.i.d.), which the distribution theorems were based on. We had to assume it and see what results it would yield. Based on the results for our backlog bounds, we could either conclude if the distribution assumed was right or if none of the distributions were fitting because the increments were not i.i.d. or the distribution of our increments was none of the three we analyzed if the results were terrible.

3.6.3 Implementation

The idea was to base the whole implementation on Python. This proved to be more complex because some methods to calculate parameters stochastically were missing in Python, despite the vast amount of libraries. We found a library which returns Hurst parameter H for your increments as input, but the values we got were not the same as calculated with R code where we knew the method used was correct. Looking into the R code we also saw some methods such as "perFit" which we did not know how to implement because they were very complex but also not available in any library of Python. We were forced to calculate the Hurst parameters and also the estimators Hurst-Up in R. To achieve this, we used an interface for R, which basically is R embedded in Python. The interface is called **rpy2**. The code was provided, as mentioned before, by [KG02].

We listed all the problems we encountered in our research and showcased our solutions for them respectively. Now we can move on to the results of the research and what we can conclude from it.

4 Analysis

4.1 Results

We will begin with our Q-Q plots and the results we obtained which will be later compared to our performance bound results, to see if the best bound for a specific distribution also is the distribution suggested by the Q-Q plot. Later on we will showcase the backlog distribution we got for the different captures and see how much of our results even had backlog values > 0. Lastly we evaluate our backlog bounds and see if we can conclude which has maybe a higher robustness or is worse in total.

4.1.1 Q-Q Plots

Starting with the trace from the Italian University, we initially put the arrivals into our method which computed the corresponding Q-Q plots, to see what we can expect from the traffic and their distribution. We were limited in the distributions we could test since there were many that we did not implement. In order to achieve a definite result we made the Q-Q plots only based on the exponential distribution. The bandwidth limited distribution tries to estimate the distribution based on empirical data so there was logically no distribution to compare the data with. As seen in Figure 4.1, the increments are not exponentially distributed, so we can assume that the backlog bounds based on the exp distribution will be worse then the results of the others.

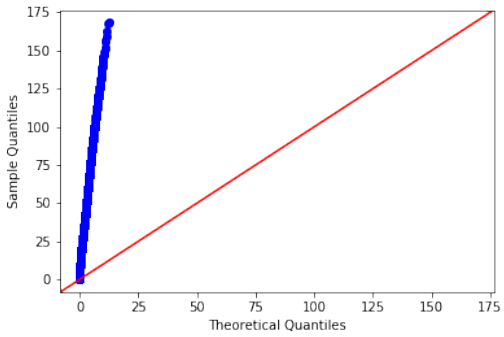


Figure 4.1: Q-Q Plot of the data compared to the exponential distribution.

This same result of the Q-Q plot is also seen in the second trace from the Wand group. On the other hand, the first trace provided a rather good result based on the Q-Q plot, for the exponential distribution, as seen in Figure 4.2.

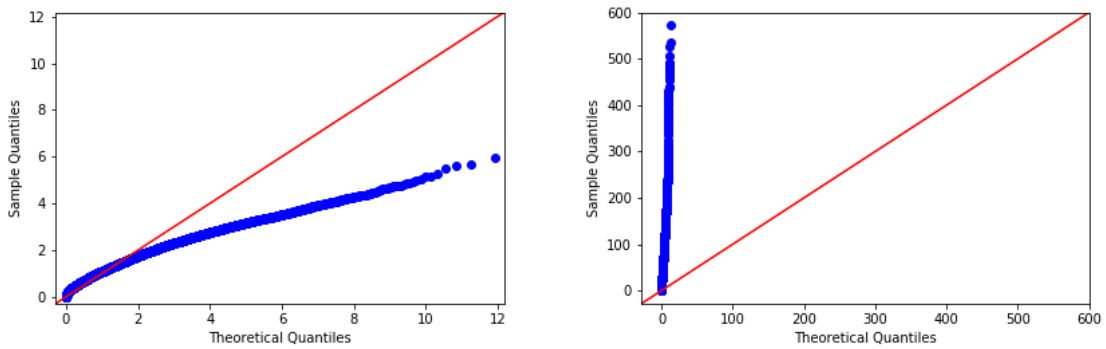


Figure 4.2: The first plot indicates more likely identical distributions, compared to the second plot which indicates different distributions.

Moving on to the three Bell Core traces which yielded negative results as seen in Figure 4.3 for the exponential distribution. We can assume bad performance bounds for all three captures, based on the exponential distribution.

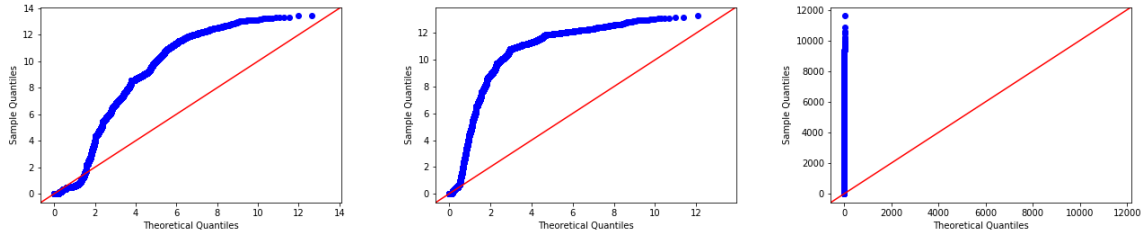


Figure 4.3: All three plots indicate the same answer, the distribution of the traffic is different from the exponential distribution. The first two have a major hump, which is unusual for identical distributions.

Now that we retrieved this information, we can continue with our analysis and see if our assumptions were right.

4.1.2 Backlog Distribution

For the next part, we analyzed our results for the backlog distribution. Although we tried to get the optimal trade-off between the parameters, as mentioned in section 3.6, we still had on average around 60% idle backlog values on average. As seen in Figure 4.4, we always start around 50-70% with our backlog distribution curve.

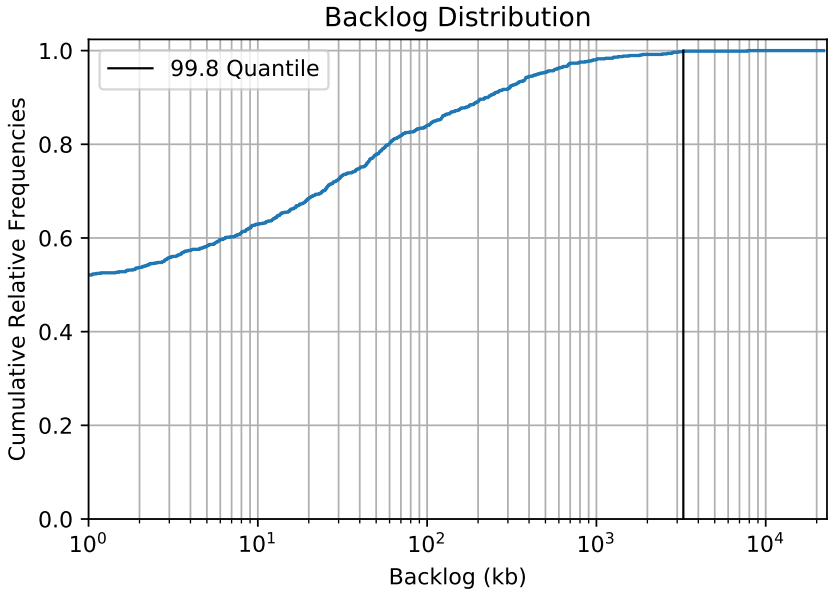


Figure 4.4: One Backlog distribution from our results.

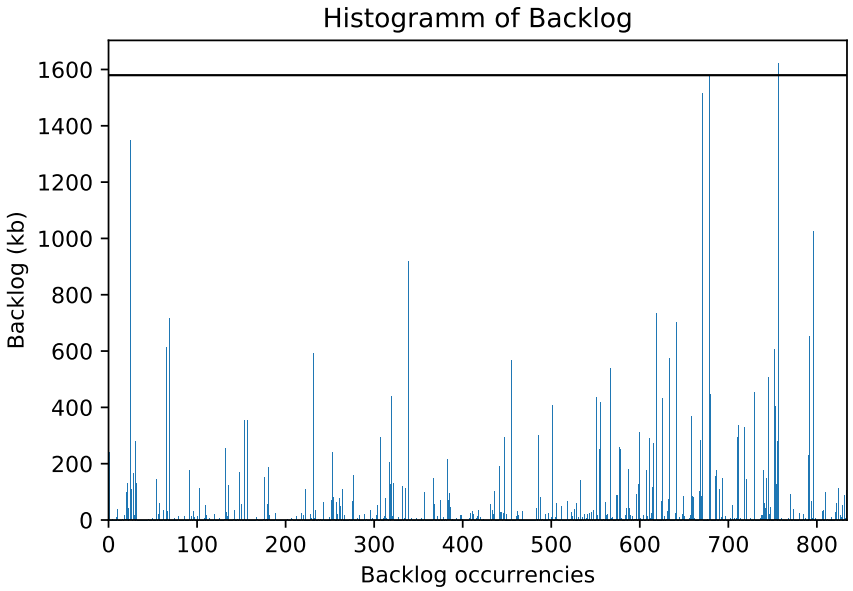


Figure 4.5: Corresponding Histogram from the previous Backlog distribution.

The reason for this is that we often have 0 values, as seen in the corresponding histogram showcased in Figure 4.5.

On the other side, considering our Backlog distribution starts at 60% based on the Cumulative Relative Frequency (CRF), we still obtained enough values to compute a clean distribution and to retrieve a consistent 99.8% quantile from the empirical

arrivals.

4.1.3 Backlog Bounds

We tried to incorporate the other formula from [NHBS18] based on the initial formula to calculate the backlog bounds, which turned out to be more complex than initially thought. The problem is that we need to calculate the optimal θ for every value k in our sum, as seen in the formula in section 2.5.3. We simplified the process and calculated just one optimal θ for all k's. The performance bound is therefore not as tight but still correct.

For all our computations we chose $\alpha = 0.001$ and a time horizon $T = 150$ for a quantile of 0.998, with a utilization of 91%.
Moving on, let us start again with the capture from the Italian University. Here we have only achieved a good result from the fBm model. On the plot below in Figure 4.6, one can see that only the fBm model for StatNC yields a good result. The bound is slightly above from the upper confidence interval of the 99.8 empirical quantile. For SNC the performance bound is identical to the empirical bound, but it is inside the confidence interval which means that the result is still not good enough. As suggested from the Q-Q plot, the performance bound for the exponential distribution is also having bad results and finally the limited bandwidth performance bound is the worst. In theory if the distribution is estimated wrongly, then the limited bandwidth distribution should still have better results because it is estimating the bound based on empirical data, which should be better then assuming the wrong distribution. The fact that the bounds of the exponential distribution are above the bound from the bandwidth limited indicates that the distribution of the traffic has maybe a slightly similar behavior as the exponential distribution. All in all, we can consider all performance bounds to be bad for this trace. The one slightly better result from the fBm bound based on StatNC, can also be considered as a coincidence, which indicates that maybe the traffic was not i.i.d. or it was a combination of distributions we did not analyze.

Next, we have the two traces from the Wand group, which gave a good insight for the Q-Q plot results. The first trace suggested an exponential distribution and as expected the performance bounds for the exponential distribution are optimal in Figure 4.7, placed over the confidence interval of the 99.8-quantile. Now contrary, the fBm model does not yield good results and the bandwidth limited performance bound is nearly exactly the same as the SNC fBm bound.

On the other side, looking at the second capture in Figure 4.8 we can see as suggested by the Q-Q plots that again the exponential approach is giving bad results, but the fBm model is giving optimal results additionally considering the confidence interval. The fact that the limited bandwidth bound is the worst of them, either comes from the fact that maybe the capture is not i.i.d. or the traffic's distribution is again slightly

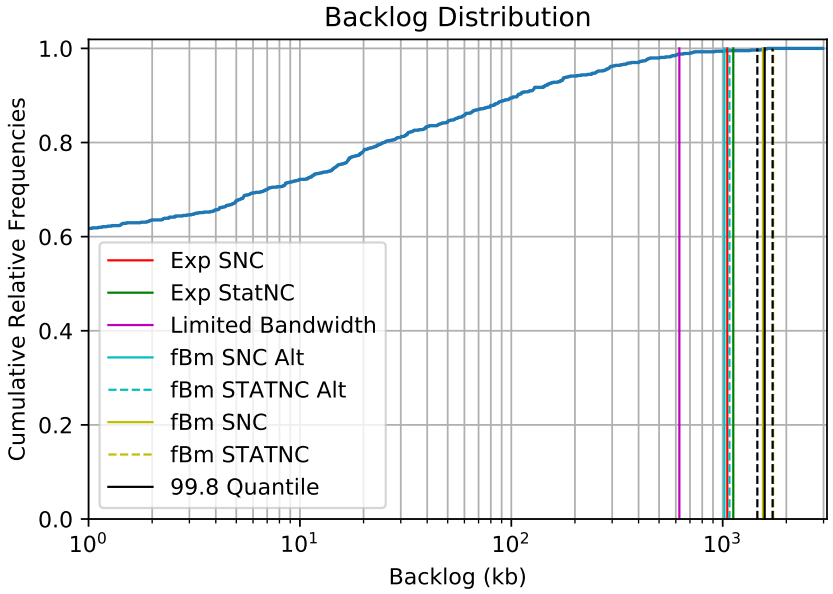


Figure 4.6: Backlog Bounds

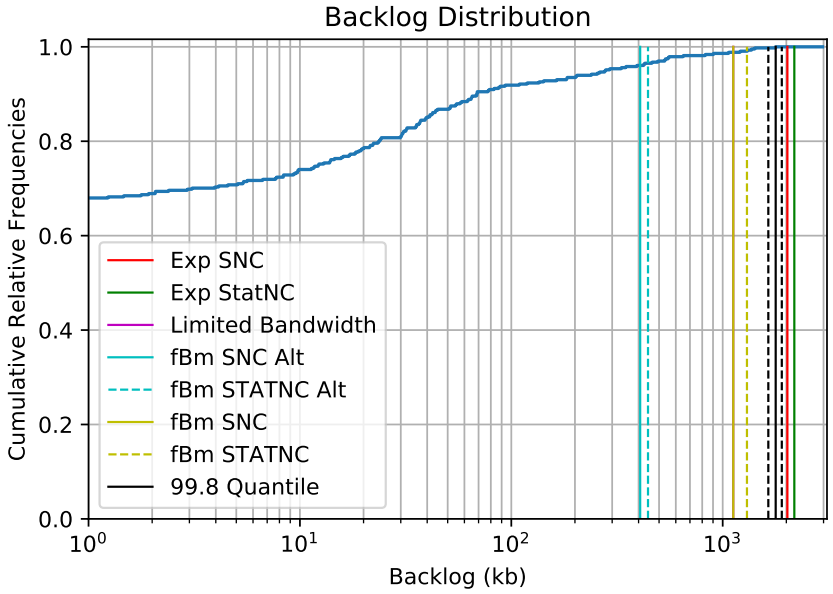


Figure 4.7: First capture which suggested good results for exponential bounds.

similar to the exponential distribution.
Lastly we arrive to the three captures from the Bell Core group. We can observe for all three backlog distributions that the fBm model is very robust and is holding good backlog bounds over the confidence interval of the 99.8-quantile. As one can see in Figures 4.9, 4.10 and 4.11, our theory that the limited bandwidth bound should be in

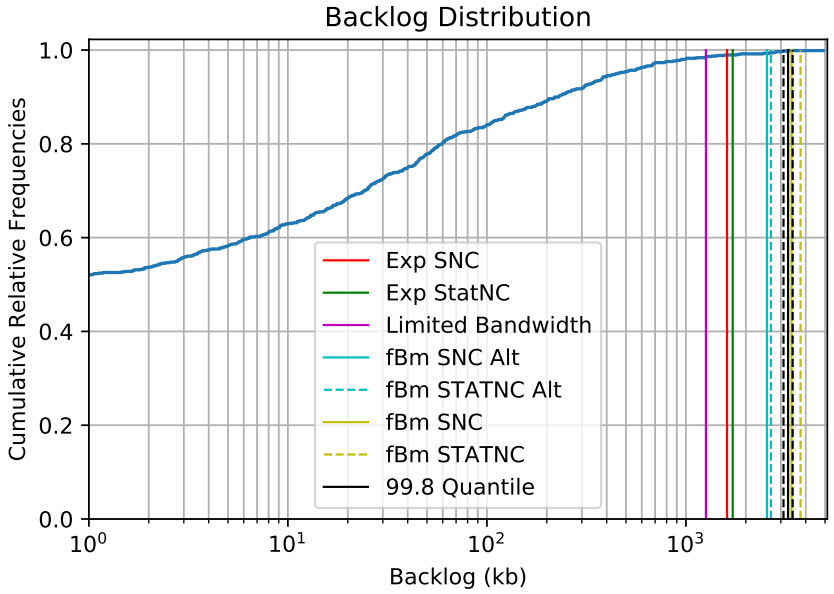


Figure 4.8: Second capture where we expected worse results for the exponential bounds.

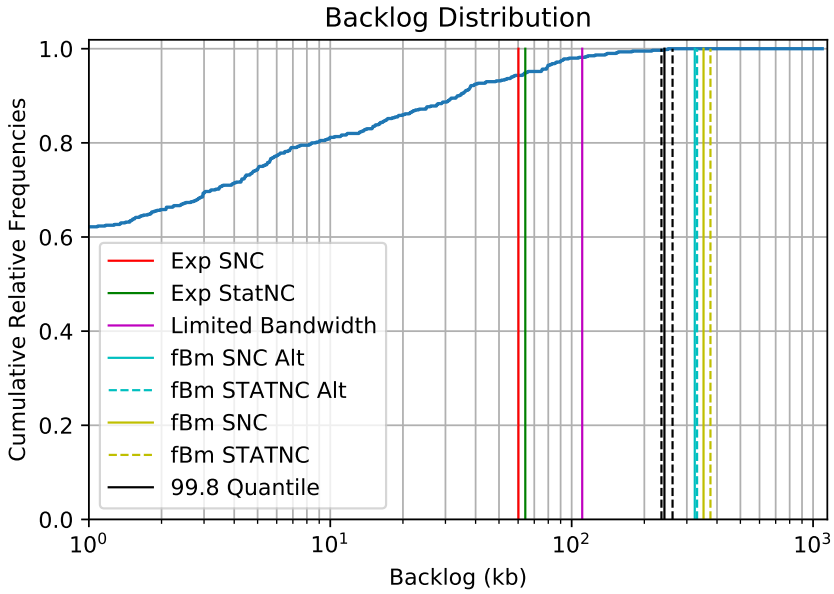


Figure 4.9: First capture from August 1989.

between, if one bound is strongly fitting, is mostly holding except for one distribution where it is nearly overlapping with the exponential distribution based bound performance. The idea behind the theory is, that if one distribution already fits, then the others should be not fitting logically. The limited bandwidth approach is based on

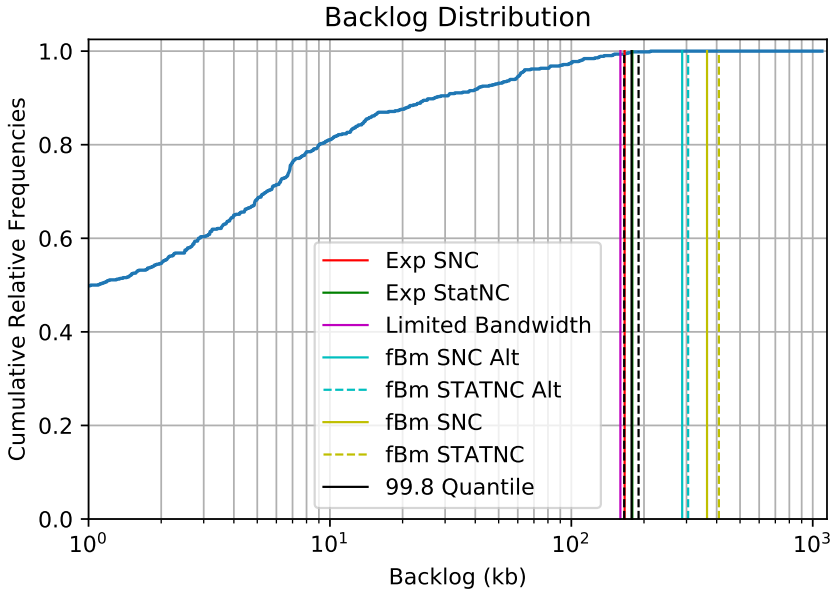


Figure 4.10: Second capture from October 1989.

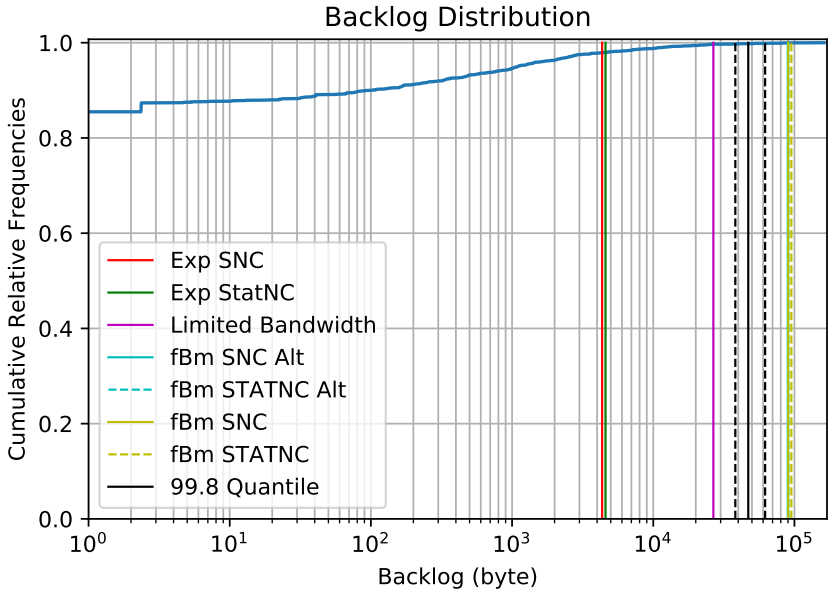


Figure 4.11: Third capture from October 1989, an extension.

empirical data so it should still fit better in theory than a wrong guessed distribution. In this chapter we showcased all of our results which can be positively interpreted since we could substantiate the fact that the limited bandwidth based performance bound is holding badly but on average better then other performance bounds where the distribution was mismatched for example, but we could also display good results

for the fBm model. We will further talk about the evaluation in the next chapter.

4.2 Evaluation

Let us begin with the Q-Q plots, these happened to be of great value, since they predicted the outcome of the exponential distribution performance bounds always right. Unfortunately, we could not generate Q-Q plots based on the fBm model but we were able to draw conclusions. Let us say the Q-Q plot suggests a exponential distribution, we can already see that the fBm performance bound will likely not fit and vice versa when the distribution is not exponential it is still possible that the fBm model could gather good results. Secondly, we also concluded that the limited bandwidth performance bound performed badly, the results were always lower than the 99.8-quantile, which is not desirable. On the other hand, we also concluded that the limited bandwidth based backlog bound is in the worst-case better on average than a performance bound with a false assumed distribution. This approach does not estimate the distribution fairly well but the method is consistent with the results. For the fBm model, it turned out that the fBm model is very robust and in most cases holds very well as seen in the Figures above. Below in the table 4.1, we compared our Hurst parameter H results with the ones from the Bell Core group. The parameters are matching except for the "Oct Ext" capture, which can be explained due to the limited amount of arrivals in this capture, in contrast to the other two captures which had one million arrivals. The chance of fluctuations is higher because of that fact.

Hurst	pAug	pOct	Oct EXT
Results	0.7988529	0.8006442	0.9143225
Bell Core	0.80	0.80	0.88

Table 4.1: Comparison between our calculated Hurst parameters and the results from BellCore [SXJY99].

Furthermore, the other fBm formula from 2.3 for the performance bound stayed close to the initial formula's bound, which can be interpreted as a proof of concept that the formula is correct. Although the bound of the initial fBm formula was always higher compared to the other bound with the discretization. Considering the theory it should have been the opposite. Other than that, it showed consistency since for the most part we got results where either both fBm bounds where above the 99.8-quantile, meaning they are a good fit, or both did not match.

All in all, the results are promising and matching mostly the theory behind it.

5 Conclusion

The are a few points for conclusion we can draw from this research. First, we want to mention that the Q-Q plots are a good preliminary to infer the distribution of a given traffic. There is still room for libraries that could support more distributions, such as a library that can generate arrivals based on a fBm model which can then be compared to other arrivals to make a Q-Q plot based on fBm.
Moving on, the limited bandwidth performance bound did not yield good results performance wise, but as already mentioned, the worst-case is better on average than the worst-case of other performance bounds. This suggests that the distribution gets estimated poorly but at least consistently, which can also give more information about the traffic in question. Every performance bound above of the limited performance bound could still have a slight relation to the traffic, which can help to narrow down possible distributions.
Lastly, this experiment was a good proof of concept for the few formulas of the StatNC framework. We could see if the models which got tested in simulations also work in real environments, considering the difficulties testing in such an environment.
We also concluded that the fBm model is very robust since it yielded always promising results such as tight backlog bounds, except for the one distribution where the exponential distribution was suggested, which makes sense. One thing worth to mention would be, that in our results, for the case when the SNC bound was in the confidence interval of the 99.8-quantile, the StatNC bound was always above with a better result. This could be interpreted as a good result for StatNC but also as coincidence, which is why we did not go into further detail on that aspect.
In conclusion, it can be said that it seems today's traffic has often the property of self-similarity, which can be perfectly expressed and bounded with the fBm model, at least in our experiment and that the StatNC framework is giving promising results which is why it should be further researched.

Bibliography

[BFH+08] Biagini, Francesca, Yaozhong Hu, Øksendal, Bernt, and Zhang Tusheng. *Stochastic Calculus for Fractional Brownian Motion and Applications*. Springer Science & Business Media, Berlin Heidelberg, 2008.

[BHBS14] M. A. Beck, S. A. Henningsen, S. B. Birnbach, and J. B. Schmitt. Towards a statistical network calculus — dealing with uncertainty in arrivals. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 2382–2390, April 2014.

[Cur42] J. H. Curtiss. A note on the theory of moment generating functions. *The Annals of Mathematical Statistics*, 13(4):430–433, 1942.

[dSdN] Universita' degli Studi di Napoli". Captures. `http://traffic.comics.unina.it/Traces/ttraces.php`.

[Fid10] M. Fidler. Survey of deterministic and stochastic service curve models in the network calculus. *IEEE Communications Surveys Tutorials*, 12(1):59–86, 2010.

[Groa] BellCore Group. Traces Source. `ftp://gaia.cs.umass.edu/pub/zhzhang/Traces-More/html/BC.html`.

[Grob] Wand Group. Waikato VIII Captures. `https://wand.net.nz/wits/waikato/8/`.

[JS20] Paul Nikolaus Jens Schmitt. Lecture notes in stochastic analysis of distributed systems, 2020.

[KG02] H. Kettani and J. A. Gubner. A novel approach to the estimation of the hurst parameter in self-similar traffic. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, LCN '02, pages 160–165, USA, 2002. IEEE Computer Society.

[MB98] L. Massoulié and A. Busson. Stochastic majorization of aggregates of leaky bucket-constrained traffic streams. 1998.

[New] Newystats. Own work. `https://commons.wikimedia.org/wiki/File:Exponential_probability_density.svg`.

[NHBS18] P. Nikolaus, S. Henningsen, M. Beck, and J. Schmitt. Integrating fractional brownian motion arrivals into the statistical network calculus. In *2018 30th International Teletraffic Congress (ITC 30)*, volume 02, pages 37–42, 2018.

[Nor95] I. Norros. On the use of fractional brownian motion in the theory of connectionless networks. *IEEE Journal on Selected Areas in Communications*, 13(6):953–962, 1995.

[PGQ18] Chen Y.-C. Peng G.-Q., Xue G. Network measurement and performance analysis at server side. *Future Internet 2018*, 2018.

[Rap] Vallat Raphael. Notes on qq plot. `https://pingouin-stats.org/generated/pingouin.qqplot.html`.

[Shi] Ji Shiyu. Own work. `https://commons.wikimedia.org/w/index.php?curid=53376119`.

[Skb] Skbkekas. Own work. `https://commons.wikimedia.org/wiki/File:Exponential_cdf.svg`.

[SXJY99] Yantai Shu, Fei Xue, Zhigang Jin, and Oliver Yang. The impact of self-similar traffic on network delay. *J. Comput. Sci. Technol.*, 14:585–589, 11 1999.