Bachelor Thesis

Generalized Process Sharing in the Stochastic Network Calculus: Heuristics, Performance Bounds, and Runtime Measurements

by

Lukas Wildberger

December 14, 2021

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

In this thesis we focus on the calculation of performance bounds in the Stochastic Network Calculus of Generalized Processor Sharing scheduled servers, through which data streams (flows) pass. For this we consider the approach of Chang from [Cha00]. This approach will allow us to decide which flows will be GPS scheduled, and which ones will not be scheduled and will be processed directly. This enables us to calculate more accurate bounds in a stochastic system (with a certain violation probability). However, the optimal and efficient choice of which of these flows are GPS scheduled and which are not is still an open problem. For this problem, we present heuristics and evaluate them with respect to the accuracy of the calculated performance bounds and the runtimes. We will see that for this choice it is essential to check the stability condition of the respective flows. Thus, we select the flows that violate the stability condition to be GPS scheduled and can calculate fairly accurate bounds in linear runtime, which is a significant improvement in runtime compared to exhaustive search.

Acknowledgement

First of all, I would like to thank Prof. Dr.-Ing. Jens B. Schmitt for giving me the opportunity to write this bachelor thesis under his supervision. I would also like to thank M.Sc. Paul Nikolaus, who always spent a lot of time helping me with questions and problems regarding the thesis.

Lukas Wildberger

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 December 14, 2021

[illegible]

Lukas Wildberger

Contents

List of Figures

1 Introduction

Anytime a system has several different participants that converge at one point and require one service at one time, we use the concept of scheduling. Writings about scheduling (in the military context) have already been written more than 2500 years ago [HW06].

This concept is still highly relevant today. Nowadays computer or network systems use scheduling algorithms at various parts of the system. The efficiency of such algorithms is crucial for the performance of the entire system for some applications. So it is only logical that these are extensively studied and optimized depending on the application in order to find the most ideal algorithm [HW06].

An ideal scheduling algorithm, called "Generalized Processor Sharing" (GPS), will be considered in this thesis (only theoretically applicable in this form). To be more precise, we want to compute performance bounds within GPS scheduled stochastic systems using the Stochastic Network Calculus (SNC). However, these bounds are quite loose for basic GPS according to [PG93]. We consider an improved approach of Chang [Cha00] to compute such performance bounds. This scheduling algorithm is considered in the scope of this thesis for a server system with multiple flows passing through it.

This improved approach gives us the option to select which flows are included in the GPS scheduling and which are not when calculating performance bounds. I.e., we have the freedom to process flows without scheduling. This option allows us, in the context of a stochastic system, to omit worst case assumptions made by the computation of performance bounds in [PG93]. Hence, we will take advantage of being in a stochastic system to disregard certain worst case assumptions in order to compute more accurate performance bounds (with a certain violation probability). The optimal and efficient choice of which flows are GPS scheduled and which ones are not is still an open problem.

In this thesis different heuristics are presented with regard to this problem. These heuristics are examined regarding the accuracy of the calculated performance bounds and the runtimes. For this purpose, we try to converge to the optimal solution of the problem in linear runtime, which can be computed in exponential runtime by exhaustive search.

We start with an introduction of Network Calculus Backgrounds in Chapter 2. Besides an introduction to Stochastic Network Calculus, the scheduling algorithm "Generalized Processor Sharing" is introduced. For GPS we also consider an improved approach by Chang [Cha00]. In Chapter 3 several heuristics are introduced that can calculate quite accurate performance bounds with efficiency. These are examined in more detail in Chapter 4. The heuristics are evaluated with respect to the accuracy of the calculated performance bounds as well as with respect to the efficiency in the run-times, for different scenarios. At the end of the thesis the results of the investigations are summarized and an overview of possible future work is given in Chapter 5.

2 Network Calculus Background

This chapter introduces the basics of the Network Calculus (NC) as well as the basics of the scheduling algorithm "Generalized Processor Sharing" (GPS), which are necessary for the understanding of the thesis and the results. During the 1990s, Network Calculus emerged as a deterministic theory for analyzing the quality of service of packet data networks. In a networked system, traffic arrivals are modeled using upper envelope functions [FR15]. Service curves are used to characterize minimum service guarantees of systems such as a router, a scheduler, or a link. The network calculus provides convolution forms based on these ideas that allow the derivation of worst-case performance bounds including backlog and delay [FR15]. Besides the original consideration of deterministic models in NC where only the worst case is considered and the statistical facts of flows are not taken into account, there is also the so-called Stochastic Network Calculus (SNC), which is also in the focus in this thesis. It is a flexible mathematical framework that is used to calculate probabilistic end-to-end delay bounds [NS20] . For the SNC, a variety of stochastic processes (including long range dependent, self-similar and heavy-tailed traffic) are considered. This general approach leads to less accurate results. Instead, performance bounds of the type $\mathrm{P}(\text{backlog} > x) \leq \epsilon$ or $\mathrm{P}(\text{delay} > x) \leq \epsilon$ (for specified x and ϵ) are computed [FR15].

2.1 Stochastic Network Calculus (SNC)

In this section, Moment Generating Functions are defined first (Subsection 2.1.1). Afterwards, the required basics of arrival and service processes are introduced (Subsection 2.1.2 and Subsection 2.1.3). At the end of this section we introduce for the further thesis important aspects for performance bounds and leftover service (Subsection 2.1.4 and Subsection 2.1.5).

2.1.1 Moment Generating Functions

Since this thesis uses moment-generating functions-based SNC, moment-generating functions (MGF) are defined initially, and examples are considered. A random process can be uniquely determined using the MGF. Thus, arrival processes or service processes can be characterized by an MGF as we will see later [FR15]. In addition, the

MGF-based SNC is used in order to bound the probability that the delay exceeds a specific value $T \geq 0$. For that, Chernoff's bound (Theorem 2.17) is used to determine the MGF-bound on a probability [NS20].

Definition 2.1. (Moment Generating Function [SN20]). The *moment generating function* (MGF) $\phi_X(\theta)$ of the random variable X is defined for $\theta \geq 0$ by

$$\phi_X(\theta) := \mathrm{E}\left[e^{\theta X}\right]$$

Remark 2.2 ([SN20]). If $\mathrm{E}\left[e^{\theta X}\right] = \infty$ the MGF of X does not exist at θ. $\phi_X(\theta)$ is called the moment generating function because all of the moments of X can be obtained by successively differentiating $\phi_X(\theta)$. In general, the nth derivative of $\phi_X(\theta)$ evaluated at $\theta = 0$ equals $\mathrm{E}[X^n]$, that is,

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

Example 2.3. (MGF of the Poisson Distribution with Parameter λ [SN20]).

$$\phi_X(\theta) = e^{\lambda(e^\theta - 1)}$$

Example 2.4. (MGF of the Exponential Distribution with Parameter λ [SN20]). Let $\theta < \lambda$. Then

$$\phi_X(\theta) = \frac{\lambda}{\lambda - \theta}$$

2.1.2 Arrival Processes

In order to be able to calculate performance bounds for a certain network or server, the arrivals must be precisely defined and sufficiently analyzable. The following definitions are available for this purpose.

Definition 2.5. (Arrival Process [SN20]). Let $(a_i)_{i\in\mathbb{N}} \geq 0$ be a sequence of real random variables, the so-called increments. So an arrival process is defined by the stochastic process A with time space $\mathbb{N}$ and state space $\mathbb{R}^+ := [0, \infty)$ as

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

The bivariate version is defined as

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

A data stream, which consists of an arrival process, that follows a given route is called "flow" [SN20].

Definition 2.5 can be easily illustrated with the help of Figure 2.1. Here, $A(T_1 - 1, T_3)$ can be calculated as the sum of the sizes of $T1$, $T2$ and $T3$.

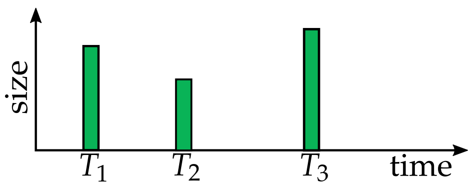


Figure 2.1: Example diagram of an arrival process in discrete time.

Definition 2.6. ((σ_A, ρ_A)-bounded Arrivals [SN20]). An arrival process $A(s,t)$ is (σ_A, ρ_A)-bounded if

$$\phi_{A(s,t)}(\theta) = \mathrm{E}\left[e^{\theta A(s,t)}\right] \leq e^{\theta\rho_A(\theta)\cdot(t-s)+\theta\sigma_A(\theta)}, \quad \text{for all } 0 \leq s \leq t$$

The definition can be interpreted as follows: the ρ_A can be considered as arrivals' rate and the σ_A as arrivals' burst.

Proposition 2.7. (Multiplexing of Independent Arrivals [SN20]). *Let A_1 and A_2 be $(\sigma_{A_1}, \rho_{A_1})$-bounded and $(\sigma_{A_2}, \rho_{A_2})$-bounded arrival processes, respectively. Further, assume that A_1 and A_2 are independent. Then the sum is $(\sigma_{A_1+A_2}, \rho_{A_1+A_2})$-bounded with*

$$\sigma_{A_1+A_2}(\theta) = \sigma_{A_1}(\theta) + \sigma_{A_2}(\theta),$$
$$\rho_{A_1+A_2}(\theta) = \rho_{A_1}(\theta) + \rho_{A_2}(\theta).$$

Proof. See, [SN20, p.49].

In the simulations performed in the context of this thesis, different types of arrival traffic are considered or used. For a better understanding of some of them and to show that they are (σ, ρ)-bounded, the following examples are given.

Example 2.8. (M/M/1-Queues and M/D/1-Queues [SN20]).

This example shows how arrivals and the service are modeled to emulate M/M/1 or M/D/1- queues. For this *Kendall's notation* is used. I.e., the first letter stands for the arrival process, the second for the service time distribution and the third number for the number of servers. The M stands for "Markovian" or "memoryless", and the D is used if it is deterministic. For the arrival process, the M means Poisson distributed arrivals, which is equivalent to an exponentially distributed service time. Regarding the service time distribution, the M means an exponential distribution [SN20].

A Poisson point process, that is a counting process $N(s,t)$ that gives us the number of arrivals between s and t, is needed to model both. Assume that

$$\mathrm{P}(N(a,b)=k)=\frac{(\lambda(b-a))^k}{k!}e^{-\lambda(b-a)}$$

with intensity parameter λ [SN20]. Then the arrivals are given by

$$A(s,t)=\sum_{i=1}^{N(s,t)} a_i,$$

with packet size a_i, that is chosen based on the desired service process. This means that the service behavior is modeled into the arrival process too [SN20].

According to [Ciu07] the following (σ,ρ)-bounds for M/M/1 and M/D/1 respectively can be derived.

<u>M/M/1:</u> For this a memoryless arrival process is assumed, that means that a_i are exponentially distributed with parameter μ. Then A is (σ_A,ρ_A)-bounded for $\theta \in (0,\mu)$, with

$$\begin{aligned}\sigma_A(\theta)&=0,\\ \rho_A(\theta)&=\frac{\lambda}{\mu-\theta}.\end{aligned}$$

<u>M/D/1:</u> Under the assumption of a constant packet size $\frac{1}{\mu}$, A is (σ_A,ρ_A)-bounded with

$$\begin{aligned}\sigma_A(\theta)&=0,\\ \rho_A(\theta)&=\frac{1}{\theta}\lambda\left(e^{\frac{\theta}{\mu}}-1\right).\end{aligned}$$

Arrivals that occupy these parameters will be called "M/M/1 traffic" respectively "M/D/1 traffic" in the following, as it would actually be an M/M/1 (respectively M/D/1)-queue when considering one flow.

Example 2.9. (Markov Modulated On-Off (MMOO) Traffic [SN20, Cha00]).

Assume the following discrete-time Markov chain (DTMC)

$$\begin{tikzcd}
0 \arrow[loop left, "p_{00}"] \arrow[r, bend left, "p_{01}"] & 1 \arrow[loop right, "p_{11}"] \arrow[l, bend left, "p_{10}"]
\end{tikzcd}$$

.

The DTMC has two states, 0 (off) and 1 (on) with transition probability matrix

$$\mathrm{P} = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix}.$$

If the chain is in state "on", traffic with peak rate $b \geq 0$ is sent, whereas in state "off", no data is sent.

Now Assume such an Markov-modulated arrival traffic A with finite signal space S. In other words, the distribution of the increments of A is determined by the current state of an underlying Markov chain Y. The Markov chain is described by S and transition matrix $P = [p_{ij}]$ such that $t_{ij} > 0$ for all $i, j \in S$. The matrix with entries $E_i := E_{ii} := \mathrm{E}\left[e^{\theta a(t)} | Y_t = i\right]$ for all states $i \in S$ is denoted by $E \in \mathrm{Diag}(S)$ [BHBS14].

For this it holds that

$$\mathrm{E}\left[e^{\theta A(s,t)}\right] \leq e^{\theta \rho_A(\theta) \cdot (t-s) + \theta \sigma_A(\theta)},$$

with

$$\begin{aligned}
\sigma(\theta) &= \frac{1}{\theta} \left(\max_{s \in S} E_s \cdot \frac{\max_{s \in S} x_s}{\min_{s \in S} x_s} \cdot \frac{1}{\mathrm{sp}(E \cdot P)} \right), \\
\rho(\theta) &= \frac{1}{\theta} \log(\mathrm{sp}(E \cdot P)),
\end{aligned}$$

where $\mathrm{sp} : Mat(S) \to \mathbb{R}$ is the spectral radius of some matrix over S and the vector x is a positive eigenvector of $E \cdot P$ [BHBS14].

However, this example can be extended by aggregating the MMOO traffic. For this, we aggregate n MMOO arrival processes, with $n \in \mathbb{N}$ arbitrary but fixed and with the following DTMC

$$\begin{tikzcd}
0 \arrow[loop left, "p_{0,0}"] \arrow[r, bend left, "p_{0,1}"] & 1 \arrow[loop above, "p_{1,1}"] \arrow[l, bend left, "p_{1,0}"] \arrow[r, bend left, "p_{1,2}"] & 2 \arrow[loop above, "p_{2,2}"] \arrow[l, bend left, "p_{2,1}"] \arrow[r, bend left, dashed] & n-1 \arrow[loop above, "p_{n-1,n-1}"] \arrow[l, bend left, dashed] \arrow[r, bend left, "p_{n-1,n}"] & n \arrow[loop right, "p_{n,n}"] \arrow[l, bend left, "p_{n,n-1}"]
\end{tikzcd}$$

.

Out of this results the transition matrix

$$P = \begin{pmatrix} p_{00} & p_{01} & \cdots \\ p_{10} & \ddots & \\ \vdots & & p_{nn} \end{pmatrix}.$$

To show that aggregated MMOO traffic is also (σ, ρ)-bounded, the result of the bounds for single flow traffic and Proposition 2.7 can be used. For this the same conditions apply as above, except that A is the aggregate MMOO traffic and P is the corresponding transition matrix.

For this it holds that

$$\mathrm{E}\left[e^{\theta A(s,t)}\right] \leq e^{\theta \rho_A(\theta) \cdot (t-s) + \theta \sigma_A(\theta)},$$

with

$$\begin{aligned}
\sigma(\theta) &\overset{(2.7)}{=} \underbrace{\frac{1}{\theta}\left(\max_{s \in S} E_s \cdot \frac{\max_{s\in S} x_s}{\min_{s \in S} x_s} \cdot \frac{1}{\mathrm{sp}(E \cdot P)}\right) + \cdots + \frac{1}{\theta}\left(\max_{s \in S} E_s \cdot \frac{\max_{s\in S} x_s}{\min_{s \in S} x_s} \cdot \frac{1}{\mathrm{sp}(E \cdot P)}\right)}_{n \text{ times}} \\
&= \frac{n}{\theta}\left(\max_{s \in S} E_s \cdot \frac{\max_{s\in S} x_s}{\min_{s \in S} x_s} \cdot \frac{1}{\mathrm{sp}(E \cdot P)}\right), \\
\rho(\theta) &\overset{(2.7)}{=} \underbrace{\frac{1}{\theta} \log(\mathrm{sp}(E \cdot P)) + \cdots + \frac{1}{\theta} \log(\mathrm{sp}(E \cdot P))}_{n \text{ times}} \\
&= \frac{n}{\theta} \log(\mathrm{sp}(E \cdot P)),
\end{aligned}$$

where $\mathrm{sp} : Mat(S) \rightarrow \mathbb{R}$ is the spectral radius of some matrix over S and the vector x is a positive eigenvector of $E \cdot P$.

2.1.3 Service Processes

Before defining the service process, the general concept of the interaction of arrival and service process will be illustrated briefly using Figure 2.2.

Let there be a server. This server receives requests, processes them and forwards the result. These requests, which the server receives at a time t, are called arrivals $A(t)$ as introduced before. However, the server that wants to process these arrivals has only a certain capacity to do so. Thus, it cannot process an arbitrary number of requests at once. This rate with which the server can process arrivals and send them out at a time t as departures $D(t)$ is what we will look at in this subsection. If we now assume that $A(t)$ is greater than the rate at which the server can process them at this time, so when so-called "bursts" arrive at the server, then the arrivals must first be buffered. For this purpose they are queued until they can be scheduled and processed by the server.

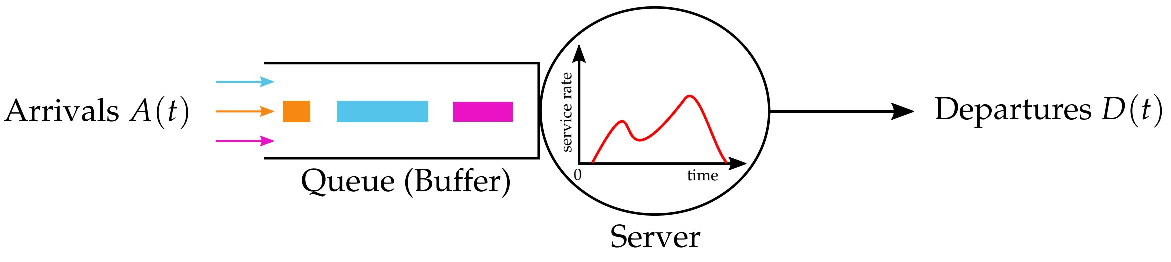


Figure 2.2: Arrivals, queue, server and departures.

In network calculus the "min-plus algebra" is used [BCOQ92]. It allows the concise notation of complicated expressions. More significantly, the operators have a direct relation to network operations (e.g., Definition 2.12) [SN20]. To make use of this, it is defined in the following.

Definition 2.10. (Bivariate (De-)Convolution in Min-Plus Algebra [SN20]). Let $x(s,t)$ and $y(s,t)$ be two real-valued bivariate functions. Then we define the *min-plus convolution* for $0 \leq s \leq t$ as

$$x \otimes y(s,t) := \inf_{s \leq \tau \leq t} \{x(s,\tau) + y(\tau,t)\}$$

and the *min-plus deconvolution* as

$$x \oslash y(s,t) := \sup_{0 \leq \tau \leq s} \{x(\tau,t) - y(\tau,s)\}$$

respectively.

Definition 2.11. (Backlog [SN20]). The *backlog* of a server at time t is the difference between arrival process $A(t)$ and according departure process $D(t)$ at time t,

$$q(t) := A(t) - D(t).$$

We use a service process $S(s,t)$ to describe the service between time s and time t [SN20]. We can use a dynamic S-server to describe such a service process.

Definition 2.12. (Dynamic S-Server [SN20]). Assume $A(0,t)$ and $D(0,t)$ are the arrival respectively departure process of a lossless server. Let $S(s,t) \geq 0$ for $0 \leq s \leq t$ be a positive, random process. The server is called a *dynamic S-server* if for any fixed sample path it holds for all $t \geq 0$ that

$$D(0,t) \geq A \otimes S(0,t).$$

Definition 2.13. (((σ_S, ρ_S))-Bounded Service). A dynamic S-server is ((σ_S, ρ_S))-bounded for $\theta > 0$ if

$$\phi_{S(s,t)}(-\theta) = \mathrm{E}\left[e^{-\theta S(s,t)}\right] \leq e^{-\theta\rho_S(-\theta)\cdot(t-s)+\theta\sigma_S(-\theta)}, \quad \text{for all } 0 \leq s \leq t.$$

Definition 2.14. (Work-Conserving Server [SN20, Fid06]). For any $t \geq 0$, let

$$\tau := \sup\{s \in [0,t] : D(s) = A(s)\}$$

be the beginning of the last backlogged period before t (such a τ always exists since $A(0) = D(0) = 0$). Assume the service $S(s,t) \geq 0$, $0 \leq s \leq t$, to be a stochastic process that is increasing in t with $S(\tau,\tau) = 0$. A server is said to be *work-conserving* if, for any fixed sample path, the server is non-idling and uses the entire available service $S(\tau,t)$ to serve backlogged data, i.e.,

$$D(t) = D(\tau) + S(\tau,t) = A(\tau) + S(\tau,t).$$

Remark 2.15. It can be seen that any work-conserving server is also a dynamic S-server [SN20].

2.1.4 Performance Bounds

With the introduction of arrival and service process, we are now able to derive performance bounds for a server. Due to the formal mathematical definitions and the upper bounds by (σ,ρ)-bounds, arrival and service can be well analyzed and controlled.

Definition 2.16. (Virtual Delay). Let $A(t)$ and $D(t)$ be the arrival and departure process. The *virtual delay* at time t is defined as

$$d(t) := \inf\{s \geq 0 : A(t) \leq D(t+s)\}.$$

Theorem 2.17. (Chernoff Bound [SN20]). *Let X be a random variable and $\theta > 0$. Then*

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

Proof. See, [SN20, p.33].

Theorem 2.18. (Violation Probability of Delay [SN20]). *Let $\theta > 0$. Suppose there are (σ_A, ρ_A)-bounded arrivals and a (σ_S, ρ_S)-bounded dynamic S-server. Additionally, the arrivals and the service are required to be independent. Further, assume the stability condition*

$$\rho_A(\theta) < \rho_S(-\theta). \tag{2.1}$$

Let $T \geq 0$. For the virtual delay, it holds for all $t \geq 0$ that

$$\mathrm{P}(d(t) > T) \leq e^{-\theta\rho_S(-\theta)T} \frac{e^{\theta(\sigma_A(\theta)+\sigma_S(-\theta))}}{\theta(\rho_S(-\theta) - \rho_A(\theta))}.$$

Proof. See, [SN20, pp.58].

Corollary 2.19. (Stochastic Delay Bounds [SN20]). Let $\theta > 0$. Assume (σ_A, ρ_A)-bounded arrivals and a (σ_S, ρ_S)-bounded dynamic S-server. Independent arrivals and service is required. Further assume the stability condition in Equation (2.1). Then, with violation probability $\epsilon \in (0, 1]$ a bound on the virtual delay is given by

$$T_\epsilon = \frac{1}{\rho_S(-\theta)} \cdot \left(\sigma_A(\theta) + \sigma_S(\theta) + \frac{1}{\theta}\log\left(\frac{1}{\epsilon \cdot \theta(\rho_S(-\theta) - \rho_A(\theta))}\right)\right) \tag{2.2}$$

Remark 2.20. The virtual delay bound calculated in Equation (2.2) should be optimized in θ to provide the tightest result possible.

2.1.5 Leftover Service

We assume a network topology where we have one server and multiple flows (as shown in Figure 4.1). We are interested in a performance bound of a specific flow, which we call flow of interest (foi) here and in the following. In order to be able to calculate such performance bounds in such a scenario, the "leftover service" is defined.

Theorem 2.21. (Leftover Service under Arbitrary Multiplexing [SN20]). *Consider two flows, f_1 and f_2, with respective arrival processes $A_1(s,t)$ and $A_2(s,t)$, that receive service process $S(s,t)$ (as can be seen for $n = 2$ in Figure 4.1). Further, we assume f_1 to be our foi and the scheduling to be arbitrary multiplexing. If the server is work-conserving, then the foi sees the dynamic S-server*

$$S_{l.o.}^{ARB}(s,t) = S_{l.o.}(s,t) = [S(s,t) - A_2(s,t)]^+, \quad 0 \leq s \leq t$$

the so-called leftover service.

Remark 2.22. The previous consideration of 2 flows in Theorem 2.21 can easily be generalized to

$$S_{l.o.}(s,t) = \left[S(s,t) - \sum_{i=2}^{n} A_i(s,t)\right]^+$$

with cross-flows $f_2, \ldots, f_n$.

2.2 Generalized Processor Sharing (GPS)

If we want to calculate performance bounds for a foi when there are other cross-flows, we also have to specify the order in which the arrivals are processed. For this task a so-called scheduling is required. So we have to take a look at the leftover service for a certain scheduling algorithm. In this section we want to define and consider a specific scheduling algorithm: Generalized Processor Sharing (GPS).

GPS is an ideal scheduling algorithm, which was initially known as "Weighted Fair Queuing" and a server that uses this algorithm is only a theoretical concept [PG94]. It is only a theoretical concept because it is based on a model that does not consider packets. However, since this is the case in reality, this algorithm cannot be applied directly in this form [PG93]. Also, GPS achieves perfect fairness and all other fair scheduling algorithms used in practice approximately emulate GPS [LBH09]. What fairness means in this context and the Metric used to examine it is described in more detail in [BCPV96].

With a GPS scheduled server, the service capacities are distributed according to defined weights ϕ for the respective flows and thereby the server acts efficiently. In addition, GPS scheduled servers can act flexible and are well analyzable [PG93]. For this let's have a look at the basics and definitions of GPS in order to be able to calculate the leftover service or (σ, ρ)-bounds for such a server.

2.2.1 GPS Basic

Definition 2.23. (GPS Resource Allocation [PG93, SN20]). Let n flows cross a GPS server (which is work conserving and operates at a constant rate c). Then, GPS applies a weighted fair allocation of resources according to weights $\phi > 0, i = 1, \ldots, n$. I.e., for any interval (s, t) in which flow f_i is (continuously) backlogged, it holds that

$$\frac{D_i(s,t)}{\phi_i} \geq \frac{D_j(s,t)}{\phi_j} \qquad \text{for all } j = 1, \ldots, n. \tag{2.3}$$

Theorem 2.24. (GPS Leftover Service Process [PG93, SN20]). *Consider a node that implements GPS for n flows. Assume that the node is work-conserving server (Definition 2.14) with service process $S(s,t)$ that services an aggregate of flows $f_1, \ldots, f_n$. Further, we assume f_i to be our foi. Then the foi sees the dynamic S-server*

$$S_{l.o.}^{GPS}(s,t) = \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} \cdot S(s,t), \quad 0 \leq s \leq t.$$

Proof. See [PG93] or [Cha00, p.68].

Work-conserving server [BBLC18]

The results of Theorem 2.24 can be generalized to work-conserving servers.

Let the server be a work-conserving server S to the aggregate. Consider the backlogged period $(s_i, t]$ of flow i. For the aggregate, this is also a backlogged period and thus

$$\sum_{j=1}^{n} D_j(s_i, t) \geq S(s_i, t) \tag{2.4}$$

by the definition of a work-conserving server (Definition 2.14).

Proof. By Equation (2.3) it is known that

$$\left(\sum_{j=1}^{n} \phi_j \right) D_i(s_i, t) \geq \phi_i \sum_{j=1}^{n} D_j(s_i, t). \tag{2.5}$$

Combining the results from Equation (2.4) and Equation (2.5) yields

$$\left(\sum_{j=1}^{n} \phi_j \right) D_i(s_i, t) \geq \phi_i \sum_{j=1}^{n} D_j(s_i, t) \geq \phi_i S(s_i, t)$$

and hence

$$D_i(s_i, t) \geq \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} S(s_i, t).$$

In summary, during the backlogged period of flow i, the output is given by $\frac{\phi_i}{\sum_{j=1}^{n} \phi_j} S(s_i, t)$. This is the proof, that $S_i(s_i, t) := \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} S(s_i, t)$ is a work-conserving server for flow i.

□

$(\sigma(\theta), \rho(\theta))$-Bounds [PG93]

According to [PG93], define

$$S_1(s, t) := \frac{\phi_1}{\sum_{j=1}^{n} \phi_j} \cdot S(s, t).$$

Assume that a work-conserving service element S is (σ_S, ρ_S)-bounded. Then, assuming n flows with weights $\phi_j > 0, j = 1, \ldots, n$, one can obtain under GPS

$$\begin{aligned}
\mathrm{E}\left[e^{-\theta S_1(s,t)}\right] &= \mathrm{E}\left[e^{-\theta\left(\frac{\phi_1}{\sum_{j=1}^n \phi_j}\right)S(s,t)}\right] \\
&\leq e^{-\left(\frac{\phi_1}{\sum_{j=1}^n \phi_j}\theta\right)\left(\rho_S\left(\frac{\phi_1}{\sum_{j=1}^n \phi_j}\theta\right)(t-s)-\sigma_S\left(\frac{\phi_1}{\sum_{j=1}^n \phi_j}\theta\right)\right)} \\
&= e^{-(\bar{\phi}_1\theta)\rho_S(-\bar{\phi}_1\theta)(t-s)+(\bar{\phi}_1\theta)\sigma_S(-\bar{\phi}_1\theta)} \\
&= e^{-\theta\bar{\phi}_1\rho_S(-\bar{\phi}_1\theta)(t-s)+\theta\bar{\phi}_1\sigma_S(-\bar{\phi}_1\theta)},
\end{aligned}$$

where

$$\bar{\phi}_1 := \frac{\phi_1}{\sum_{j=1}^n \phi_j}.$$

One obtains a (σ_S, ρ_S)-bound with

$$\begin{aligned}
\sigma_S(-\theta) &= \bar{\phi}_1\sigma_S(-\bar{\phi}_1\theta) = \frac{\phi_1}{\sum_{j=1}^n \phi_j}\sigma_S\left(-\frac{\phi_1}{\sum_{j=1}^n \phi_j}\theta\right), \\
\rho_S(-\theta) &= \bar{\phi}_1\rho_S(-\bar{\phi}_1\theta) = \frac{\phi_1}{\sum_{j=1}^n \phi_j}\rho_S\left(-\frac{\phi_1}{\sum_{j=1}^n \phi_j}\theta\right).
\end{aligned}$$

2.2.2 GPS M

In this subsection we consider another method to calculate performance bounds for GPS scheduled flows. This is the method of Chang, presented in [Cha00], which is called GPS M in the context of this thesis. With this approach we get the additional possibility to select flows that are not GPS scheduled, but are guaranteed to get the service they need. This allows us to calculate more accurate upper performance bounds, as will be shown later. For the choice of which flows to consider in the GPS scheduling, a set M was used in the original work of Chang [Cha00], which also explains the choice of the name GPS M. As we will see, all flows that are elements of the set M are scheduled by GPS, the others are not.

In the following, we will look at the statements that can be made about a GPS M scheduled server.

Work-conserving Server [PG93, Cha00]

Let the server be a work-conserving server with service process $S(s,t)$ that services an aggregate of flows $f_1, \ldots, f_n$. Assume f_i to be our foi. For $\varnothing \neq M \subset N := \{1, \ldots, n\}$

and $0 \leq s \leq t$ the foi sees the dynamic S-server

$$S_i(s,t) = \max_{\varnothing \neq M \subset N} \left\{ \frac{\phi_i}{\sum_{j \in M} \phi_j} \left(S(s,t) - \left(\sum_{j \notin M} D_j(s,t) \right) \right) \right\}$$

Proof. See [Cha00, p.69].

This result can be generalized to work-conserving servers.

Proof. Let s_i be the beginning of the backlogged period up of flow i up to time t. By the observation from above we have

$$D_i(t) - D_i(s_i) \geq \frac{\phi_i}{\sum_{j \in M} \phi_j} \left(S(s,t) - \left(\sum_{j \notin M} D_j(s,t) \right) \right)$$

for all $M \subset N$. Hence

$$D_i(s_i,t) \geq \max_{\varnothing \neq M \subset N} \left\{ \frac{\phi_i}{\sum_{j \in M} \phi_j} \left(S(s,t) - \left(\sum_{j \notin M} D_j(s,t) \right) \right) \right\}.$$

□

The question of obtaining a bound on the departures $D_j(s_i,t)$ can be approached in a variety of ways. [Fid10] suggests to use

$$\sum_{j \notin M} D_j(s_i,t) \leq \sum_{j \notin M} \left\{ A_j \oslash \left(\frac{\phi_j}{\sum_{k=1}^n \phi_k} S \right)(s_i,t) \right\} \tag{2.6}$$

$(\sigma(\theta),\rho(\theta))$-Bounds [PG93, Cha00]

To obtain the (σ,ρ)-bounds we use Equation (2.6)

$$\sum_{j \notin M} D_j(s_i,t) \leq \sum_{j \notin M} \left\{ A_j \oslash \left(\frac{\phi_j}{\sum_{k=1}^n \phi_k} S \right)(s_i,t) \right\}.$$

Let $\varnothing \neq M \subset N$ be arbitrary but fixed. We receive the work-conserving server with service process

$$\begin{aligned} & \frac{\phi_i}{\sum_{j \in M} \phi_j} \left(S(s,t) - \left(\sum_{j \notin M} D_j(s,t) \right) \right) \\ \geq & \frac{\phi_i}{\sum_{j \in M} \phi_j} \left(S(s,t) - \left(\sum_{j \notin M} \left\{ A_j \oslash \left(\frac{\phi_j}{\sum_{k=1}^n \phi_k} \right)(s,t) \right\} \right) \right) \\ =: & S_1(s,t). \end{aligned}$$

Further, we assume the stability condition

$$\rho_{A_j}(\bar{\phi}_{1,M}\theta) < \hat{\phi}_j \rho_S(-\bar{\phi}_{1,M}\hat{\phi}_j\theta) \tag{2.7}$$

For independent flows it can be calculated that

$$\begin{aligned}
&\mathrm{E}\left[e^{-\theta S_1(s,t)}\right] \\
=&\mathrm{E}\left[e^{-\theta\left(\frac{\phi_i}{\sum_{j\in M}\phi_j}\left(S(s,t)-\left(\sum_{j\notin M}\left\{A_j\oslash\left(\frac{\phi_j}{\sum_{k=1}^{n}\phi_k}\right)(s,t)\right\}\right)\right)\right)}\right] \\
\leq& e^{-\theta\rho_S(-\theta)\cdot(t-s)+\theta\sigma_S(-\theta)}
\end{aligned}$$

with

$$\begin{aligned}
\sigma_S(-\theta) &= \bar{\phi}_{1,M}\sigma_S(-\bar{\phi}_{1,M}\theta) + \bar{\phi}_{1,M}\sum_{j\notin M}\Big\{\sigma_{A_j}(\bar{\phi}_{1,M}\theta) + \hat{\phi}_j\sigma_S\left(-\bar{\phi}_{1,M}\hat{\phi}_j\theta\right) \\
&\quad - \frac{1}{\theta}\log\left(1 - e^{\bar{\phi}_{1,M}\theta(\rho_{A_j}(\bar{\phi}_{1,M}\theta)-\hat{\phi}_j\rho_S(\bar{\phi}_{1,M}\hat{\phi}_j\theta))}\right)\Big\} \\
\rho_S(-\theta) &= \bar{\phi}_{1,M}\left(\rho_S(-\bar{\phi}_{1,M}) - \sum_{j\notin M}\rho_{A_j}(\bar{\phi}_{1,M}\theta)\right),
\end{aligned} \tag{2.8}$$

where

$$\begin{aligned}
\bar{\phi}_{1,M} &:= \frac{\phi_1}{\sum_{j\in M}\phi_j}, \\
\hat{\phi}_j &:= \frac{\phi_j}{\sum_{k=1}^{n}\phi_k}, \quad \text{for } \varnothing \neq M \subset N.
\end{aligned}$$

Remark 2.25. Note that GPS Basic is a special case of GPS M. In the case where all flows are in the set M, we have the consideration of GPS Basic. When all flows are element of set M, it is assumed that all flows are permanently backlogged and actually consume their allocated part of the service process. If some flows do not fully use their allocated resources, then the remaining resources are divided among the other backlogged flows according to the ratio of their weights [Fid06].

3 GPS Analysis Heuristics

After introducing the background to the Stochastic Network Calculus and the Generalized Processor Sharing, we come to the analysis that is the focus of this thesis. In general, we want to calculate performance bounds for GPS scheduled servers using SNC. For this, we have already seen the GPS Basic (Subsection 2.2.1) and the GPS M method (Subsection 2.2.2). GPS M gives us the additional option of not including flows in the GPS scheduler.

The chapter begins with defining the precise purposes of the research. Subsequently, the methods respectively the used heuristics will be described in detail.

3.1 Purpose of the investigations

The option to select a set M for GPS M that contains the flows that are not included in the GPS scheduler, in order to be able to determine more accurate upper bounds raises new questions. Intuitively, the question arises what criteria should be used to select the flows to be included in this set M. To be more precise, the question is how to select them efficiently. It is possible to calculate the optimal flows for this set M that minimize the upper performance bounds (see Section 4.2). However, this method is not efficient, since all possible subsets for M have to be considered, which leads to exponential runtimes. I.e., for $|\mathcal{P}(M)| = 2^{|M|}$ possibilities the delay bounds have to be calculated (including the respective optimizations for θ, as mentioned in Remark 2.20). Since the question how to select flows for the set M efficiently is still an open problem, different heuristics are presented and investigated in one of the following sections.

3.2 Assumptions for the analysis

For the analysis in this thesis we assume, without loss of generality, f_1 to be our foi for a network topology given in Figure 4.1.

Since the delay bounds are to be computed for the foi f_1, it is assumed that this flow has to be in the set M of flows for GPS M, i.e., it holds for the thesis foi $\in M$. This assumption is crucial, since otherwise no computation of performance bounds for the foi would be possible and required.

3.3 Heuristics

After explaining the reasons for the need for heuristics and the general assumptions, heuristics are going to be introduced in this section. The heuristics follow different approaches in each case but follow a certain basic pattern, which is introduced as "Stability condition preprocessoing" and "GPS heuristic algorithm".

Stability condition preprocessing

Before the basic algorithm is executed on the respective heuristics, there is a preprocessing calculation where the stability condition (see Equation (2.7)) is checked for each flow. These preprocessing calculations are used by most heuristics below, what we will see in detail later.

Remarkably, the $\rho(\theta)$ for arrival and service process needs to be calculated, which depends on θ in each case. However, the optimization of θ only takes place in the further steps when calculating the performance bounds. I.e., the θ must be cleverly selected for the stability condition check. For this purpose, θ is chosen to be equal to 1 for the thesis, since 1 as a multiplicative neutral element does not influence the multiplication in any way.

If the stability condition for a certain flow is not fulfilled, i.e., if holds

$$\rho_{A_j}(p_j\bar{\phi}_{1,M}\theta) \geq \hat{\phi}_j\rho_S(-\bar{\phi}_{1,M}\hat{\phi}_j\theta),$$

then the flow has to be an element of the set M. If a flow that does not satisfy the stability condition is not element of set M, then the arrival rate of the flow would be greater than (or equal to) the service rate. If this is the case, the server could be permanently fully utilized for this flow. Thus, it is not possible to calculate performance bounds for the GPS scheduled flows, in particular for the foi.

From a mathematical point of view, it can be seen that in this case the $\sigma_S(-\theta)$ (see Equation (2.8)) for the server cannot be calculated, since the logarithm is calculated from a negative number and it is known that the logarithm is only defined for positive input.

GPS heuristic algorithm

The following algorithm is applied for all heuristics used in the thesis (except for the heuristic Minimized GPS Set (Subsection 3.3.5)):

Algorithm 1 GPS Heuristic

```
Input  : List of flows $F$, set $M \neq \varnothing$ and a sorting criteria function $scf$
Output: Performance bound for the foi
1 Sort the List $F$ according to $scf$ and we get $F_{sorted}$.
2 while $F_{sorted}$ is not empty do
3     insert the first flow from $F_{sorted}$ into the set $M$
4     calculate performance bounds for the foi with $M$ and save it in a list
        $list\_of\_bounds$
5     remove the first flow from $F_{sorted}$
6 end
7 return min($list\_of\_bounds$)
```

Remark 3.1. Note here that despite the calculation of the minimum of list B in Algorithm 1, the calculated performance bound is not necessarily the most accurate possible bound for the foi. Is it only the minimum of the performance bounds for the considered sets M.

3.3.1 Sorted Randomly

This heuristic differs from the others by not executing stability condition preprocessing (see Section 3.3). As the name suggests, the flows are sorted randomly. I.e., for the application of Algorithm 1 a function is used as sorting criteria function, which sorts the flows randomly. Algorithm 1 is executed for this heuristic and a performance bound is obtained.

3.3.2 Sorted Weights

This heuristic focuses on the weights of the respective flows ϕ for GPS scheduling. However, before the flows are sorted by their weights ϕ, stability condition preprocessing is performed. That means that the flows that do not satisfy the stability condition become elements in the set M before Algorithm 1 is executed. The set M with the foi and the flows added in the preprocessing step is used for the algorithm. As mentioned above, sorting by the weights ϕ of the flows for GPS scheduling (ascending) is used as the sorting criteria function. Finally, Algorithm 1 is executed to calculate a performance bound.

3.3.3 Sorted Arrival-Rates

For this heuristic the stability condition preprocessing check is performed and the according flows are added to the set M. The sorting criteria function used here is the sorting by arrivals-rates $\rho_A(\theta)$ (descending).

Remarkably, the arrival rates are dependent on θ. Analogous to the stability condition preprocessing (see Section 3.3), the optimization of θ takes place at a later time and is therefore set to 1 (of course only for sorting the list and not for calculating performance bounds).

We now run Algorithm 1 and obtain a performance bound.

3.3.4 Sorted Arrival-Bursts

This heuristic has the same procedure as the Sorted Arrival-Rates heuristic, but for the sorting criteria function we replace the sorting by the arrival rates by the sorting by the arrival bursts $\sigma_A(\theta)$ (ascending).

Since $\sigma_A(\theta)$ depends on θ we have the same problem as above. θ is set to 1 for sorting the list.

3.3.5 Minimized GPS Set

This heuristic only performs the stability condition preprocessing. With the resulting set M a performance bound is calculated and returned. I.e., with this heuristic it can be compared how decisive the further filling of the set M with the flows according to the respective sorting criteria really is.

3.3.6 Minimized GPS Set (Random Fill up of M)

The last heuristic is a combination of Minimized GPS Set (see Subsection 3.3.5) and Sorted Randomly (see Subsection 3.3.1). For this, the stability condition preprocessing is executed first and then Algorithm 1 is executed with a sorting criteria function that sorts randomly. With this heuristic it can be compared how much impact the sorting criteria function really has after the stability condition has been checked.

Remark 3.2. It should be noted that the computed delay bound of the Minimized GPS Set heuristic upper bounds the computed delay bounds of the Sorted Weights, Sorted Arrival-Rate, Sorted Arrival-Bursts, and Minimized GPS Set (Random Fill up of M) heuristics. This is due to the fact the M of Minimized GPS Set used to compute a delay bound is also used by the other four heuristics and, in addition, even further different sets M are used to calculate delay bounds by the heuristics. Consequently, Minimized GPS Set cannot find more accurate bounds than the other heuristics, it could compute the same bounds at best.

4 Numeric Evaluation

In order to benchmark the heuristics described in Chapter 3, they must be numerically evaluated. The results are visualized and analyzed by graphs. Thus, the heuristics are examined under different conditions and then compared to the previously known performance bounds of GPS Basic in Subsection 2.2.1 or the performance bounds calculated by exhaustive search. But before the exact results are evaluated, we first take a closer look at the scenario for the analysis.

4.1 Scenario, topology and performance metric

For the analysis of the heuristics the scenario code provided by Paul Nikolaus [Nik21] was used and extended accordingly. The code for the program is written in Python, version 3.8 [VRD09].

For the analysis, we assume an n flows - single server network topology (see Figure 4.1).

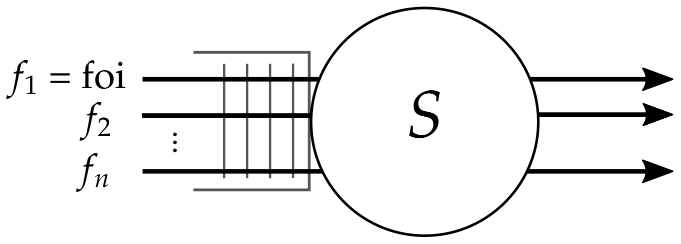


Figure 4.1: Network Topology: n flows - single server

For the entire investigations of the heuristics, we calculate stochastic delay bounds with a violation probability of $\epsilon = 10^{-6}$ (see Corollary 2.19).

This means that for the foi with the given network topology (Figure 4.1) upper delay bounds are calculated, which will not be exceeded with a probability of 99.9999%.

4.2 Benchmark

To benchmark the heuristics we consider the delay bounds on the one hand, which are calculated with the GPS Basic method as a maximum upper bound and on the other hand the minimum upper bound, which can be calculated with the help of exhaustive search.

When executing exhaustive search every possible subset of the set of flows is used for the calculation of a delay bound as M for GPS M. This ensures that the minimal delay bound is always found.

Using GPS Basic, we obtain a maximum upper bound for the heuristics. The heuristics (except heuristic "Minimized GPS Set", see Subsection 3.3.5) also compute a delay bound for the set M containing all flows. GPS Basic therefore also is a special case of GPS M. In GPS Basic we implicitly assume that all flows are backlogged (see Remark 2.25). This worst-case assumption, which is made in GPS Basic, does not have to be true in a stochastic system, which we are in here.

Consequently, it holds that for the minimal delay bound computed by using exhaustive search "es", by the heuristics "heu" and by GPS Basic "gps-b"

$$\text{es} \leq \text{heu} \leq \text{gps-b}.$$

4.3 Evaluation of the heuristics

For the evaluation of the heuristics, we should first be conscious of the settings of the considered system that could affect the quality of the heuristics.

In order to keep the overview, we list the entire setting options below:

- service rate / utilization,
- number of flows that passes through the server,
- arrival types configurations,
- weights ϕ in the list of weights of the flows for GPS scheduling.

And for the sake of completeness, the choice of the *network topology* and the *performance metric* are also to mention here, since these could also be changed in principle. However, these were fixed for the analysis (see Section 4.1).

Note here that the server utilization can be derived from the service rate with the given arrivals. It should also be noted that the heuristics can be examined with a fixed utilization, even if arrivals with random rates ρ_A are used. The service rate can be calculated for a given set of arrivals and the desired utilization. E.g., for a utilization of u and the set of all flows F, the service rate is given by

$$\text{service rate} = \frac{\sum_{f \in F}(\rho_f)}{u}.$$

Since different arrival types can lead to different results in the delay bound calculations, a number of possible types of arrivals were taken into account. Now, we take a closer look at the list of the types used:

- M/M/1 (see Example 2.8),
- M/D/1 (see Example 2.8),
- Discrete-time MMOO traffic (see Example 2.9),
- Aggregated discrete-time MMOO traffic (see Example 2.9),
- D/M/1 (see [SN20]),
- D/Gamma/1,
- D/Weibull/1,
- A uniform distributed random mix of the above arrival types.

For each of the arrival types, uniformly distributed random numbers are used for the parameters in the following evaluation. I.e., the arrivals that are considered for this are randomly generated. As an example, we take a look at M/M/1 traffic (see Example 2.8). For this, λ and μ are each uniformly distributed random numbers between 0 and 10 and between 1 and 10, respectively, used to generate the arrivals. In this way, the parameters of all arrivals are chosen in adjusted number intervals.

The choice of weights ϕ for the list of weights for GPS scheduling can also be decisive for the accuracy of delay bounds calculated by heuristics. For this purpose, we consider two different configurations for the evaluation:

- all weights are equal,
- uniform distributed random numbers between 0 and 1.

The plots presented in the following subsections consider the number of flows or server utilization on the x-axis and the "delay ratio" on the y-axis.

The delay ratio is defined as follows:

$$\text{delay ratio} := \frac{\text{delay bound heuristic}}{\text{delay bound exhaustive search}}. \tag{4.1}$$

The delay ratio is calculated for the delay bounds of GPS Basic in analogy to Equation (4.1). This ratio is used to achieve a better comparability of many data sets (in the following, averages of thousands of computations are considered). Due to the high degree of randomness in the collection of data, the absolute values of the calculated delay bounds can sometimes be very different and would therefore not be readily comparable.

The average of several delay ratios is used for the data points. For the calculations of the delay ratios, new random parameters are generated. I.e., each calculation is independent of each other.

In order to be able to overview the deviation of the calculated averages of the sample from the actual mean value, we use the standard error of the mean (SEM). This is also plotted in the graphs for the respective data points (averages). The SEM is added to the average to give an upper bound for the deviation and subtracted to give the lower bound [AB05]. The SEM can be calculated for a given sample with sample size n and sample standard deviation σ as follows [AB05]:

$$\mathrm{SEM} = \frac{\sigma}{\sqrt{n}}.$$

4.3.1 Investigation of different arrival types

In this section we investigate the influence of different arrival types on the calculated delay bound of the heuristics. For this we vary the number of flows that pass through the server as well as the server utilization for two independent considerations.

Impact of number of flows

For each plot of a graph of the respective arrival types, the average of 5000 calculated delay ratios (see Equation (4.1)) was used for every data point.

We define the following settings for the analysis:

- Server utilization: 60% ,
- Weights ϕ for GPS scheduling: uniform distributed random numbers between 0 and 1.

For discrete-time MMOO traffic, we consider the delay ratios of the heuristics and GPS Basic depending on the number of flows passing through the server, as shown in Figure 4.2.

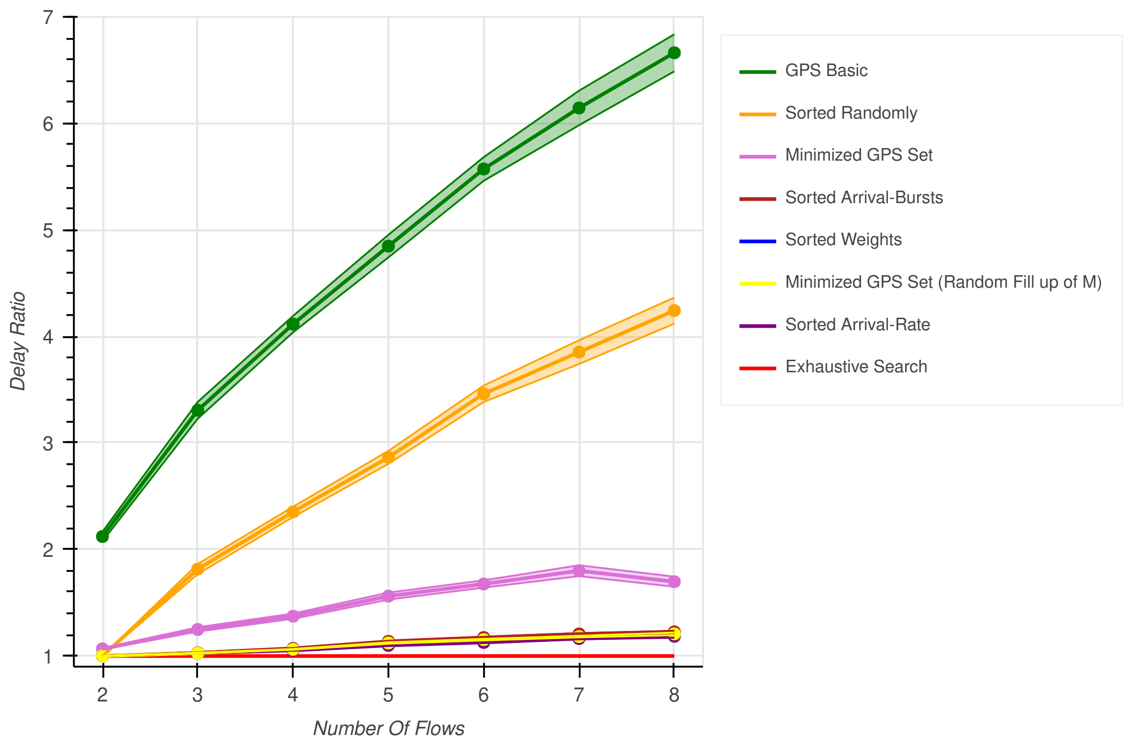


Figure 4.2: Graph of the delay ratio of discrete-time MMOO traffic depending on the number of flows.

For this graph in Figure 4.2 we can observe the following :

- The delay ratio of all heuristics and GPS Basic increases with the number of flows. This is generally in line with expectations, since with the number of flows running through the server the service must be shared for more flows. Accordingly, it becomes more difficult to specify accurate bounds, as a larger number of flows can take away service time for the foi. The growth appears to be roughly linear.
- It can be seen that GPS Basic forms the maximum of the delay ratios towards the top, as we already expected, see Section 4.2.
- A special feature considered in this plot is the Sorted Randomly heuristic, which is omitted in the following plots, since the statement about it would be analogous in each case. On average less accurate delay bounds are obtained in comparison with the other heuristics that check the stability condition for the respective flows. The gap between the delay ratios of Sorted Randomly and Minimized GPS Set (by a factor of 2.7) highlights this point very well. It can therefore be assumed that this difference is due to the use of the stability condition check, because the only difference between Minimized GPS Set (Random Fill up of M) is the stability condition preprocessing and with the use of this heuristic more accurate delay bounds can be calculated.

- If we take a look at the delay ratio for two flows, we see that it is 1 for all heuristics (except Minimized GPS Set). E.g., this can also be observed in Figure 4.3 or Figure 4.4. In general we can assume that due to the definition of the heuristics this behavior is always observed for a system with two flows. With two flows there is only the foi and one other flow. Since the foi is an element of the set M, there are only two possibilities - the set M consists only of the foi or of foi and the other flow. Both possibilities are considered by the heuristics, so we can calculate the minimal upper bound with certainty. For Minimized GPS Set this does not have to be the case, since the second flow does not always violate the stability condition.

- The four heuristics Sorted Arrival-Bursts, Sorted Weights, Minimized GPS Set (Random Fill up of M) and Sorted Arrival-Rate, provide the best delay ratios, with about 1.2 for 8 flows, which is about $\frac{1}{5}$ of the delay ratio of GPS Basic. The noticeable difference to the heuristic "Minimized GPS Set" can be explained by Remark 3.2. What is also noticeable at first glance is that the heuristics differ much less from each other than they do to the others. Let's take a closer look at these four heuristics. To do this, we "zoom in" on Figure 4.2 and consider the heuristics again in Figure 4.3. Now the differences between the delay ratios are more obvious. In the case of discrete-time MMOO traffic, the Sorted Arrival-Rate heuristic provides the most accurate bounds and the Sorted Arrival-Bursts heuristic provides the least accurate bounds of of these heuristics.

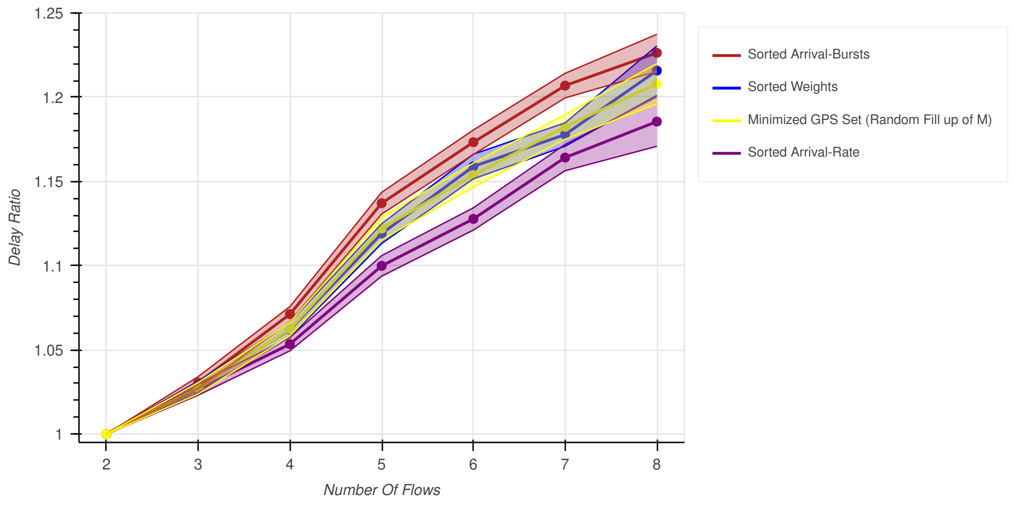


Figure 4.3: Graph of the delay ratio of discrete-time MMOO traffic ("zoomed in") depending on the number of flows.

However, we do not want to make a general conclusion as the order of heuristics differs, e.g., when considering other traffic classes (see Figure 4.4 with D/M/1 traffic). I.e., the differences in the two-digit comma range of the delay ratios of these four heuristics depends on the arrival types used.

What is noticeable for the graph in Figure 4.4, however, is that Sorted Arrival-Bursts and Minimized GPS Set (Random Fill up of M) appear to behave almost identically and provide slightly less accurate delay bounds than the other two heuristics. This is due to the fact that all traffic classes considered in this thesis have $\sigma_A = 0$, with the notable exception of discrete-time MMOO. For M/M/1 and M/D/1 this is shown in Example 2.8.
This means that sorting by the bursts σ_A does not work and can be considered as a special case of random fill up for Minimized GPS Set (Random Fill up of M). For this reason, for further analysis, we omit the Sorted Arrival-Bursts heuristic for the arrival types for which $\sigma_A = 0$ holds. Since this is not the case for discrete-time MMOO traffic, we do not see any clear differences or similarities between the heuristics (see Figure 4.3). A clustering of Sorted Arrival-Bursts with Minimized GPS Set (Random Fill up of M) and Sorted Arrival-Rates with Sorted Weights, where the latter providing more accurate bounds, could have been observed for the other arrivals where $\sigma_A = 0$. This can be explained by the fact that by considering the weights or the arrival rates, the flows can already be brought into a more meaningful order for the calculation of the bounds, because the rates and weights play a central role and can influence with the order they provide. This, in consequence, leads to more accurate bounds.

However, Sorted Weights does not necessarily always yield better delay bounds (see Figure 4.5b with M/D/1 traffic).

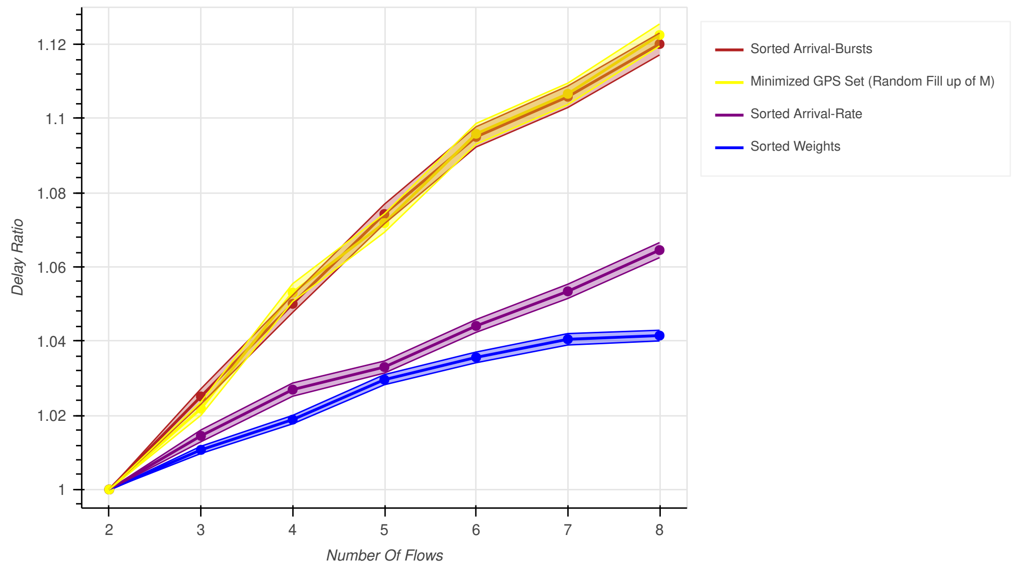
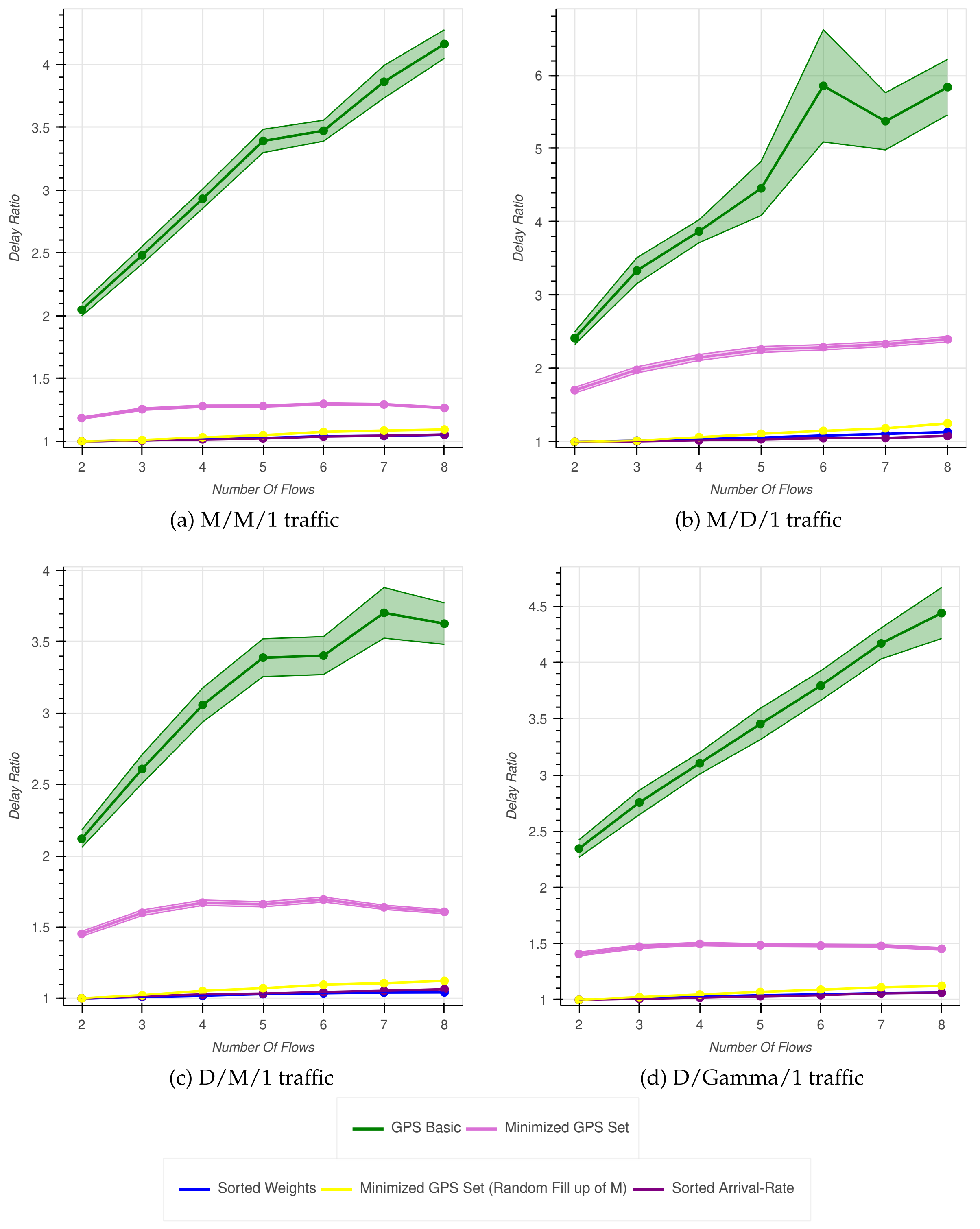


Figure 4.4: Graph of the delay ratio of D/M/1 traffic ("zoomed in") depending on the number of flows.

Thus, one can say that the considered heuristics behave roughly similar with regard to the delay ratios, but on closer inspection they show small differences, especially for arrivals with $\sigma_A = 0$. This behavior can be confirmed by looking at other arrival types in Figure 4.5 and Figure 4.6.

When looking at Figure 4.5 and Figure 4.6 we see that for all arrival types the delay ratio of the heuristics and GPS Basic increases with the number of flows. This can be interpreted analogous to above. The further descriptions and interpretations of the plots are analogous to those for Figure 4.2. However, some graphs show some minor differences to Figure 4.2. These will now be examined and clarified:

- There are occasional spikes in the data, such as Figure 4.5b at flow 6. Such spikes cannot be easily explained, but one can assume that they occur due to the high degree of randomness and too small sample size.
- Differences in "Minimized GPS Set":
  - The relations of the delay ratio of "Minimized GPS Set" to the delay ratios of other heuristics varies. It can be assumed that these differences arise when the stability condition is violated more often for some arrival types compared to others. This in turn means that a higher (or lower) proportion of all flows for Minimized GPS Set are elements of the set M, which consequently affects the accuracy of the bounds.
  - Interestingly, the delay ratio of "Minimized GPS Set" does not increase significantly in the number flows in most cases (e.g., Figure 4.5a). Why this heuristic behaves this way could not be found out within the scope of this thesis. This investigation would be part of future work.



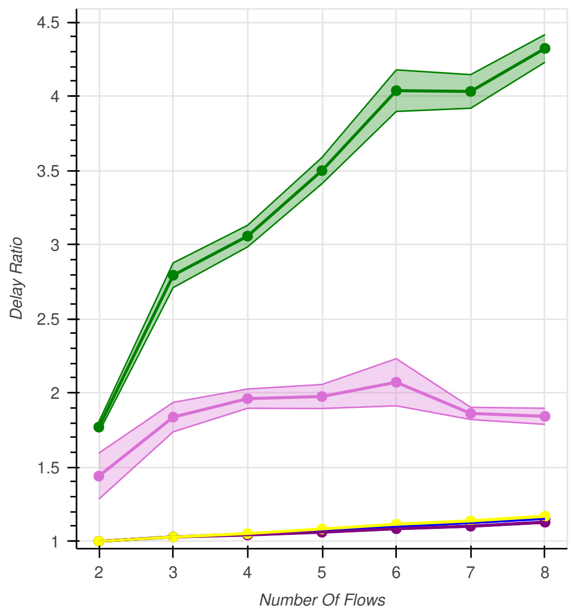
(a) M/M/1 traffic

(b) M/D/1 traffic

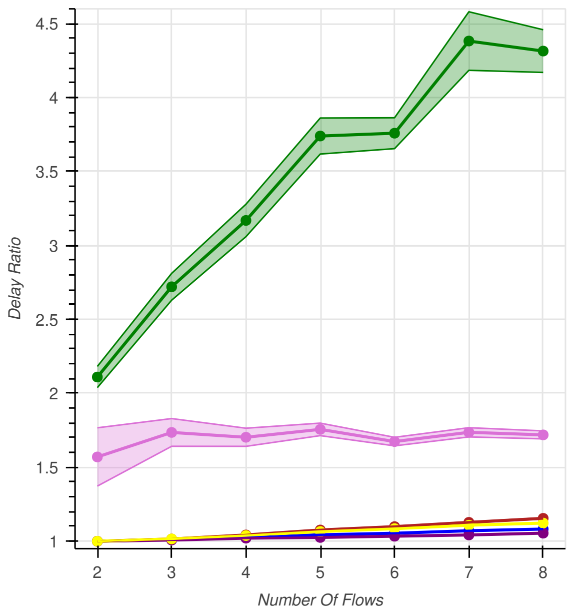
(c) D/M/1 traffic

(d) D/Gamma/1 traffic

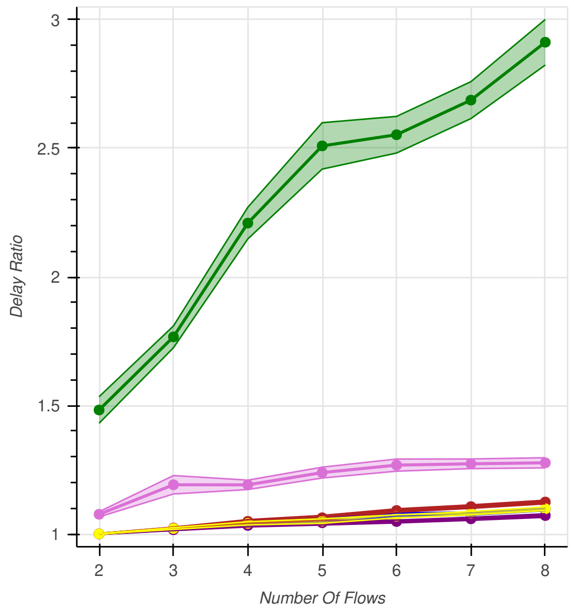
Figure 4.5: Graphs of the delay ratio of M/M/1, M/D/1, D/M/1 and D/Gamma/1 traffic depending on the number of flows.



(a) D/Weibull/1 traffic



(b) Mixed traffic



(c) Aggregated discrete-time MMOO traffic (with $n = 5$)

Figure 4.6: Graphs of the delay ratio of D/Weibull/1, Mixed and aggregated discrete-time MMOO traffic depending on the number of flows.

To sum it up, one can say that the check of the stability condition is crucial for the heuristics, which is shown by the difference between Sorted Randomly and the other heuristics. We can also assume that after the check of the stability condition a further insertion of the remaining flows is meaningful, which the comparison between Minimized GPS and the other heuristics shows. However, it is not necessarily possible to say according to which sorting criteria, also because "Minimized GPS Set (Random Fill up of M)" does not deliver much less accurate results compared to the other heuristics with the corresponding sorting of the flows. It could be concluded that the check of the stability condition is decisive, the following fill up can happen randomly to reduce the runtime in case of a very high number of flows (see Section 4.4).

In the following we will see that these results can be confirmed by looking at the delay ratios of the heuristics in dependence of the server utilization for different arrival types.

Impact of server utilization

In this part of the numeric evaluation we will take a look at the behavior of the heuristics depending on the server utilization. First of all, an important aspect that will be seen in the data should be explained. The delay ratios (see Equation (4.1)) are considered for server utilization of 100% or even 110%. This could be a bit counter intuitive at first. But this is a special characteristic of GPS scheduling. Even at such high utilization's the foi is still stable. The remaining flows use that much capacity of the server, which does not matter when considering delay bounds for the foi, since the allocated service by GPS can be used by the foi. Hence, we are still able to calculate delay bounds.

For each plot of a graph of the respective arrival types, the average of 2000 calculated delay ratios was used for every calculated data point.

We define the following settings for the analysis:

- Number of flows: 7 ,
- Weights ϕ for GPS Scheduling: uniform distributed random numbers between 0 and 1.

For the following analyses, we choose the same approach as already used for the investigation of the delay bounds depending on the number of flows in Subsection 4.3.1. Accordingly, we take a look at the data for discrete-time MMOO traffic in Figure 4.7.

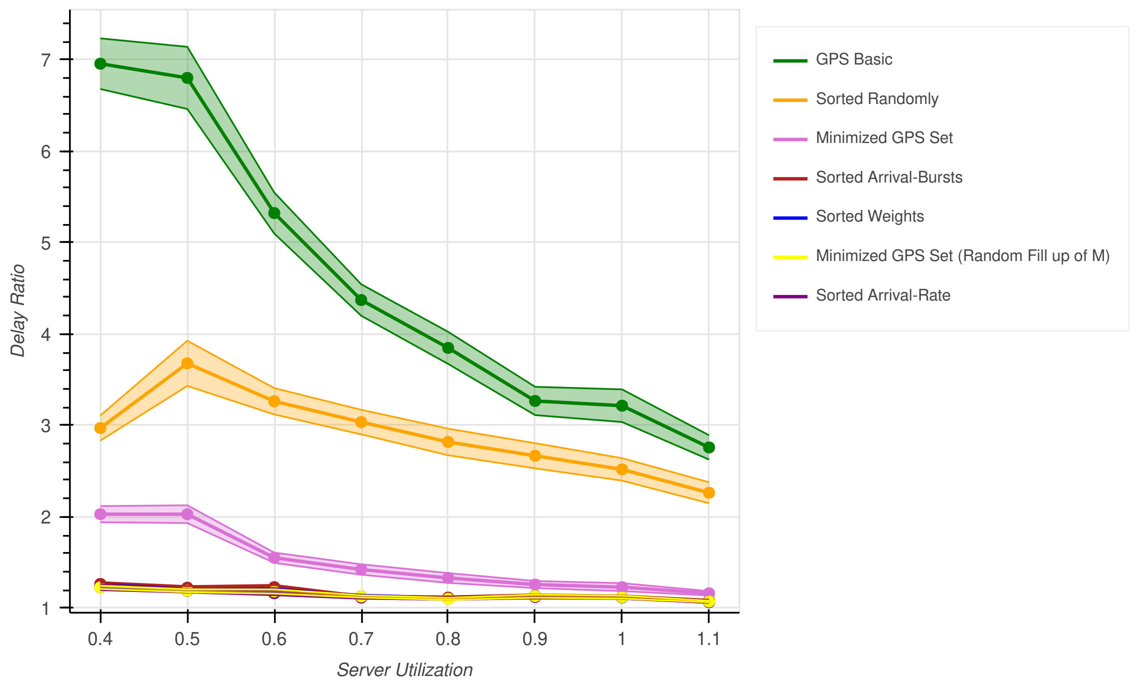


Figure 4.7: Graph of the delay ratio of discrete-time MMOO traffic depending on the server utilization.

We can state the following:

- The delay ratio of all heuristics and GPS Basic decreases with increasing server utilization. How can this observation be explained? With increasing server utilization more flows need to be element of set M to avoid the violation of the stability condition. Hence, with increasing utilization the calculated delay bounds of GPS Basic become more accurate, since only one set M with all flows is considered. But also the heuristics provide better bounds. Since the heuristics run through a stability condition preprocessing step, these flows become elements of the set M for the heuristics anyway. This reduces the degree of freedom for further filling with flows (see Algorithm 1), which means that there are basically fewer possibilities to find a suitable set M and thus the probability to make a good choice for M increases.
- Again, we see here that GPS Basic calculates the least accurate delay bounds and that the delay ratios of Sorted Randomly are between those of GPS Basic and the other heuristics that check the stability condition. These results can be interpreted in the same way as above.
- A Further observation that the delay ratios of Minimized GPS Set are higher or equal than the other heuristics (except Sorted Randomly) is in line with it from above and has already been explained.
- Looking at the remaining four heuristics, we can see that they again give the

best delay ratios in this plot, but with decreasing difference to delay ratios of the other two heuristics or GPS Basic. As already seen in Figure 4.2 from above, we can see here that the four heuristics Sorted Arrival-Bursts, Minimized GPS Set (Random Fill up of M), Sorted Arrival-Rates and Sorted Weights differ only slightly from each other. We "zoom in" on Figure 4.7 and thus obtain the view in Figure 4.8. The differences are still relatively small, but we see they are basically existing.

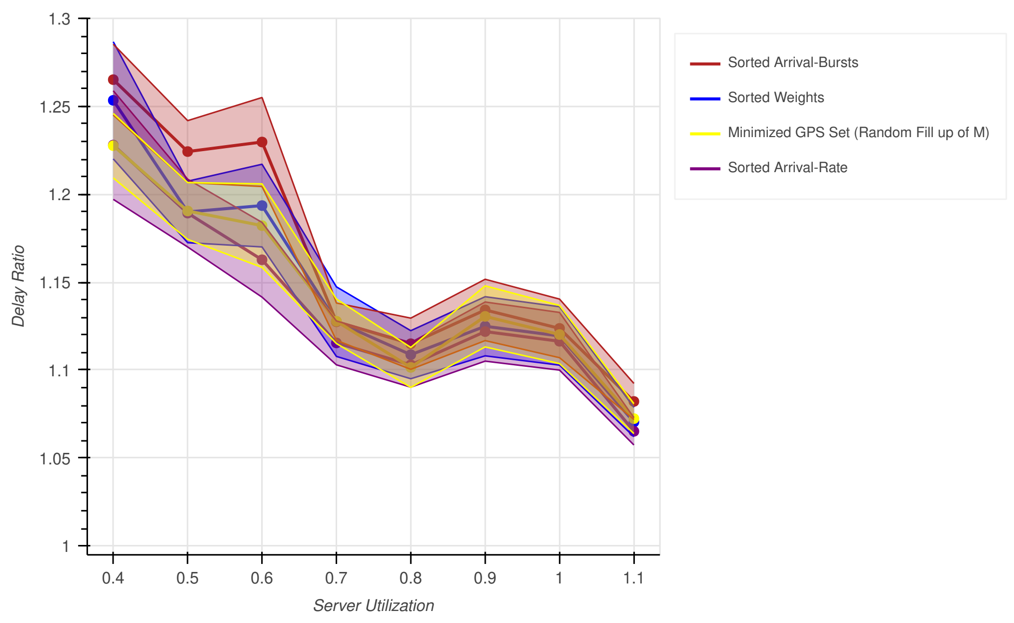


Figure 4.8: Graph of the delay ratio of discrete-time MMOO traffic ("zoomed in") depending on the server utilization.

The differences between the heuristics can be seen more clearly in Figure 4.9. Here we see the delay ratios for D/M/1 traffic. The difference of the delay ratios between Sorted Arrival-Bursts respectively Minimized GPS Set (Random Fill up of M), which behave approximately the same, and the other two heuristics can be seen here again. Again, the Sorted Arrival-Bursts heuristic is not considered further for the arrival types for which $\sigma_A = 0$ applies (holds for all arrival types considered in the thesis except discrete-time MMOO traffic), since it is a special case of Minimized GPS Set (Random Fill up of M). This can be interpreted in the same way as the results from Figure 4.4 above.

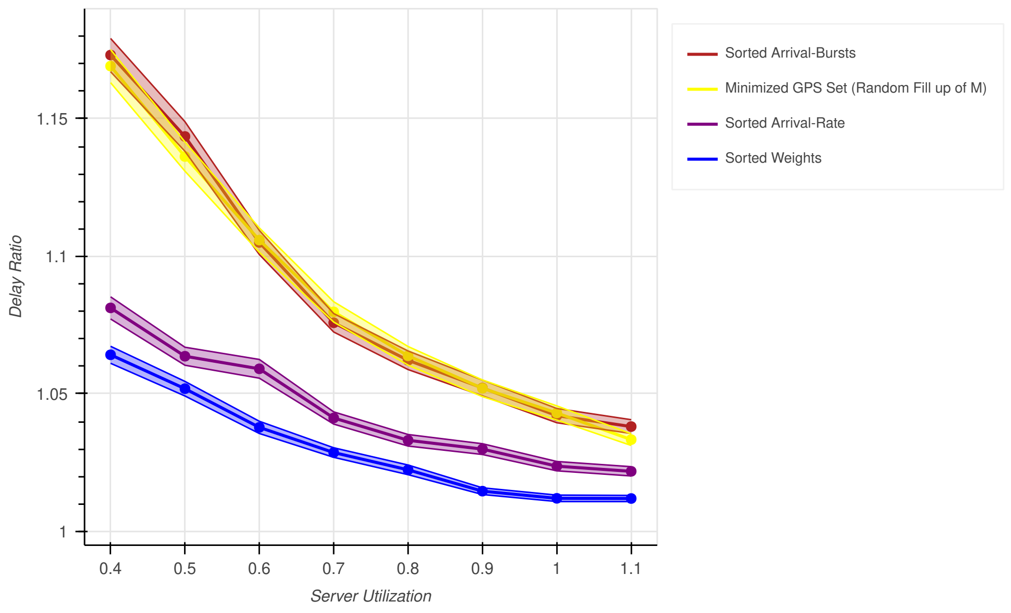
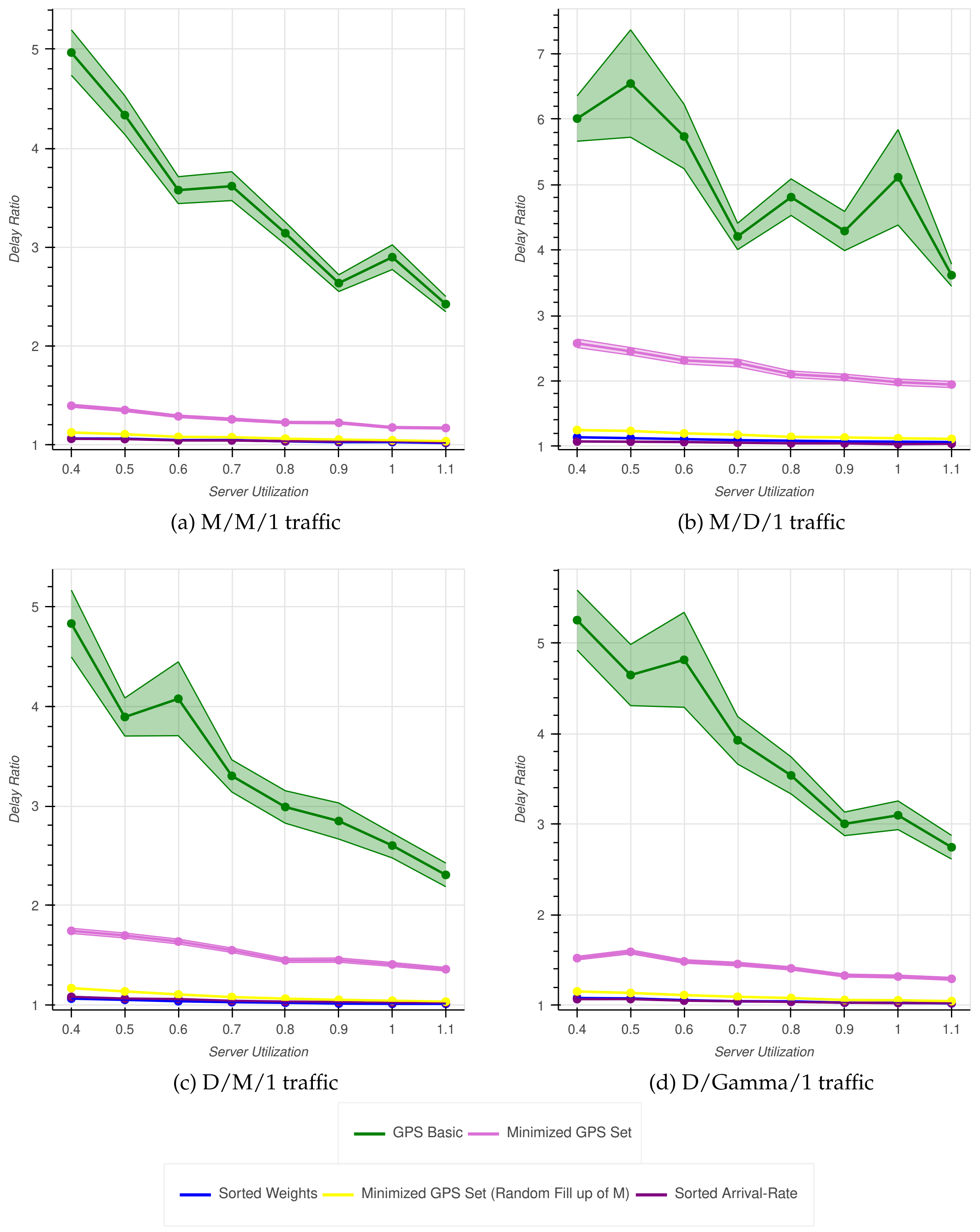


Figure 4.9: Graph of the delay ratio of D/M/1 traffic ("zoomed in") depending on the server utilization.

The results are also confirmed when looking at the other arrival types in Figure 4.10 and Figure 4.11, since analogous behavior can be seen in each case. Differences that can also be seen here from Figure 4.7 to Figure 4.10 and Figure 4.11, as well as the inaccuracies of the individual data can be explained analogously to the case with dependency on the number of flows in Subsection 4.3.1.



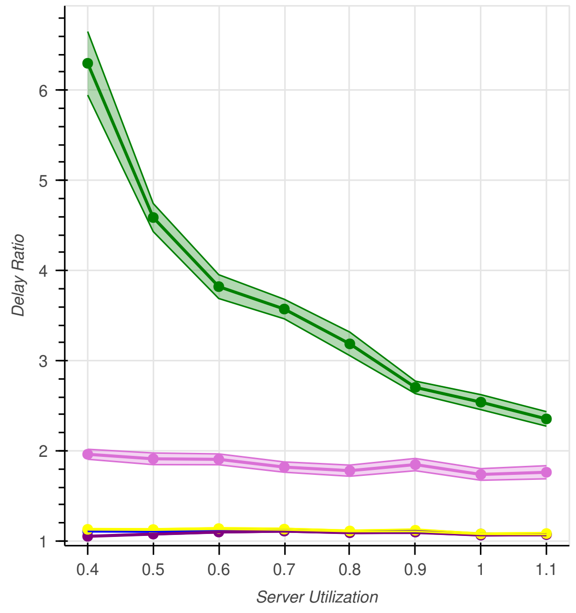
(a) M/M/1 traffic

(b) M/D/1 traffic

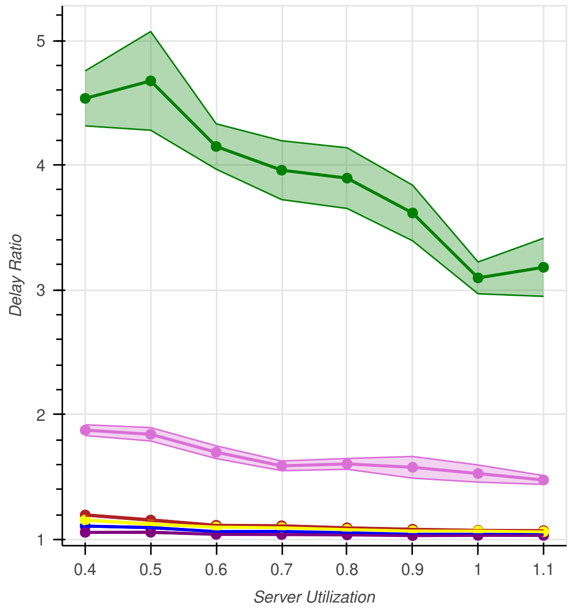
(c) D/M/1 traffic

(d) D/Gamma/1 traffic

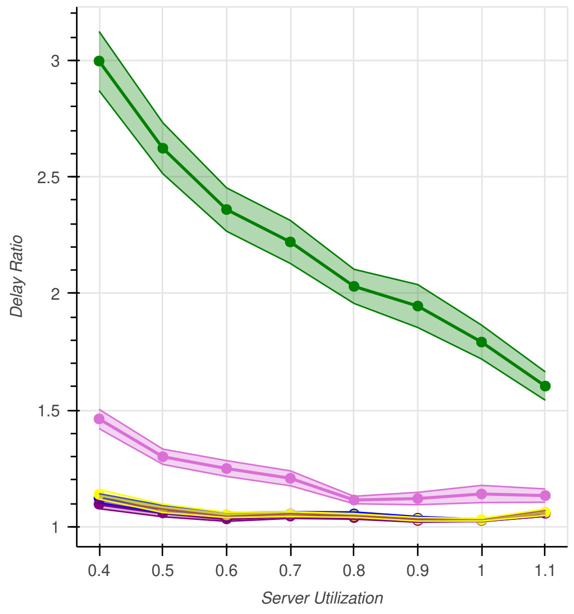
Figure 4.10: Graphs of the delay ratio of M/M/1, M/D/1, D/M/1 and D/Gamma/1 traffic depending on the server utilization.



(a) D/Weibull/1 traffic



(b) Mixed traffic



(c) Aggregated discrete-time MMOO traffic (with $n = 5$)

🟩 GPS Basic 🟪 Minimized GPS Set 🟥 Sorted Arrival-Bursts

🟦 Sorted Weights 🟨 Minimized GPS Set (Random Fill up of M) 🟪 Sorted Arrival-Rate

Figure 4.11: Graphs of the delay ratio of D/Weibull/1 and Mixed traffic depending on the server utilization.

In summary, for the investigation of the delay ratios in dependence of the server utilization, it can be said that the conclusions of the investigations in dependence of the number of flows in Subsection 4.3.1 have been confirmed.

4.3.2 Investigation of different GPS Weights configurations

In the previous considerations, we have chosen uniformly distributed random values for the weights ϕ for the GPS scheduling. But what happens if we do not generate them randomly, but set them all to one fixed value? We end up in a special case called "uniform processor sharing" [PG93]. It should be noted that the concrete value of the weights does not matter, since only the ratio between the weights would be crucial (see Equation (2.3)).

Now, we take a look at the following scenario depending on the number of flows. We assume a server utilization of 60% for discrete-time MMOO arrivals. For each data point the average of 1250 delay ratios (see Equation (4.1)) is given. I.e., in comparison to the observation from above for Figure 4.2, only the weights ϕ changes, which are now equal for all flows. Moreover, the number of calculated delay ratios for the average per data point is reduced, but this does not affect the basic result.

That means, in the following we can compare the graph in Figure 4.12 to the graph in Figure 4.2 above.

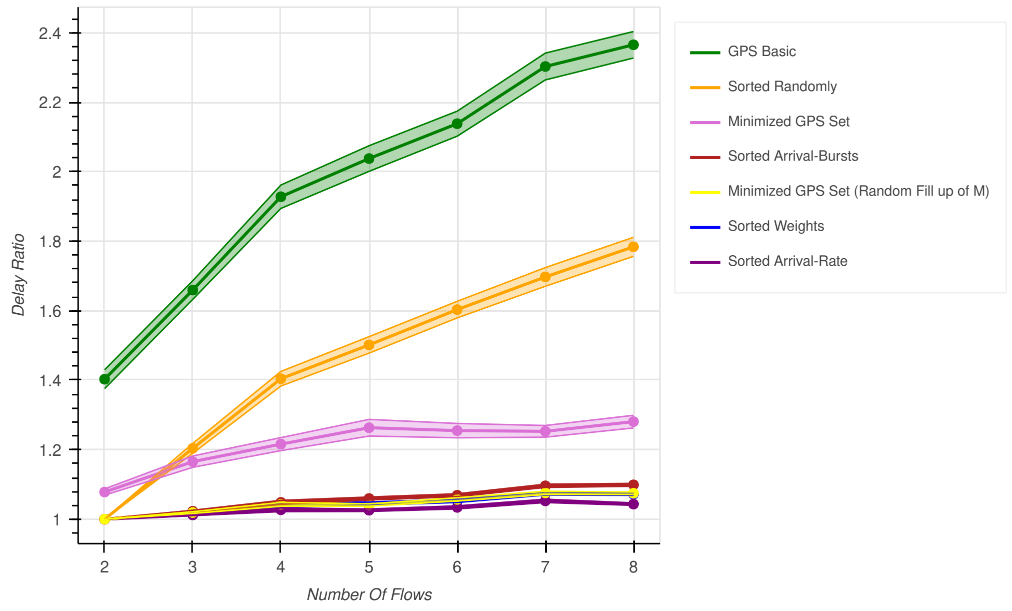


Figure 4.12: Graph of the delay ratio of discrete-time MMOO traffic depending on the number of flows with equal weights for GPS scheduling.

We still see that the delay ratio of GPS Basic and the heuristics increases with the number of flows. Also, we can consider that the "basic order" of the delay ratios has not changed, i.e., that GPS Basic provides higher average delay ratios than Sorted Randomly. Sorted Randomly offers higher delay ratios than Minimized GPS Set, which in turn provides higher average delay ratios than the other four heuristics. In addition, all further observations and interpretations of those can be adopted from above for Figure 4.2. So we could assume that in these aspects the choice of the values of the weights does not matter. However, the delay ratios for all heuristics and GPS Basic are significantly lower, which can be seen at first sight. With GPS Basic, the value of the delay ratio is even only about $\frac{1}{3}$ compared to the value in Figure 4.2 for uniform distributed random value weights ϕ. Thus, the ratio between the heuristics and GPS Basic is reduced from about 5.5 times higher to about 2.2 times higher for 8 flows. How can this be explained?

For uniformly distributed random values for the weights ϕ for the GPS scheduling, it is possible that ϕ_1 for the foi is significantly smaller in relation to other weights. Let us consider the meaning of this in the example of GPS Basic, where all flows are elements of the set M, since it has the most influence there, which is also suggested by the factor of improvement of the delay ratio.

If all flows are elements of the set M, this means that all flows are handled by the GPS scheduling. If the weight ϕ_1 of the foi is relatively low to some other weights, then the other flows have more service available (compare with Subsection 2.2.1) and accordingly the calculated delay bounds for the foi do not have to be close to the minimum upper delay bound [PG93]. If there is no longer a difference between the weights, then it generally leads to more accurate bounds.

Taking a look at the delay ratios of the four heuristics in Figure 4.13, we could see that the distances of the delay ratios between the heuristics seem to be increased slightly compared to Figure 4.3 for uniform distributed random value weights ϕ from above. However, this is a consequence of the scaling of the y-axis, as can be seen on closer inspection of the relations to each other.

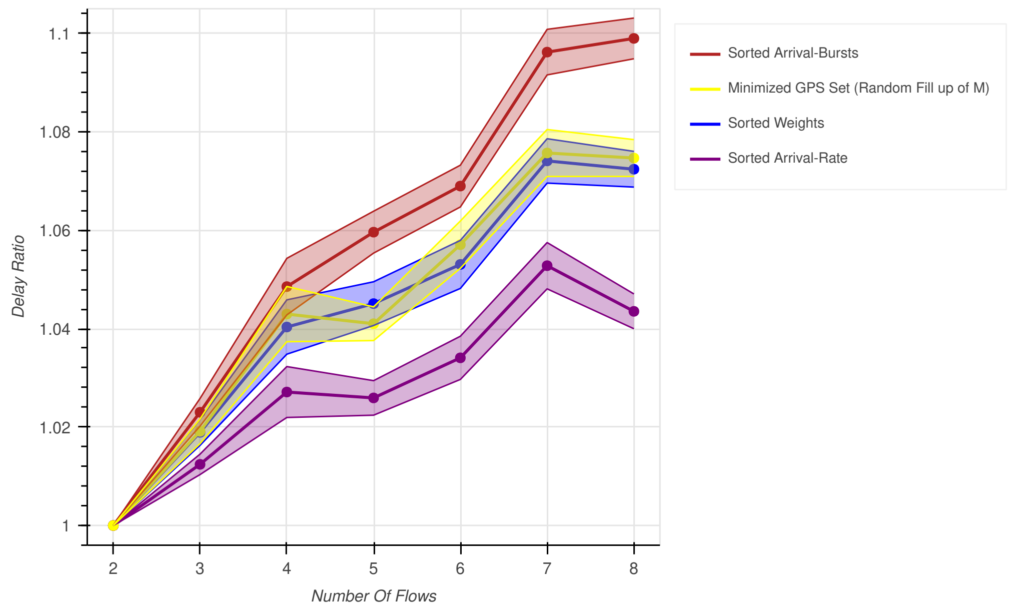


Figure 4.13: Graph of the delay ratio of discrete-time MMOO traffic ("zoomed in") depending on the number of flows with equal weights for GPS scheduling.

To sum up, we can state that even for equal weights, the accuracy of the bounds do not change in comparison to each other. And the key findings from the last Subsection 4.3.1 can be confirmed for this scenario too.

4.3.3 Example: failed approach for the investigation of the heuristics

Within the scope of the thesis other approaches for the investigation of the heuristics were pursued. These did not generate data that could have been used constructively. In the following an example is briefly touched upon. Thus it was an attempt to analyze the heuristics by studying the behavior for discrete-time MMOO traffic with upper bounded burst size σ. I.e., having arrivals generated with uniform distributed random parameters that cannot have bursts σ larger than 5, 10, 15 etc.. The idea behind the observation was to see if the sorted arrival bursts heuristic can provide more accurate delay bounds than the other heuristics, which could not be confirmed in this way.

4.4 Runtime investigation

Before looking at the results of the runtime measurements, we first consider the theoretical runtimes using the Bachmann-Landau notation [Lan00, p.31].

4.4.1 Theoretical view

Let n be the number of flows without the foi. In order to make theoretical observations about the runtimes of the heuristics and the benchmarks, we first need to look at the runtime of the individual subprocesses of the respective heuristics or benchmarks. Processes that run in constant time, such as optimization in θ, which is in $\mathcal{O}(1)$, will not be listed specifically.

We start with the *stability condition preprocessing*. For this we run once through the respective flows and do arithmetic operations in constant time, i.e., the stability condition preprocessing is in

$$\mathcal{O}(n) \cdot \mathcal{O}(1) = \mathcal{O}(n).$$

For the sorting made in the specific heuristics, we assume according to [Wir75] for a suitable *sorting algorithm* a runtime of

$$\mathcal{O}(n \log(n)).$$

Since the remaining operations run in constant time, we only consider the stability condition preprocessing, the runtime for sorting and the runtime for the calculation of Performance bounds for different subsets M (which can be derived from the knowledge from above) for the total time complexity.

Exhaustive Search

Stability condition preprocessing: -
Sorting: -
Calculation of Performance bounds for different subsets M: $\mathcal{O}(2^n)$

Runtime: $\mathcal{O}(2^n)$

GPS Basic

Stability condition preprocessing: -
Sorting: -
Calculation of Performance bounds for different subsets M: $\mathcal{O}(1)$

Runtime: $\mathcal{O}(1)$

Sorted Randomly

Stability condition preprocessing: -
Sorting: -
Calculation of Performance bounds for different subsets M: $\mathcal{O}(n)$

Runtime: $\mathcal{O}(n)$

Sorted Weights, Arrival-Rates and Arrival-Bursts

Stability condition preprocessing: $\mathcal{O}(n)$
Sorting: $\mathcal{O}(n\log(n))$
Calculation of Performance bounds for different subsets M: $\mathcal{O}(n)$

Runtime: $\mathcal{O}(n) + \mathcal{O}(n\log(n)) + \mathcal{O}(n) = \mathcal{O}(n\log(n))$

Minimized GPS Set

Stability condition preprocessing: $\mathcal{O}(n)$
Sorting: -
Calculation of Performance bounds for different subsets M: $\mathcal{O}(n)$

Runtime: $\mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$

Minimized GPS Set (Random Fill up of M)

Stability condition preprocessing: $\mathcal{O}(n)$
Sorting: -
Calculation of Performance bounds for different subsets M: $\mathcal{O}(n)$

Runtime: $\mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$

4.4.2 Runtime measurements

After analyzing the runtime theoretically, we will take a look at the runtime in practice. For this we used the "time" module provided by the Python Standard Library [VRD09]. Only the actual time of the process for the execution of the calculations was measured during the entire measurements.

The runtime measurements were performed on the following system.

• Processor	:	Intel Core i5-8600, 6C/6T, 3.10-4.30GHz
• Graphic card	:	Gainward GeForce GTX 1060 Phoenix GS, 6GB GDDR5
• RAM	:	G.Skill RipJaws V, 16GB, DDR4-3200
• Mainboard	:	MSI Z370-A Pro
• Hard disk	:	Western Digital, 1TB, WD10EXEX
• Operating system	:	Microsoft Windows 10 Pro, Version 10.0.19043

For the runtime measurements, we consider all cases of different theoretical runtimes and select one representative each for runtime in $\mathcal{O}(2^n)$, $\mathcal{O}(1)$, $\mathcal{O}(n\log(n))$ and $\mathcal{O}(n)$ from the section above. For this we select Exhaustive Search and GPS Basic, of course. In addition we consider Sorted Arrival-Rates and Minimized GPS Set (Random Fill up of M). For the measurements, the time for the calculation of one delay bound was considered in each case, according to the number of flows. Also, M/M/1 traffic, a server utilization of 60%, and uniformed random values for the weights ϕ for the GPS scheduling were used.

We take a look at the results of the measurements in Figure 4.14. We can observe the following:

- *Exhaustive Search:* An exponential increase of the runtime can be seen here. Already with 10 flows, the runtime is roughly 23 times higher compared to the heuristics considered. Of course, it is obvious that this ratio grows with increasing number of flows.
- *GPS Basic:* For GPS Basic we have a constant runtime of 0.015625s, independent of the number of flows.
- For the heuristics *Sorted Arrival-Rates* and *Minimized GPS Set (Random Fill up of M)*, we see that the runtime is not significantly different. This could be due to the fact that the sample size is not large and the theoretical runtime of $\mathcal{O}(n\log(n))$ is not very influential yet. Another explanation could be that in practice the worst-case runtimes for the sorting do not always occur and in the best case one also has a runtime of $\mathcal{O}(n)$ [Wir75]. This can also reduce the runtime differences in some cases.

The interpretations of these results of the measurements can be derived right from the respective definitions in the chapters above.

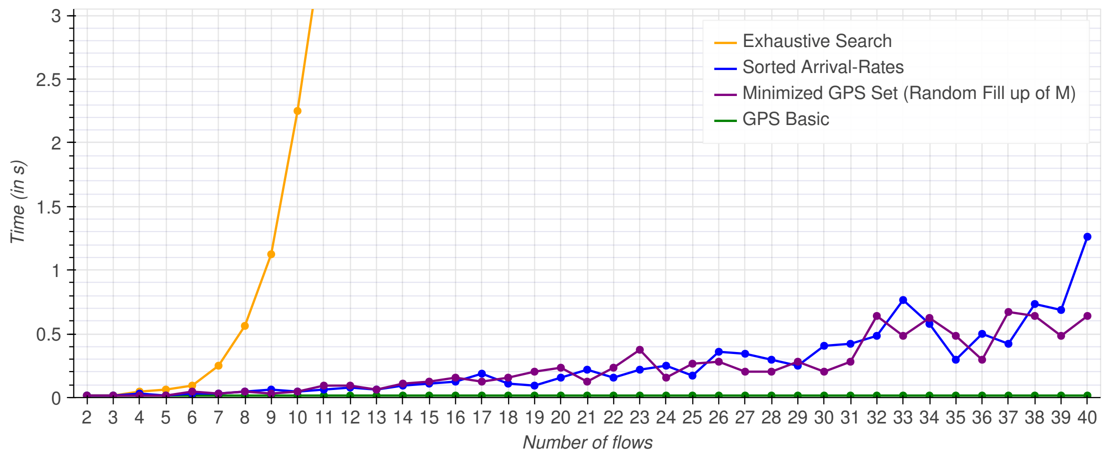


Figure 4.14: Runtime Measurements for Exhaustive Search, GPS Basic, Sorted Arrival-Rates and Minimized GPS Set (Random Fill up of M) for up to 40 flows.

The measurements of Figure 4.14 were continued for a higher number of flows. The results are shown in Figure 4.15, in which we consider the runtime of the delay bound calculation for up to 250 flows. In this graph, the exponential growth of the runtime of Exhaustive Search can also be observed for further measurements. Additionally, the runtime for GPS Basic remains constant at 0.015625s. A significant difference of the runtimes between the heuristics Sorted Arrival-Rates and Minimized GPS Set (Random Fill up of M) is not to be recognized also with this number of the flows.

From this it can be concluded that the runtimes of the heuristics considered above are all approximately the same, at least for a number of flows up to 250. Beyond that, of course, no information can be given.

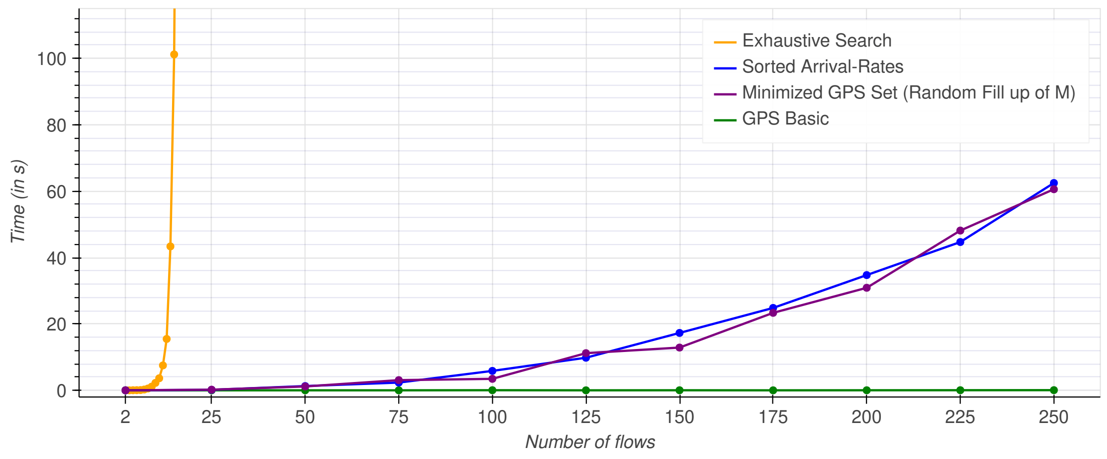


Figure 4.15: Runtime Measurements for Exhaustive Search, GPS Basic, Sorted Arrival-Rates and Minimized GPS Set (Random Fill up of M) for up to 250 flows.

It is also interesting to measure the time it takes for different arrival types to calculate a certain number of delay bounds. I.e., we want to see if it matters for the runtime for which arrival type we calculate delay bounds. Using the example of calculating 2000 delay bounds each with a server utilization of 60%, 7 flows and uniformly distributed random values for the weights ϕ for GPS scheduling. We consider the runtimes of the different arrival types in Table 4.1.

It should be noted that the runtime for discrete-time MMOO arrivals was measured by calculating 750 delay bounds for 6 flows only, in contrast to the other arrival types. This reduction was made since the runtime would have been significantly longer otherwise.

Considering Table 4.1, it can be seen that the runtime for discrete-time MMOO arrivals is the highest, although significantly fewer delay bounds were calculated for one flow less. This could be due to the fact that the stability condition is violated more often and no delay bound could be calculated. The consequence is that random arrivals and the random weight list for GPS have to be generated repeatedly for which the stability conditions have to be checked over and over again. This costs a lot of time with such a high number of repetitions, which is also clearly reflected in the measurements. These stability condition violations occur with every arrival type, it just happens to some types more than to others. This could also explain the differences in the runtimes that we can observe in Table 4.1. Therefore, we can assume that the runtime is dependent on the arrival type used.

Table 4.1: Runtime measurements for different arrival types.

Arrival Type	**Time (in min)**
Disc.-time MMOO	48.61
D/Weibull/1	46.75
Mixed Arrival Types	33.91
M/D/1	33.16
D/M/1	25.79
D/Gamma/1	25.48
M/M/1	25.02

5 Conclusion and Future Work

In this thesis we have presented several heuristics for the still open problem of optimal and efficient choice of flows that should or should not be GPS scheduled for the approach of computing performance bounds by Chang in [Cha00]. These heuristics were evaluated with respect to the accuracy of the calculated performance bounds and the runtimes.

The numerical evaluations demonstrated that we can compute quite accurate delay bounds using the heuristics in comparison to the delay bounds obtained by exhaustive search. However, for this loss in accuracy, we were able to significantly reduce the runtime from exponential for exhaustive search to linear for the heuristics. Although we do not achieve the constant runtime of the approach in [PG93], we calculate significantly more accurate delay bounds in comparison. We could also demonstrate that the essential step of the heuristics is to check the stability condition of the individual flows and the following fill up of the flows to the set of GPS scheduled flows could be done randomly. Thus, we could decide that a flow is GPS scheduled if it does not fulfill the stability condition. The following steps of the heuristics could also cause small differences depending on the considered arrival types, but they were not as crucial as the stability condition preprocessing.

Since these heuristics have already yielded promising results, future work could address further investigations of them. For example, other network topologies could be considered, such as a tandem of two servers, two concatenated servers, or even completely different network topologies. The use of other performance metrics could also be investigated, since only delay bounds were considered in this thesis. Another interesting approach would be the use of other arrival types, especially those that have bursts $\sigma_A > 0$, as in discrete-time MMOO traffic.

Bibliography

[AB05] Douglas G Altman and J Martin Bland. Standard deviations and standard errors. *Bmj*, 331(7521):903, 2005.

[BBLC18] Anne Bouillard, Marc Boyer, and Euriell Le Corronc. *Deterministic Network Calculus: From Theory to Practical Implementation*. John Wiley & Sons, 2018.

[BCOQ92] François Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. Synchronization and linearity: an algebra for discrete event systems. 1992.

[BCPV96] Sanjoy K Baruah, Neil K Cohen, C Greg Plaxton, and Donald A Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, 1996.

[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.

[Cha00] Cheng-Shang Chang. *Performance guarantees in communication networks*. Springer Science & Business Media, 2000.

[Ciu07] Florin Ciucu. Network calculus delay bounds in queueing networks with exact solutions. In *International Teletraffic Congress*, pages 495–506. Springer, 2007.

[Fid06] Markus Fidler. An end-to-end probabilistic network calculus with moment generating functions. In *200614th IEEE International Workshop on Quality of Service*, pages 261–270. IEEE, 2006.

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

[FR15] Markus Fidler and Amr Rizk. A guide to the stochastic network calculus. *IEEE Communications Surveys Tutorials*, 17(1):92–105, 2015.

[HW06] Canberra Hyatt and Patrick Weaver. A brief history of scheduling. *Melbourne, Australia: Mosaic Project Services Pty Ltd*, 2006.

[Lan00] Edmund Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*, volume 1. , 2000.

[LBH09] Tong Li, Dan Baumberger, and Scott Hahn. Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin. *ACM Sigplan Notices*, 44(4):65–74, 2009.

[Nik21] Paul Nikolaus. snc-mgf-toolbox. Website, 2021. Online available at `https://github.com/paulnikolaus/snc-mgf-toolbox`; accessed December 13, 2021.

[NS20] Paul Nikolaus and Jens Schmitt. Improving delay bounds in the stochastic network calculus by using less stochastic inequalities. In *Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools*, pages 96–103, 2020.

[PG93] Abhay K Parekh and Robert G Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM transactions on networking*, 1(3):344–357, 1993.

[PG94] Abhay K Parekh and Robert G Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM transactions on networking*, 2(2):137–150, 1994.

[SN20] Jens Schmitt and Paul Nikolaus. Lecture notes in stochastic analysis of distributed systems, 2020.

[VRD09] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[Wir75] Niklaus Wirth. *Algorithmen und Datenstrukturen*, volume 2. Springer, 1975.