Challenges for Modelling of Software-based Packet Processing in Commodity-Hardware using Queueing Theory

Christian Thieme
Advisor: Daniel Raumer
Seminar Innovative Internet Technologies and Mobile Communications
Chair of Network Architectures and Services SS2017
Departments of Informatics, Technical University of Munich
Email: christian.thieme@tum.de

ABSTRACT
In the past few years, a trend of softwarization in the networking world has emerged. Software defined networking and the virtualization of networking functions, which can be realized in off-the-shelf hardware, help network operators to meet the new challenges that arise from nowadays traffic demands. The demand on routing traffic can partly be traced back to the rise of a digital society and hypes like internet of things, streaming platforms and others.
In order to encounter these challenges software defined networking and virtualized networking functions are required to be implemented efficiently on commodity hardware, so they can offer a high quality of service, while also saving expenses for hardware and energy consumption. These and other reasons motivate modelling the system of commodity hardware devices.
This paper introduces the basics of Queueing Theory Models and then identify challenges which arise from modelling commodity hardware. Furthermore, recent literature is surveyed in respect to how or whether the proposed models map hardware and software properties.

Keywords
Queueing Theory, NFV, network function virtualization, Model, SDN, software defined networking

1. INTRODUCTION
The internet has led to the creation of a digital society, where almost everything is connected and accessibly from anywhere [11]. This effect has also been intensified by the rise of the internet of things. The demand on processing traffic has therefore risen significantly. To meet those demands while saving costs (e.g. costs for hardware, software and energy consumption) and staying environmentally aware, new approaches to meet these demands have been developed.

One step to engage these challenges is software-defined networking (SDN). With this new paradigm, the control plane and the data plane are decoupled and are more bundled within single networking devices. In SDN the control plane decides how to handle network traffic. While the data plane, which forwards traffic according to decisions made by the control plane, can be leveraged and managed in a simpler way (i.e. network switches become simple forwarding devices). [11]

At the same time, a complementary trend has emerged: network function virtualization (NFV). In NFV networking functions like packet-forwarding, routing, applying firewall rules etc. are decoupled from their physical devices. Furthermore, NFV has the ability to facilitate the new deployment of new services with increased agility and faster time-to-value [15].

Both paradigms, namely SDN and VNF, offer the opportunity to be flexible and considerably cheap compared to dedicated hardware. Both approaches allow well-designed networking-systems to meet and balance demands, while staying cost-efficient and maintaining a high quality of service (e.g. for an ISP). In order to develop, optimize and understand certain behaviours of the new systems, modelling is crucial. "Modeling is an important and useful approach for performance evaluation and system validation and it can provide prediction and comparison of design alternatives" [1].

The focus of this paper lies on queueing theory models and is structured as follows: an introduction to the basics of queueing theory is presented in chapter 2. Chapter 2 also provides insight on how a device (i.e. a router) can be mapped to a queueing theory model. Furthermore, Chapter 2 provides reasons and properties, why queueing theory can be an appropriate tool to model networking systems or devices. Chapter 3 lists some challenges and real-world effects that might come with modelling commodity hardware computers. Chapter 4 covers recent literature, in which queueing models have been developed and surveys them regarding the collected challenges from chapter 3. Chapter 5 concludes this paper and summarizes the results of this work.

2. QUEUEING THEORY
"Queueing theory deals with one of the most unpleasant experiences of life, waiting. Queueing is quite common in many fields, for example, in telephone exchange, in a supermarket, at a petrol station, at computer systems, etc." [18]

It was mentioned first by A.K. Erlang at the beginning of the 20th century, when he studied the behaviour of telephone networks [5, 6]. Erlangs works inspired engineers and mathematicians to deal with queueing problems using probabilistic methods [18]. And now, over a hundred years later, hundreds of articles and books have been published and queueing theory is still a widely-used modelling technique.

Queueing theory is used to compute mean values and predictions of a system. This chapter is mostly inspired by the books of Kobayashi et al [10] and Sztrik [18].

2.1 Basic Queueing Theory

To characterize a queueing system, the probabilistic properties of the incoming flow of requests, service times and service disciplines have to be identified [18]. The time between incoming requests is also called interarrival time. In the following the term customer is used to describe an entity that is being served or processed by a service or server [10]. The interarrival time A(t) is usually given as a stochastic distribution, which can, in example, express the probability of customers arriving to our system within a specific time t.

$$A(t) = P(\text{Customer enters system} < t)$$

There is also the property of service time. The service time B(x) is a probabilistic distribution, and denotes, for how long a request is served. Following the example from above, this distribution expresses the probability of a customer being served within time x.

$$B(x) = P(\text{Customer is served} < x)$$

The interarrival times and service times are commonly supposed to be independent random variables [18]. The service discipline describes in which manner the requests are being processed. Some commonly used service strategies are:

FIFO: First In First Out
The first to arrive at the queue, is served first.

LIFO: Last In First Out
The last to enter the queue, is served first.

PS: Processor Sharing
Entities arriving at the Service are served immediately and receive service in the following way: C/n. Where C equals the total servicing capacity of the server and n is the number of entities. This is an idealization of the Round-Robin scheduling scheme. [10]

Priority: Some other priority, i.e. Random
Enqueued entities are being processed according to some other priority.

Queueing Systems can have multiple service units offering the same service that serve customers. The number of service units within a queueing system is denoted as m.

Finally, the capacity of the system and the population size of the queueing system can be identified. The capacity K of the system represents the maximum number of customers within the system, including the ones being served [18]. The population size n basically describes whether the source of incoming customers is limited by a variable, number or ∞.

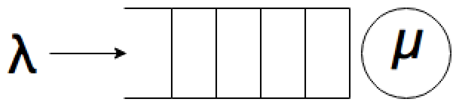

Figure 1: A typical depiction of a single queue, with arrival rate λ and a single service node with service rate μ.

Figure 1 shows the usual visual representation of a simple single queueing system. Customer arrive at arrival rate λ and are enqueued into the waiting area. The customers are served at rate μ.

2.2 Kendall's Notation

In 1953 Kendall [9] introduced a notation to describe a queueing system, which is commonly used nowadays. He denoted the system by

$$A \,/\, B \,/\, m \,/\, K \,/\, n \,/\, D$$

where

A: distrubtion function of the interarrival times,

B: distribution function of the service times,

m: number of service units (offering the same service),

K: capacity of the system,

n: size of the source, and

D: service discipline.

Throughout literature, the notation A/B/m is used when the capacity of the system and the size of the source are assumed to be ∞ and the service discipline is FIFO. Commonly used options for A and B are:

M: A Poisson arrival distribution (an exponential distribution) or an exponential service time distribution. This usually denotes a Markovian process, which possesses the memoryless property. The memoryless property means, regardless of how long ago a customer arrived, the probability of a customer arriving now stays the same. [10]

D: A deterministic or constant value.

G: A general distribution with a known mean and variance formulas. [10]

Example 1: M/M/1/∞/∞/FIFO is abbreviated to M/M/1, denoting a system with poison arrivals, exponentially distributed service times and a single service unit.

Example 2: M/G/m denotes an m-server system with Poisson arrivals and generally distributed service times.
Example 3: M^x/G/M denotes the same system as in Example 2, but additionally defines that there are x sources from which customers can arrive with the same arrival distribution.
Example 4: M/G/m-S denotes the same system as in Example 2, but additionally defines the queue length S.
Example 5: M/M/r/K/n describes a system with a finite-source of n elements, where they stay for an exponentially distributed time and the service times are exponentially distributed. Furthermore the service is carried out according to the request?s arrival by r servers, and the system capacity is K. [18]

More examples, explanations and theory can be found in [18] and [10] or any other book on queueing theory.

2.3 Properties

From the queueing system the following properties can be derived:

- The arrival rate of customers (denoted by λ) which can also be referred to as birth rate.
- The service rate (denoted by ţ) which can also be referred to as death rate.
- The server utilization, throughput and response time.
- The number of customer in the system at time t.
- The probability of n costumers in the system at time t.
- The length of an idle or a busy period.
- The traffic intensity (usually denoted as $\rho = \lambda \ / \ \mu$), which represents the expected number of arrivals during the service of a customer.

Furthermore, there are some more properties that can be calculated only if the the system is in steady-state. This is the case, if the system is not overloaded, e.g. when requests arrive faster than the system can processes them. The steady-state is also called equilibrium. "A queue is defined as stable if $\mu > \lambda$ or $\rho > 1$. When $\rho > 1$, requests entering the queue accumulate faster than they can be served and, in theory, latency increases to infinity. In practice, the system saturates, limiting the request rate to be $\lambda = \mu$, (or $\rho = 1$) and stabilizes the queue with high latency." [16]

Using theorems and formulas, the following properties of a M/M/1, M/M/n or M/M/∞ queueing system can be derived if the system is in steady state:

- The mean number of customers in the queue.
- The mean number of customers in the system.
- The mean total waiting time in the system.

3. CHALLENGES IN MODELLING X86 DEVICES USING QUEUEING THEORY

This chapter lists and discusses some features that can come with commodity hardware. Software effects are also considered. Each of the features is explained and if necessary an example is given. Then an it is pointed out, how the features could be taken into account by a queueing theory model. At the end, there is also an example given, how a commodity hardware computer could be modelled as a queueing system.

3.1 Multi-Core

One of the main reasons, why off-the-shelf hardware is able to compete with dedicated hardware is because of the ability to process data in parallel due to multiple central processing units (CPUs). Cores or threads are mapped to service units in every queueing model and therefore represented by the number of service units. The CPU is therefore modelled as a queueing system.

Furthermore the CPU-frequency is modelled and represented by the service time distribution. The frequency usually has the biggest impact on the service time distribution.

3.2 Bus

Data from any NIC usually has to be transferred to memory or cache, so it can be accessed and processed by a processing unit. Direct Memory Access (DMA) is a way to transfer data from a device via a bus system to the main memory without involving the CPU. This provides an efficient way of moving data into the main memory for processing. [17]

A step further than DMA is the direct cache access (DCA), which basically does the same as DMA, just that the data is directly passed to a CPU cache. Though, these techniques are not available on every off-the-shelf hardware. In any case, a bus system moves the data into the cache or main memory.

The bus system, which transfers the data to the cache or main memory can be modelled as a single queue. Therefore, the queueing system of the bus and the queueing system of the CPU can be modelled as a queueing network [1].

3.3 Number of NICs

Using commodity hardware, systems have to be equipped with Network Interface Cards (NICs) in order to receive and send packets. To fully utilize the system, as many NICs as supported should be installed. As shown by Runge et al. in [19], the number of NICs can have a significant impact on the performance of a networking device.

As NICs are part of a computer, their output is an input to the next queueing system within the computer (e.g. a bus or CPU). A NIC itself can be modelled as a single queueing system. The input from NICs can be implicitly modelled by adding an exponent to the interarrival distribution of the next queueing system entity in the queueing network (e.g. M^x/M/1).

[1]When multiple queues are plugged together, e.g. the output of a queue is the input of the next queue, then this is called a queueing network.

3.4 Memory and Software Latencies

There are several latency effects that can arise from memory and software. In the following some effects which can introduce latencies are listed:

- moving data between cache hierarchies (L1, L2, L3 and main memory)
- interrupts, which can be triggered by hardware (e.g. a packet arrives at a NIC and need to be stored in main memory) or software (e.g. a process with higher priority forces a context switch)
- resource contention (e.g. threads are fighting over CPU time and force context switches)
- queue implementation (e.g. a queue must be implemented in a way so that parallel processing, retrieving and inserting of data is efficient, which is also pointed out by Orozco et al. in [16])

These latencies have an impact on the time a packet is spending in the system. One could model these effects by adding a term or constant value to the service time distribution. Even though, the listed effects are usually neglected for simplicity by the models proposed in literature.

3.5 Finite Memory

Any physical device has limited resources regarding space (caches, buffers and main memory) available. And since they represent queue sizes (and capacity of the system) they should be modelled and not assumed to be limitless. A infinite or infinite buffer has a direct impact on properties like the waiting time for a packet in the system. Though, most proposed models neglect this fact. The capacities of the system are assumed to be infinite.
When queueing models in networking assume infinite buffers, they can become vulnerable to attacks (i.e. Denial of Service Attacks) as shown by [3].

3.6 Batch-Processing

In computer hardware, it is common to load and transfer batches of data in order to hide loading latencies. I.e. a NIC does not instantly transfers a packet on its arrival, but waits until a certain time has passed or the batch is filled up. Also CPUs tend to load blocks of data into higher cache hierarchies at once for processing.This behavior can influence the performance of a system significantly. Consequently, batch processing has an impact on the time a packet stays within the system and the service time of a packet. Batch processing is sometimes modelled by recent literature.

3.7 Packet Size

When dealing with traffic in the internet, it is very likely to encounter packets of different sizes. I.e. when processing a big packet (or a batch of big packets), the total number of packets processed is comparatively lower, compared to processing small packets. On the other hand, the throughput (in Mbit) is higher that way.

Therefore, the size of different packet sizes should be considered in the service time of the queueing model. This also has an impact on the time the packet spends in the system. The packet size is usually not considered in recent models.

3.8 Example

In general, commodity hardware can be modelled as a tandem queueing network [14]. A tandem queueing network is a subclass of queueing networks. In case of a tandem queueing network, customers, who arrive at the system, can encounter multiple queue-service pairs, until they leave the system again. E.g. this means that after a packet has entered a queue and has been serviced, it enters yet another queue to a service and so on, until it finally leaves the system again. It should also be noted, that customers cannot skip one or multiple queues.

In literature, a computer system is modelled either as a single queue, usually representing a bottleneck, or as a network of queues. In Figure 2, an example of computer modelled as a queueing network is shown. The figure shows the packet flow $\lambda+$ through the computer. λ represents the incoming

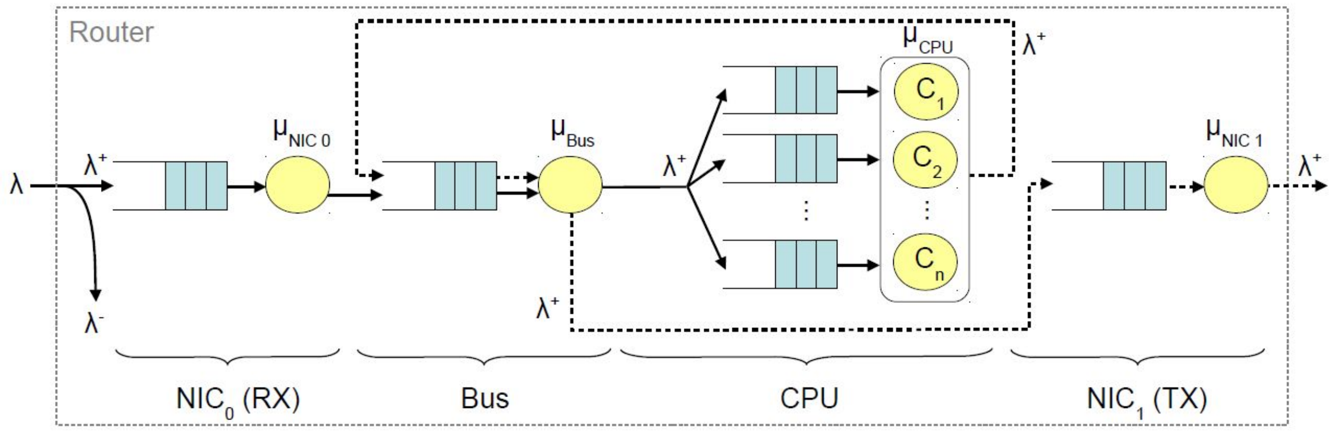

Figure 2: A Router Model of [14] which shows, the flow of packets through a computer. The NICs, bus and CPU are each modelled as a queueing system, forming a queueing network.

traffic which is split into $\lambda+$ (packets that enter the system) and $\lambda-$ (packets that are dropped at the NIC due to overload). Solid arrow depict the packet flow towards the CPU, while dashed arrows depict the flow of processed (outgoing) traffic. Mayer et al [14] show this figure, but finally model only the CPU as an M/M/n queue.

A possible alternative for the modelling of Figure 2 could be as follows: The system can be modelled as a queueing network, starting with an M/M/1 queue, which represents the NIC that receives the incoming traffic. Packets leaving the NIC are then enqueued into another M/M/1 queue for the bus system. The CPU can be modelled either as n * M/M/1 queues or as M/M/n queue, where n is the number of CPU cores or threads. Finally, packets leaving the system must pass through the outgoing NIC, which again could be modelled as an M/M/1 queue. The here proposed queueing network results in a jackson network [2].

4. QUEUEING THEORY MODELS IN X86 INTERCONNECT DEVICES

In this chapter, some recent literature is presented and surveyed in respect to whether the proposed models consider the aspects presented in chapter 3, or not. The focus is on research, which uses commodity hardware. At the end of this chapter a comprehensive overview within Table 1.

Meyer et al. [14] propose a tandem queuing model to model a software router. This model is then simplified to the extend where the model basically only includes the packet size and multiple service units (CPU cores) and the services times. This is done in order to measure the performance at the bottleneck, namely the CPU. Their goal is to measure and predict the maximum throughput of a general software router.
The authors do not provide a notation for their queueing system, but given from the information in the text, it boils down to a M/M/n queue for the entire router. In the formulas, the packet size and multi-cores are taken into account. Other effects are neglected or ignored for simplicity.

In the work of Jemaa et al. [2] a model for VNF placement and provisioning optimization strategies over an edge-central carrier cloud infrastructure is introduced, which takes QoS requirements into account. Since the paper is dedicated to spawning VNFs in cloud environments, it could be considered off-topic. Nonetheless, the reason this paper was taken in, was because the paper includes the delay, which introduced by virtualization layer of a virtual machine (VM) in the cloud.
The authors model each VNF as a single M/M/1 queue. The VNFs are hosted in a VM, which is therefore modelled as a jackson network. Packets that are transferred between two VMs are modelled as a M/GI/$infty$ queue which acts like a bus. Arriving packets at the queue are instantly processed and sent to the next VM. In the formulas the latencies of spawning a VNF is taken into account.

An energy-aware resource allocation scheme to manage virtual machines, dedicated to perform certain virtualized network functions is proposed by Bruschi et al. in [4]. The authors M^x/M/1/SET model assumes batch arrivals with a constant packet size and take this into account in their formulas. They also model multiple packet arrival sources and a wake up time for servers that have been idle.

Gebert et al. [7] propose a general analytical model for generic virtualized network functions running on commodity hardware. They model NICs as a external queues which send incoming traffic into one GI^x/GI/1-L queue, which represents the CPU, using batch-processing. A packet in the batch arriving at the CPU queue is dropped if the packet has stayed in queue for too long. Packets are also rejected, when they arrive and would have to wait longer than L-1 packets need for processing.

Jarschel et al. [12] derive a basic model for the forwarding speed and blocking probability of an OpenFlow switch combined with an OpenFlow controller. In their model, they assume a queue of infinite size, which collects incoming traffic. That traffic is then forwarded and passed into a queue with limited buffer size (denoted by the parameter S in their queueing system).

In [8] Mahmood et al. extends the case in which a controller in a OpenFlow network is only responsible for a single node in the data plane to multiple nodes. They approximate the case as an open Jackson network and provide formal proof for their approximation for the case of infinite buffer and finite buffer sizes.

[2]A jackson network is a special case of queueing networks where all external arrivals into the queueing system are a Poisson Process, service times are exponentially distributed, and the utilization of all queues is less than one [18]

Table 1: Summary of the surveyed papers in respect to whether they take the challenges of chapter 3 into account, or not.

	3.1 Multi-Core	3.2 Bus	3.3 Number of NICs	3.4 Memory & Software Latencies	3.5 Finite Memory	3.6 Batch Processing	3.7 Packet Size	Queueing System(s)
[14]	yes	no	no	no	no	no	yes	M/M/n
[2]	yes & no	yes	no	yes	no	no	no	N * M/M/1, M/GI/∞
[4]	no	no	no	yes	no	yes	yes	M^x/G/1/SET
[7]	no	no	yes	no	no	yes	yes	GI^x/GI/1-L
[12]	no	no	no	no	yes & no	no	no	M/M/1, M/M/1-S
[8]	no	no	no	no	yes & no	no	no	M/M/1

As can be seen in table1 the models in recent literature do not take all challenges listed in Chapter 3 into account. In most papers the packet-processing system was modelled as a single queue. This might be an sufficient assumption as long as the CPU stays the bottleneck of such systems.

5. CONCLUSION

This paper introduced basic queueing theory and provided challenges on how such a model can be applied to a commodity hardware computer. Queueing theory has proved to be a suitable model to predict long-term and average behaviours.

Judging from the validation of the proposed models that have been conducted in the presented papers, the models seem to be approximating the real world systems sufficiently. Which is impressive, given the fact that most of the models only take some of the challenges listed in Chapter 3 into account. This on the other hand, just shows how powerful this modelling technique is.

It general, is also reassuring to see that research can already provide queueing theory models for recent advances in the softwarization hype, e.g. towards flexible and scalable networking using SDN and VNF.

6. REFERENCES

[1] S. Balsamo and A. Marin. Performance engineering with product-form models. In S. Kounev, V. Cortellessa, R. Mirandola, and D. Lilja, editors, *ICPE'11*, page 437, [New York], 2011. ACM.

[2] F. Ben Jemaa, G. Pujolle, and M. Pariente. Analytical models for qos-driven vnf placement and provisioning in wireless carrier cloud. In M. . C. COMMITTEE, editor, *MSWIM 16 19TH INTERNATIONAL CONFERENCE ON MODELING, ANALYSIS AND SIMULATION OF WIRELESS AND... MOBILE SYSTEMS*, pages 148–155, [S.l.], 2016. ACM.

[3] D. S. Berger, M. Karsten, and J. Schmitt. On the relevance of adversarial queueing theory in practice. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):343–354, 2014.

[4] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo. Joint power scaling of processing resources and consolidation of virtual network functions. In *Proceedings, 2016 5th IEEE International Conference on Cloud Networking*, pages 70–75, Los Alamitos, California, 2016. IEEE Computer Society, Conference Publishing Services.

[5] A. Erlang. The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B 20*, B 20:33–39, 1909.

[6] A. Erlang. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *The Post Office Electrical Engineers' Journal*, 10:189–197, 1918.

[7] S. Gebert, T. Zinner, S. Lange, C. Schwartz, and P. Tran-Gia. Performance modeling of softwarized network functions using discrete-time analysis. In *2016 28th International Teletraffic Congress (ITC 28)*, pages 234–242. IEEE, 2016.

[8] M. Jarschel, O. Østerbø, A. Chilwan, and K. Mahmood. Modelling of openflow-based software-defined networks: The multiple node case. *IET Networks*, 4(5):278–284, 2015.

[9] D. G. Kendall. Some problems in the theory of queues. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(2):151–185, 1951.

[10] H. Kobayashi, B. L. Mark, H. M. Kobayashi, and analysis. *System modeling and analysis: Foundations of system performance evaluation / Hisashi Kobayashi, Brian L. Mark*. Pearson Education Ltd, London, pearson international ed. edition, 2009.

[11] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[12] M. Jarschel and S. Oechsner and D. Schlosser and R. Pries and S. Goll and P. Tran-Gia, editor. *2011 23rd International Teletraffic Congress (ITC): Modeling and performance evaluation of an OpenFlow architecture*, 2011.

[13] T. M. Runge, B. E. Wolfinger, S. Heckm?uller, and A. Abdollahpouri. A modeling approach for resource management in resource-constrained nodes. *Journal of Networks*, 10(01), 2015.

[14] T. Meyer, F. Wohlfart, D. Raumer, B. E. Wolfinger, and G. Carle. Validated model-based performance prediction of multi-core software routers. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 37(2), 2014.

[15] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. de Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[16] D. Orozco, E. Garcia, R. Khan, K. Livingston, and G. R. Gao. Toward high-throughput algorithms on many-core architectures. *ACM Transactions on Architecture and Code Optimization*, 8(4):1–21, 2012.

[17] A. Rubini and J. Corbet. *Linux device drivers: Chapter 13: mmap and DMA - Direct Memory Access and Bus Mastering*. O'Reilly & Associates, Sebastopol, 2nd ed. edition, 2001.

[18] J. Sztrik. *Basic Queueing Theory: Foundations of System Performance Modeling, pages 7-15*. GlobeEdit, Saarbrücken, 1. auflage edition, 2016.

[19] Torsten M. Runge and Bernd E. Wolfinger. How do multiple network cards influence the software router performance?