A Taxonomy of Performance Models of Network Interconnect Devices

Joline Bui
Supervisor: Daniel Raumer
Seminar Future Internet SS2017
Chair of Network Architectures and Services
Departement of Informatics, Technical University of Munich
Email: Joline.Bui@tum.de

ABSTRACT

A major objective of computer network performance and optimization models is to reduce energy consumption as much as possible, thus lowering costs for electricity and the CO_2 footprint while still offering a high service quality to the user. To this extent, the study of queueing packets within a network, as well as resource modeling, workload modeling, and task scheduling are considered suitable to explain the basics of some network performance models. The presented models are compared to each other in terms of their performance measurement aspects. Furthermore, the comparison of several modeling techniques shows that the selection of a model or a model technique depends on the expected results, parametrized constraints, and the execution in real world systems.

Keywords

Queueing Theory, Network Calculus, Resource Modeling, Workload Modeling, Task Scheduling Problem, List Scheduling, Dynamic Programming, Linear Programming, Integer Programming, Simplex Method, Branch and Bound.

1. INTRODUCTION

The research and study of quality of service guarantees has been motivated by the increasing demand in transmitting multimedia and other real time applications over the Internet [25]. *Packet switching* and *circuit switching* are two networking methods for transferring data between two nodes or hosts. In packet-switched networks the route is not exclusively determined. Routing algorithms are used, which allows many users to share the same data path in the network. This type of communication between sender and receiver is called *connectionless* [39]. On the other hand, in circuit-switched networks the route is setup and established prior to initializing connections to the host. Circuit-switched networks are used in telephone systems [49]. In contrast, most traffic over the Internet uses *packet switching* because the Internet is basically a connectionless network [39] and because it performs better than circuit switching in terms of average delay and buffer requirements [2]. For the communication channels in the network, queues are formed inside the switching nodes, such as routers, bridges, and switches [33]. Many aspects significantly influence the performance within many networks and they can be modeled with various performance models.

In this paper, the focus is on three major fields of computer networks: The first field deals with the study of queueing of packets and its performance models. The second field covers *workload modeling* and the third field is concerned with the *task scheduling* if many processors are available in a network node for parallel programming. Hence, taxonomies of performance models of network interconnect devices are presented. These give instructions under which conditions which models should be selected. First, section 2 gives insights into the *queueing theory* and queueing systems. Then section 3 compares two performance model techniques which deal with queueing type problems encountered in computer networks: *queueing theory* and *network calculus*. Section 4 covers *resource modeling* and subsequently *workload modeling* is presented in section 5. Finally, model techniques for schedule optimization and task allocation within computer networks are presented in section 6.

2. QUEUEING THEORY

2.1 Background Information

The origin of *queueing theory* can be traced back to the early 1900s when A. K. Erlang, a Danish engineer, applied this theory extensively to study the behaviour of telephone networks [41]. In a network node, tasks are queued for CPU in various stages of their processing [41]. Queueing systems are dynamic and contain a flow of network data. There are two different types of flows [29]:

1. ***Steady flow***: These systems contain exact numbers of constraints and do not depend on time. For example, there is a network of channels, each with a channel capacity. To analyze those systems, graph theory, combinatorial mathematics, optimization theory, mathematical programming and heuristic programming are used to deal with scheduling problem and task allocation. This will be explained in more detail in section 6.
2. ***Unsteady flow***: These systems do not have exact numbers of constraints and must be predicted. Stochastic tools are often used in this context and will be described further in section 3.

Queueing theory can be clustered into three major components: input process, system structure and output process [41]. These components are shown in Figure 1. The input process, also called the arrival process, deals with three aspects: the size of the arriving population, the arriving pattern of the customers and the behaviour of the arriving customers. The system structure is concerned with the systems resources, which is described by the physical number and layout of servers, as well as constraints like the system capacity. The output process, also called the departure

doi: 10.2313/NET-2017-09-1_01

process, is characterized by the type of queueing discipline and the service-time distribution. These characteristics are demonstrated in Figure 2 as they contribute to modeled performance aspects.

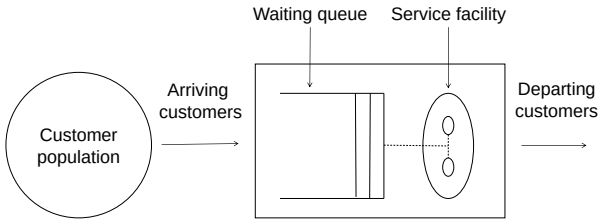

Figure 1: Schematic diagram of a queueing system [41]

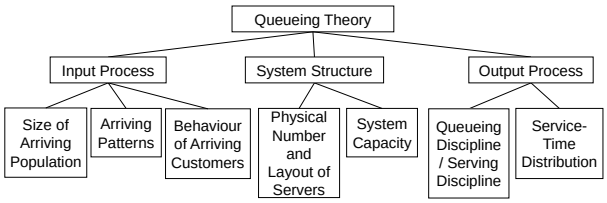

Figure 2: Components and characteristics of queueing theory [41]

The following components, derived from the characteristics presented in Figure 2, can be calculated with various mathematical tools [41]:

- average number of customers in the system
- distribution of number of customers in the system
- average waiting time of a customer in the system
- distribution of waiting time
- length of busy period
- length of an idle period
- current work backlog expressed in units of time

These components represent performance measurement aspects that vary in quantity in different queueing systems.

2.2 Kendall Notation

Queueing systems can have various compilations of different elements. They are described as shorthand notations, known as the *Kendall notation*, which was developed by David G. Kendall, a British statistician, to describe a queueing system containing a single waiting queue which has been further extented [28]. The following notations apply:

A / B / X / Y / Z

A : Customer arriving pattern (inter-arrival-time distribution)
B : Service pattern (service-time distribution)
X : Number of parallel servers
Y : System capacity
Z : Queueing discipline

Example: The classical queue $M/M/1/\infty/FCFS$, but known as $M/M/1$, represents a queuing system where customers arrive according to *Poisson process* and request exponentially distributed service times from the server. The system has only one server, an infinite waiting queue and customers are served on a First Come First Serve (FCFS) basis. If only the first three parameters are shown, the default values for the last two parameters are Y = ∞ and Z = FCFS.

2.3 Queueing Systems

Queueing systems are clustered into two groups of systems, which have many variations of further queueing characteristics within them [41]:

Markovian queueing system: A markovian queueing system is characterized by a *Poisson* arrival process and exponentially distributed service times. Both the arrival and service process are *memoryless*.

Semi-markovian queueing system: Semi-markovian queueing systems refer to those queueing systems in which either the arrival or service process is not *memoryless*. They include $M/G/1$, $G/M/1$, their priority variants and the multi-server counterparts.

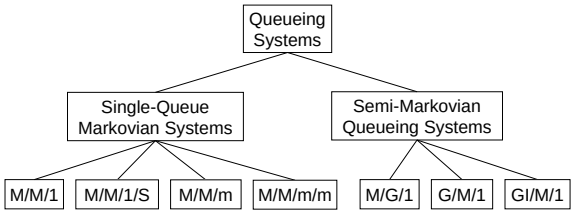

Figure 3: Components and characteristics of queueing theory [41]

Figure 3 gives an overview of the two groups of systems and basic types of queueing systems, whereas $M/M/1$ is referred to as the classical queueing system. For detailed explanation of the other queueing systems that are presented in Figure 3, [41] gives clear definitions and demarcations. Queueing networks are collections of interactive queueing systems, whereby departures of some queues enter some other queues. This can happen deterministically or probabilistically. From the network topology point of view, queueing networks can be categorized into two classes [41]:

Open queueing networks: In this queueing network, customers arrive from external sources outside the domain of interest, go through several queues, or even revisit a certain queue more than once and finally leave the system. Open queueing networks are good models for analyzing circuit-switching and packet-switching data networks.

Closed queueing networks: In this queueing network, customers do not arrive at or depart from the system. There is a constant number of customers that behave within the network like in the open queueing networks. Other than open queueing networks, closed queueing networks are good models for analyzing window-type network flow controls as well as CPU task scheduling problems.

3. QUEUEING THEORY COMPARED TO NETWORK CALCULUS

Generally, *queueing theory* considers the average quantities in an equilibrium state and the traffic that enters a queueing system (or queueing network) is always characterized by a stochastic process, such as the *Poisson* distri-

doi: 10.2313/NET-2017-09-1_01

bution. However, unique customer and service characteristics and requirements in such networks as the Internet often make the application difficult to analyze [26]. Beginning with [15], it has been shown several times that the *Poisson* distribution is not necessarily a realistic assumption for Internet traffic [43]. In contrast, in network calculus the arrival traffic is assumed to be *unknown* as long as it satisfies the following regularity constraints: "The amount of work brought along by the arriving customers within a time interval is less than a value that depends on the length of that interval" [41]. *Network calculus* is a new paradigm of queueing analysis, which is part of the deterministic queueing [35]. This paradigm is able to put deterministic bounds on network performance measures. This new methodology was pioneered by Rene L. Cruz [10] for analyzing delay and buffering requirements of network elements and was further developed by Jean-Yves Boudec and Thiran [35]. The main difference between these two is that *queueing theory* allows us to make statistical predictions of performance measures whereas *network calculus* establishes deterministic bounds on performance guarantees. The idea of *network calculus* is that service guarantees can be achieved by regulating the traffic and deterministic scheduling. Analog to the classical *queueing theory*, a system is classified into the same major components as classical queueing theory (see Figure 1). The input, mostly referred to as an arrival curve, is an abstraction of the traffic regulation, and the transfer function, mostly referred to as a service curve, is an abstraction of the scheduling [43]. The difference with classical *queueing theory* is that non-traditional algebra, such as *min-plus-algebra* and *max-plus-algebra*, is used to transform complex network systems into analytically tractable systems, as well as focusing on the worst case [26]. *Network calculus* has developed into two branches:

Deterministic network calculus: R. L. Cruz introduced in [10] the idea of using a function to deterministically upper-bound the cumulative arrival process. For arrival modeling he developed the arrival curve model, which specifies a traffic envelope to arrivals. In contrast, for server modeling a function to deterministically lower-bound the cumulative service process is used [44],[26]. However, the service curve model evolved and out of it emerged the idea to compare the actual departure time with a virtual time function and to use the difference together with the rate parameter to define the *virtual clock scheduling* algorithm [57],[26]. A server model, called *guaranteed rate* was developed in [18],[26]. Based on these concepts, the derivation of worst-case performance bounds including backlog and delay can be calculated [14].

Stochastic network calculus: Latest techniques try to bring stochastic approaches into *network calculus* [3]. The arrival and server models of stochastic *network calculus* can be considered as probabilistic extension or counterparts of the deterministic *network calculus* [26]. The approach's goal is to comprehend statistical multiplexing and scheduling of non-trivial traffic sources in a framework for end-to-end analysis of multi-node networks [14]. Fidler stated in [14] that compared to classical *queueing theory*, the stochastic *network calculus* comprises a much larger variety of stochastic processes, including long range dependent, self-similar [47], and heavy-tailed traffic [36].

The backlog as a function of time is shown as the vertical marked as b(t) in Figure 4. It is the amount of bits that is held inside the system. The virtual delay at time t is the

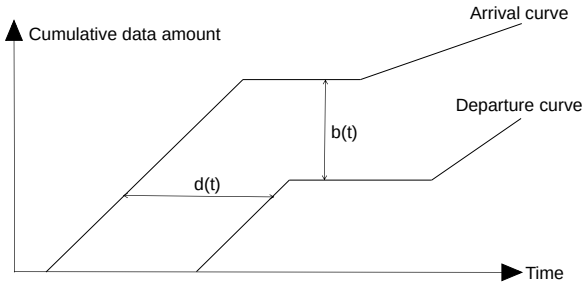

Figure 4: Example of an input and an output function with horizontal and vertical deviations, that demonstrate delay and backlog in network calculus theory [35, Page 5]

delay that would be experienced by a bit arriving at time t if all bits received before it are served before it. It is the horizontal deviation, marked as d(t) in Figure 4 [34].

4. RESOURCE MODELING

In this section, further performance models describe resources of which limitations can be analyzed. Packet processing systems make extensive use of resources like cashes, memory controllers, buses and NICs [12]. In [16] the performance of the packet processing application built for high-speed IO frameworks is modeled. As resources have limits and bottlenecks, in [16] four distinctive characteristics that limit the performance of the packet processing are listed:

1. The maximum transfer rate of the used NICs
2. The link capacity, such as the capacity of a PCI express, which is used to connect the NICs to the rest of the system
3. The restriction of the possible network bandwidth in a RAM
4. The maximum processing load on the CPU, which can be kept low due to modern offloading features of NICs

For these characteristics, the high-performance prediction model can be used to provide upper bounds for the capabilities of a software-based packet processing system [16]. According to Rizzo [48], packet processing costs can be divided into per-byte and per-packet costs, whereas the latter has a deeper impact on IO frameworks. This model assumes that if the limit of the NIC is reached and a constant load per packet times the number of packets is lower than the available CPU cycles, the cycles are spent waiting for new packets in the busy wait loop. By adding these costs, a new value of the constant cost per packet is calculated [16]. The constant cost per packet consists of the sum of the cost used for sending and receiving packets (c_{IO}), the cost of the complexity of the task (c_{task}) and the cost that is introduced by the busy wait on sending or receiving packets (c_{busy}). The available CPU cycles are the main limiting factor of software packet processing. The throughput of a packet processing application heavily depends on the amount of CPU cycles available for its processing task. According to [16], this amount is influenced by the following factors for real world applications:

doi: 10.2313/NET-2017-09-1_01

- the complexity of packet processing
- the time the CPU spends waiting for data to arrive in cache
- the effect of different batch sizes

These factors can be determined precisely by various measurement methods. Experiments in [16] show following performance characteristics predicted by this model: Further measurements demonstrate that this model can be applied to estimate processing tasks, which can be approximated with a constant average load. A possible use case for this model is to evaluate the eligibility of PC systems for specific packet processing tasks. In addition, the results illustrate the trade-off between throughput and latency with different queue sizes. The larger the batch sizes, the bigger the throughput but also the higher the average latency, even though performance increases. The reason is, that packets spend more time queued the larger the batch sizes are. However, batch sizes also have a lower bound due to overloading frameworks. Hence, smaller batch sizes are more suitable for applications that are sensitive to high latency, whereas larger batch sizes are more suitable for application where raw performance is critical.

5. WORKLOAD MODELING

As the resource model describes how to deal with resource limits, workload modeling gives insights about characteristics of the load, which have an impact on resource modeling as well as on task scheduling.
"The workload behavior of the system can be defined as the disposition of resource usage over a period of time. The disposition allows us to characterize the behavior of the system during that period of time [23]."

As shown in Figure 5, workload modeling can be classi-

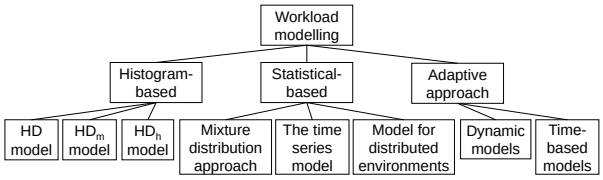

Figure 5: Taxonomy of workload modelling [23],[4],[46]

fied in three different branches: models that are based on histograms, models that are statistical-based, and adaptive approaches. Workloads can be visualized as histograms [23].

The *histogram model* [53], [50], was introduced by Skelly to predict buffer occupancy and loss rate for multiplexed streams. The loss rate and network delay distribution can be obtained using the buffer occupancy distribution [23]. In [23] three traffic models are proposed:

1. *HD* ***model*** [53],[50]: This model has only one histogram for a given sample period and is short-range dependent. It uses an analysis method based on a M/D/1/N queueing system. Compared to the other models and to the real traffic, this model reduces the number of cells. Although, the inaccuracy increases with the number of cells, the model is simple and compact and the results are still accurate. Therefore, it is a good approximation.

2. HD_m ***model*** [38]: This method is based on the modification of the Mean Value Analysis (MVA) algorithm. It is based on several histograms using different time scales. The disadvantage of this model is that they resolve each histogram's groups independently. Therefore, no dependencies between the histograms groups are taken into account. Furthermore, the selection of the number and values of the sample periods can be one problem. The model can be too complex with too many sample periods. It is more accurate but is not as compact as the *HD model*. But compared to MVA, the HD_m loss curve is more precise.

3. HD_h ***model*** [22]: This model is based on a *Hurst parameter* and is long-range dependent. The underlying idea is to modify the variance of the histogram depending on the *Hurst parameter*. In addition, this model needs only one histogram.

The results of the experiment in [23] showed that HD_m has the best results. However, this approach can be cumbersome. Therefore, the best approach is the HD_h *model*, which is compact and very precise.

In general, loss ratio and node delay are traffic quality of service parameters that are obtained. So, the probability that a packet is delayed is higher than a certain value can be predicted. For example, in application areas like video and audio transmission, the delay histogram can be useful in the end nodes to adapt their transmissions rates or to configure the buffer in the reception nodes. Further application areas are e.g., admission control, traffic provisioning, and network configuration [23].

The main objective of workload models is the optimal provisioning of network resources, so service cost of network transmission can be reduced. According to [4], "the workload is considered to be only that processing which is specifically required to satisfy the user request".

In [4], three models are presented, which are used to characterize the system workload and are constructed in a statistical framework: Therefore, any user request R made to a distributed computing system may be characterized by the quadruple (T, L, X, F): T = time, L = location, X = amount of service requested, F = flag.

1. ***A mixture distribution approach*** (A): This model takes only X and F into account and ignores the effects of time and spatial distribution of the requests. The advantage of this formulation is that a very complex probability distribution can be represented in terms of several simpler distributions. When no reasonable assumptions about the conditional probability distributions can be made, clustering is a viable approach for construction of probabilistic models of workload. All clustering techniques assume that the items to be grouped are specified in terms of certain parameters. Therefore, a workload characterization study using the clustering approach can be effected significantly by judicious choice of such parameters [4].

Stochastic process workload models (B, C): The goal is to create a validated probabilistic workload model first, and then to use that model to construct a test workload [4].

2. ***The time series model*** (B): This model takes the time characteristics of the requests into account and ignores the source of the requests. Time series analysis requires data be collected over a long period of time. For example, the number of jobs processed per day may be adequate to analyze the load on a system over a year. To study the time-

doi: 10.2313/NET-2017-09-1_01

dependent nature of steps in the job a markovian model can be used [4].

3. ***Model for distributed environments*** (C): The requests of this model are characterized by the quadruple. If location is allowed to have only a few discrete values, then this model may be considered to be a multivariate point process over time and space. In an actual distributed computing environment, more complex models may be required. The techniques of multivariate point process analysis are likely to be useful. These models may then be used for generating representative test workloads [4].

The third branch of workload modeling in Figure 5 are ***adaptive workload models***:
These are models that can learn and adapt to any environment. There are dynamic models [21] and time-based models [7]. In [46], the adaptive model, Support Vector Regression (SVR) model, predicts the short-term future as a string. The information content in the string conveys as a substring that reflects the future. It has been successfully applied in real environments, such as web traffic generation and virtual path bandwidth management in ATM networks. Therefore, these models could also be applied in areas such as MPEG video stream modeling, intrusion detection, and intelligent prefetching [46].

6. SCHEDULE OPTIMIZATION AND TASK ALLOCATION

In chapter 2, CPU task scheduling was mentioned in terms of *closed queueing networks*. This chapter deals with task scheduling problems in multiprocessor systems. The increasing demand for efficient parallel programming algorithms leads to this issue gaining importance [54]. The *multiprocessor scheduling theory* is concerned with optimally allocating sets of agents to complete a set of tasks over time [54]. Scheduling is a fundamental hard problem (an *NP-hard* optimization problem [31]), "as the time needed to solve it optimally grows exponentially with the number of tasks" [55]. In related works, e.g. [37],[42],[20],[45],[52],[56],[9],[24], good but not optimal scheduling algorithms are presented [55]. Optimal schedules can make a fundamental difference e.g. for time critical systems, or to enable the precise evaluation of scheduling heuristics [55].

When computing an optimal schedule, the system may be trying to achieve one or more of the following objectives [54]:

- minimize the service time of the schedule
- minimize the weighed completion time of each service
- minimize the total number of processors used to solve the problem
- minimize the amount of communication bandwidth used
- if there is a value associated with assigning a task to a particular processor, maximize the aggregate value
- if there is a cost (other than time) associated with assigning a task to a particular processor, minimize the aggregate cost

In general, schedules are constrained by the execution delay of each service, the task release times and precedence relations, as well as communication delays between processors [54]. As many variations of scheduling problems exist, many techniques were proposed to solve them. There are some techniques that assume that processors are a scarce resource, while others suppose the opposite. Some techniques assume that the execution time for each task on each processor is constant [13], while others do not consider the communication delay, which implies that the communication time between processors is zero [8]. Some fundamental scheduling techniques are presented in the Figure 6 to give and overview of the different approaches:

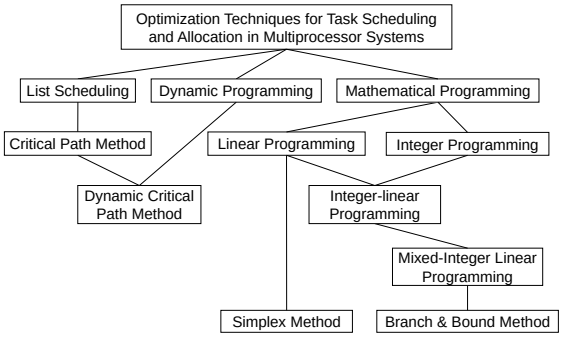

Figure 6: Optimization techniques for task scheduling and allocation in multiprocessor systems [54]

List scheduling works with priority levels by which tasks are ordered in a list. Therefore, the first step is the task prioritizing phase, where a priority is computed and assigned to each task. The second step is the processor selection phase, where the prioritized tasks are assigned to processors that minimizes a suitable cost function. The processor selection phase can be static or dynamic:

Static: The two phases happen sequentially. Hence, the processor selection phase starts after the completion of the task prioritizing phase [5],[54].

Dynamic: The processor selection phase and the task prioritizing phase are interleaved [19],[11].

Most scheduling heuristics algorithms are based on *list scheduling* [5],[6],[8],[54], but they differ mainly in different prioritizations of the tasks. However, the main goal of all *list scheduling* algorithms is to find and place the most critical tasks high on the priority list, so they can be executed quickly [55]. For example, the *critical path* method is a technique, where priorities are assigned to tasks according to how far away they are from the starting point, therefore giving preference to tasks that are likely to be on the *critical path*. This is the longest chain of tasks, where its length is a lower bound on the optimal schedule that can be produced [54]. Ronald Graham, a mathematician also known for his work in *scheduling theory*, first examined *list scheduling*, a stochastic approximation, in detail in 1966 [19]. He looked at a classification that assumes that each processor is identical, therefore execution delay is a function of the task only but no communication delay between the processors is considered [54]. By assuming the latter, Kohler [30] and Sih [51] pointed out, that this can be a problem, because large communication delays between processors can often result in sub-optimal makespan [54]. R. E. Bellmann, the scientist who coined the term *dynamic programming* developed the theory in its initial stages [5]. *Dynamic programming* is a technique that breaks the scheduling problem into simpler sub-problems at different points in time. By getting optimal solutions to the smaller sub-problems, optimal solutions to the bigger scheduling problem can be found [1]. A variation of the introduced *critical path* method is the *dynamic critical path* method [17], that works on following steps [32]:

doi: 10.2313/NET-2017-09-1_01

1. Determine the *critical path* of the task graph and select the next node to be scheduled dynamically.
2. Rearrange the schedule on each processor dynamically. The positions of the nodes in the partial schedules are not fixed until all nodes have been considered.
3. Select a suitable processor for a node by looking ahead the potential starting times of the remaining nodes on that processor.

According to [54] "*dynamic programming* may be used to develop polynomial-time approximation algorithms or optimal solutions for very simple scheduling classifications". However, it is only suitable to solve schedules optimally if it runs in exponential time, which is no better than mixed *integer linear programming* techniques [54]. The third branch in the Figure 6 is *mathematical programming*. This tool solves complex problems such as those that can be modeled as an objective function with a set of mathematical constraints [54]. Generally, the objective is to minimize or maximize a linear function of these variables. *Linear programming* (LP) problems are *mathematical programming* formulations where the objective and each constraint is formulated as a linear function of X_1, X_2,... , X_n. Therefore, an LP-formulation would have the following structure of constraints [54]:

$MIN(orMAX)$:
$c_1X_1 + c_2X_2 + ... + c_nX_n$

$Subject\ to$:
$a_{11}X_1 + a_{12}X_2 + ... + a_{1n}X_n \leq \mathrm{b}_1$
...
$a_{k_1}X_1 + a_{k_2}X_2 + ... + a_{k_n}X_n \geq \mathrm{b}_k$
...
$a_{m_1}X_1 + a_{m_2}X_2 + ... + a_{m_n}X_n = b_m$

One widely used algorithm for solving *linear programming* problems is the *simplex method*. It has been proven to find optimal solutions (if there are any) to all instances of *linear programming* problems. The underlying idea of the algorithm is that there is always a single optimal solution to any *linear programming* problem, which can be found on a corner point of the feasible region [6]. Another method of *mathematical programming* is *integer programming*, in which some or all the variables are restricted to being integer. *Integer linear programming* (ILP) adds the rule, that the objective function and the constraints must be linear. These models are for cases, where the decision variable represents a nonfractional entity such as people or cars, or where a decision variable is needed to model a logical statement. For instance, 5.23 cars cannot be produced - only 5 or 6. Another example is to set integers for binary decisions, such as yes (1) and no (0).

For multiprocessor scheduling with communication delays (MSPCD) optimal ILP solutions are presented in [55] which aim to maximize the utilisation of each available processor and minimize the task schedule length. To achieve this goal, the algorithms need to find where and when each task will be executed, so that the total completion time is minimal. Node costs as required time for task completion, edge costs as communication time between two tasks on different processors as well as data transfer times as communication delays are considered as constraints. A variation of LP-formulations based on diverse logic systems can be derived from this provided information. In [55] the formulation of ILP-RevisedBooleanLogic (ILP-RBL) and ILP-TranssivityClause (ILP-TC) are proposed. While the formulation ILP-RBL reworked the logic for Boolean multiplication, ILP-TC enforces the partial ordering of the processor indices with the help of an additional transivity clause. Both formulation methods manage to reduce the number of variables and constraints by effective linearization of the bilinear equation arising out of communication delays in the MSPCD model. The results of the experimental evaluation show that in the ILP-RBL the formulation runs faster over a small numbers of processors and the linearization in ILP-TC runs faster over a larger number of processors.

Other cases exist where some decision variables must be integers but the remaining decision variables are allowed to be non-integers. In those cases, *mixed integer-linear programming* can be applied [54]. The *branch and bound* technique is used to find those integer solutions [54]. The idea behind *branch and bound* (B&B) is to optimally solve the problem via a relaxation of the problem. A relaxation is a simplification of a problem, which makes the calculation easier.

In [27] a parametrized B&B algorithm for scheduling real-time tasks on a multiprocessor system is presented to minimize the maximum task lateness in the system. Task lateness is defined as the difference between a task's completion time and its deadline. The algorithm uses a search tree that represents the solution space of the problem. Each vertex in the search tree represents one specific task-to-processor assignment with schedule ordering. The optimal solution is presented as one or many vertices (if one exist), where all tasks have been scheduled on the processors. With the aid of intelligent rules for selecting vertices to explore/expand and pruning vertices that do not lead to an optimal solution, the complexity of the search can be reduced. However, not only the choice of a vertex selection rule, e.g. last-in-first-out (LIFO) or least-lower-bound (LLB), but also the determination of a lower-bound funtion and an approximation strategy of the B&B method, have impacts on the optimal solution.

In a maximization problem, the optimal objective function value of an ILP-relaxation is always an upper bound on the optimal integer solution, whereas any integer feasible point found is always a lower bound on the optimal integer solution [6],[54]. The values are used to compare and update the upper and lower bounds at every step to narrow down the solution area, so the optimal integer solution is found [54]. Tompkins stated in [54] that *branch and bound* works better for *mixed-integer-linear programming* problems than listing every possible integer solution. One problem of *branch and bound* is that the running time in some cases may grow exponentially because of the size of the problem. However, he stated that "the closer the linear programming relaxation is to the bound, the less time it takes to find the optimal integer solution, as less division can be made" [54].

7. CONCLUSION AND OUTLOOK

Many different parts within a computer network can be modeled to give information about the network performance. Throughout this paper, many complex models and techniques are presented to obtain specific results to improve network performance. The challenge here is to figure out

what results are expected and how constraints need to be parametrized. Furthermore, models and techniques often do not provide very precise results. Therefore, taxonomies are presented to propose subordinate topics which lead to techniques that match the problem and present their advantages and disadvantages.

In the future, optimization problems within queueing and scheduling in networks will become more relevant due to the increasing amount of data and therefore also increasing data traffic. The long-term objective is to save as much energy as possible, thus reducing the electricity costs and lowering the CO_2 footprint. This leads towards the concept of *Green NFV Infrastructure*, which deals with placing as many virtual network functions as possible on the smallest set of physical servers [40]. Here as well, optimization models, such as mixed-integer optimization, with e.g. the *robust optimization technique*, are used to solve the virtual network function placement problem to optimality [40]. Therefore, more complex performance models that are used in many areas of different layers of a network are required and gain importance.

8. REFERENCES

[1] Dynamic Programming. Available online at `https://www.cs.cmu.edu/~avrim/451f09/lectures/lect1001.pdf`; last accessed on 2017/04/02.

[2] Queueing delay. Available online at `http://www.cs.toronto.edu/~marbach/COURSES/CSC358_S14/delay.pdf`; last accessed on 2017/04/01.

[3] Systemperformanz. Available online at `https://www.net.in.tum.de/pub/systemperformanz/ss2012/skript/networkcalculus.pdf`; last accessed on 2017/04/01.

[4] A. K. Agrawala, J. M. Mohr, and R. M. Bryant. An approach to the workload characterization problem. *Computer*, 9(6):18–32, 1976.

[5] R. Bellman. Dynamic programming (dp). 1957.

[6] S. Bradley, A. Hax, and T. Magnanti. Applied mathematical programming. 1977.

[7] M. Calzarossa and G. Serazzi. A characterization of the variation in time of workload arrival patterns. *IEEE Transactions on Computers*, C-34(2):156–162, Feb 1985.

[8] C. Chekuri and M. Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *Journal of Algorithms*, 41(2):212–224, 2001.

[9] E. G. Coffman and R. L. Graham. Optimal scheduling for two-processor systems. *Acta informatica*, 1(3):200–213, 1972.

[10] R. L. Cruz. A calculus for network delay. i. network elements in isolation. *IEEE Transactions on information theory*, 37(1):114–131, 1991.

[11] O. Deutsch, M. B. Adams, and J. Lepanto. Closed-loop hierarchical control of military air operations. Technical report, DTIC Document, 2002.

[12] M. Dobrescu, K. Argyraki, and S. Ratnasamy. Toward predictable performance in software packet-processing platforms. Technical report, 2012.

[13] D. W. Engels, J. Feldman, D. R. Karger, and M. Ruhl. Parallel processor scheduling with delay constraints. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 577–585. Society for Industrial and Applied Mathematics, 2001.

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

[15] J. L. Florin Ciucu, Almut Burchard. A network service curve approach for the stochastic analysis of networks. *SIGMETRICS Perform. Eval. Rev.*, 2005.

[16] S. Gallenmueller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle. Comparison of frameworks for high-performance packet io. In *Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on*, pages 29–38. IEEE, 2015.

[17] M. J. Gonzalez Jr. Deterministic processor scheduling. *ACM Computing Surveys (CSUR)*, 9(3):173–204, 1977.

[18] P. Goyal, S. S. Lam, and H. M. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 273–284. Springer, 1995.

[19] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Labs Technical Journal*, 45(9):1563–1581, 1966.

[20] T. Hagras and J. Janeček. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. *Parallel Computing*, 31(7):653–670, 2005.

[21] G. Haring. *On state-dependent workload characterization by software resources*, volume 11. ACM, 1982.

[22] O. Hashida, Y. Takahashi, and S. Shimogawa. Switched bath bernoulli process (sbbp) and the discrete-time sbbp/g/1 queue with application to statistical multiplexer performance. *IEEE Journal on Selected Areas in Communications*, 9(3):394–401, 1991.

[23] E. Hernández-Orallo and J. Vila-Carbó. Network performance analysis based on histogram workload models. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS'07. 15th International Symposium on*, pages 209–216. IEEE, 2007.

[24] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee. Scheduling precedence graphs in systems with interprocessor communication times. *SIAM Journal on Computing*, 18(2):244–257, 1989.

[25] Y. Jiang. A basic stochastic network calculus. In *SIGCOMM Comput. Commun. Rev.*, 2006.

[26] Y. Jiang. Network calculus and queueing theory: Two sides of one coin: Invited paper. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 37:1–37:12, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[27] J. Jonsson and K. G. Shin. A parametrized branch-and-bound strategy for scheduling precedence-constrained tasks on a multiprocessor system. In *Parallel Processing, 1997., Proceedings of the 1997 International Conference on*, pages 158–165. IEEE, 1997.

[28] D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[29] L. Kleinrock. Queueing systems, volume i: theory. 1975.

[30] W. H. Kohler. A preliminary evaluation of the critical path method for scheduling tasks on multiprocessor systems. *IEEE Transactions on Computers*, 100(12):1235–1238, 1975.

[31] Y.-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE transactions on parallel and distributed systems*, 7(5):506–521, 1996.

[32] Y.-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE transactions on parallel and distributed systems*, 7(5):506–521, 1996.

[33] S. S. Lam and J. Wong. *Queueing network models of packet switching networks*. Faculty of Mathematics, University of Waterloo, 1981.

[34] J.-Y. Le Boudec and P. Thiran. A short tutorial on network calculus. i. fundamental bounds in communication networks. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 4, pages 93–96. IEEE, 2000.

[35] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.

[36] J. Liebeherr, A. Burchard, and F. Ciucu. Delay bounds in communication networks with heavy-tailed and self-similar traffic. *IEEE Transactions on Information Theory*, 58(2):1010–1024, 2012.

[37] W. Löwe and W. Zimmermann. Scheduling iterative programs onto logp-machine. *Euro-ParâĂŹ99 Parallel Processing*, pages 332–339, 1999.

[38] J. Luthi, S. Majumdar, and G. Haring. Mean value analysis for computer systems with variabilities in workload. In *Computer Performance and Dependability Symposium, 1996., Proceedings of IEEE International*, pages 32–41. IEEE, 1996.

[39] Margaret Rouse. packet-switched. Available online at `http://searchnetworking.techtarget.com/definition/packet-switched`; last accessed on 2017/04/01.

[40] A. Marotta and A. Kassler. A power efficient and robust virtual network functions placement problem. In *Teletraffic Congress (ITC 28), 2016 28th International*, volume 1, pages 331–339. IEEE, 2016.

[41] C.-H. Ng and S. Boon-Hee. *Queueing modelling fundamentals: With applications in communication networks*. John Wiley & Sons, 2008.

[42] A. Palmer and O. Sinnen. Scheduling algorithm based on force directed clustering. In *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on*, pages 311–318. IEEE, 2008.

[43] K. Pandit, J. Schmittt, and R. Steinmetz. Network calculus meets queueing theory-a simulation based approach to bounded queues. In *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on*, pages 114–120. IEEE, 2004.

[44] A. K. Parekh and R. 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.

[45] A. Radulescu and A. J. Van Gemund. Low-cost task scheduling for distributed-memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 13(6):648–658, 2002.

[46] S. Raghavan, N. Swaminathan, and J. Srinivasan. Predicting behavior patterns using adaptive workload models [computer networks]. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999. Proceedings. 7th International Symposium on*, pages 226–233. IEEE, 1999.

[47] A. Rizk and M. Fidler. Non-asymptotic end-to-end performance bounds for networks with long range dependent fbm cross traffic. *Computer Networks*, 56(1):127–141, 2012.

[48] L. Rizzo. Netmap: a novel framework for fast packet i/o. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112, 2012.

[49] M. Rouse. circuit-switched. Available online at `http://searchnetworking.techtarget.com/definition/circuit-switched`; last accessed on 2017/04/01.

[50] N. Schroff and M. Schwartz. Video modeling within networks using deterministic smoothing at the source. In *INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE*, pages 342–349. IEEE, 1994.

[51] G. C. Sih and E. A. Lee. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE transactions on Parallel and Distributed systems*, 4(2):175–187, 1993.

[52] O. Sinnen. *Task scheduling for parallel systems*, volume 60. John Wiley & Sons, 2007.

[53] P. Skelly, M. Schwartz, and S. Dixit. A histogram-based model for video traffic behavior in an atm multiplexer. *IEEE/ACM Transactions on Networking (TON)*, 1(4):446–459, 1993.

[54] M. F. Tompkins. *Optimization techniques for task allocation and scheduling in distributed multi-agent operations*. PhD thesis, Massachusetts Institute of Technology, 2003.

[55] S. Venugopalan and O. Sinnen. Optimal linear programming solutions for multiprocessor scheduling with communication delays. *Algorithms and architectures for parallel processing*, pages 129–138, 2012.

[56] T. Yang and A. Gerasoulis. List scheduling with and without communication delays. *Parallel Computing*, 19(12):1321–1344, 1993.

[57] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *ACM SIGCOMM Computer Communication Review*, volume 20, pages 19–29. ACM, 1990.