The Offered Load in Fork-Join Networks: Calculations and Applications to Service Engineering of Emergency Department.

M.Sc. Research Proposal

Itamar Zaied

Advisors: Prof. Haya Kaspi and Prof. Avishai Mandelbaum

The Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology

March 18, 2010

Contents

1 Introduction

This research proposal deals with Offered Load calculations in various queueing networks, which relate to health care systems. We first recall the Offered Load of the $M_t/GI/N_t$ queue (Subsection 1.1). In Subsection 1.2, we shortly explain the complexity of queueing networks in health care. We elaborate on that topic in Section 2. In Subsection 1.3, we explain, in more details, the significance of the Offered Load for the capacity problem.

Navigating through the Document

For the reader's convenience, this document contains hyper-links that direct the reader to a page relevant to the topic. For example, if we quote a theorem, the reader will have a hyper-link to that theorem. After using the hyper-link, a recommended way to return to the originating page is using the Adobe Acrobat feature "go to previous view", which can be found in the menu under "documents".

1.1 The Offered Load

The first step in understanding the service capacity of time-varying systems is to understand the Offered Load at time t in the $M_t/GI/N_t$ model. Here M_t indicates that the arrival process is assumed to be non-homogeneous Poisson with arrival rate $\lambda(t)\,, t \geqslant 0$; GI indicates that the service times are iid with cdf G. Finally, N_t denotes the number of servers at time t, which can vary over time.

We introduce the $M_t/GI/\infty$ modeling: it differs from the $M_t/GI/N_t$ model only by having infinitely many servers at all times, which means that each customer who joins the system is encountering an idle server and does not need to wait. In our study, we use the $M_t/GI/\infty$ model for several reasons. First, it is remarkably tractable, as will be described later. Second, one can use the $M_t/GI/\infty$ model to analyze the level of required service capacity, namely N_t. Moreover, we can use this model to get an upper bound on the performance that could be achieved if the staffing level was as high as needed. But most important, as explained below, analyzing $M_t/GI/\infty$ yields the "right" definition of the Offered Load for the corresponding $M_t/GI/N_t$.

We now define the Offered Load of the $M_t/GI/N_t$ Model:
To do that, one must introduce formally its corresponding $M_t/GI/\infty$ queue.

Definition 1. *Let $M_t/GI/\infty$ be a model with the same arrival process and the same service time distribution as the $M_t/GI/N_t$ model; let its number of servers be ∞ at all times. Then $M_t/GI/\infty$ is the corresponding $M_t/GI/\infty$ queue of $M_t/GI/N_t$.*

Definition 2. *For the $M_t/GI/N_t$ queue, the* offered load $R = \{R(t),\, t \geqslant 0\}$ *is given by the function $R(t) = E[L(t)]$, where $L(t)$ is the number of customers/patients (number of busy servers) at time t, in the corresponding $M_t/GI/\infty$ queue. The stochastic process $L = \{L(t),\, t \geqslant 0\}$ will be referred to as the* workload *process.*

It is well known (see [2]) that in $M_t/GI/\infty$ the number of busy servers at time t, $L(t)$, has a Poisson distribution with a time-varying mean, $R(t) = E[L(t)]$. The calculation of $R(t)$ will be detailed in Subsection 3.1.

1.2 The Complexity of Hospitals as Service Systems

Hospitals are among the most complex service systems, providing service that must adhere to three often conflicting dimensions: Clinical, i.e. providing the best possible medical care; Operational - matching static capacity (beds) and dynamic capacity (doctors, nurses) with demand, and Financial - controlling the cost of care.
We focus on the operational dimension where, in order to set staffing levels, one must calculate the Offered Load which is, roughly speaking, the work offered to the system for processing. Thus, we seek to calculate the Offered Load in complex networks.
The capacity problem, namely that of matching capacity to demand, is key for providing satisfactory service. Green (pages 15-42 in[4]) describes the general background and issues involved in hospital capacity planning and explains how Operations Research models can be used to provide important insights into operational strategies and practice.

1.3 The Importance of the Offered Load for Solving the Capacity Problem

In our study, we focus on calculating the Offered Load of a single queue within a queueing network. Section 3 elaborates on the topic of Offered Load calculation in general and the Offered Load of a single isolated queue in particular. The Offered Load represents the amount of work that the system has to handle. Thus, calculating the Offered Load of a single queue is important to set the staffing level of it. An example of applying the Offered Load for staffing is the *"square root rule"*. The *"square root rule"* is a rule used in systems with time-homogeneous arrivals, with a large number of agents. The *"square root rule"* commends that the number of agents n be given by $n \approx R+\beta\cdot\sqrt{R}$, where R is the Offered Load and β is a parameter set by the sought after service level. Clearly, in health care one cannot assume time-homogeneous arrivals, nor large number of agents, but that does not mean that the the Offered Load is insignificant in the case of setting health care staffing levels. Indeed, in our study we seek to understand its relevance for staffing health care personal-its advantages and limitations.

2 Preliminary Material

2.1 Fork-Join Networks

A fork-join network consists of a group of service stations, which serve arriving customers simultaneously and sequentially according to pre-designed deterministic precedence constraints. More specifically, one can think in terms of "jobs" arriving to the system over time, with each job consisting of various tasks that are to be executed according to some preceding constraints. The job is completed only after all its tasks have been completed.
The distinguishing features of this model class are the so-called "fork" and "join" constructs. A "fork" occurs whenever several tasks are being processed simultaneously. In the network model, this is represented by a "splitting" of a task into multiple tasks, which are then sent simultaneously to their respective servers. A "join" node, on the other hand, corresponds to a task that may not be initiated until several prerequisite tasks have been completed. Components are joined only if they correspond to the same job; thus a join is always preceded by a fork. If the last stage of an operation consists of multiple tasks, then these tasks regroup (join) into a single task before departing the system.

Example:

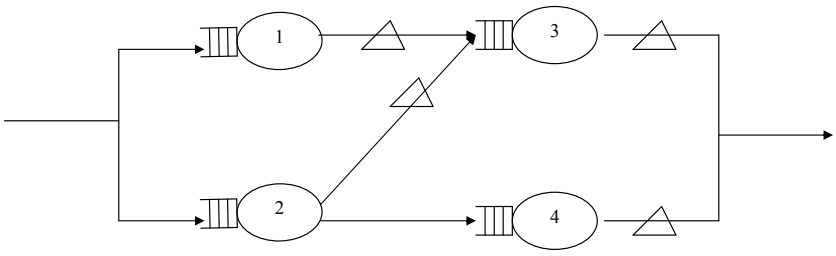


Figure 1: A Fork-Join Network

A simple example of a fork-join network is presented in Figure 1. A "fork" occurs at Station 2, when task 2 splits into tasks 3 and 4. A "join" occurs at Station 3, when tasks 1 and 2 are joined into task 3.

2.1.1 Fork-Join Networks in Hospitals

Fork-join networks can be found frequently in the health-care system in general, and hospitals in particular. Here, the patients and their medical files, test results and insurance policies fork and join at different parts of the process, in order to get to the final task, which may be admitting a patient to a ward, starting an operation, etc. Another reason for the need of a fork-join network in hospitals is the necessity to join and synchronize many separate resources such as doctors, nurses, room/bed, special equipment and test results in order to perform one integrated operation. An example of a fork-join network is the process of transferring a patient from the Emergency Department to an internal ward. This process is depicted in Figures 2-5, which are taken from [10].

Figure 2: Activities Flow Chart

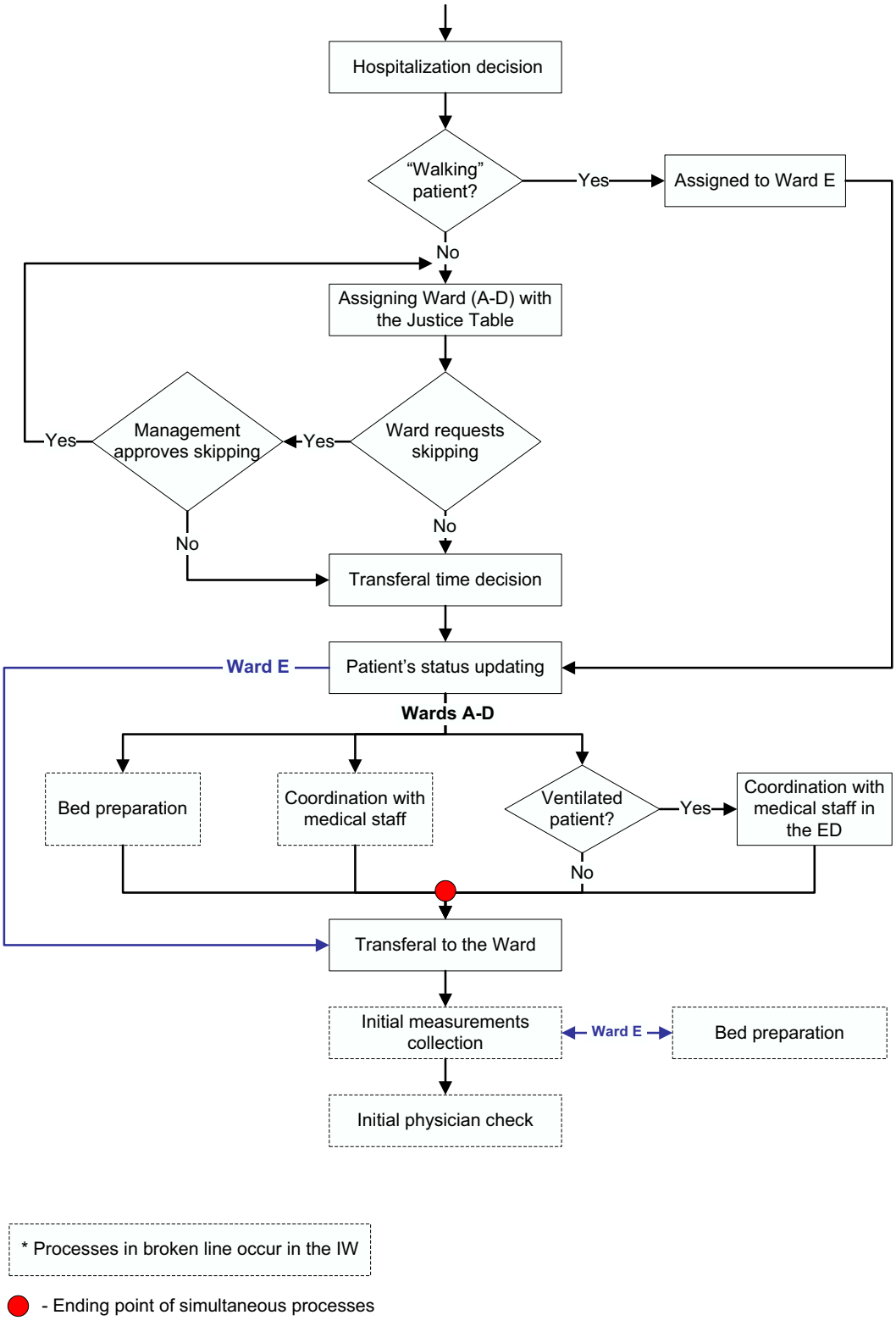


Figure 3: Resources Flow Chart

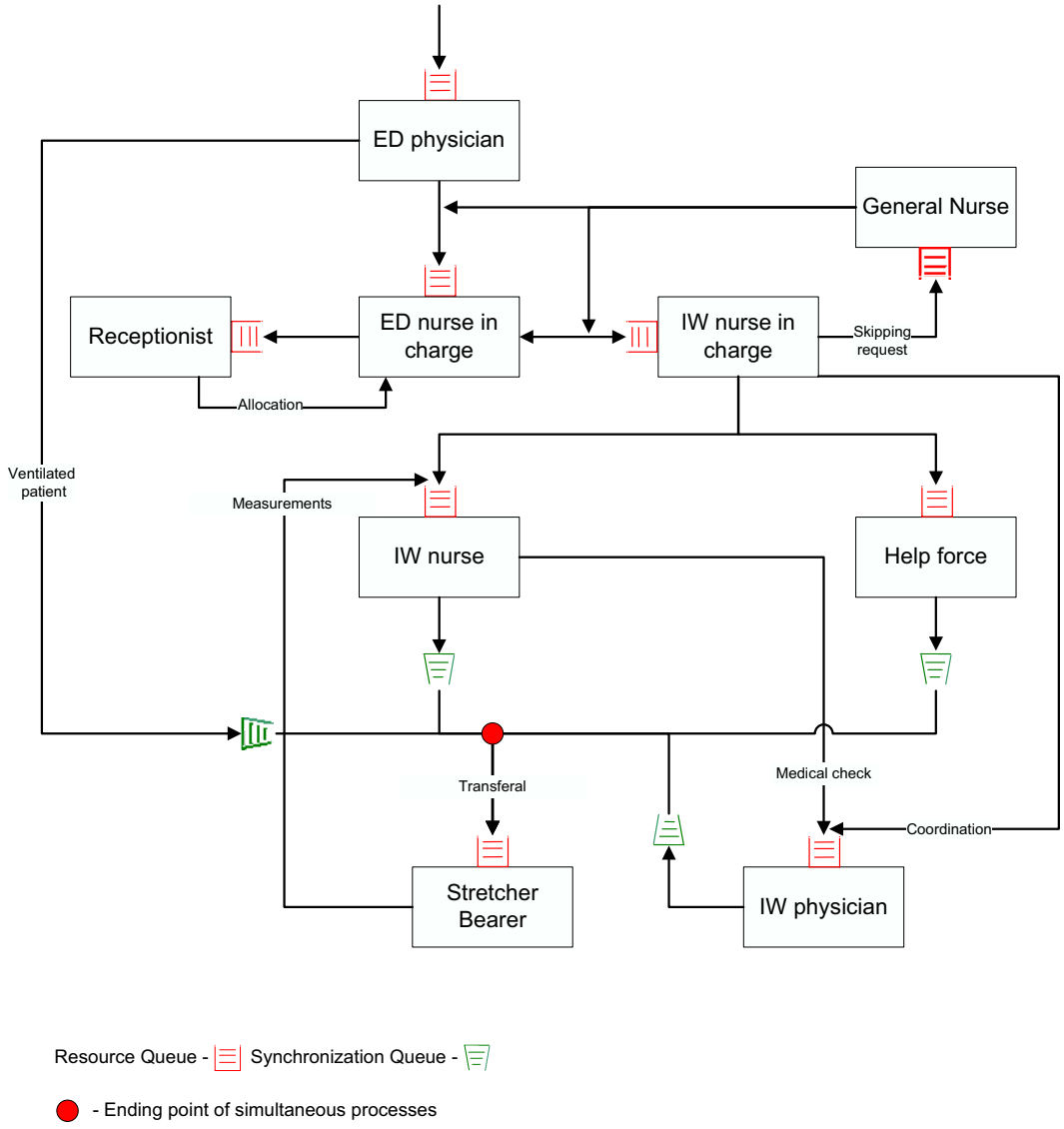


Figure 4: Combined Activities and Resources Flow Chart

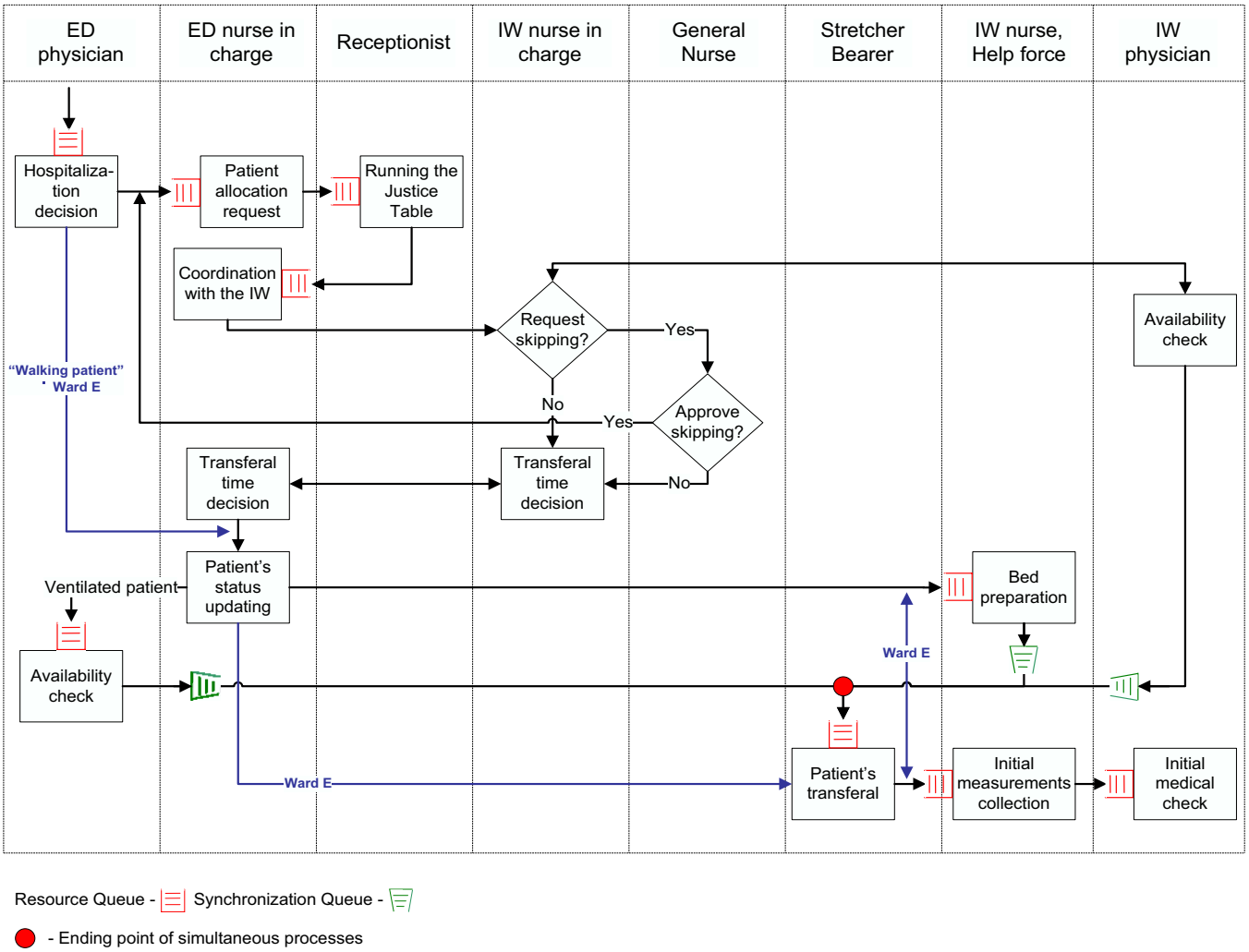
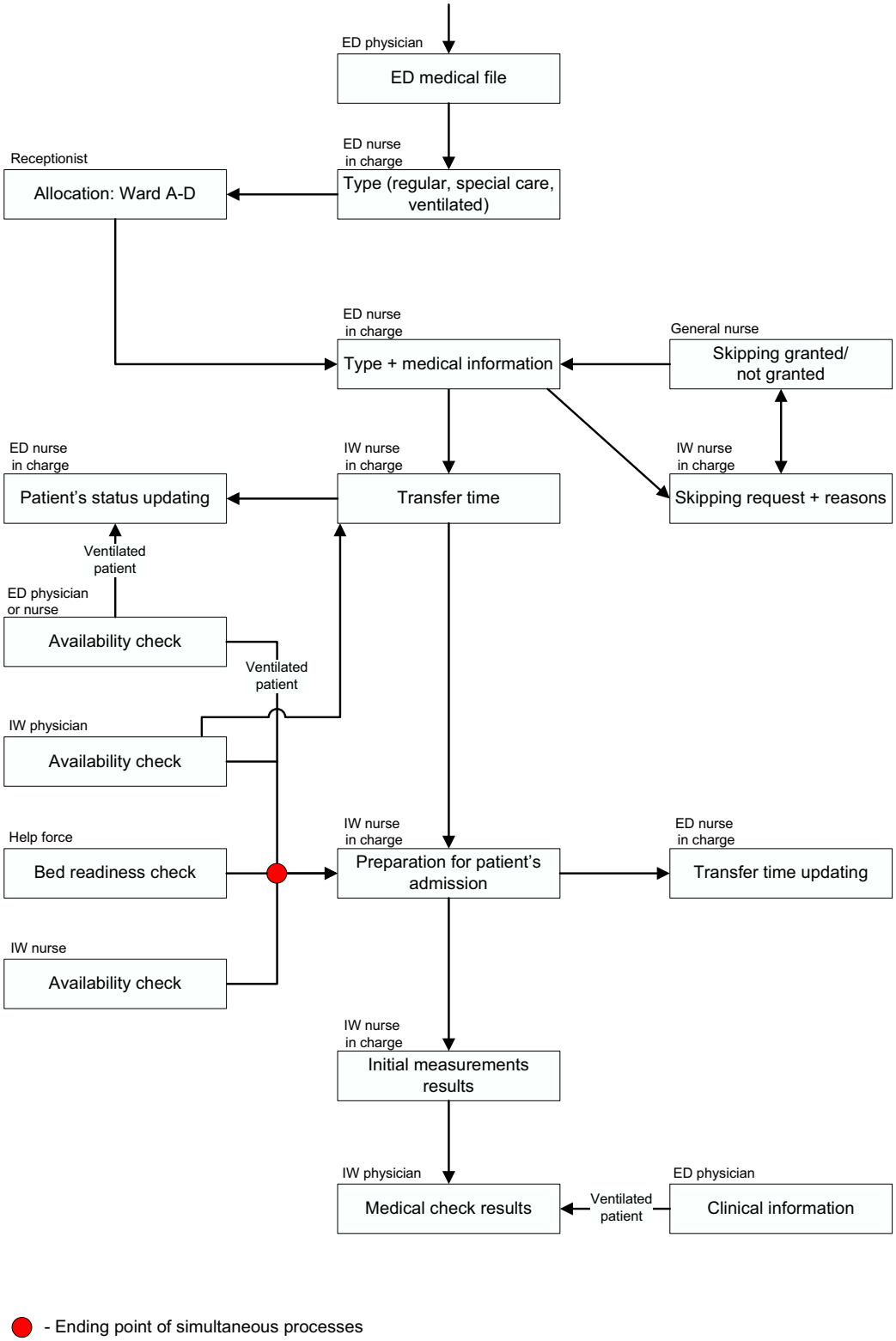


Figure 5: Information Flow Chart



To elaborate, Figure 2 represents the activities carried out when a patient is admitted to the hospital wards. Figure 3 represents the different stations that a patient visits at the Emergency Department, starting with the "ED Physician". An example of a "fork" can be found immediately after the "ED Physician" service. An example of a "join" can be found just before the station "Stretcher Bearer". Note that service in this station starts only after the completion of all the tasks in the stations that proceed it ("ED Physician", "IW Nurse", "Help Force"), thus creating what we call "Synchronization Queues" - these will be discussed later on in more detail.

2.2 PERT Networks

Our discussion of fork-join networks has not taken the stations service times into consideration-in this subsection we add these to our discussion.

Definition 3. *Let G=(V,A) be a graph with V its vertices and A its arcs. Then G is called a PERT Network if it is a directed acyclic network with a single source s and a single sink t.*

Note that in a PERT network, the source s and the sink t do not have service times, as opposed to the rest of the stations. The source s represents the beginning of the "job", the sink "t" represents the completion of the job, while the rest of the stations represent a task/service that is being carried out.

2.2.1 Static Stochastic PERT Networks

In our research we study queueing networks with several service stations; each station is identified with a vertex of the graph. A PERT network that represents the progress of one patient/customer through the system is specified as a Static PERT Network. Note that in a Static PERT one has only synchronizing queues due to the fact that the PERT has only one patient/customer. A Static Stochastic PERT Network is a Static PERT Network with stochastic (random) service times.

2.2.2 Dynamic Stochastic PERT Networks

A Dynamic PERT Network is a PERT Network that represents the progress of several patients/customers through the system. Finally a Dynamic Stochastic PERT Network is a Dynamic PERT Network with stochastic service times.
In our research, we will sometime use models that assume that every service station has an infinite number of agents; in such cases, every patient/customer who enters the network will enter a Static Stochastic PERT Network with no interaction with any of the other patient/customer. That is why, in some cases, analyzing the Static Stochastic PERT Network is enough even though it seems more natural to model a queueing network as a Dynamic Stochastic PERT Network. Moreover, we intend to demonstrate that the Static Stochastic PERT network constitutes an important ingredient in calculating the Offered Load of the originating Dynamic Network.

Definition 4. *Let G=(V,A) be a Static PERT network with exponentially distributed service times. Then G is called a Static Markov PERT Network.*

Definition 5. *Let G=(V,A) be a Dynamic PERT network with Poisson arrivals. If the service times are exponentially distributed, then G is called a Dynamic Markov PERT Network.*

2.2.3 A PERT Network Corresponding to a Fork-Join Network

We now take a fork join network and define its corresponding PERT Network: we add additional vertices s and t which serve as the beginning and the end of the PERT respectively.
Let V' be all the vertices of the fork join network and now let $V = V' \bigcup \{s, t\}$.
Let A' be all the arcs from the fork join network.
Define start(V') to be all the vertices of V' that do not have an arc directed to them. Define end(V') to be all the vertices of V' that do not have an arc which is directed from them to another vertex.
Now let A" be all the arcs that begin at s and end at a vertex from start(V'), and all the arcs that begin at a vertex from end(V') and end at t.
Finally, let $A = A' \bigcup A''$.

The graph G=(V,A) is the PERT network corresponding to the fork-join network. Note that G=(V,A), for V and A as defined above, is a PERT network since G has a single source s and a single sink t, and G is a directed acyclic graph (by its definition).

Example: In Figure 1 we presented a fork-join network. Its Corresponding PERT Network is simply

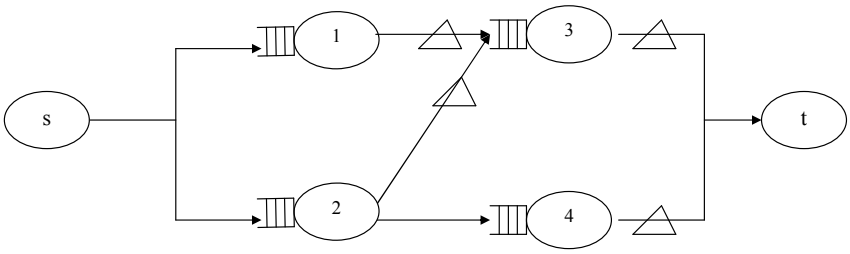
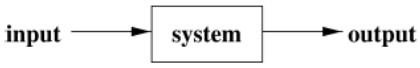


Figure 6: The Corresponding PERT Network of Figure 1

2.3 Little's Law

Little's law is a conservation law that applies to the following general setting:



Consider a system that carries out a certain task, and a flow of units that goes through the system at rate λ. Now let L be the number of units in the system and let W be the average task completion time. We assume one of the following two scenarios:

- A system during a finite cycle, meaning that the system has started the cycle empty and finished it empty (empty $\rightarrow$ empty, finite horizon);
- A system in steady state/in the long run.

Then, under either of the above conditions, we have the following:

$$L = \lambda \cdot W$$

In plain English, Little's Law says that the average number of customers in a stable system (over a specific time interval) is equal to their average arrival rate multiplied by their average time in the system.
In our study we will refer to the units as customers or patients (regarding our health-care study). The system will be either a service station or a network of service stations. And of course the output is the flow of customer/patients who have finished their service and are leaving the system. The "task completion time" in regard to queueing theory is the station's service time. For a proof of Little's Law and further applications of it see Chapter 5 in [9].

2.3.1 Transient Little's Law

In this subsection we discuss Little's law for systems with time-dependent arrivals. We present the "Transient Little's Law", which is an alternative to Little's Law for such systems. The next theorem, by Bertsimas and Mourtzinou [1], gives us the Transient Little's Law.
Let $N_a(t)$ be the number of arrivals in $(0, t]$ for all $0 < t \leqslant \infty$. We also define $\lambda(t)$ to be the "arrival rate"

$$\lambda(t) \triangleq \lim_{\Delta t \to 0} E[N_a(t) - N_a(t - \Delta t)].$$

In the special case of renewal processes, for the limit to exist, the inter-arrival distribution has to be absolutely continuous. For a non-homogeneous Poisson arrival process, $\lambda(t)$ is its rate function.

Theorem 1. ***Transient Little's Law:*** *(Theorem 3 in [1]) Denote by $L(t)$ the number of customers in the system at time t, and by $S(u)$ the time spent in the system for a customer that arrived at $(u - du, u]$. Then, if the system starts empty ($L(0) = 0$), then we have that*

$$E[L(t)] = \int_0^t \lambda(u) P\{S(u) > t - u\} du, \quad t \geqslant 0.$$

2.3.2 Little's Law in Systems With Time-Dependent Arrivals

In our research, we shall study systems with arrival processes that are nonhomogeneous Poisson processes; in such systems, clearly one cannot use Little's law in a steady-state because of the non homogeneous arrival process.
One can use Little's law for those kinds of systems if the system is periodically empty, but that is not the case in the emergency department.
In our research we try to find an alternative formula to Little's law for time changing arrivals; a good example of such is presented in the Transient Little's Law (from Theorem 1). In Section 3.1, we will present such a formula; calculated for a specific kind of network. In the rest of the research we shall try to find similar results for more complex systems.

3 Offered-Load: Calculations and Applications

In this section, we demonstrate methods for calculating the Offered Load on various kinds of networks. First we show a method of calculating the Offered Load of the $M_t/GI/N_t$ queue. In order to calculate the Offered Load we use the corresponding infinite-server model. Infinite-server queues represent the (usually) highly idealized situation in which different customers do not interfere with each other. Infinite-server models are obviously not appropriate to describe systems in which customers spend time waiting before being served. Nevertheless, networks of infinite server queues are interesting both in their own right and as approximations of networks of light-to-moderately loaded multi-server queues, possibly with finite waiting space. The elegant theory of infinite-server models with time-dependent arrival rates is a useful frame of reference for examining more difficult finite-server models with time-dependent arrival rates. We can usefully view the time-dependent behavior of the finite-server model in relation to the more tractable analytical descriptions of its infinite-server counterpart.

3.1 The Offered Load of $M_t/GI/N_t$

The following theorem provides four representations for $R(t) = E[L(t)]$, with respect to the service time distribution. They are proved in [2].

Theorem 2. *For each t, $L(t)$ has a Poisson distribution with mean*

$$R(t) = E[L(t)] = E[\lambda(t - S_e)] \cdot E[S] = E\left[\int_{t-S}^{t} \lambda(u)du\right] = \int_{-\infty}^{t} [1 - G(t-u)]\lambda(u)du, \qquad (1)$$

where
S is a generic service time with cdf G;
S_e is a generic "excess service time", having the following cdf:

$$P(S_e \leqslant t) = \frac{1}{E(S)} \int_0^t [1 - G(u)]du, \qquad t \geqslant 0. \qquad (2)$$

Remark:

$$E\left[\int_{t-S}^{t} \lambda(u)du\right] = E(A(t) - A(t-S)), \qquad (3)$$

where A(t) is the cumulative number of customers/patients that arrived to the system up to time t.
This remark provides us with an insight on the expression $E\left[\int_{t-S}^{t} \lambda(u)du\right]$: the number of customers/patients in the systems are those who arrived to the system minus the ones who came into the system and already left (due to their end of service). From the result of Theorem 2, one can observe that when $\lambda(t) \equiv \lambda$, the expression for $R(t)$ becomes the Offered Load R of a homogeneous arrival rate $R = \lambda \cdot E[S]$. Hence, we also deduce that the expression of $R(t)$ is similar to that of the homogeneous arrival case, except for a random time lag S_e in $\lambda(t)$.

3.2 Offered Load of Open Networks

In this section we will study the Offered Load of open networks. In order to do that we use the $(M_t/GI/\infty)^N/M$ model, The N means that there are N stations with independent Markovian routing according to a substochastic matrix $P = [p_{ij}]$, where p_{ij} is the probability of going next to station j immediately after completing service at station i.

We assume that all arrivals eventually leave; hence we will assume that for all i and j $\Sigma_{n=1}^{\infty} p_{ij}^n < \infty$, or equivalently the matrix $[I - P]$ is non-singular. In this model the external arrival process is nonhomogeneous Poisson. External arrivals are initially assigned to station i with probability π_i, with successive assignments being mutually independent. In this model, the service times are mutually independent and independent of the arrival process; the service time distribution in station i is denoted S_i with cdf G_i, M indicates independent stationary Markov routing.

Let $(\alpha_1(t), ..., \alpha_N(t))$ be the deterministic external-arrival-rate function, where $\alpha_i(t)$ is the arrival rate to station i from outside the network. Let $\lambda_i^+(t)$ be the minimal nonnegative solution (or, equivalently, the unique integrable solution) to the system of flow equations

$$\lambda_i^+(t) = \alpha_i(t) + \sum_{j=1}^{N} E[\lambda_j^+(t - S_j)]p_{ji}, \quad 1 \leqslant i \leqslant N; \tag{4}$$

$\lambda_i^+(t)$ corresponds to the aggregate-arrival-rate function to station i.
Now let $\lambda_i^-(t)$ be

$$\lambda_i^-(t) = E[\lambda_i^+(t - S_i)](1 - \sum_{j=1}^{N} p_{ij}), \quad 1 \leqslant i \leqslant N; \tag{5}$$

$\lambda_i^-(t)$ corresponds to the aggregate-departure-rate function from station i.
Since we do not have stationarity, $\lambda_i^+(t)$ need not be equal to $\lambda_i^-(t)$.
Existence of a minimal nonnegative solution to Equation (4) is proved in [8].

We assume that the network started empty in the infinite past. Let $\{Q_i(t), \quad t \geqslant 0\}$ denote the number of customers in station i at time t, and define $Q(t) = (Q_1(t), ...Q_N(t))$. From Massey and Whitt [8] we know that, in the $(M_t/GI/\infty)^N/M$ model, the steady state distribution of $Q(t)$ is product-form, and its mean is given in the next theorem.

Theorem 3. *(Theorem 1.2 in [8]) In the $(M_t/GI/\infty)^N/M$ model, for each i, the number of patients in station i at time t, $Q_i(t)$ $1 \leqslant i \leqslant N$,are independent Poisson random variables with finite means:*

$$m_i(t) \equiv E[Q_i(t)] = E[\int_{t-S_i}^{t} \lambda_i^+(u)du] = E[\lambda_i^+(t - S_{ie})]E[S_i]. \tag{6}$$

In addition, for each t, the vector $(Q_1(t), ..., Q_N(t))$ is independent of the external departure processes (from the network from each station) before time t. The external departure processes are independent Poisson processes with integrable time-dependent rate functions

$$\lambda_i^-(t) = E[\lambda_i^+(t - S_i)](1 - \sum_{j=1}^{N} p_{ij}), \quad 1 \leqslant i \leqslant N. \tag{7}$$

Moreover, the aggregate arrival process to station i (counting arrivals from other stations as well as from outside the network) and the aggregate departure process from station i (counting flows to other stations as well as to outside the network) are Poisson processes if and only if no customer can visit station i more than once.

Example: An Open Queueing Network of Three Queues

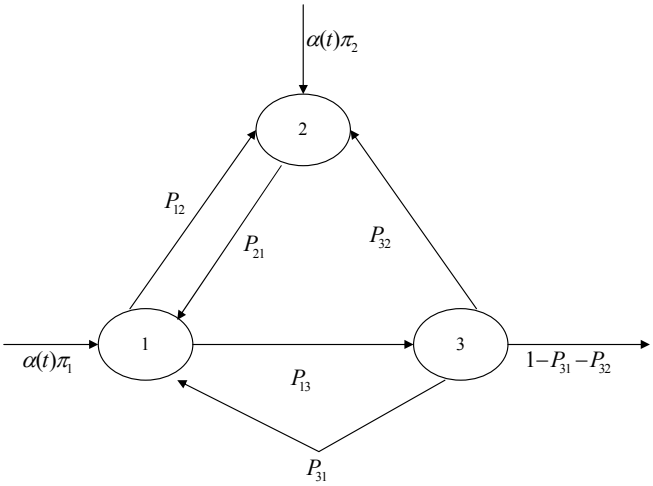


Figure 7: An Open Queue Network

Here $\alpha_1(t)$ and $\alpha_2(t)$ represent the rates of the external arrivals to station 1 and 2 respectively, and P_{ij} is the probability of routing to station j after service in station i. In this example, one can only leave the network from station 3.

Another interesting way of calculating the Offered Load of an open network is given by the next theorem of Keilson and Servi [5]. Here we use the service time p.d.f and survival function for the Offered Load representation, as opposed to the formulas given in Theorem 2.

Theorem 4. *(Theorem 15 in [5]) Consider an $(M_t/G/\infty)^N$ model with:*
(i) an external independent Poisson arrival process with time dependent rate function $\alpha_i(t)$ to station i;
(ii) a service time with p.d.f. $a_{S_i}(t)$ and survival function $\overline{A}_{S_i}$ for station i (independent of the entry point into the network and the number of revisits to station i);
(iii) routing probabilities P_{ji} specifying the probability that station i will be visited after facility j and P_{jout} for the probability of leaving the network after service at station j.

Let $Q(t) = (Q_1(t), Q_2(t), ..., Q_K(t))$ be the vector of populations at the service station with $Q(0) = 0$. Then the multivariate population distribution of $Q(t)$ is Poisson, with all stations populations independent.

In addition, the mean of $Q_i(t)$ *is* $E[Q_i(t)] = \lambda_i(t) * \overline{A}_{S_i} = E(\int_{t-S_i}^{t} \lambda_i(t)dt)$,

where $\lambda_i(t)$ *is the solution to*

$\lambda_i(t) = \alpha_i(t) + \sum_j P_{ji}[\lambda_j(t) * a_{S_j}(t)]$.

Theorems 3 and 4 both represent the Offered Load of the same system. The proof of Theorem 3 relies on the result of Theorem 2, which provides us with an insightful representation of the Offered Load. We elaborate on this representation in Subsection 4.2

3.3 Offered Load of Parallel Queues

A special case of a fork-join network is one which consists of only one "fork" and one "join", where all of the stations work simultaneously. We refer to that kind of network as "Parallel Queues" Network.

Example: A Set of k Parallel Queues

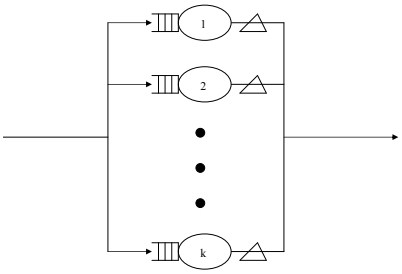


Figure 8: A set of Parallel Queues

In Figure 8 we have a set of k stations; at each station an activity is being carried out. All of the activities are being carried out simultaneously, the service is completed when all of the activities are completed. We have referred to each activity's execution time as the station's service time.

We now present a more general structure of Parallel Queues. Suppose every patient comes into the system with a set of stations P_k that determines which of the stations the patient will visit, in that case the patient visits all of the stations from the list P_k simultaneously. Here the patient does not have to visit all the stations but only the ones in the set P_k. Let $K = 2^M - 1$, now let $\{1, ..., K\}$ be the group of all the possible indexes for the sets P_k. Meaning that $\{P_k : 1 \leqslant k \leqslant K\}$ are all the possible subsets of $\{1, ..., M\}$, which are all the stations of the network. The next result by Keilson and Servi [5] provides us with a simple way to calculate each station's Offered Load in this case.

Theorem 5. *(Theorem 16 in [5]) Let $M_t/GI/\infty$ be a network with M stations and, for $1 \leqslant k \leqslant K$, let $\lambda_k(t)$ be the rate of arrivals that simultaneously use the set of stations P_k for a duration with survival function $\overline{A}_{Sk}(t)$. Then, if the system is initially empty, the joint probability generating function of $Q_m(t)$, namely the number of customers/patients using station m at time t, is*

$$\pi(u_1, u_2, ..., u_M, t) = E[\prod_{m=1}^{M} u_m^{Q_m(t)}] = \prod_{k=1}^{K} exp[-\zeta_k(t)(1 - \prod_{m \in P_k} u_m)]$$

where

$$\zeta_k(t) = \lambda_k(t) * \overline{A}_{Sk}(t).$$

In this case, we refer to $E(Q_m(t))$ as the Offered Load of station m at time t.

3.4 Offered Load of Tandem Queues

The term tandem-queues means that a patient must visit station number 1 first; only after service at station number 1 is finished then service at station number 2 starts. The patient keeps progressing in that manner at the network until the service of the last station of the tandem is finished; then the patient's service process in the network is complete.

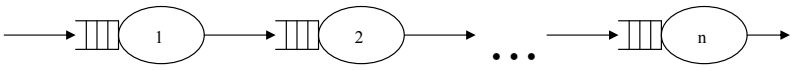


Figure 9: n Queues in Tandem

In Figure 9, one can see a tandem queue, consisting of n stations in series.

We begin our study of Tandem Queues by defining the Offered Load of each station of the Tandem Queues. Consider a network of Tandem Queues, $(M_t/GI/N_t)^N$. Here N means that there are N stations in the network. Let $(M_t/GI/\infty)^N$ be the corresponding Tandem Queues network with the same arrival process and service time distribution; where every station has infinity many servers.

Definition 6. *Consider a network of Tandem Queues $(M_t/GI/N_t)^N$, with a corresponding Tandem Queues network $(M_t/GI/\infty)^N$. The Offered Load of station k, $k \leqslant N$, $R_k = \{R_k(t), t \geqslant 0\}$ is given by $R_k(t) = E[L_k(t)]$, where $L_k(t)$ is the number of customers/patients in station k at time t in the corresponding Tandem Queues network $(M_t/GI/\infty)^N$.*

In this section we present an insightful way to calculate the Offered Load of a station which is a part of Tandem Queues. This representation of the Offered Load uses only the service time and the arrival rate function, which are model's primitives.

Theorem 6. *In a queueing network of two tandem queues, assume that service times of each station are independent, the service time of station i is S_i and the arrival at the network is from a non-homogeneous Poisson process with time dependent arrival rate function $\lambda(t)$ (also independent of the service times).*
The Offered Load at time t of the second station is given by

$$R_2(t) = E(\lambda(t - S_1 - S_{2e})) \cdot E(S_2), \quad t \geqslant 0.$$

(The first station is simply an $M_t/GI/\infty$ queue.)

The proof of Theorem 6 will be presented in the thesis.

Conclusion: Using this result, one can calculate the Offered Load of any station of the tandem network, by considering all the stations that precede it as one station where its service time is the sum of all those stations' service times. Specifically if one wishes to calculate the Offered Load of station number k ($k \leqslant n$) in Figure 9. One can regard all of the stations previous to station k as a single station with service time $S = S_1 + S_2 + ... + S_{k-1}$. Now we have constructed a new tandem network with two stations, with service times S and S_k of the first and the second station respectively. By using Theorem 6, therefore, the Offered Load at time t of station number k is:
$R_k(t) = E(\lambda(t - S - S_{ke})) \cdot E(S_k)$=$E(\lambda(t - (S_1 + ... + S_{k-1}) - S_{ke})) \cdot E(S_k)$.

3.5 Offered Load of Static Markov PERT Networks

In this section, we will discuss the Offered Load calculation in a PERT network with independent and exponentially distributed activity durations. We model such networks as a finite-state, absorbing continuous-time Markov chains with upper triangular generator matrices. The state space is related to the network structure. We will use a simple algorithm by Kulkarni and Adlakha [6] to calculate the distribution of the network service time which will help us calculate the Offered Load.

PERT networks were defined in Subsection 2.2. In this subsection we will present an algorithm to calculate the Offered Load of a Static Markov PERT Network. Doing so requires a few definitions regarding the graph; we will define them now and illustrate some of them using the next example.

Example:

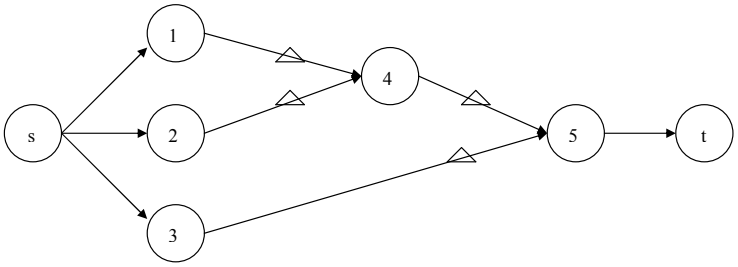


Figure 10: A Markov PERT Network

In Figure 10 we see a Markov PERT network of 7 nodes. We identify the nodes of the graph with the activities of the queueing network. We assume that the activities have an exponential service time with rate μ_i for node i respectively.

For every $a \in A$, let $\alpha(a)$ be the starting node of a, and let $\beta(a)$ be the ending node of a. We now define:

(i) $L(v) = \{a \in A : \alpha(a) = v\}$; $L(v)$ consists of all arcs that start at node v.

(ii) $O(v) = \{a \in A : \beta(a) = v\}$; $O(v)$ consists of all arcs that end at node v.

(iii)$\beta(L(v)) = \{\beta(a) \in V : a \in L(v)\}$; $\beta(L(v))$ consists of all the nodes that end an arc that starts at node v.

(iv) $\alpha(O(v)) = \{\alpha(a) \in V : a \in O(v)\}$; $\alpha(O(v))$ consists of all the nodes that start an arc that ends at node v.

Let $B \subseteq V$ be a set of nodes. Define:

(v) $L(B) = \{L(v) : v \in B\}$; $L(B)$ consists of all the arcs starting at a node $v \in B$.

(vi) $O(B) = \{O(v) : v \in B\}$; $O(B)$ consists of all the arcs ending at a node $v \in B$.

Definition 7. *A directed (s,t) path in the network is a sequence of arcs $(a_1, ..., a_k)$ such that $a_i \in A$ for $i = 1, ...k$; $\alpha(a_1) = s$ and $\beta(a_k) = t$. And $\alpha(a_{i+1}) = \beta(a_i)$, for i=1,2,...,k-1. Henceforth, a path will always mean a directed (s,t) path.*

Definition 8. *A (u,v) node-path is a group of nodes $\{v_1, ..., v_k\}$ such that $v_i \in V$ for $i = 1, ...k$; $v_1 = u$ and $v_k = v$. And there exists $a \in A : \alpha(a) = v_i$, $\beta(a) = v_{i+1}$ for i=1,2,...,k-1.*

Definition 9. *For every $X \subset V$, define a set $(X, \overline{X}) = \{a \in A : \alpha(a) \in X, \beta(a) \in \overline{X}\}$ where $\overline{X} = V \setminus X$. The set of arcs $(X, \overline{X})$ is called an (s,t) cut if $s \in X$ and $t \in \overline{X}$.*

Definition 10. *An (s,t) cut $(X, \overline{X})$ is called a uniformly directed cut (UDC) if $(\overline{X}, X)$ is empty.*

Definition 11. *A pair (E,F) where $E, F \subset A$ is called an admissible 2-partition of a UDC D , if $E \cup F = D$, $E \cap F = \phi$, and for every $a \in F$, $O(\beta(a)) \nsubseteq F$.*

Definition 12. *A node-path (u, v) passes through a set of arcs E if $\exists v_1, v_2 \in (u, v)$ such that there is an arc $a \in E$ that satisfies $\alpha(a) = v_1 \quad \beta(a) = v_2$.*

Our goal is to calculate the Offered Load of a service station. We have defined it to be the number of patients in the station in a corresponding network with infinity number of servers. In order to find this number, one has to identify which node is serving a patient, meaning which node task is active. Thus in the next definition we will define *active* nodes. Another goal is to calculate the number of patients in each "synchronizing queue". To this objective we add the definition of a *dormant* node. In plain English: A dormant node represents a service station with a task that should begin its service due to the end of the service of a prior service station. However this station's task cannot start due to another prior service station with an active task.

Definition 13. *Let G=(V,A) be a PERT network. Let us assume that the project that is modeled by the network G starts at time zero and the project ends at time S, Define:*

(i) Active: a node v is active if the task of the node is being executed.

(ii)Dormant: a node v is dormant if the task of the node should be active due to a nodes path that reaches the node v where every node in the path has finished its activity, but the activity cannot start due to another path that has not finished all its activities yet.

(iii) Idle: a node v is idle if it is neither active nor dormant.

During the course of project execution, each node can be in one of the following three states: active, dormant or idle.

Corollary 1. *Given all of the active nodes of the project the collections of dormant nodes and idle nodes are uniquely determined*

Proof:
We first show that given the set of active nodes the set of dormant nodes are uniquely determined. To this end we start with the following definition: a node v is called *suspected dormant* if there exists a nodes path $(s, v)_1$ where $(s, v)_1 \setminus \{v\}$ includes an active node, and there exists a nodes path $(s, v)_2$ where every node in $(s, v)_2$ is not active.
With this definition at hand we can now identify the dormant nodes. A node v is dormant if there exists a nodes path $(s, v)_1$ where $(s, v)_1 \setminus \{v\}$ includes an active node, and there exists a nodes path $(s, v)_2$ where every node in $(s, v)_2 \setminus \{v\}$ is not active and not suspected dormant.
First, we show that a node cannot be active and dormant at the same time; if a node v is dormant then there exists a nodes path (s, v) where $(s, v) \setminus \{v\}$ includes an active node. Thus v cannot be active. Thus a dormant node cannot be an active node at the same time. Finally we show that we can indeed identify the dormant nodes as above. By demanding the existence of a "nodes path $(s, v)_1$ where $(s, v)_1 \setminus \{v\}$ includes an active node", we demand a path which hasn't finished all its activities yet. The demand of the existence of a "nodes path $(s, v)_2$ where every node in $(s, v)_2 \setminus \{v\}$ is not active and not *suspected dormant"* is to make sure that there is a path that has finished all the activities prior to node v. We have thus shown that given the

set of all of the active nodes, the dormant nodes are uniquely determined. The idle nodes are determined uniquely by the definition of an idle node.■

We now use the above definitions in order to define the state space of the CTMC. Since our activities are identified with the nodes of the graph, we will define the state space using the nodes.
Our state space will be composed of two sets of nodes B and D; B will be all the active activities while D will be all the dormant activities.
Finally we will add the state (ϕ, ϕ) which will be the absorbing state of the CTMC. This state represents the project completion.

Definition 14. *Let $\overline{M}$ be all the pairs of nodes (B, D) where $B, D \subseteq V$, such that B are active nodes D are dormant nodes. Our state space M will be $M = \overline{M} \bigcup (\phi, \phi)$.*

Note that by Corollary 1, one can define the same state space using only the set B of all the active nodes. However, defining the state space as in Definition 14, exposing D, will make our definition of the transition matrix easier and will also help us later to study the synchronizing queues.

Our model is slightly different from Kulkarni and Adlakha's model in the fact that the activities are identified with the nodes of the graph G and in Kulkarni and Adlakha's model they are identified with the edges. We shall show that there is a bijective map that leads from their state space to ours, thus enabling us to use some of their results.
Identifying the activities with the nodes is natural for our objective. Furthermore, one can also obtain useful information from doing this. Indeed, if one would like to calculate the number of patients that are waiting for their blood test before they can see a doctor, then one would simply sum up the Offered Load on all the states where the specific doctor's station is dormant and the blood test station is active. Calculating the Offered Load of these states will be presented in the course of this section.

We now define the starting state of our process. Let $X_0 = (B, \phi)$ be starting state which is defined by $B = \beta(L(s))$. Finally we define the transition matrix of those states.
Suppose $v \in B$, the state of the process just before the transition is (B,D) and the transition occurs due to the completion of activity v. We will look at $\beta(L(v))$ as all the activities that can become active or become dormant after activity v is completed. Note that an activity in $\beta(L(v))$ can also stay dormant.
The transition will be from state (B,D) to the state (B',D') where $B' = ((B \setminus \{v\})) \bigcup \{u \in \beta(L(v)) : for \ \ every \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) = \phi\}$.
and $D' = (D \bigcup \{u \in \beta(L(v)) : for \ \ some \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) \neq \phi\}) \setminus B'$.
By stipulating that $for \ \ every \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) = \phi$ we make sure that the activity is not waiting for another activity in order to become active. Thus the active activities are now $((B \setminus \{v\})) \cup \{u \in \beta(L(v)) : for \ \ every \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) = \phi\}$. While $B \setminus \{v\}$ are the activities that remains active.
By stipulating $for \ \ some \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) \neq \phi$ we make sure that activity u is waiting for another activity besides v to be completed thus it is labeled as a dormant activity; the activities that remain dormant are those which were dormant minus the ones that became active.

Thus $D' = (D \bigcup \{u \in \beta(L(v)) : for \ \ some \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) \neq \phi\}) \setminus B'$.

Theorem 7. *Define* $\{X(t), t \geqslant 0\}$ *to be the state that the project is at time t.*
Assuming that all activity durations are independent exponentially distributed random variables. Then $\{X(t), t \geqslant 0\}$ *is a CTMC (absorbing, continues-time Markov chain) on M. with the following infinitesimal matrix* $Q = [q\{(B,D),(B',D')\}]$ *for* $(B,D),(B',D') \in M$ *where* $q\{(B,D),(B',D')\}$
$= \mu(v)$ *if* $B' = ((B \setminus \{v\})) \bigcup \{u \in \beta(L(v)) : for \ \ every \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) = \phi\}$.
And $D' = (D \bigcup \{u \in \beta(L(v)) : for \ \ some \ \ node \ \ path \ \ (s,u) : (s,u) \bigcap ((B \setminus \{v\}) \bigcup (D \setminus \{u\})) \neq \phi\}) \setminus B'$.
$= -\sum_{v \in B} \mu(v)$ *if* $B' = B, D' = D$
$= 0$ *otherwise.*

Example:

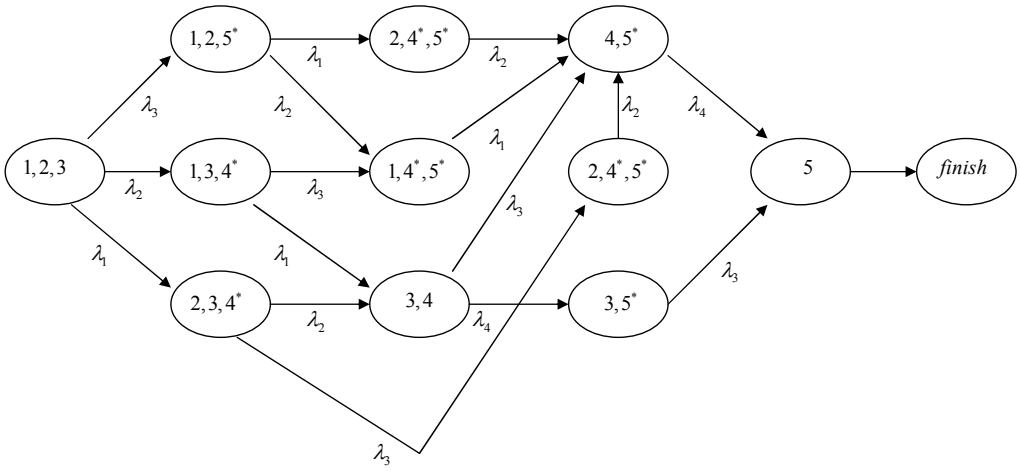


Figure 11: The rate diagram of the Continuous-Time Markov Chain of Figure 10

Here, at each state, a number i means that the activity numbered i in the original network is active; $i \in B$. i^* means that activity i in the original network is dormant; $i^* \in D$.

After defining this CTMC one can calculate the network service time distribution , and then with the service time distribution calculate the Offered Load.
The following are two algorithms by Kulkarni and Adlakha [6] that will allow us to calculate the service time distribution of a Markov PERT network.

Backward Algorithm

Define $p_i(t) = P(X(t) = N|X(0) = i); \quad 0 \leqslant i \leqslant N$. Then the cdf of the PERT network service time will be $F(t) = p_0(t)$.
$p_i(t)$ are given by

$$p_i'(t) = \sum_{j \leqslant i} q_{ij} p_j(t)$$

$$p_i(0) = \delta_{iN} \quad 0 \leqslant i \leqslant N,$$

where $\delta_{ij} = 1$ if i=j, and 0 otherwise, and q_{ij} are taken from the infinitesimal generator matrix of the CTMC. We will start the algorithm with $p_N(t) = 1$ for $t \geqslant 0$ and compute $p_{N-1}(t), ..., p_1(t), p_0(t)$ recursively.

Forward Algorithm

Define $\overline{p_j}(t) = P(X(t) = j|X(0) = 0); \quad 0 \leqslant j \leqslant N$. Then $P(S \leqslant t) = \overline{p_N}(t)$. The differential equations for $\overline{p_j}(t)$ are given by $\overline{p_j'}(t) = \sum_{i \leqslant j} \overline{p_i}(t) q_{ij}, \qquad \overline{p_j}(0) = \delta_{j0} \quad 0 \leqslant j \leqslant N$ Now we start with $\overline{p_0}(t) = e^{-q_{00}t}$ and compute $\overline{p_1}(t), \overline{p_2}(t), ..., \overline{p_N}(t)$ in that order.

Offered Load Calculation

The above is an algorithm to calculate the distribution of S the project completion time. In terms of service, one means that the total completion time is the service time of a patient who has to go through the whole network. Since we have ∞ number of servers at each station, every patient can look at his situation as if a whole network is assigned to him alone, that is why the project completion time is the time it takes a project to go through the PERT network. In order to calculate the Offered Load one must find the distribution of the service time. Let S be the service time calculated with the above algorithm , then the Offered Load R(t) will be calculated by: $R(t) = E(\lambda(t - S_e)) \cdot E(S)$, when $P(S_e \leqslant t) = \frac{1}{E(S)} \int_0^t P(S > u)du$.

3.6 Station by Station Calculations

In this section we shall propose a method of calculating the Offered Load at each individual station, and not only of the whole system.

The Black Box Method

In Subsection 3.5, we have shown a way to calculate a project completion time distribution (we called it the service time). Section 3 provides us several ways of using the service time distribution in order to calculate the Offered Load of the entire network.

Here we will use this knowledge to calculate the Offered Load of a single specific node.

Suppose one has a fork join network with V as its set of nodes and A its set of arcs. To calculate the Offered Load of station i, one can define a new fork join network i^- as follows:
• Let V^- be all the nodes of V which have a directed path to station i.
• Let A^- be all the arcs in A which start at a node from V^- and finish at a node from $V^- \cup i$.
• i^- will be the fork join network with V^- its set of nodes and A^- its set of arcs.

Now let $G^{(i)}$ be the Corresponding PERT network of i^- . We will consider $G^{(i)}$ as a single station with its project completion time S as our service time. Since we look at the whole PERT as one large black box, we can look at the system as a $M_t/G/\infty$ station which means that the departure process is a Poisson process with time dependent rate function $\delta(t) = E(\lambda(t - S))$. Since we took $G^{(i)}$ to be a network that ends just before station i, the departure process of $G^{(i)}$ is the arrival process to station i.

Now we have an $M_t/G/\infty$ where $\delta(t)$ is the arrival rate function. Let S_i be the service time of station i and S_{ie} the equilibrium-residual-lifetime of the service of station i.
Finally Theorem 2 shows us that the Offered Load of station i at time t is $E(\delta(t - S_{ie})) \cdot E(S_i)$

Another elegant way to calculate the Offered Load is by viewing the network as a "tandem of two queue's", where all of the Corresponding PERT network of i^- is considered as one station with service time S_{i^-}, while the second station is simply station number i. Then from Theorem 6 one can represent the Offered Load in a more elegant way using only service times and the arrival rate function:
The Offered Load of station i at time t will again be $E(\lambda(t - S_{i^-} - S_{ie})) \cdot E(S_i)$.

Example

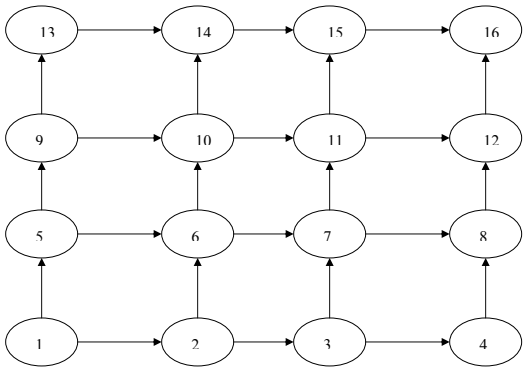


Figure 12: A Fork-Join Network

To calculate the Offered Load of station 10 in the system in Figure 12; one first need to identify the Corresponding PERT Network of 10^-.

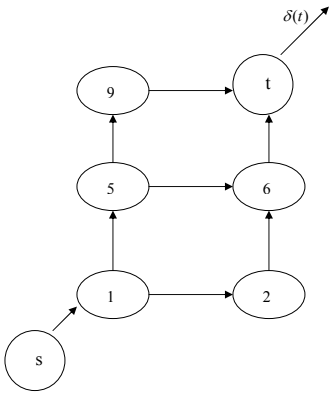
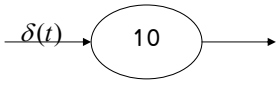


Figure 13: The Corresponding PERT Network of 10^-

Figure 14: Now we view 10^- as a single black box with service time S.

The departure process of 10^- is a Poisson process with mean $\delta(t) = E(\lambda(t - S))$.
That departure process is the arrival process to station 10; so now we have a simple network of only one station, station number 10:



And from Theorem 2 the Offered Load of station 10 is $E(\delta(t - S_{10e})) \cdot E(S_{10})$ when S_{10} is the service time of station 10.

3.6.1 Calculating the Offered Load of each Station of the Static Markov PERT Networks

We now present a simple method to calculate a node's (station) Offered Load in a Static Markov PERT Network, in which each node's service time is exponentially distributed. When the service time of the stations are all exponentially distributed, then one can use Kulkarni and Adlakha's algorithm and find the Offered Load of each state in $\overline{M}$. In order to find a single station Offered Load, we need to sum up all the Offered Loads of all the states that include the current station as active.
A wise choice would be to use the forward algorithm which is slightly harder but gives us more information.
If one wants to calculate the Offered Load of phase i, then one can stop the forward algorithm

at $\overline{p_i}(t)$ which gives us the service time of a network that ends at phase i. In that manner, one can continue with the black box method in order to calculate phase i Offered Load.
Our goal is to calculate the Offered Load of the stations and not those of the phases but, given the phases' Offered Load, one can calculate the station's Offered Load by summing up all the phases where the wanted station is active.
In the example illustrated in Figure 12, if one wishes to calculate the Offered Load of Station 4 then one can find the Offered Load of phases $(3, 4)$ and $(4, 5^*)$; the sum of them is the Offered Load of Station 4.
In this manner we calculate only the expectation of the number of projects at Station 4; however, we have stated before that the distribution of the number of projects at time t of each station is Poisson distributed and we have just found its mean.

4 Our Proposed Research

In this section, we present the topics that we propose to investigate.
We shall focus our research on the following:

- **Developing Methods for Calculating the Offered Load in Emergency Department Networks**
- **Representing the Offered Load using Model Primitives**
- **Setting staffing requirements in the Emergency Department, based on the Offered Load results**
- **Estimating the Offered load using an Emergency Department Simulation**

4.1 Calculating the Offered Load in Emergency Department Networks

The Emergency Department network is a very complicated queueing network. In Subsection 2.1.1, we elaborated on the topic and illustrated it in Figures 2-5. We proposed methods for calculating the Offered Load of different kinds of networks; for this topic we now focus on two of them. In Section 3.2, we have discussed Open Networks and presented Theorem 3, which provides us with a calculation of the Offered Load of such networks. In Section 3.5, we discussed PERT Networks and offered a couple of methods for Offered Load calculations of such networks. However, as indicated in Subsection 2.1.1, the Emergency Department's network is neither a pure Open Network nor a pure PERT Network but rather a combination of the two.
One characteristic of the Emergency Department network is "fork and join"; we have dealt with this characteristic in the study of PERT Networks. Another characteristic of this Emergency Department network is "stochastic routing"; a characteristic that we dealt with in the study of Open Network. Thus, in our research we shall develop methods designated for Offered Load calculations of Emergency Department Networks, which are a combination of the Open Networks and PERT Networks.

4.2 Representing the Offered Load Using Model Primitives

One of our main goals is to develop efficient and insightful methods of calculating the Offered Load. In Section 3 we discussed a few of those methods; however, some calculations may end up complex, causing the loss of one's intuition. We are seeking representations that utilize only elements that are "model primitives". A good example is given in Subsection 3.1, where the Offered Load at time t is presented as $R(t) = E[\lambda(t - S_e)]E[S]$. Here the Offered Load is represented as a function of S, the service time, and $\lambda(t)$ the arrival rate function, both "model primitives" . Note that the distribution of S_e is determined by S.
In Subsection 3.4 we implemented the above and presented the Offered Load of the "Tandem Queues Network"; for example we represented the Offered Load of the second queue of the tandem as $R_2(t) = E[\lambda(t - S_1 - S_{2e})] \cdot E(S)$. Here S_1 and S_2 are the service time of Queue 1 and Queue 2 respectively, again "model primitives". In Section 3, we presented several queueing networks that are more complicated then the Tandem network. In our research, we plan to

consider such systems and try to represent the Offered Load, again, in terms of model primitives: arrivals, services and routing. Thus, we seek to represent the Offered Load of a station in a complicated system, such as the Emergency Department network, using the model primitives in an insightful manner .

4.3 Staffing of the Emergency Department

The Offered Load is prerequisite for setting staffing levels (see [3]). The staffing problem is to determine the minimal required nurses/doctors subject to pre-specified quality of service constraints. In addition, the staffing problem must satisfy constraints such as nurses' and doctors' time-shifts, standard number of nurses per number of beds etc.
In systems where the arrival rate is time-homogeneous and the number of agents is large, it is customary to use the "square-root rule", $n = R + \beta \cdot \sqrt{R}$. Here n is the number of agents, R is the Offered Load and β is a parameter determined by the sought after service level. The problem at the Emergency Department is that the arrivals are not time-homogeneous and the number of agents is usually small. The work in [3] proposes a way to extend the "square-root rule" to a time varying environments: simply set $n(t) = R(t) + \beta \cdot \sqrt{R(t)}$, where $\{R(t),\ t \geqslant 0\}$ is our time-varying Offered Load. Such staffing remarkably, stabilizes performance over time. But Feldman [3] treated a single station. Our challenge is to extend it to a network setting, both Open and Fork-Join networks, and hopefully their combination. We shall then test our staffing procedures on a realistic Emergency Department scenarios, via its simulation, as described in the next research item.

4.4 Offered Load Estimation Using Emergency Department Simulation

We will use a generic Emergency Department simulation developed by Yariv Marmur in [7]. We will use the simulation with real data from a large hospital in Israel. Using this simulation, we will estimate the Offered Load of different service stations of the Emergency Department; then we will use those results in order to determine staffing levels as described previously. These will be validated, via the simulation, in order to develop guidelines for Emergency Department staffing.

References

[1] D. Bertsimas and G. Mourtzinou. Transient laws of non-stationary queueing systms and their applications. *Queueing Systems*, 25:115–155, 1997.

[2] S.G. Eick, W.A. Massey, and W. Whitt. The physics of the $M_t/G/\infty$ queue. *Operations Research*, 41(4):731–742, Jul. - Aug. 1993.

[3] Z. Feldman, A. Mandelbaum, W.A. Massey, and W. Whitt. Staffing of time-varying queues to achieve time-stable performance. *Management Science*, 2007.

[4] L. Green. *Operations Research and Health Care*. Kluer Academic Publisher, 2004.

[5] J. Keilson and L.D. Servi. Networks of non-homogeneous $M/G/\infty$ systems. *Operations Research*, 31:157–168, 1994.

[6] V.G. Kulkarni and V.G. Adlakha. Markov and Markov-regenerative PERT networks. *Operations Research*, 34(5):769–781, 1986.

[7] Y. Marmur. Developing a simulation tool for analyzing Emergency Department performance. *M.Sc. Thesis, Technion*, 2003. Supervised by the late Prof. David Sinreich.

[8] W.A. Massey and W. Whitt. Networks of infinite-server queues with nonstationary Poisson input. *Queueing Systems*, 13:183–250, 1993.

[9] R. Serfozo. *Introductiob to Stochastic Networks*. Springer-Verlang New York. inc., 1999.

[10] Y. Tseytlin. Queueing systems with heterogeneous servers: On fair routing of patients in emergency departments. *M.Sc. Thesis, Technion*, 2009.