Methods for airport terminal passenger flow simulation

Gábor Kovács, István Harmati, Bálint Kiss, Gábor Vámos, Péter Maráczy

Abstract — Increased air traffic has also caused major rise of passenger flow at airport terminals. In order to provide efficient and comfortable service at airports, passenger flow has to be improved, which has to be based on analysis of simulation results. This paper presents an evaluation of two methods for simulating passenger flow of an airport terminal. The terminal is decomposed to several zones, referred to as cells, each having its own behavior. Passenger flow between these cells is defined as a directed graph. The paper presents the difference equation based store–and–forward model and a Petri net based model for the simulation of passenger flow. Principles of passenger flow modeling by the two methods are presented, and detailed description of typical cell models are given for both approaches. The methods are evaluated on the simulation of a small-scale example. Based on the results, comparison on the two methods is given and a conclusion is drawn.

Keywords — Airport, Passenger flow modeling, Petri net, Simulation, Store–and–forward model

I. Introduction

Air traffic has been rising significantly in the last decades, resulting a daily average of about 10 000 scheduled commercial flights over the world. It is straightforward that the increase of traffic has also affected the passenger facilities of airport terminals. The most crowded airport, London Heathrow (UK) accommodates a daily average of 190 000 passengers.

As the first impression of passengers about their flight is the airport terminal, and how they can get from the entrance to the boarding gate, passenger handling facilities of airports have a great importance. Airport developers try to do their bests to guide the passengers smoothly from check-in to boarding, providing commercial and entertaining facilities for their comfort. However, inconveniences like long queues at security screening or long transfer routes may cause the passengers to leave the terminal with a bitter taste in their mouth. In addition, increased and improperly handled passenger flow might lead to several problems: increase of security threat, delay of connecting flights, passengers missing their flights etc. Moreover, IATA classifies airports based on factors like passenger density and transfer times [1], also depending largely on handling the passenger flow, and a lower grade might cause major airlines to avoid the use of the given terminal. Also, tenants of commercial facilities are interested in a high number of passengers visiting their locations, affecting the rental fees.

Evidently, the most effective moment to influence passenger flow is the phase of terminal planning or reconstruction. Careful design of the floor plan and extension of present terminals by new wings can significantly improve the speed and quality of passenger flow, although it is hard to predict the change of traffic in a timespan of decades. However, passenger routing and service can be improved also by modifications not affecting significantly the architectural basis. For example, by installing more security screening checkpoints, the queues can be remarkably shortened for a moderate cost. On the other hand, airports are interested in reducing costs without deteriorating passenger satisfaction, which can be achieved by careful resource planning, e.g. using cost–effective self–service check-in kiosks, closing some of the security screening checkpoints in off–peak periods or replanning the assignment of gates to connecting flights.

The aforementioned procedures have a common feature: they can not be achieved without modeling the passenger flow of the terminal. The effects of the modification of the floor plan or resource allocation have to be predicted before carrying out the physical work. However the expertise of airport employees might serve as a basis for prediction, they can only provide a rough qualitative estimation. In order to study the passenger flow precisely, an adequate numerical model has to be used. A simulation tool based on such a model can help a lot in the airport development process. Such simulation methods might help not just in finding optimal passenger paths, but also in discovering bottlenecks or designing adequate emergency evacuation routes.

Such simulation tools have to be based on well-formulated models. The main requirements against such models are that they should capture the behavior of passenger flow, allow the modeling of uncertainties and stochastics, be understandable for airport design experts and, not at least, they should support analysis based on simulation. Also, these models have to be flexible enough to deal with different airport terminal configurations (e.g. size, types and number of facilities), and computationally effective to allow fast simulation.

Due to the importance of the problem, various methods concerning the modeling, simulation and optimization of air traffic [3], [8] and passenger flow [14], [4], [16], [13], [20] ,[9], [18], [2] have been already proposed. Based on simulation results, approaches for improving airport passenger flow by design and optimization have been presented in [17], [15],

This work was supported in part by the Hungarian National Scientific Research Foundation grant OTKA K71762. Also, it is connected to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project, supported by the New Hungary Development Plan (Project ID: TMOP-4.2.1/B-09/1/KMR-2010-0002). Authors would like to thank the staff of Budapest Airport Ltd the information provided on the operation of airport passenger terminals.

Gábor Kovács, István Harmati, Bálint Kiss and Gábor Vámos are with the Department of Control Engineering and Information Technology, Budapest University of Technology and Economics, Budapest, Hungary. E-mail: {gkovacs,harmati,bkiss,vamos} iit.bme.hu

Péter Maráczy is with Budapest Airport Ltd., Budapest, Hungary. E-mail: peter.maraczy@bud.hu

[12].

This paper presents and compares two methods for passenger flow modeling adapted to airport terminals. The store–and–forward model, based on nonlinear difference equations, provides a macroscopic view on passenger flow using a global sampling time. The other method, using Petri net models, is a microscopic one, allowing the study of asynchronous evolution of passenger flow. Both methods provide a way to include uncertainties in the model, and they are both flexible enough to be adapted for different airport terminals. The latter property is related to cell–based modeling, which is a common feature of the two models.

The remaining part of the paper is organized as follows. Section II presents a common feature of the approaches, namely the modeling of an airport passenger terminal based on functional cells and their interconnections. Section III introduces background of store–and–forward models and models of typical terminal facilities, while Section IV discusses the Petri net-based approach and gives the models of the facilities. Section V presents the simulation results on the same small–scale terminal model using the two methods and compares their properties. Section VI concludes the paper.

II. Cell–based modeling of a passenger terminal

In order to adapt modeling methods to the needs of airport development, a flexible and unified approach has to be constructed. Such a method should allow the modeling of various sized airport terminals from smallest ones to large hubs, while being easy to parametrize and simulate. To meet the latter requirements, it is straightforward that some kind of decomposition of the terminal model should be carried out.

A. Cell–based modeling

Passenger flow, like any other flow type, should be defined between nodes. However, in the case of modeling airport terminals, nodes are not just artifacts for connecting flow directions, but they should serve as objects for affecting passenger flow. In order to develop an adequate model, their internal dynamics has to be also represented.

Therefore, the first step of modeling, independent of the method chosen, is to decompose the terminal to cells. Cells are basic building blocks of the terminal, and are described by their types and parameters. Their type (e.g. check-in counter, security screening etc.) defines how they affect passenger flow, while their parameters can be used to adapt the given general model to the properties of the given terminal. Each cell has its internal model, depending on its type, which can be defined in various ways. Formally, internal models of cells can be collected to the set $\mathcal{M}$, and its elements $M_i \in \mathcal{M}$ depend on the chosen modeling method, e.g. M_i is a set of difference equations when using store–and–forward models, or a Petri net when dealing with Petri net models.

Independent from the modeling method used, a cell can be formally defined as $C_i = (M_i, P_i)$, where M_i is the internal model of cell dynamics while $P_i = (p_{i,1}, p_{i,2}, \dots p_{i,n_i})$ is the set of the parameters associated to the cell with n_i as the number of parameters associated to the cell C_i. The set of the cells of which the model is built up from is denoted by $\mathcal{C}$.

Cells are nodes of the passenger flow graphs, which are connected by directed edges. Passenger flow is defined along these edges, so passengers are only allowed to pass to a drain cell connected to the source by a directed arc. Passengers choose freely the next cell to visit from reachable neighboring ones. Their habits are represented by branching rates assigned to the arcs. The branching rate of an arc from cell C_i to C_j is given by the mapping $B : \mathcal{C} \times \mathcal{C} \to [0, 1]$, so $B(i, j)$ defines the probability that a passenger leaving C_i will go to C_j. It is straightforward that if there is no arc from C_i to C_j, then $B(i, j)$ is zero and that $\sum_j B(i, j) = 1, \forall C_i, C_j \in \mathcal{C}$. Note that branching rates might be time-dependent, i.e. $B : \mathcal{C} \times \mathcal{C} \times \mathbb{R}^+ \to [0, 1]$ for continuous-time models ($t \in \mathbb{R}^+$) or $B : \mathcal{C} \times \mathcal{C} \times \mathbb{Z} \to [0, 1]$ for sampled models ($k \in \mathbb{Z}$).

Therefore, the model of a terminal is a pair $\mathcal{T} = (\mathcal{C}, B)$, consisting of the cells and branching rates associated to the flow between them.

B. Common cell types

Comparing the floor plans of different terminals, from smallest ones to large hubs, one can divide their parts to functional units, i.e. cells. However the parameters of the general cell models are different, their internal models are the same, so these cells can be used for modeling various terminals. In the followings the most common cell types a passenger passes by on his way from the entrance to the boarding gate will be summarized.

1) Check–in counter: Traditionally, check–in counters are the input points of the terminal, where passengers' tickets are checked and their hold baggages are processed. The check–in affects passenger flow as a delay, as the passengers have to wait in a queue until they can proceed to a free counter. Parameters commonly associated to check–in counters are the time the check–in procedure takes and the number of counters in operation.

2) Self–service check–in and baggage drop–off: Beside traditional check–in counters, more and more self–service check–in kiosks can be found at international airports, where passengers can choose their seats and print their boarding cards by using a computer terminal. Their main advantages compared to traditional check–in counters are their lower cost, as they can operate without using human resources, and their moderate floor need. However the procedure of self–service check–in for one passenger might need more time compared to check–in counters providing assistance of a trained employee, the high number of kiosks can reduce the average waiting time. Similarly to check–in counters, a waiting time and the number of operating kiosks can be used as parameters.

Since the handling of baggages is generally not possible at self–service check–in kiosks, passengers with hold baggage have to pass to one of the baggage drop–off counters, where they can check in their luggage. As it is a relatively fast procedure, one drop–off point is capable to handle the output of many check–in kiosks. Parameters associated to drop–off points are the time of the drop–off procedure and the number

of counters in operation. Passengers without hold baggage can pass by the drop–off counter, so they are handled faster.

3) Security screening: After checking in, passengers have to go through a security screening, where they and their carry–on luggages are searched for security threats. Depending on the regulations, the screening procedure might take several minutes, so these checkpoints are common bottlenecks of passenger flows. To the security screening the number of checkpoints in operation and the time of the screening procedure can be associated as parameters.

4) Hall: The hall serves as an area for passengers to move between other cells, therefore its parameters, corresponding to the time the passengers spend there, can significantly affect the passenger flow. These parameters depend largely on architectural factors (i.e. floor plan of the terminal), which can hardly be changed.

5) Retail unit: The importance of retail units like shops, cafes, restaurants is rising as airport terminals are becoming not only transportational, but also commercial facilities. Beside affecting the passenger flow significantly, their rental fees provide an important income to the airport operator, so their modeling is of paramount importance. After entering a retail unit, passengers usually spend some time browsing amongst the goods or studying the menu, and then decide to become customers or not. In the former case, passengers proceed to the cashier's desk, while in the latter case, they leave the cell. Parameters associated to retail units are the capacity of the given cell, the browsing time (time spent before deciding to be actual customer or not), customer ratio (ratio of passengers becoming actual customers), the number of cashiers desks in operation, and the length of the payment procedure.

6) Boarding gate: Boarding gates are the exit points of the passengers from the terminals. Passengers gather at the gate area, and after boarding, they leave the terminal one by one. Parameters associated to the boarding gates are the their capacities and the time of boarding procedure.

C. Example

Figure 1 shows the floor plan of the departure side of a small airport terminal, which will be used to evaluate the methods presented in the forthcomings. Beside the check–in counters, the airport operator provides a self–service check–in area with check–in kiosks and a baggage drop–off counter. After the check–in procedure, passengers proceed to security screening. Screened passengers arrive to the hall, where they can choose to visit one of the two shops, or proceed directly to the boarding gates. It is assumed that passengers pass only towards the boarding gates after the announcement of call for boarding.

The flow graph of the terminal is illustrated by Figure 2, with the cells listed by Table I. Note that the cell C_0 is not a real cell, it only represents the passengers arriving to the terminal, and used to be coherent with the notations. The branching rates $B(0,1)$ and $B(0,2)$ therefore correspond to the ratio of passengers choosing traditional or self–service check-in procedure. For example, $B(1,2) = 0$, since there is no arc leading to C_2 from C_1, while $B(1,3) = 1$, since all

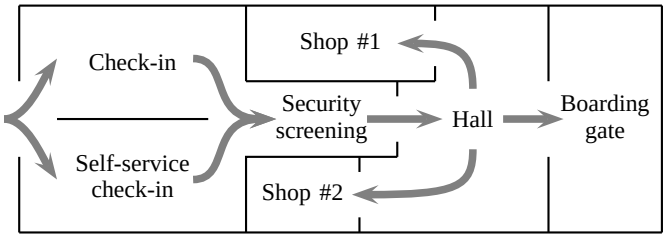

Fig. 1. Floor plan of a simple terminal

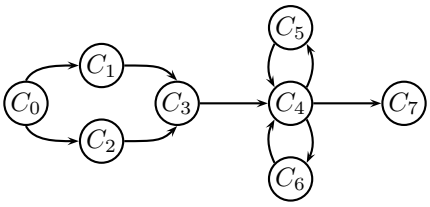

Fig. 2. Flow graph of the terminal

passengers leaving the check-in counters proceed to security screening, i.e. C_3. Branching rates, which are not zeros, are given by Table II.

III. Store-and-forward models

Store-and-forward models are widely used in urban and highway traffic modeling and control [7], [10], [11]. However, the versatility of the models allows their easy adaptation to passenger flow modeling needs.

A. Modeling background

The store–and–forward models use difference equations to describe the evolution of cells. Since these equations are well known formalisms of mathematics and control engineering, the modeling procedure is familiar to system engineers. However, on the other hand, they might be hard to understand for airport professionals.

Due to the use of difference equations, store–and–forward models are sampled ones. It means that the state of the cells is refreshed synchronously, at time instances defined by the global sampling time T. The synchronous property allows the tuning of the simulation: by choosing a large sampling time, the model will be less accurate but can be simulated faster, on the other hand, small sampling time results in higher execution time but detailed results.

Cell number	Cell type
C_0	*Generator*
C_1	Traditional check–in (check–in counters)
C_2	Self-service check-in (check–in kiosks and baggage drop–off)
C_3	Security screening
C_4	Hall
C_5	Shop #1
C_6	Shop #2
C_7	Boarding gate

TABLE I
CELLS OF THE EXAMPLE

Parameter	Value
B(0,1)	0.6
B(0,2)	0.4
B(1,3)	1
B(2,3)	1
B(3,4)	1
B(4,5)	$\begin{cases} 0.6 & \text{if } t < t_{boarding} \\ 0 & \text{if } t \geq t_{boarding} \end{cases}$
B(4,6)	$\begin{cases} 0.35 & \text{if } t < t_{boarding} \\ 0 & \text{if } t \geq t_{boarding} \end{cases}$
B(4,7)	$\begin{cases} 0.05 & \text{if } t < t_{boarding} \\ 1 & \text{if } t \geq t_{boarding} \end{cases}$

TABLE II
NON-ZERO BRANCHING RATES OF THE EXAMPLE

For each cell, an input queue and a functional unit is defined. The functional unit represents actions taken in the current cell (e.g. screening of passengers, sell of goods at a store), while the input queue represents passengers waiting for the given action.

The general store–and–forward model of the cell C_i is defined formally as follows:

$$\begin{aligned} \overline{x}_i(k+1) &= x_i(k) + T\sum_j B(j,i,k)u_j(k) - \\ &\quad -T\sum_j B(i,j,k)u_i(k) \\ x_i(k+1) &= \max(\min(Q_i, \overline{x}_i(k+1)), 0) \end{aligned} \tag{1}$$

where $x_i(k)$ is the state of the given cell, i.e. the number of passengers in the cell at the time instance kT, $B(i,j,k)$ is the branching rate of passengers leaving cell i and entering cell j. The parameter $u_i(k)$ represents the processing rate of the cell i in passengers per minute (PAX/min). The first equation describes the number of passengers in the given cell at the next sampling instance without considering any constraints, while the second equation assures that the number of passengers in the cell will be between zero and the capacity of the cell, denoted by Q_i. For the reason of simplicity, the time-dependece of branching rates will be omitted in the followings.

However, if one of the drain cells is saturated, passengers might choose to pass to another, not saturated cell accessible from the given cell. Therefore the actual branching rate can be expressed by

$$\begin{aligned} B'(i,j) &= \begin{cases} 0 & \text{if } B(i,j) = 0 \vee x_j \geq Q_j \\ B(i,j) & \text{otherwise} \end{cases} \\ \overline{B}(i) &= \sum_j B'(i,j) \\ B^*(i,j) &= \begin{cases} \frac{B'(i,j)}{\overline{B}(i)} & \text{if } \overline{B}_i > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

Then the state equation of the cell can be rewritten as

$$\begin{aligned} \overline{x}_i(k+1) &= x_i(k) + T\sum_j B^*(i,j)u_j(k) - \\ &\quad -T\textstyle\sum_j B^*(i,j)u_i(k) \\ x_i(k+1) &= \max(\min(Q_i, \overline{x}_i(k+1)), 0) \end{aligned} \tag{3}$$

The processing speed of a cell might depend on many factors corresponding to the given cell. These factors can be divided into two groups, namely constant parameters and decision variables. Constant parameters depend only on the model and they can not be changed on–line, i.e. during the simulation. Factors like the area of the check–in zone or the processing time of a security screening checkpoint correspond to the layout and operational rules of the terminal, so they are considered to be constant parameters. On the other hand, decision variables can be changed on–line, i.e. during the simulation, which allows the use of different control strategies. These decision variables represent the way how airport operators can influence the passenger flow. Factors like the number of check–in counters or security screening checkpoints in operation are considered as control variables.

Formally, the processing speed of a cell C_i, depending on these factors, is

$$\overline{u}_i(k) = f_i(x_i(k), p_{i1}, \ldots, p_{im_i}, d_{i1}, \ldots, d_{in_i}), \tag{4}$$

where $p_{i1}, \ldots, p_{im}$ are the constant parameters associated to the cell, while $d_{i1}, \ldots, d_{in}$ are the decision variables corresponding to the cell C_i. Note that f_i can be any arbitrary positive semidefinite nonlinear function. However, the processing speed of a cell is zero if all of its drain cells (i.e. the cells to which the output of the given cell flows) are saturated, so the processing speed is given as

$$u_i(k) = \begin{cases} 0 & \text{if } \forall j : B(i,j) \neq 0, x_j(k) \geq Q_k \\ \overline{u}_i(k) & \text{otherwise} \end{cases} \tag{5}$$

According to the definition of occupancy and processing speed, the parameter set of the cell C_i can be formally given as $P_i = (p_{i1} \ldots, p_{im}, d_{i1}, \ldots, d_{in})$.

B. Cell models

In the followings, store–and–forward models of the cells the example terminal consists of are given.

1) Check–in counters: The overall processing speed of the ensemble of check–in counters at the terminal depends on two factors: the processing speed of a single counter (i.e. the number of passengers checking in at a single counter in per minute) and the number of check–in desks in operation. The former factor will be denoted by p_{11} while the latter by d_{11}.

The processing speed of the cell reads

$$u_1(k) = \begin{cases} \min(\frac{x_1(k)}{T}, p_{11}d_{11}) & \text{if } x_3(k) < Q_3 \\ 0 & \text{if } x_3(k) \geq Q_3 \end{cases} \tag{6}$$

The first equation defines the normal operation, while the second expresses that processing speed of the check–in counters is zero if its output cell, the security screening is fully saturated.

Then, by denoting the arriving passenger flow by v, and the rate of passengers choosing traditional check–in by $B(0,1)$, the state equation of the check–in cell reads

$$\begin{aligned} \overline{x}_1(k+1) &= x_1(k) + TB^*(0,1)v(k) - Tu_1(k) \\ x_1(k+1) &= \max(\min(Q_1, \overline{x}_1(k+1)), 0) \end{aligned}, \tag{7}$$

Note that the factor $B^*(1,3) = 1$ has been omitted at the last term of the first equation.

2) Self–service check–in: To calculate the processing speed of the self–service check–in area, parameters including the average processing speed of a check–in kiosk (p_{21}) and a baggage drop–off counter (p_{22}) have to be considered. These parameters have to be carefully approximated as they depend largely on the expertise of the given passenger. Another important parameter is the ratio of passengers with hold baggage, denoted by $p_{23} \in [0,1]$. Decision variables corresponding to the self–service check–in are the numbers of check–in kiosks and drop–off counters in operation, denoted by d_{21} and d_{22}, respectively.

Since not every passenger has hold baggage, i.e. not all of them will pass to baggage drop–off, the cell should be decomposed to two sub–cells, one corresponding to the check–in kiosks, and the other to the drop–off counters. Variables corresponding to these two sub-cells will be denoted by the indices 1 and 2, respectively. The output flow of the cell is the sum of the output of the baggage drop-off point and the output of the check–in kiosks weighted by the ratio of passengers without hold baggage. However, since the queues passengers form can occupy the space allocated to the self–service check–in zone, no individual capacities are assigned to the sub-cells. Therefore, the processing speeds of the subcells and the self–service check–in cell reads

$$\begin{aligned} u_{2,1}(k) &= \min(\tfrac{x_{2,1}(k)}{T}, p_{21}d_{21}) \\ u_{2,2}(k) &= \min(\tfrac{x_{2,2}(k)}{T}, p_{22}d_{22}) \end{aligned} \tag{8}$$

$$u_2(k) = \begin{cases} (1-p_{23})u_{2,1}(k) + u_{2,2}(k) & \text{if } x_4(k) < Q_4 \\ 0 & \text{if } x_4(k) \geq Q_4 \end{cases} \tag{9}$$

The state equation of the subcells and the cell reads

$$\begin{aligned} \overline{x}_{2,1}(k+1) &= x_{2,1}(k) + TB^*(0,2)v(k) - Tu_{2,1}(k) \\ \overline{x}_{2,2}(k+1) &= x_{2,2}(k) + Tp_{23}u_{2,1}(k) - Tu_{2,2}(k) \\ \overline{x}_2(k+1) &= x_2(k) + TB^*(0,2)v(k) - Tu_2(k) \\ x_2(k+1) &= \max(\min(Q_2, \overline{x}_2(k+1)), 0) \end{aligned} \tag{10}$$

3) Security screening: The only parameter corresponding to the security screening zone is the processing time of the screening procedure, p_{31}. The airport operator can decide to open or close checkpoints, so the number of checkpoints in operation is considered to be a decision variable, and will be referred to as d_{31}.

The processing speed of the security screening cell reads

$$u_3 = \begin{cases} \min(\tfrac{x_3(k)}{T}, p_{31}d_{31}) & \text{if } x_4(k) < Q_4 \\ 0 & \text{if } x_4(k) \geq Q_4 \end{cases} \tag{11}$$

The number of passengers in the security screening zone can be computed as follows:

$$\begin{aligned} \overline{x}_3(k+1) &= x_3(k) + T(u_1 + u_2) - Tu_3 \\ x_3(k+1) &= \max(\min(Q_3, \overline{x}_3(k+1)), 0) \end{aligned} \tag{12}$$

Note that since the security screening is the only drain cell of the check–in cells and it emits passengers only to the hall, the branching rates can be omitted in the state equation.

4) Hall: The hall serves as an area for passengers to move between other cells, therefore its parameters correspond to the speed of passenger flow. Parameter p_{41} denotes the minimal speed of flow (in PAX/min) in case the hall is saturated, while p_{42} corresponds to the maximal free flow speed. Parameter p_{43} describes the limit of free flow, i.e. the number of passengers under which the flow speed is considered to be p_{42}. The processing speed of the hall is as follows:

$$\begin{aligned} \overline{u}_4(k) &= p_{41} + \left[1 - \tfrac{\max(x_4(k) - p_{43}, 0)}{Q_4 - p_{43}}\right](p_{42} - p_{41}) \\ u_4(k) &= \begin{cases} 0 & \text{if } x_5(k) \geq Q_5 \wedge x_6(k) \geq Q_6 \wedge \\ & \wedge x_7(k) \geq Q_7 \\ \overline{u}_4(k) & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

The equations above assure that until reaching the number of the free–flow limit, the hall will operate with its full processing speed. Then, being populated between the free–flow and its saturation limit, the processing of the hall speed decreases as the number of passengers increases.

The state equation of the cell reads

$$\begin{aligned} \overline{x}_4(k+1) &= x_4(k) + Tu_3(k) - TB^*(4,5)u_5(k) - \\ &\quad -TB^*(4,6)u_6(k) - TB^*(4,7)u_7(k) \\ x_4(k+1) &= \max(\min(Q_4, \overline{x}_4(k+1)), 0) \end{aligned} \tag{14}$$

5) Shop: Passengers entering a shop might become customers or just look around and leave without buying anything. Therefore, like in the case of the self–service check–in, the queue of the cell is divided into two queues, namely of the passengers browsing amongst the goods ($x_{5,1}$) and the passengers waiting at the cashier's desk ($x_{5,2}$). All passengers entering the shop are added to $x_{5,1}$ and stay there for a given time, representing the activity of looking around in the shop. Then, a given ratio of these passengers, who become actual customers, are put into $x_{5,2}$, while the others leave the shop. The ratio of customers is defined by $p_{53} \in [0,1]$. Processing speed of a cashier is given by p_{51} (PAX/min) while average time spent in the shop in sampling times is defined by p_{52}. The decision variable d_{51} denotes the number of cashier's desks in operation. The processing speed of the shop is as follows.

$$\begin{aligned} u_{5,1}(k) &= \begin{cases} (1-p_{53})\frac{x_5(k - floor(p_{52}))}{T} \\ \quad \text{if } k > \text{floor}(p_{52}) \wedge x_4 \leq Q_4 \\ 0 \\ \quad \text{otherwise} \end{cases} \\ u_{5,2}(k) &= \begin{cases} \min\left(\frac{x_{5,2}(k)}{T}, p_{51}d_{51}\right) \\ \quad \text{if } x_4 \leq Q_4 \\ 0 \\ \quad \text{otherwise} \end{cases} \\ u_5(k) &= (1 - p_5 3)u_{5,1}(k) + u_{5,2}(k) \end{aligned} \tag{15}$$

Then the state equation of the cell reads

$$\begin{aligned} \overline{x}_{5,1}(k+1) &= x_{5,1}(k) + TB^*(4,5)u_4(k) - \\ &\quad -Tu_{5,1}(k) \\ \overline{x}_{5,2}(k+1) &= x_{5,2}(k) + Tp_{53}u_{5,1}(k) - \\ &\quad -T_u 5,2(k) \\ \overline{x}_5(k+1) &= \overline{x}_{5,1} + \overline{x}_{5,2} \\ x_5(k+1) &= \max(\min(Q_5, \overline{x}_5(k+1)), 0) \end{aligned} \tag{16}$$

6) Boarding gate: The only parameter of the boarding gate is the speed of the boarding (p_{71} in PAX/min) and the corresponding decision variable is the number of the attendants handling the boarding procedure (d_{71}).

Since passengers arriving to the gate can leave only towards their flight after the boarding time and stay at the gate until the boarding, the processing speed of the cell reads

$$u_7(k) = \begin{cases} 0 & \text{if } k < \frac{T_{boarding}}{T} \\ \min(\frac{x_7(k)}{T}, p_{71}d_{71}) & \text{if } k \geq \frac{T_{boarding}}{T} \end{cases} \tag{17}$$

Then the state equation of the boarding gate is as follows

$$\begin{aligned} \overline{x}_7(k+1) &= x_7(k+1) + TB^*(4,7)u_4 - Tu_7 \\ x_7(k+1) &= \max(\min(Q_7, \overline{x}_7(k+1))) \end{aligned} \tag{18}$$

IV. Petri net models

Petri nets [6] are used widely for modeling and controlling concurrent systems from manufacturing lines [5] to traffic networks [19]. Petri nets allow the simulation of various systems by using a simple token game.

A. Modeling background

Petri nets are bipartite directed graphs, and its nodes can be divided to the sets of places and transitions, which are connected by weighted arcs. To each place, a marking (a nonnegative integer number) can be assigned and the state of the Petri net is represented by the marking vector $m = [m_i]^T$ consisting of the markings of each state m_i. Graphic representation of Petri nets uses circles for denoting places and bars for denoting transitions (see, for example, Figure 3). Directed arcs are evidently represented by arrows with their weights written next to them (omitting the weight corresponds to an arc with a weight of one). The marking of a state is represented by drawing an adequate number of tokens (filled circles) inside the symbol of the state. When modeling passenger flow, tokens might represent either passengers or resources, depending on the place they are associated to.

Formally, a Petri net can be described by a 5-tuple $N = (P, T, Pre, Post, m_0)$, where P is the set of places, T is the set of transitions with $P \cap T = \emptyset$, $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the input and output incidence mappings and m_0 is the initial marking.

Input and output incidence mappings define the weight of the arcs form places to transitions and vice versa, respectively. The place p_i is said to be the input place of the transition t_j if $Pre(p_i, t_j) > 0$, i.e. there exists an arc with a weight of at least one from p_i to t_j. Similarly, the place p_i is the output place of transition t_j if $Post(p_i, t_j) > 0$. In order to describe the relations in a more compact form, the mappings Pre and $Post$ can be given by the input and output incidence matrices, I^- and I^+, respectively. The elements of the matrices are $I^-_{(i,j)} = Pre(p_i, t_j)$ and $I^+_{(i,j)} = Post(p_i, t_j)$. The matrix $I = I^+ - I^-$ will be referred to as the incidence matrix of the net.

Dynamics of Petri nets are connected to the firings of transitions, which might (and usually) change the marking of a net. A transition t_i is said to be enabled, if $m_j > Pre(p_j, t_i), \forall p \in P$, i.e. if each of its input places contains at least as much tokens as the weight of the arc leading from it to t_i. The enabling degree of a transition is the number $e_i = \text{floor}(\min(m_j / Pre(p_j, t_i)) : Pre(p_j, t_i) > 0)$, i.e. the number of firings allowed by the marking of its input places. An enabled transition t_i might fire, which means that it takes away $Pre(p_j, t_i)$ tokens from each of its input place p_j and places $Post(p_j, t_i)$ tokens to its output places, thus changing the marking, i.e. the state of the net. If transitions firing at a time are assembled to a firing vector $s = [s_i]^T$ with s_i is the number of firings of the transition t_i (note that a transition might fire e_i times simultaneously), then the marking after firing is given by

$$m' = m + Is, \tag{19}$$

which means that the dynamics of Petri nets can be expressed as linear equations.

It is possible that two transitions t_j and t_k are enabled simultaneously, but their simultaneous firing is not enabled, e.g. they share a common input place p_i such that $m_i \geq Pre(p_i, t_j)$ and $m_i \geq Pre(p_i, t_k)$ but $m_i < Pre(p_i, t_j) + Pre(p_i, t_k)$. This situation is called a conflict, and needs to be handled. One of the possibilities to overcome conflicts, used in this paper, is to choose the firing transition in a random way. A probability $P(t_i)$, corresponding to the branching rates, is assigned to each transition, and the transition to fire from a conflicting set is selected according to it.

Time, which plays paramount importance in passenger flow modeling, can be included to Petri nets in two ways. T-timing means that a delay is attached to each transition, while P-timing deals with delays associated to places. In this paper, the latter will be used (however, note that P-timed and T-timed Petri nets can be converted to each other), and a (possibly zero) sojourn time $\tau_i \geq 0$ is associated to each place t_i. If a token is added to the place t_i at time t by the firing of a transition, it is not available for firing (i.e. it can not be counted when determining enabled transitions) prior to the time $t + \tau_i$. The sojourn time τ_i might also be stochastic, and will be represented by its lower and upper bounds in this paper, between which the actual value will be generated according to normal distribution. At $t = 0$ all tokens are considered to be enabled for firing.

B. Petri net models of cells

In the followings, the Petri nets of different cell types the example terminal consists of will be given. In order to carry out the simulation, these models have to be interconnected according to the flow graph of the terminal. To do so, input and output transitions of the neighboring cells have to be merged.

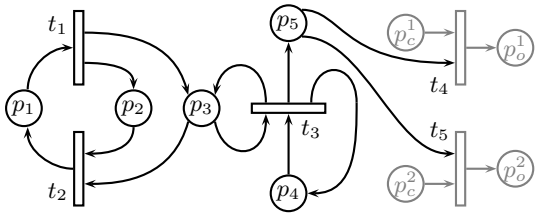

Fig. 3. Petri net model of the generator

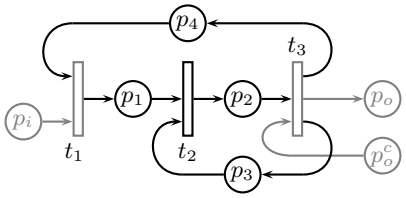

Fig. 4. Petri net model of the check–in counters

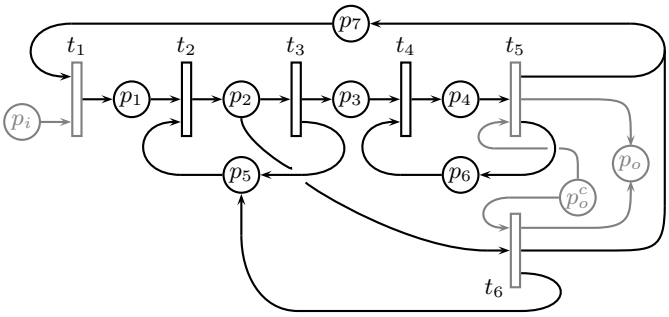

Fig. 5. Petri net model of the self–service check–in area

In the following figures, the input and output transitions are denoted by gray color, and places of neighboring cells connected to these transitions appear also in gray.

1) Generator: In order to simulate the arrival of the passengers at the airport, the net representing the terminal is extended by a generator cell C_0. Illustrated by Figure 3., the tokens representing passengers enter the terminal by transition t_3. The firing of t_3 removes a token from place p_4 ($m_{4,0} = 1$), but beside placing a token to p_5, it immediately replaces the token at p_4. The next time t_3 will fire depends on the sojourn time τ_4, so it can be used to define the density of passengers arriving to the terminal. From p_5, tokens proceed through transitions t_5 or t_6 towards the output places p_o^1 and p_o^2, respectively. By setting the probabilities of transitions t_5 and t_6, the ratio of passengers choosing traditional or self–service check–in can be given.

The other part of the net is responsible for enabling the inflow of passengers only for a given time. The initial markings are $m_{1,0} = 1$, $m_{2,0} = m_{3,0} = 0$, therefore transition t_1 fires at time $t = 0$, and tokens are placed to p_2 and p_3, making the firing of transition t_3 possible. As the sojourn time of p_3 is zero, the time tokens are removed from p_3 and p_2 is determined by τ_2. It means that t_3 is enabled periodically in the time intervals $[n\tau_1, n\tau_1 + \tau_2), n \in \mathbb{N}$. Thus, by setting e.g. $\tau_1 = 30$ min, $\tau_2 = 120$ min and the simulation time to 150 min, passengers will enter the terminal in the first 120 minutes.

2) Check–in: The model of the check–in counters, depicted by Figure 4, represents a multi-server queue. The model contains two capacity-places, namely p_3 with its marking corresponding to the number of free check–in desks, and p_4 with its marking denoting the number of free places in the queue. The initial marking of these places therefore corresponds to the parameters of the check–in area: $m_{3,0}$ gives the number of check–in counters in operation, while $m_{4,0}$ denotes the maximal capacity of the check–in area.

Passengers enter the banking queue of the check–in counters through transition t_1, which is only enabled if there is at least one token at p_4, i.e. the area is not saturated. The marking of place p_1 represents the number of passengers waiting in the queue, who can proceed to the check–in procedure of one of the counters is free, represented by the presence of a token at p_3. Tokens at p_2 correspond to check–in operations in progress, and the length of the check–in procedure can be given by the sojourn time τ_2. Finishing the procedure is represented by the firing of the transition t_3, which places a token to p_3 (meaning that a counter has become free), to p_4 (i.e. a passenger has left the queue) and to the output place, p_o. Note that the firing of t_3 requires also a token at p_o^c, i.e. a free place in the queue of the next cell.

3) Self-service check–in: The model of the self–service check–in area (see Figure 5) is similar to the one of the check–in counter. However, in this case there are two queues. Passengers waiting in the first queue are represented by tokens at p_1, while the self–service check–in procedure is taken place at p_2 (note that marking of p_5 corresponds to the number of free check–in kiosks). However, from p_2, the tokens can be moved by two transitions, t_3 and t_6. The former corresponds to the flow of passengers with hold luggage, who proceed to the queue of baggage drop–off (p_3). The handling of baggages is represented by the place p_4, while p_6 corresponds to to the number of free drop–off counters. Tokens representing passengers with hold luggage proceed from p_4 to the output place p_o through transition t_5, while passengers with only cabin baggage pass directly from p_2 to the output place through transition t_6.

Parameters corresponding to the average time of the check–in and baggage drop–off procedure are represented by the sojourn times τ_2 and τ_4, respectively. The number of check–in kiosk and baggage drop–off points can be given by the initial markings $m_{5,0}$ and $m_{6,0}$, while the capacity of the self–service check–in area is specified by $m_{7,0}$. The ratio of passengers with or without booked luggage can be set by the probabilities $P(t_3)$ and $P(t_6)$ of transitions t_3 and t_6.

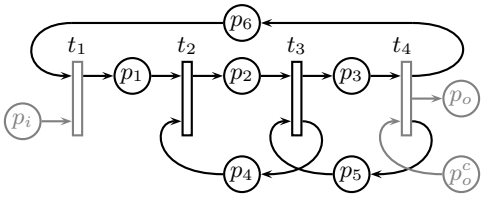

Fig. 6. Petri net model of security screening

4) Security screening: As shown by Figure 6., the queue of security screening is represented by the place p_1, where tokens can pass through transition t_1, guarded by the capacity place p_6. The screening procedure is represented by p_2, fed by the transition t_2, needing also a passenger in the queue (a token at p_1) and an available screening checkpoint (a token at p_4) to be fired. The end of the screening procedure is represented by the firing of transition t_3, placing a token at p_4 denoting that a checkpoint has become free and another token to p_3, representing the benches where passengers pack their baggages, put on their shoes and belts etc. after the screening. Note that t_3 is guarded by the capacity place p_5, with its marking corresponding to the free places at the benches area. After a short time passengers leave the security screening zone, represented by transition t_4, placing tokens also to capacity places p_5 and p_6.

The capacity of the security screening zone and the benches can be given by the initial markings $m_{5,0}$ and $m_{6,0}$, respectively, while $m_{4,0}$ corresponds to the number of security screening checkpoints in operation. Parameters of the screening procedure and behavior of the passengers are represented by the sojourn times τ_2 and τ_3.

5) Hall: The Petri net model of the hall is depicted by Figure 7. Tokens representing passengers enter the hall through transition t_1, guarded by the capacity place p_2. As the hall serves as a lobby for accessing other cells, tokens at place p_1 might leave the cell through various transitions (for the sake of simplicity, only two of such transitions are present at Fig. 7, however, the arcs of the transition leading to Shop #2 is similar to the one leading to Shop #1, i.e. t_3). The difference between the output transitions t_2 and t_3 is their availability in time. Places p_3, p_4 and p_5 realize an enabling subnet, similar to the one of the generator. Its role is to enable the transitions towards the shop cells (t_3 at the figure) only until the boarding is announced. Prior to the call for boarding, passengers might pass towards the shops or the gate (represented by transitions t_3 and t_2, respectively) according to the prescribed probabilities $P(t_2)$ and $P(t_3)$, but after the boarding call they proceed to the boarding gate with a probability of 1.

The capacity of the hall can be defined by the initial marking of p_2, while the probabilities of transitions t_2 and t_3 (and possibly others) are set according to the corresponding branching rates. The average time spent at the hall is represented by the sojourn time τ_1. The place p_3 of the enabling subnet is marked during the boarding call, while p_4 and p_5 should be marked to enable the transition t_3 (and possibly others) at the time interval $[0, T_{call})$, where T_{call} is the time of the announcement. The initial markings should be set to $m_{3,0} = 1$, $m_{4,0} = m_{5,0} = 0$, and since transition t_4 is enabled at $t = 0$, the token at p_3 will be immediately removed, and tokens will be placed at p_4 and p_5. The sojourn time τ_3 should be set to the length of a boarding call, τ_5 according to the time between the boarding calls and $\tau_5 = 0$.

6) Shop: The model of a shop is shown by Figure 8. The entry of passengers to the shop is represented by transition t_1, guarded by the capacity place p_6. Tokens at place p_1 represent passengers browsing in the shop, who might decide

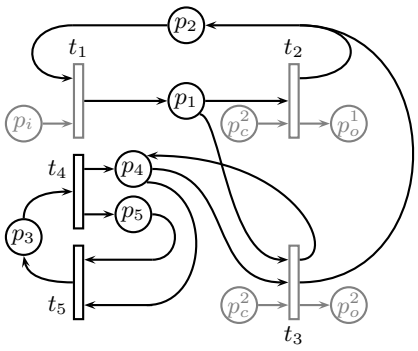

Fig. 7. Petri net model of the hall

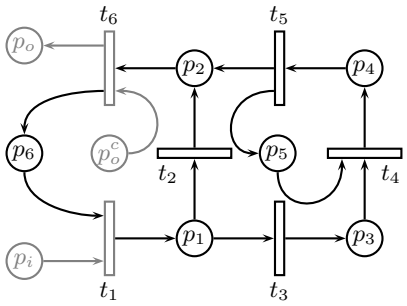

Fig. 8. Petri net model of a shop

to buy something (transition t_3) or to leave the shop without becoming actual customers (transition t_2). Former passengers proceed to the cashier queue, represented by place p_3. The payment procedure is modeled by place p_4, which can be reached through transition t_4, requiring also a free cashier's desk (token at p_5). After the payment, tokens representing passengers are transferred to the place p_2, and the marking of p_5 is increased to denote that a cashier's desk have been freed. From p_2, the tokens can leave the cell through t_6 if the hall is not fully saturated.

Parameters of the shop include its capacity (initial marking $m_{6,0}$ and the number of cashiers (initial marking $m_{4,0}$). The time payment takes can be specified by the sojourn time τ_4, while the ratio of the actual customers can be set by assigning appropriate probabilities to the transitions t_2 and t_3.

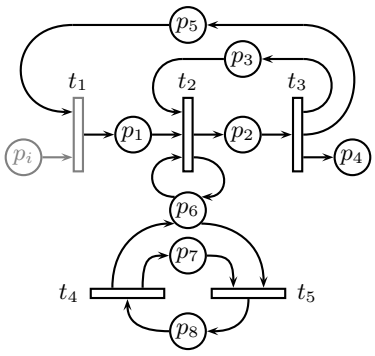

Fig. 9. Petri net model of the boarding gate

7) Boarding gate: As shown by Figure 9, passengers arriving to the boarding gate are represented by the marking of place p_1. The boarding procedure is modeled by the place p_2, which needs additional resources (i.e. an attendant) beside the passenger itself, represented by the tokens at p_3. Tokens representing boarded passengers are collected to the drain place p_d. Note that passing the tokens from p_1 to p_2 is possible through transition t_3, guarded by an enabling subnet (places p_6, p_7, p_8). This subnet allows the firing of t_3 only if there is a token at p_6, corresponding to the fact that the boarding is allowed only in a predefined time window. The initial marking of places p_6 and p_7 are $m_{6,0} = m_{7,0} = 1$, while $m_{8,0}$=0, allowing transition t_5 to fire at $t = 0$, thus disabling the firing of t_2. The transition t_2 will be enabled again only after the sojourn time τ_8, for the time period defined by τ_7 ($\tau_6 = 0$).

The capacity of the boarding gate and the number of attendants handling the procedure can be given by the initial markings $m_{5,0}$ and $m_{3,0}$, respectively. Timing of the boarding can be specified by the sojourn times τ_7 and τ_8, while the time the procedure takes can be given by τ_2.

V. Simulation results

In order to compare the methods, the passenger flow of the terminal presented in Section II-C has been simulated with the same parameter settings using both store–and–forward and Petri net models.

The length of the simulation was chosen to be 240 minutes. Passengers arrive to the terminal at the first 150 minutes, and the incoming flow is 3 passengers per minutes. The boarding call is active from $t = 120$ minutes, but the boarding procedure starts at $t = 130$ minutes.

It is assumed that 60% of the passengers choose traditional check–in at one of the counters, while 40% prefers self–service check–in. The ratio of passengers with hold luggage is 70%. Parameters of the store–and–forward and Petri net models are given by Table III and IV, respectively. In case of paramteres of the Petri net models, the first indices denote the cell, while the second indices correspond to place and transition indices of Sections IV-B.1 – IV-B.7.

The occupancy of cells, i.e. number of passengers in the cells is shown by Fig. 10 and Fig. 11 for the store–and–forward and Petri net models, respectively. Flow rates of the store–and–forward model are depicted by Fig. 12 while Fig. 13 shows the flow rates based on the simulation of the Petri net model.

Looking at the figures, it can be observed at first sight that shapes of the plots are similar, but there are some minor differences. First of all, results of the Petri net simulation show oscillatory behavior, while curves presenting the results of the store–and–forward model are smooth. This phenomenon results mainly from the fact that store–and–forward models are sampled ones, i.e. the occupancy of cells are calculated at discrete time instances, and curves are displayed using an interpolation between sampling instances. On the other hand, the Petri net model is an asynchronous one with stochastic sojourn times, so the change of markings caused by all aperiodically fired transitions are shown by the plots.

Plots corresponding to the check–in cell show that passengers are handled slower at the check–in counters as they arrive.

Parameter	Value	Description
		Incoming flow
v	3	Incoming passenger flow (PAX/min, $t \leq 150$)
	0	Incoming passenger flow (PAX/min, $t > 150$)
		Check–in
p_{11}	0.5	Processing speed of a counter
d_{11}	3	Number of counters in operation
Q_1	50	Capacity of the check–in (PAX)
		Self–service check–in
p_{21}	0.4	Processing speed of a check–in kiosk
p_{22}	1	Processing speed of a drop–off counter
p_{23}	0.7	Ratio of passengers with hold baggage
d_{21}	3	Number of check–in kiosks in operation
d_{22}	1	Number of drop–off counters in operation
Q_2	30	Capacity of the self–service check–in area (PAX)
		Security screening
p_{31}	0.8	Processing speed of a checkpoint
d_{31}	3	Checkpoints in operation
Q_3	50	Capacity of security screening (PAX)
		Hall
p_{41}	4	Processing speed of saturated hall
p_{42}	8	Free flow speed
p_{43}	50	Limit of free flow (PAX)
Q_4	200	Hall capacity (PAX)
		Shop #1
p_{51}	1	Processing speed of a cashier
p_{52}	2	Browsing time (in samples)
p_{53}	0.3	Ratio of customers
d_{61}	2	Number of cashiers' desks in operation
Q_5	80	Capacity of Shop #1 (PAX)
		Shop #2
p_{61}	0.3	Processing speed of a cashier
p_{62}	1	Browsing time (in samples)
p_{63}	0.9	Ratio of customers
d_{61}	1	Number of cashiers' desks in operation
Q_6	20	Capacity of Shop #2 (PAX)
		Boarding gate
p_{71}	3	Processing speed of boarding
d_{61}	2	Number of attendants at boarding
Q_7	120	Capacity of the boarding gate (PAX)

TABLE III
PARAMETERS OF THE STORE–AND–FORWARD MODEL (PROCESSING SPEEDS GIVEN IN PAX/MIN)

This fact corresponds the parameter set as an average of 3 passengers arrive per minute, of which 60% proceeds to the traditional check–in, but the three counters in operation can handle only 1.5 passengers per minute. Therefore, the queue at the check–in fills up, and it gets saturated at around $t = 140$ minutes. Then, as no passangers arrive from $t = 150$ minutes, the occupancy of the check–in decreases to zero.

On the other hand, plots show that even self–service check–in is slower that the traditional one, three kiosks and a single baggage drop–off counter can handle the incoming flow of passengers. The store–and–forward model shows a constant occupancy between $t = 10$ min and $t = 140$ min, and the plot of the Petri net simulation oscillates also around the value of 10. However, at $t = 140$ minutes, there is a significant peak at the occupancy plot of the store–and–forward model. The reason for it is the saturation of the traditional check–in cell, which causes that all arriving passengers pass to

Parameter	Value	Description
		Generator
$m_{01,0}$	1	Initial marking of p_{01}
$m_{04,0}$	1	Initial marking of p_{04}
τ_{01}	90 min	Sojourn time of p_{01}
τ_{02}	150 min	Sojourn time of p_{02}
τ_{04}	[10,30] sec	Sojourn time of p_{04}
		Check–in
$m_{13,0}$	3	Check–in counters in operation
$m_{14,0}$	50	Capacity of check–in
τ_{12}	[90,150] sec	Length of the check–in procedure
		Self–service check–in
$m_{25,0}$	3	Check–in kiosks in operation
$m_{26,0}$	1	Baggage drop–off counters in operation
$m_{27,0}$	30	Capacity of self–service check–in
τ_{22}	[100,200] sec	Length of self–service check–in
τ_{24}	[30,90] sec	Length of baggage drop–off
		Security screening
$m_{34,0}$	3	Checkpoints in operation
$m_{35,0}$	3	Number of benches
$m_{36,0}$	50	Capacity of security screening
τ_{32}	[60,90] sec	Length of the screening procedure
τ_{33}	[30,60] sec	Length of arrangement after screening
		Hall
$m_{42,0}$	200	Hall capacity
$m_{43,0}$	1	Initial marking of p_{43}
τ_{41}	[8,10] min	Time spent in the hall
τ_{43}	120 min	Sojourn time of p_{43}
τ_{45}	120 min	Sojourn time of p_{45}
		Shop #1
$m_{55,0}$	2	Number of cashiers' desks in operation
$m_{56,0}$	80	Capacity of Shop #1
τ_{51}	[300,900] sec	Browsing time
τ_{54}	[30,90] sec	Length of payment procedure
		Shop #2
$m_{65,0}$	1	Number of cashiers' desks in operation
$m_{66,0}$	20	Capacity of Shop #1
τ_{61}	[200,400] sec	Browsing time
τ_{64}	[100,300] sec	Length of payment procedure
		Boarding gate
$m_{73,0}$	2	Number of attendants at boarding
$m_{75,0}$	120	Capacity of the boarding gate
$m_{76,0}$	1	Initial marking of p_{76}
$m_{77,0}$	1	Initial marking of p_{77}
τ_{72}	[10,30] sec	Length of boarding procedure
τ_{77}	110 min	Sojourn time of p_{77}
τ_{78}	130 min	Sojourn time of p_{78}

TABLE IV
PARAMETERS OF THE PETRI NET MODEL

the self–service check–in zone. Also the security screening cell saturates at $t = 140$ minutes, so passengers can not leave the check–in zone, increasing also its occupancy and leading to a transient decrease of processing speed to zero. Due to its asynchronous and stochastic properties, the Petri net model does not show such abrupt changes, but an increased occupancy can be observed in the period of $t = 120 - 150$ minutes.

The bottleneck of the terminal is the security screening. As shown by the plots, it can not handle the flow of passengers checked in, and it gets to the limit of saturation. According to the store–and–forward model, the cell gets actually saturated, but the Petri net model shows that even the security screening is working on its limits, it does not get actually saturated. The difference between the two results is also related to the different philosophy of the two approaches.

The plots of the hall corresponding to the two methods show significant differences. Although the figures corresponding to the two methods both show the increase of occupancy from the beginning of the simulation with a notable peak around the time of the boarding call, the slopes and numerical values are different. The reason for these differences is that in case of the store–and–forward model, the processing speed of the hall depends on the number of passengers in the cell (see Section III-B.4), while the sojourn times of the Petri net model are independents from the occupancy. Looking at Fig. 12, it can be seen that the processing speed decreases at $t = 100$ min, where the occupancy of the hall passes the limit of free flow, i.e. passenger can pass by a lower speed. The peak at around $t = 130$ minutes corresponds to the announcement of the boarding call. As described above, passengers are assumed to proceed to the gate after the boarding call and not pass towards the shops. Since the capacity of the gate is limited, many of them have to stay in the hall for a while.

Clearly, the passenger flow of the hall influences its neightbouring cells. Both methods show that the number of passengers at Shop #1 increases, and due to the differences in the processing speed of the hall, results of the Petri net simulation even show saturation behavior. Shop #2 has a much lower capacity, so it gets saturated early, and stays saturated until the boarding call, from when no passengers arrive to the shops.

As 5% of the passengers proceeds towards the boarding gate when arriving to the hall, the occupancy of the gate starts increasing from the first minutes of the simulation. However, as it can be clearly observed on the plot of the store–and–forward model, the slope changes significantly after the announcement of the boarding call. Since from that time 100% of the passengers pass towards the hall, its occupancy rises near the limit of saturation. In case of the Petri net model, the boarding gate itself gets saturated, and since passengers from the hall take the place from those who board the airport, the cell remains saturated until around 175 minutes. On the other hand, plots of the store–and–forward model show that the decreased speed of passenger flow of the hall prevents the boarding gate from being saturated. Moreover, since the boarding speed is higher than the processing speed of the hall, the occupancy of the gate shows a decreasing trend from $t = 130$ min. However, as passengers leave the hall, its processing speed rises, and when it reaches the speed of boarding, the occupancy of the gate start rising again.

Simulations were carried out in Matlab environment. The simulation time was 13 milliseconds for the store-and-forward model and 12.41 seconds for the Petri net model on a laptop computer with an Intel Core 2 Duo processor, which shows a significant benefit of the former method. Also, while the computational time of the store–and–forward model depends on the number of cells and the sampling time, the computational time of the Petri net based simulation depends largely on the number of tokens, i.e. the number of passengers.

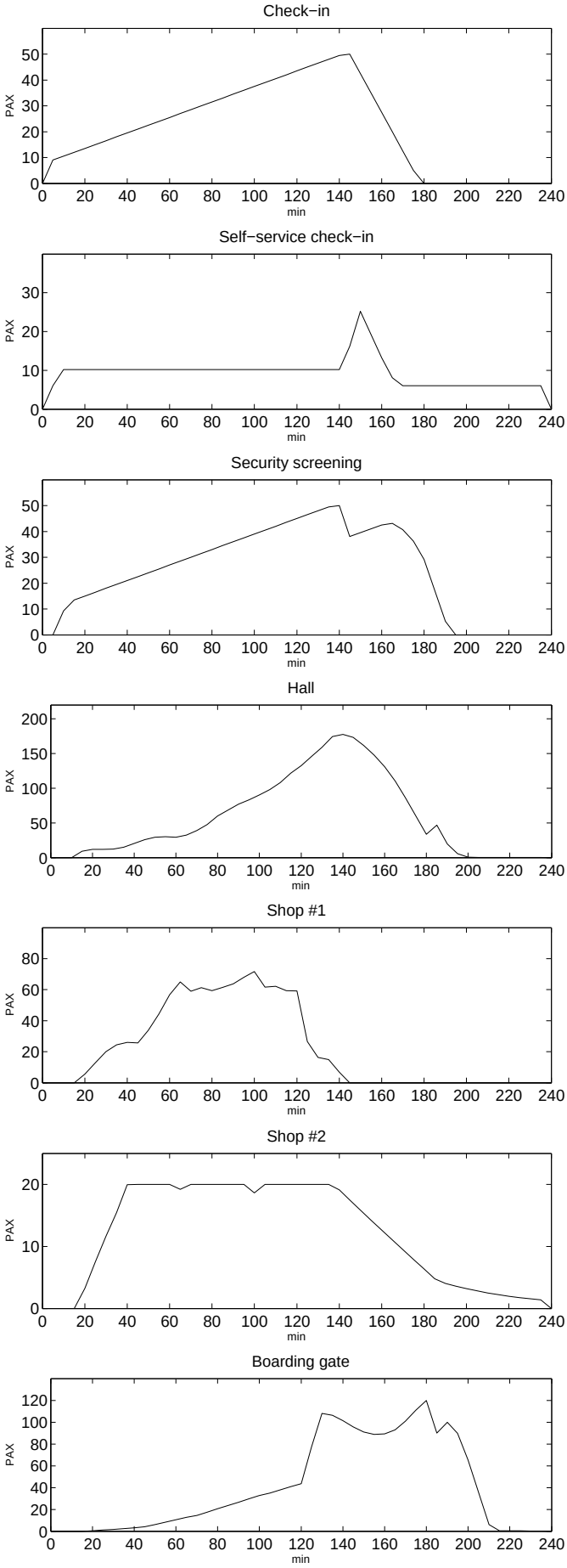

Fig. 10. Cell occupancy according to the store–and–forward model

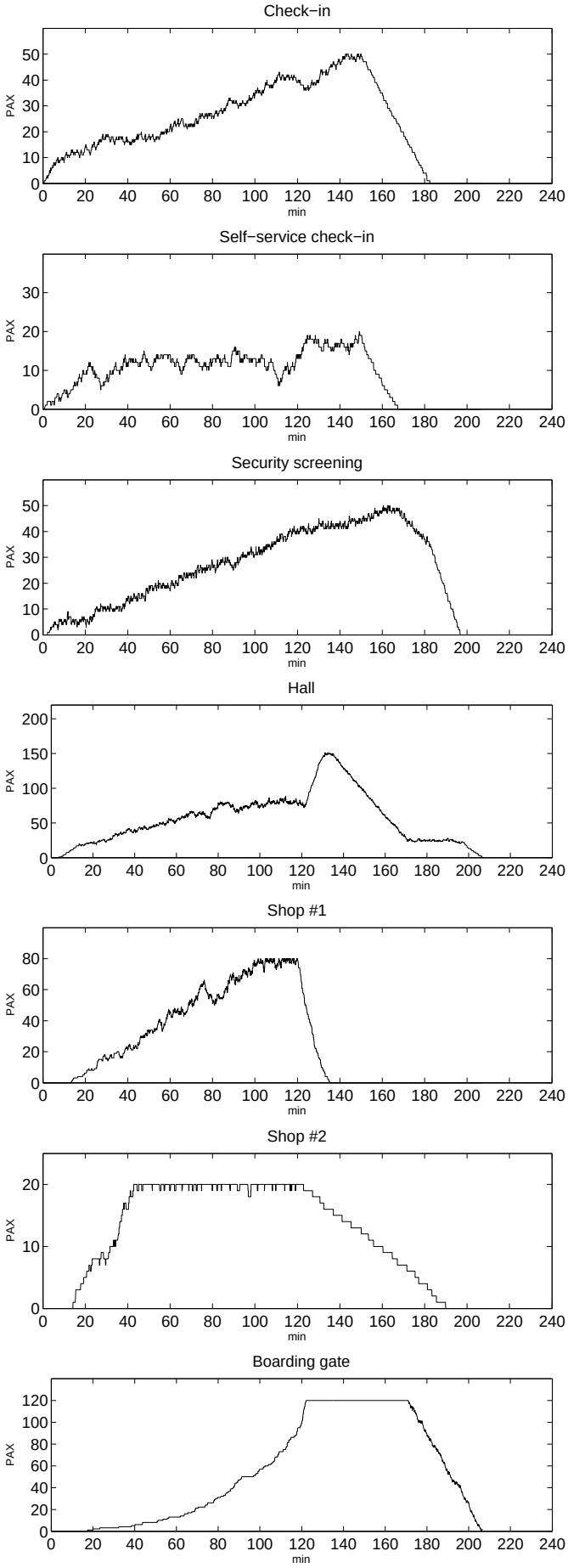

Fig. 11. Cell occupancy according to the Petri net model

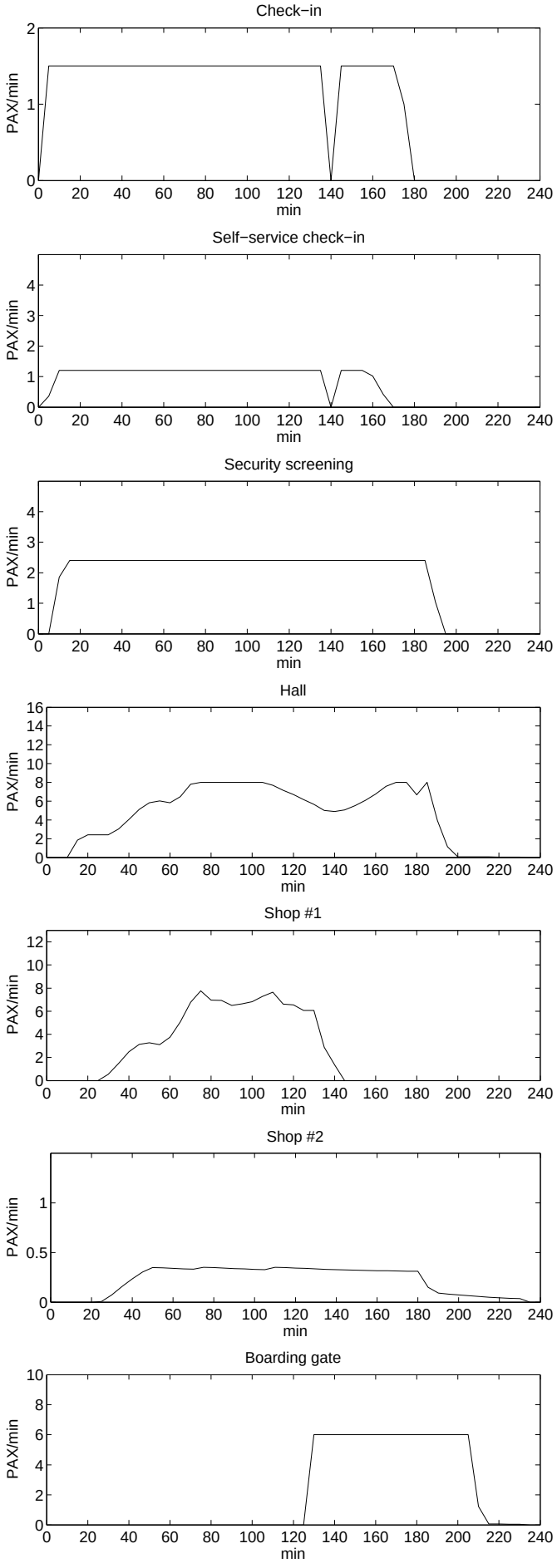

Fig. 12. Flow rates according to the store–and–forward model

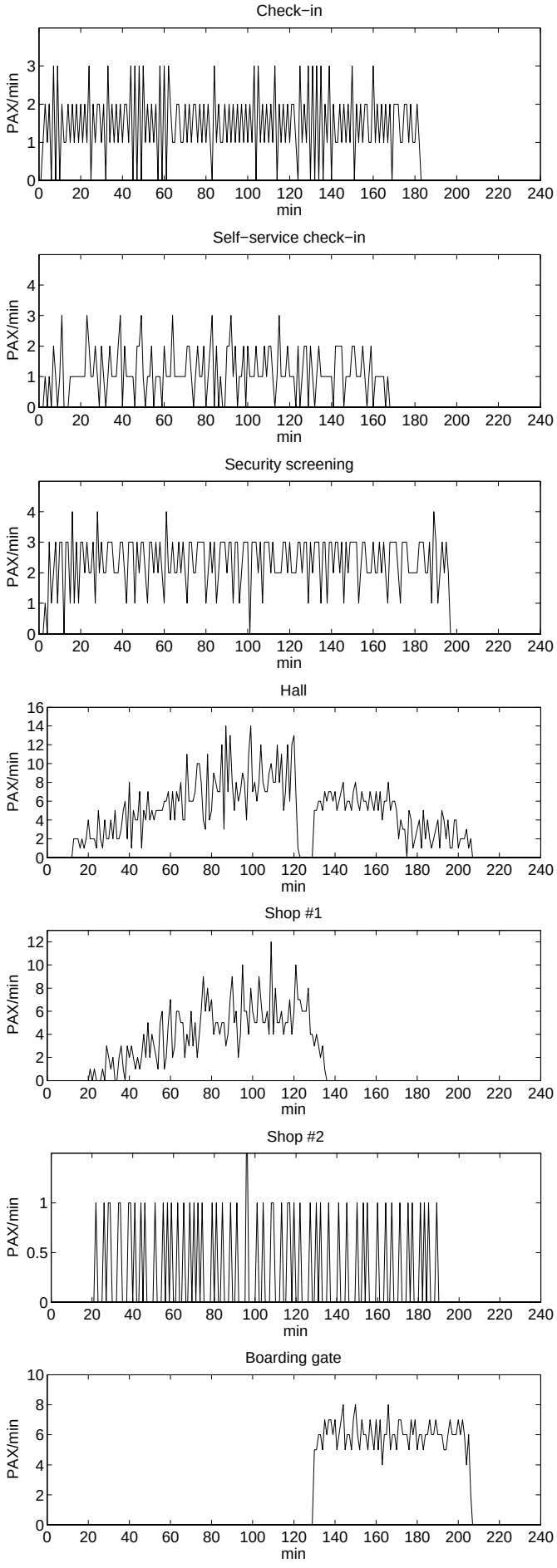

Fig. 13. Flow rates according to the Petri net model

VI. Conclusion

Two methods for passenger flow modeling were evaluated on the example of a small-scale terminal model. Results obtained by simulations based on store–and–forward and Petri net models are similar, however, some differences arise from the details of modeling principles. The Petri net model is more detailed, providing a way to include non-deterministic timing parameters, but its computational need rises with the number of passengers. Store–and–forward models needs low computational resources, however their accuracy is limited due to their sampled behavior. Therefore a store-and-forward model based simulation might serve for a draft macroscopic evaluation of passenger flow, while final, accurate results should be deduced from the simulation of the corresponding Petri net model.

Such simulations might help airport operators to analyze the passenger flow of the terminals, and to optimize routing and resource allocation to improve the efficiency of passenger handling. Future work includes evaluation of the methods on large-scale models, and improving their modeling capabilities by introducing variable step size for store-and-forward models and colored Petri nets. By introducing cost factors, the study of optimization methods adapted to passenger flow is also planned.

References

[1] *Airport development reference manual, 9th edition*. IATA - International Air Transport Association, 2004.

[2] E. Valentin A. Verbraeck. Simulation building blocks for an airport terminal modeling. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[3] Branko Bubalo and Joachim Daduna. Airport capacity and demand calculations by simulation - the case of berlin-brandenburg international airport. *NETNOMICS*, pages 1–21. 10.1007/s11066-011-9065-6.

[4] C. A. Chung and T. Sodeinde. Simultaneous service approach for reducing air passenger queue time. *Journal of Transportation Engineering*, 126:53–74, 2000.

[5] Marius Gigi Coman Daniela Coman, Adela Ionescu. A stochastic petri net approach for the manufacturing system design. In *Proceedings of 14th WSEAS International Conference on Systems (Volume II)*, 2010.

[6] R. David and D. Alla. *Discrete, Continuous and Hybrid Petri Nets*. Springer, 2005.

[7] V. Dinopoulou, C. Diakaki, and M. Papageorgiou. Applications of urban traffic control strategy tuc. *European Journal of Operational Research*, 175:1652–1665, 2006.

[8] P. Zoppoli E. Romano, L.C. Santillo. A static algorithm to solve the air traffic sequencing problem. *WSEAS Transactions on systems*, 7:682–695, 2008.

[9] G. Guizzi, T. Murino, and E. Romano. A discrete event simulation to model passenger flow in the airport terminal. In *Proceedings of the 11th WSEAS Conference on Mathematical Models and Computational Techniques in Electrical Engineering*, 2009.

[10] I. Harmati. Game theoretic control algorithms for urban traffic network. *WSEAS Transactions on systems and control*, 1:141–148, 2006.

[11] I. Harmati. Urban traffic control in game theoretic framework. In *Proceedings of the 5th WSEAS International Conference on System Scienece and Simulation Engineering*, 2006.

[12] Chin-Wu Chu Jin-Ru Yen, Hsiu-Fen Lin. Designing departure facility layout at airport passenger terminals. In *Proceedings of the 2nd WSEAS International Conference on Urban Planning and Transportation*, 2009.

[13] P. E. Joustra and N. M. V. Dijk. Simulation of check-in at airports. In *Proceedings of the the 2001 Winter Simulation Conference*, 2001.

[14] G. Romanin-Jacur L. Brunetta. Passenger and baggage flow in an airport terminal. In *Proceedings of the Industrial Simulation Conference*, 2004.

[15] A. R. Odoni M. A. Stamatopoulos, K. G. Zogragos. A decision support system for airport strategic planning. *Transportation Research C*, 12:91–117, 2004.

[16] S. van der Weij M. R. Gatersleben. Analysis and simulation of passenger flows in an airport terminal. In *Proceedings of the 1999 Winter Simulation Conference*, 1999.

[17] R. Moriyama N. Doshi. Application models in airport facility design. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[18] T. Oyama S. Takakuwa. Simulation analysis of international-departura passenger flows in an airport terminal. In *Proceedings of the 2003 Winter Simulation Conference*, 2003.

[19] Anthony Spiteri Staines. A colored petri net model for the france paris metro system. In *Proceedings of the 11th WSEAS International Conference on. Software Engineering, Parallel and Distributed Systems*, 2012.

[20] H. Che Y. Ju, A. Wang. Simulation and optimization for the airport passenger flow. In *Proceedings of International Conference on Wireless Comunications, Networking and Mobile Computing*, 2007.