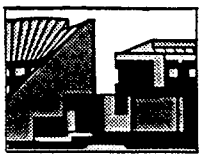
TEKNILLINEN KORKEAKOULU
DIGITAALITEKNIIKAN LABORATORIO
OTANIEMI

TEKNISKA HÖGSKOLAN
LABORATORIET FÖR DIGITALTEKNIK
OTNÄS



HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **B:** Technical Reports
No. **19**; December 1998

ISSN 0783–540X
ISBN 951–22–4424–1

PERFORMANCE ANALYSIS OF A TRAFFIC CONTROL SYSTEM USING STOCHASTIC PETRI NETS

MARKO MÄKELÄ, JANI LAHTINEN AND LEO OJALA

Digital Systems Laboratory
Department of Computer Science and Engineering
Helsinki University of Technology
Otaniemi, FINLAND

Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
P.O.Box 1100
FIN–02015 HUT, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Reports of the DIGITAL SYSTEMS LABORATORY

Series **A** : Research Reports ISSN 0783–5396

Series **B** : Technical Reports ISSN 0783–540X

The format of the report series was designed 1987 by Dr. *Jussi Orava* and covers with the aid of Mr. *Harri Rimmes* and Otapaino, Oy Dipoli Ab, Otaniemi
Offset printing: Picaset Oy, Lauttasaari
FIN-00210 HELSINKI, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **B**: Technical Reports
No. **19**; December 1998

ISSN 0783–540X
ISBN 951–22–4424–1

Performance Analysis of a Traffic Control System Using Stochastic Petri Nets

MARKO MÄKELÄ, JANI LAHTINEN AND LEO OJALA

Abstract: This paper discusses the modelling and performance analysis of signalized traffic control systems using the Deterministic and Stochastic Petri Net (DSPN) formalism. We develop a formal DSPN model for a group of interconnected signalized traffic intersections and present performance indices obtained from simulation runs of example systems.

Keywords: performance analysis, traffic control, Stochastic Petri Nets

Printing: Picaset Oy 1999

Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
P.O.Box 1100
FIN–02015 HUT, FINLAND

Phone: $\frac{09}{+358\text{–}9}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–9–451 3369
E-mail: lab@saturn.hut.fi

This talk was given at the Fourteenth UK Computer and Telecommunications Performance Engineering Workshop (UK PEW 1998), July 9–10, 1998, Edinburgh.

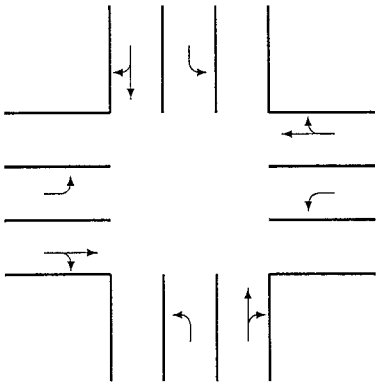


Figure 1: A simple road junction.

1 Introduction

Traffic control is an important topic that has great impact on the life in urban areas. Proper scheduling of traffic lights reduces congestion, speeds up the movement, prevents accidents, and improves the air quality by avoiding unnecessary use of engines.

Generally the performance analysis of complex traffic systems is done either by monitoring traffic flows near road junctions, or with simulators. Basic Petri Net formalisms [1] are rarely seen in this area. They are typically used for proving freedom of deadlocks and other important properties of distributed systems, which are not so interesting in traffic control. However, using the Deterministic and Stochastic Petri Net formalism [2, 3], it is possible to obtain performance indices either with analytic means or by simulation. To our knowledge, this was first done in [5], where a network of intersections with four-phase signal control was modelled with coloured DSPNs.

In this paper, we develop a formal DSPN model for a simple road junction with two-phase signal control (one phase for vertical traffic and another for horizontal traffic). The DSPN formalism has in the past decade shown to be a very effective tool in analyzing the performance of parallel systems. We show how the behaviour of a road junction can be intuitively described with a DSPN model, and how the model can be extended to describe networks of several interconnected traffic junctions.

In order to keep the crossroad model compact and readable, we use a coloured DSPN, just like [5]. Our notation and terminology are those of SWN (*Stochastic Well-Formed Nets* [4]), but our net does not belong to the class of SWNs because it contains deterministic timed transitions, which the SWN model

doesn't have.

We present a model of a generic four-road junction (Figure 1), with sensors for arriving vehicles. The model and its parameters are based on the conditions in Finland, where two-phased traffic control is common for small intersections.[6].

The parameters of our model must be set according to measured data (vehicles' arrival rates and time needed to pass the junction) or to the control system specification (green phase length etc.) In Section 2.5 we will show how junctions can be chained together.

This paper is organized as follows. In Section 2, a coloured DSPN model of a single intersection is developed and extended to cover multiple intersections. Simulation results from the model are represented in Section 3, and discussion can be found in Section 4.

2 DSPN model of a single intersection

The traffic signal control mechanism is rather simple, based on the following principles:

1. The signals are initially red for all directions.
2. When a car shows up, the signal is switched green for it to pass the junction.
3. Even if vehicles keep coming, the *maximum green time* will not be exceeded.
4. If there is a gap long enough in the queue of incoming vehicles, the signal will be switched to red.

Figure 2 shows our coloured DSPN model of the intersection given in Figure 1. A DSPN consists of places (round circles), which can contain tokens (tuples of integers in our model), and transitions (rectangles or thin lines) that can fire, consuming tokens from its input places (places from where an arrow-headed arc leads to the transition) and putting tokens to its output places (places where an arc leads to from the transition). A transition can only fire if the proper combination of tokens is present in its input places, and in the case of timed transitions, if enough time has elapsed. Our model also has inhibitor arcs, arcs with a small circle attached to a transition. The presence of a token in the inhibitor place can inhibit the firing of the transition. In coloured Petri Nets, arc expressions specify what kind of tokens the transitions require as

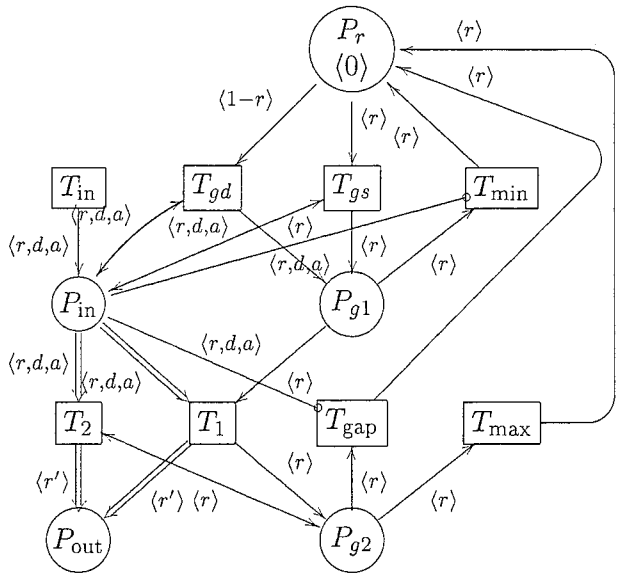


Figure 2: A high-level Petri Net model for the junction of Figure 1.

Table 1: Variables used in the arc expressions of Figure 2.

Symbol	Description		
r	Road	0	north–south
		1	east–west
r, d	Road, direction	0,0	north
		0,1	south
		1,0	east
		1,1	west
a	Action	0	drive straight ahead or turn right
		1	turn left

input or produce as output. The arc variables used in Figure 2 are described in Table 1.

The places P_r, P_{g1} and P_{g2} represent the state of the signal controller, and the places P_{in} and P_{out} represent the queues of arriving and departing vehicles. The departing vehicles will become interesting when connecting intersections together; when modelling a single junction, the place P_{out} can be omitted.

Bidirectional arcs (arcs with arrows on both ends) are shortcuts, meaning that there are unidirectional arcs both ways with the same arc expression. This means that the transition will only fire if there is a suitable token in the place, but the firing of the transition will not change the token contents in the place.

The double arcs attached to transitions T_1 and T_2 in Figure 2 are our own notation; actually these are macro transitions which will be explained in Section 2.2. The transition T_1 represents the departure of the first vehicle(s) when the signal turns green, and T_2 takes care of the following vehicles until the signal turns red. This is useful, because [6] shows that the first vehicle passes the junction somewhat faster than the following ones. With more control places and macro transitions, it would be possible to have different passing times for vehicles in different queue positions.

There is a tradeoff between the size and the accuracy of the DSPN model. We wanted to keep the model as simple as possible, and assumed that vehicles in queue positions 2 and greater will pass the intersection using the same amount of time. Also, we assumed that no vehicle can drive through the intersection without slowing down or stopping, in which case it could pass it faster than vehicles starting from the queue. In the model, there are no privileges for public transportation. Finally, there are no pedestrians, bicyclists or trams in our example city.

2.1 Overview of the model

Initially, all queues are empty, so there are no tokens in the places P_{in} and P_{out}. The place P_r contains the token $\langle 0 \rangle$, meaning that the current road is 0 (north–south). When there is a token in P_r, all signals are red. There are two places associated with a green signal. P_{g1} contains a token right after the green signal has been switched but before any vehicle has passed the junction, and P_{g2} will contain the token after the first vehicles have passed the junction until the signal is switched red.[1]

Vehicles are represented by triples of integers $\langle r, d, a \rangle$, where r is the road (north–south, east–west), d is the direction (north (east), south (west)), and a is the action (proceed straight through or turn right, turn left).[2] Each of these fields is coded with the integer numbers 0 and 1.

The arrival of a vehicle is modelled by the transition T_{in}, whose firing causes a token to appear in P_{in}. Then, either T_{gd} or T_{gs} will become enabled and fire, corresponding to switching the signal to green. The transition T_{gs} represents switching the signal green on the same road where it was previously, and T_{gd} represents switching it green on the other road. A token appears in P_{g1} and enables the macro transition T_1, which removes the first token(s) from P_{in} and

[1] The yellow signal is not covered by our model; the firing times of the signal-changing transitions are made so long that they include the yellow phases.

[2] The discrimination of vehicles driving straight through and vehicles turning right will be made with a stochastic choice on the output side of our model.

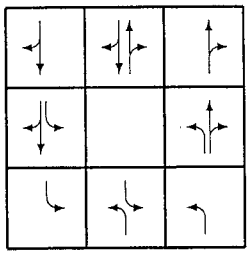


Figure 3: Turn combinations.

moves the control to place P_{g2}. Meanwhile P_{in} may have received more tokens, which will be removed by the macro transition T_2, until T_{max} fires, meaning that the maximum green phase time has elapsed. The green phase will also be terminated if there is a gap long enough in the queue (transition T_{gap}), or after a minimum green time (by transition T_{min}) if there are no vehicles in the queue.[3]

2.2 The macro transitions T_1 and T_2

Our example intersection has two-phase signal control, meaning that there is no separate signalling for left-turners. It is up to the motorists to avoid collisions when taking left turns.[4] There are eight non-interfering combinations of routes through the intersection, shown in Figure 3.

The leftmost column of combinations in Figure 3 can be omitted because of symmetry, which leaves us with five different cases. These cases are handled in our model by the five transitions of the subnet in Figure 4, representing the transitions T_1 and T_2 of Figure 2. Priorities are handled by inhibitor arcs, which prevent the transitions handling left turns from firing while there are tokens representing oncoming traffic in the queue place P_{in}. For clarity, only the places P_{in} and P_{out} of Figure 2 are included.

There is one intermediate place in the subnet, labelled P_{turn}. This place will collect tokens representing vehicles that are either proceeding straight through or turning right. The two immediate transitions leading from this place to the output place P_{out} will choose which route was taken by the vehicle. Again, we kept the model simple and assumed that right-turners need approximately the same time for passing the junction as vehicles passing straight through.[5]

[3] Actually T_{min} can never fire in our model, because the green phase will only be activated when there are vehicles in the queue. But most signalling control systems have the concept of minimum green time.

[4] Four-phase signal control, with separate phases for left-turners, has been modelled in [5].

[5] In [5], the incoming vehicles were assigned their complete route through the intersection

The place P_{turn} can contain four different tokens, numbered from 0 to 3: north (east), south (west), east (south), and west (north). The words in parentheses are the exit directions when a right turn is taken, and the non-parenthesed words are the exits when passing straight through. Likewise, the place P_{out} can hold four kinds of tokens, numbered from 0 to 3: west, east, north, and south. The arc expressions on the right side of the immediate transitions perform the mapping. To keep the formulas short, the ternary ?: operator familiar from the C programming language was used: $a < b\,?\,c : d$ evaluates to c if $a < b$, and to d otherwise.

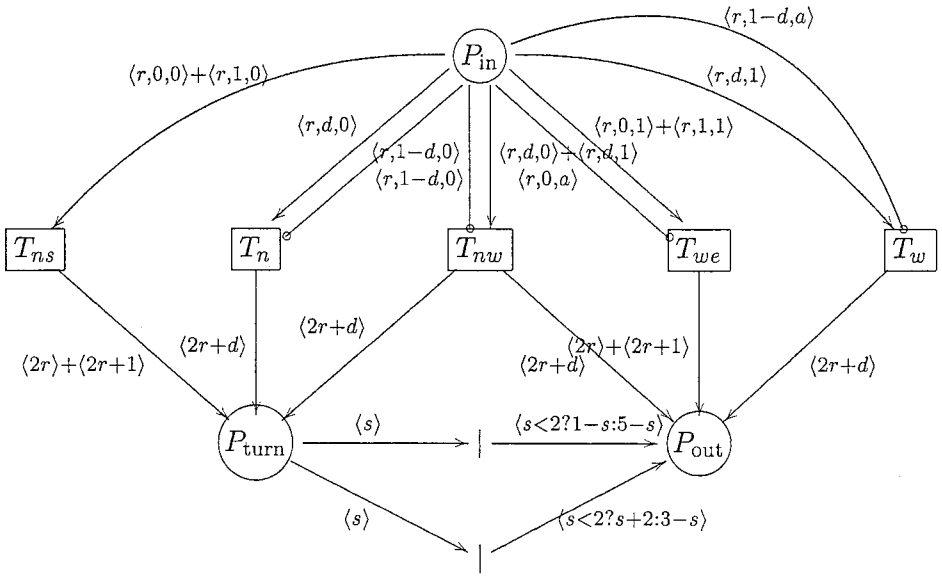


Figure 4: A subnet for the macro transitions T_1 and T_2 in Figure 2.

2.3 Invariants

An invariant is an expression that yields the same value in all markings (distributions of tokens in the model's places) reachable from the initial marking.[1] Our model describing one intersection (Figure 2) has only one invariant, stating that in the three control places of the network, there is one token circulating at all times: $\#P_r + \#P_{g1} + \#P_{g2} = 1$, where e.g. $\#P_r$ denotes the number of tokens in the place P_r. When combining several independently controlled intersections, there will be one such invariant for each intersection.

network at their arrival time. That approach may be more intuitive.

Table 2: Transition parameters of the DSPN model of Figures 2 and 4.

Transition		Time/s
T_{in}	negative exponential	1.5
T_{gd}	deterministic, resampling	5.0
T_{gs}	deterministic, resampling	5.0
T_1	deterministic, enabling memory	1.9
T_2	deterministic, enabling memory	1.9
T_{min}	deterministic, resampling	5.0
T_{gap}	deterministic, resampling	5.0
T_{max}	deterministic, enabling memory	45.0

2.4 Transition parameters

The model presented in Figures 2 and 4 contains three types of transitions: exponential, deterministic and immediate transitions. Incoming traffic is modelled with a timed transition with negatively exponentially distributed firing time. All other timed transitions have a deterministic (constant) firing time. Non-deterministic choices are made by immediate transitions, whose relative firing probabilities are determined by their weights. The parameters of the transitions in Figures 2 and 4 are presented in Table 2.

The transition firing times presented in Table 2 are merely examples, and they need not be equal for all instances of the high-level transitions. For example, the average rate of incoming traffic, in the table 1.5 seconds between vehicles, or 2,400 veh/h, is typically unevenly distributed. With that amount of traffic, if there were as many left-turners as cars passing straight through, the queue of left-turners would grow infinitely long.

Most deterministic transitions obey the resampling policy, meaning that whenever another transition fires, the clocks associated with these transitions will be reset. The T_{max} transition, which takes care of the maximum green phase time, is an exception: it must keep counting until the maximum time is reached. Similarly, the two macro transitions T_1 and T_2 obey the enabling memory policy.[6] All transitions in the model use the single-server policy.

2.5 Connecting intersections

Each four-road intersection can have up to four neighbours. We consider a system consisting of two four-road intersections, presented in Figure 5.

[6]The parameters of these macro transitions are distributed to the parameters of the timed transitions in the subnets of Figure 4.

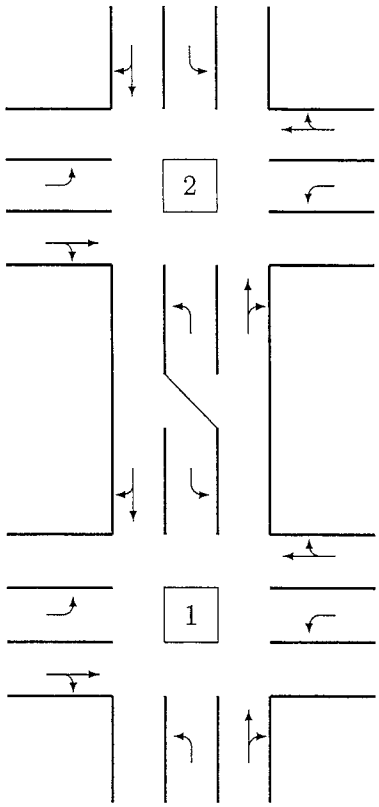


Figure 5: Two interconnected road junctions.

This system can be translated into a DSPN by translating each intersection to a DSPN, and by connecting the output place of one DSPN to the input place of the other DSPN, and vice versa. In this case, we create two DSPNs like those shown in Figure 2.

Transition T_{in} must be broken down to pieces, because the situation is not symmetric any more. Some vehicles arrive from outside the system, but others will come from other intersections' output places. For the simplicity of modelling, the vehicles stochastically choose their route in each intersection. This does not notably change the statistical properties of the model, but makes it homogenous: when intersections are connected, they will not need any information of each other. So, vehicles coming from other junctions must decide whether they take a left turn or proceed straight through or turn right. This is taken care of by the four immediate transitions in Figure 6.

In our example with two intersections, one of the input transitions in Figure 6 for each intersection will be connected to the other intersection's output place. That transition can be either deterministic or exponential, depending on the

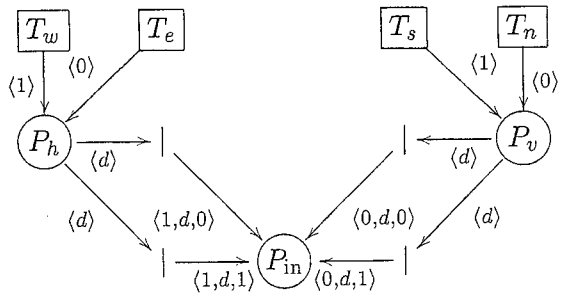


Figure 6: A subnet for the transitions T_{in} of Figure 2.

distance between the intersections, and its firing time must be set to the time it takes to travel from one intersection to the other. The other three transitions will be exponential. The proportion of the weights (in a sense firing probabilities) associated with the immediate transitions in Figure 6 transitions determines the proportion of vehicles turning left and vehicles driving stright ahead or turning right.

The timed transitions with resampling policy should not resample when transitions of subnets describing other intersections fire. If this was not the case, the arrival of a vehicle in any intersection would reset the queue gap counters of all transitions, to name one unfortunate consequence.

The approach presented here makes the intersections independent of each other. Synchronizing the signal control between intersections would make the model somewhat more complicated, or it would require a centralized control network.

3 Simulation

Deterministic and Stochastic Petri Nets where more than one deterministic transition is enabled in a marking cannot be analyzed easily.[3] Until the theory and the software tools have been developed to cope with such DSPNs, simulation is the only way to obtain performance indices from such models.

Two software tools were considered for simulating the system, DSPNexpress and its successor TimeNET (Timed Petri Net Evaluation Tool), developed at Technische Universität Berlin. Both are tools for low-level timed Petri Nets. Had we known about POSES earlier, a commercial high-level Petri Net simulator written by Gesellschaft für Prozeßautomation & Consulting mbH, we could have chosen differently. We chose TimeNET over DSPNexpress, because it is better supported and more widely available. We also experimented with a self-developed DSPN simulator in Matlab.

3.1 Creating the model

Because TimeNET is a low-level Petri Net tool, the high-level model had to be unfolded. We wrote a Perl script that constructed the TimeNET input file, because with its 14 places and 62 transitions the low-level network was too complicated to be managed with TimeNET's graphical editor. The script made it also easier to create models of multiple intersections. The low-level model for two intersections has 52 places and 154 transitions.

3.2 Results

As TimeNET's support for transitions with resampling policy is somewhat limited, the simulation results obtained from it cannot be accurate. In TimeNET, it is impossible to have a transition that is resampled only when certain other transitions fire. As a result, the transition T_{gap} is resampled too frequently, and gaps in the queues are not always detected.

Fortunately TimeNET's lack of support for group weights for immediate transitions in models with continuous time is not a problem in our model, because the input places of immediate transitions are mutually exclusive.

3.2.1 Single junction

The intersection of Figure 1 with the parameters in Tables 2 and 3 was simulated for 33 years of simulated time. Since the model parameters are symmetric, it is no surprise that also the results are. The queue length for left-turners is remarkably high, comparing it to the traffic flow. Unfortunately we were not able to obtain average waiting times from the simulator. Anyway, in real life, one or two vehicles would take a left turn after the light has turned red, and the queue would be shorter.

We ran several simulations on this junction, varying the amount of vehicles driving straight through (or turning right) and vehicles turning left, keeping the parameters symmetric. The result curves in Figure 7 and especially in Figure 8 express the exponential behaviour familiar from [6].

3.2.2 Two junctions

We simulated the two-intersection system of Figure 5, with the parameters listed in Tables 2 and 4, for 4.75 years of simulated time. The performance measures obtained from the simulation are presented in Table 4. The system

Table 3: Queue lengths for a single junction.

Direction	Rate, veh/h	Queue length, veh
north	600	6.75
south	600	6.75
east	600	6.75
west	600	6.75
north, turning left	120	8.34
south, turning left	120	8.34
east, turning left	120	8.34
west, turning left	120	8.34

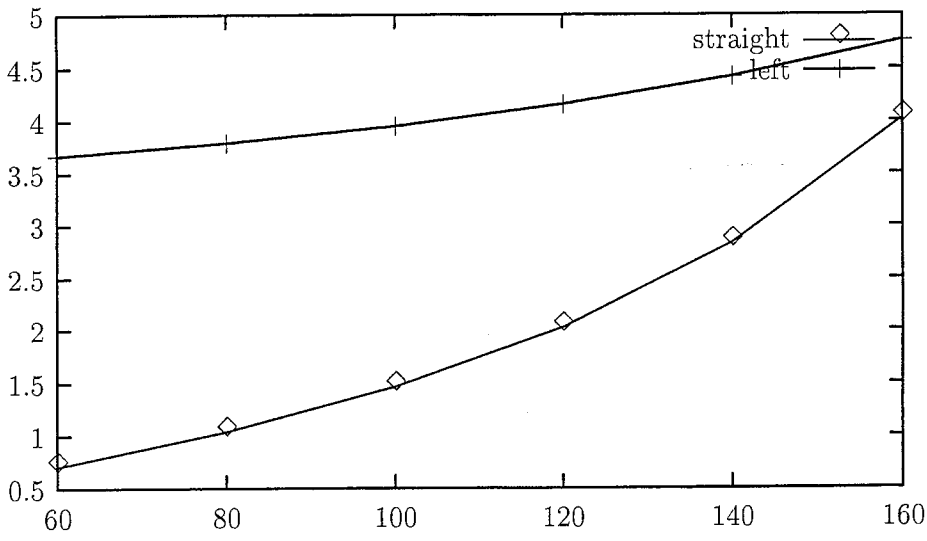


Figure 7: Expected queue lengths of a single intersection with 500 veh/h driving straight through and varying amount of left-turners.

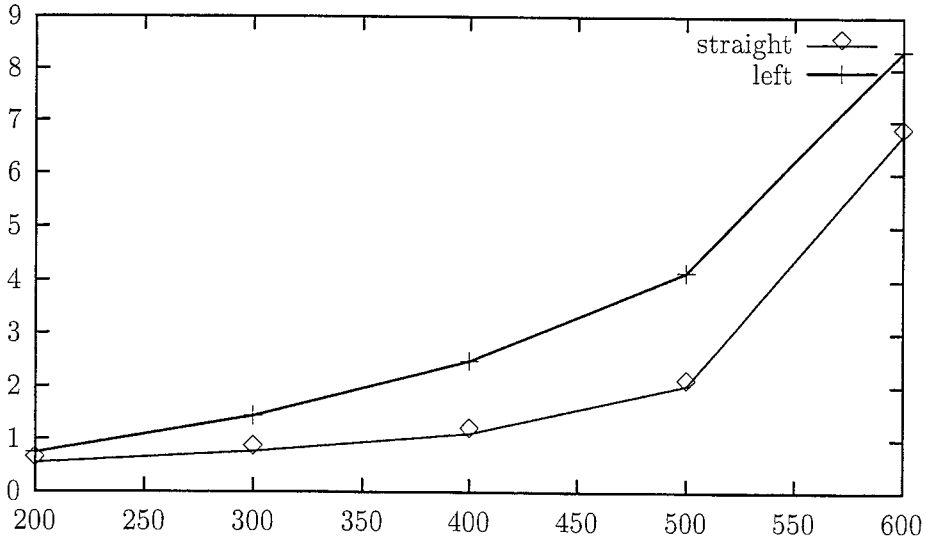


Figure 8: Expected queue lengths of a single intersection with 120 veh/h turning left and varying amount of vehicles driving straight through.

was fairly symmetrical; only the varying percentage of right-turners breaks the symmetry.

4 Discussion

Analyzing the performance of traffic signal control may seem trivial at first, but bringing all the factors of the real-life system to the simulation model is all but trivial. Our model does not include pedestrians or bicyclists.

In our models, we assumed that the vehicles are indistinguishable of each other, and thus did not need to maintain the queue position for individual vehicles. This works well, as long as most vehicles are of the same type. A truck or a bus accelerates slower than a small car and also occupies more place. Our model assumes that all vehicles pass the junctions in equal time, and that the roads have infinite capacity.[7]

The traffic is generated with timed transitions with exponentially distributed firing times. For more accurate results, the rate of incoming vehicles should be varied, to model rush hours and other events.

DSPNs are probably not the most flexible tool for modelling very sophisticated traffic control systems. Special-purpose traffic simulators take care of many

[7]If all cars were almost the same size, the capacity could be easily limited with a complement-place construction on the queue places.

Table 4: Queue lengths for two junctions.

Junction	Direction	veh/h	Queue, veh
1	north, 80% turning right	540	4.57
1	east, 70% turning right	540	4.55
1	west, 80% turning right	540	4.55
1	north, turning left	60	0.82
1	east, turning left	60	0.92
1	west, turning left	60	0.92
2	south, 45% turning right	540	4.61
2	east, 40% turning right	540	4.65
2	west, 30% turning right	540	4.65
2	south, turning left	60	1.06
2	east, turning left	60	0.95
2	west, turning left	60	0.95

things that would be very difficult to model with a DSPN, such as vehicle deceleration and acceleration, driving in a queue or priorities for public transportation. Also, fuzzy logic based traffic controllers are becoming increasingly more common, and they would be quite difficult if not impossible to model with Petri Nets.

A Petri Net based approach has the advantage over simulators [5] that the model is mathematically defined, which makes it possible to verify that the control logic enforces all required mutual exclusions, that capacities are not exceeded and that gridlock will not occur as a result of the logic. Simulators can only find errors; they cannot verify the absence of them.

Acknowledgements

We would like to thank Lic. Tech. Jarkko Niittymäki for introducing modern traffic signal control to us and Dr.-Ing. Armin Zimmermann for answering our questions regarding TimeNET.

This research was partially financed by Academy of Finland (project number 8309).

References

[1] W. Reisig. *Petri Nets – An Introduction.* Springer-Verlag, Berlin 1985.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, Inc., Chichester 1995.

[3] C. Lindemann. *Performance Modelling with Deterministic and Stochastic Petri Nets.* John Wiley & Sons, Inc., Chichester 1998.

[4] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad. *Stochastic well-formed coloured nets for symmetric modelling applications.* IEEE Transactions on Computers, 42(11):1343–1360, Washington 1993.

[5] F. DiCesare, P. T. Kulp, M. Gile, G. List. *The Application of Petri Nets to the Modeling, Analysis and Control of Intelligent Urban Traffic Networks.* In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain, 1994*, pages 2–15. Springer-Verlag, Berlin 1994.

[6] J. Niittymäki. *Isolated Traffic Signals – Vehicle Dynamics and Fuzzy Control.* Licentiate's Thesis, Helsinki University of Technology, Department of Civil and Environmental Engineering, Laboratory of Transportation Engineering, Publication 94, Espoo 1998.

HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Reports of the DIGITAL SYSTEMS LABORATORY

Series **A** : Research Reports ISSN 0783–5396

No. **46** *Tuomas Aura and Pekka Nikander:* Stateless Connections, May 1997; 25 p. ISBN 951–22–3607–9

No. **47** *Patrik Simons:* Towards Constraint Satisfaction Through Logic Programs and the Stable Model Semantics, August 1997; 49 p. ISBN 951–22–3712–1

No. **48** *Tuomas Aura:* On the Structure of Delegation Networks, December 1997; 53 p. ISBN 951–22–3889–6

No. **49** *Tomi Janhunen:* Non-Monotonic Systems: A Framework for Analyzing Semantics and Structural Properties of Non-Monotonic Reasoning, March 1998; 211 p. ISBN 951–22–3974–4

No. **50** *Ilkka Niemelä (Ed.):* Proceedings of the HeCSE Workshop on Emerging Technologies in Distributed Systems, March 1998; 46 p. ISBN 951–22–3996–5

No. **51** *Kimmo Varpaaniemi:* On the Stubborn Set Method in Reduced State Space Generation, May 1998; 105 p. ISBN 951–22–4064–5

No. **52** *Ilkka Niemelä and Torsten Schaub (Eds.):* Proceedings of the Workshop on Computational Aspects of Nonmonotonic Reasoning, May 1998; 87 p. ISBN 951–22–4072–6

No. **53** *Stefan Rönn:* Semantics of Semaphores, August 1998; 10 p. ISBN 951–22–4251–6

Series **B** : Technical Reports ISSN 0783–540X

No. **12** *Kimmo Varpaaniemi:* On Computing Symmetries and Stubborn Sets, April 1994; 16 p. ISBN 951–22–2126–8

No. **13** *Kimmo Varpaaniemi, Jaakko Halme, Kari Hiekkanen, and Tino Pyssysalo:* PROD Reference Manual, August 1995; 56 p. ISBN 951–22–2707–X

No. **14** *Tuomas Aura:* Modelling the Needham-Schröder Authentication Protocol with High Level Petri Nets, September 1995; 32 p. ISBN 951–22–2756–8

No. **15** *Eero Lassila:* ReFlEx—An Experimental Tool for Special-Purpose Processor Code Generation, March 1996; 79 p. ISBN 951–22–2993–5

No. **16** *Markus Malmqvist:* Methodology of Dynamical Analysis of SDL Programs Using Predicate/Transition Nets, April 1997; 70 p. ISBN 951–22–4007–6

No. **17** *Tero Jyrinki:* Dynamical Analysis of SDL Programs with Predicate/Transition Nets, April 1997; 54 p. ISBN 951–22–4008–4

No. **18** *Tommi Syrjänen:* Implementation of Local Grounding for Logic Programs with Stable Model Semantics, October 1998; 40 p. ISBN 951–22–4358–X

No. **19** *Marko Mäkelä, Jani Lahtinen and Leo Ojala:* Performance Analysis of a Traffic Control System Using Stochastic Petri Nets, December 1998; 14 p. ISBN 951–22–4424–1