A STUDY OF CONTROL VIA ON-LINE SIMULATION USING STOCHASTIC PETRI NETS

Matthias Becker
Thomas Bessey
Helena Szczerbicka
Institute of Systems Engineering, University of Hannover
Welfengarten 1, 30167 Hannover, Germany
{xmb,tby,hsz}@sim.uni-hannover.de

KEYWORDS
On-Line Control, Decision Making, On-Line Simulation, Transient Dynamics

ABSTRACT

Complex systems such as flexible manufacturing systems and traffic systems typically evolve with alternating periods of transient and nearly steady-state behavior; such systems often show suboptimal performance. Thus, it is desirable to optimize the system's performance on-line by adjusting the system's parameters properly before a performance drop is to occur. To this end, the system's future evolution is assessed in advance repeatedly by means of on-line simulation. However, there are several problems accompanying this approach, particularly the demand of real-time decisions, that have not been sufficiently solved yet.

Aiming at studying the dynamics of on-line control as well as its impact on the system's operation, we built a stochastic Petri net model that simulates on-line control of a simple open queueing network as it performs by means of on-line simulation. The system under control is easy to study since it has known properties and can be considered as part of a manufacturing system; jobs arriving at the system have to be dispatched to one of two machines, each providing a queue for jobs waiting to be processed. The processing times of the machines are deterministic or stochastic, while the jobs' arrival times are stochastic. With on-line simulation, the system's future performance is assessed by virtually dispatching a new job to either of the machines, based on the system's current state; the results are compared and thus lead to the real decision concerning to what machine the new job should be dispatched in order to minimize the work in progress.

In this work, we compare the quality of on-line control with that of other policies such as random choice and join the shortest queue.

INTRODUCTION

On-Line Simulation

Complex systems such as flexible manufacturing systems and urban traffic systems are usually planned through simulation before their operation actually starts. However, such systems typically evolve with high dynamics, that is, there are alternating periods of transient and nearly steady-state behavior. This is partially due to short-term changes of the requirements of their environment. Additionally, unexpected events such as machine breakdowns or accidents blocking certain routes in the network for some considerable time are also responsible for periods of transient behavior due to congestions. Thus, such systems often show suboptimal performance.

In order to overcome this problem, the idea is to optimize the system's performance on-line by repeatedly adjusting the system's parameters properly. This is referred to as on-line control. In general, on-line control is either reactive or proactive. The former type is characterized by adjusting the system's parameters only after a considerable performance drop is observed.

Proactive on-line control tries to adjust the system's parameters before a performance drop is to occur, in order to avoid this drop. To this end, the system's future evolution is assessed in advance repeatedly. The instants of time at which the assessment of the further evolution and adjustment of the parameters are done are called decision points. The assessment is done as follows: First, the system's current state is copied to several identical system models. For each of these models, certain values of the system's parameters are set, according to some appropriate policies that alternatively could control the system. Once the initialization is done, the models are analyzed in order to assess the future evolution under each policy. As the system under control typically is complex, simulation is the only feasible analysis method. This method is referred to as on-line simulation. With the results, the policy that leads to the optimal future performance of the system under control is chosen

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

to be implemented next, that is, the system is controlled by the chosen policy until the next decision point. This process is referred to as decision making.

There are several problems encountered with this approach, such as setting of the decision points, repeated validation of the system model (the search space for the parameters may vary over time) and proper analysis of the simulation results [2]. As simulation runs consume much time, the number and the length of the simulation runs become crucial. Since the system under control continues to evolve while the next policy is sought by on-line simulation, further problems arise [2]. For a detailed discussion of on-line simulation and associated proactive on-line control, see [4].

For some applications of this approach for traffic systems and (flexible) manufacturing systems, see [7] and [12, 6, 10], respectively. However, these applications are by far not suitable for widely adoption to the real world; they merely employ classical off-line simulation techniques for on-line use. By now, no strict theoretical research has been done.

This Work

This work is intended as being a first step towards a characterization of the dynamics of proactive on-line control as well as of its impact on operation of the system under control. We believe that such characterization is an important and yet open problem that has to be solved in order to be able to assess the impact of applying on-line simulation on the system's performance before actual application. With the results of this and future work, we hope that we will get hints towards theoretical aspects in the field of on-line simulation. For this purpose, we built a stochastic Petri net model that simulates on-line control of a simple open queueing network as it performs by means of on-line simulation. The system under control is easy to study, since it has known properties; it can be considered as part of a manufacturing system. The on-line control and its associated on-line simulation are modeled as a stochastic Petri net as well. Note that both the system under control and its on-line control are integrated into one single stochastic Petri net model.

Stochastic Petri nets are widely accepted and applied as a powerful and concise modeling formalism for modeling of concurrent systems [11, 5], enabling qualitative as well as performance analysis. Depending on the state space of the model, performance analysis may be conducted by means of analytical methods, while simulation of a stochastic Petri net is possible in any case. Since Petri nets have a concise graphical notation, we decided to use stochastic Petri nets for our work; the resulting model is considered to be much clearer than many lines of code in any programming language.

Furthermore, we believe that Petri nets might be a powerful tool for application in the field of on-line simulation [1]; however, to our best knowledge, no such application has been reported in the literature yet.

The remainder of this paper is organized as follows: In the next section, the stochastic Petri net model is presented in detail. Following this, we discuss the results of experiments conducted with the model. Finally, we conclude with a review of this work.

THE PETRI NET MODEL

The stochastic Petri net employed in this work is modeled and simulated using the tool TimeNET [8]. However, the nets depicted in this paper are redrawn using another tool for sake of readability, since TimeNET does not show marking dependent arc weights. For the use of marking dependent arc weights, see the subsection on on-line control.

The System under Control

The system under control consists of two queueing systems, that is, two queues Q1 and Q2 with a single server each. The two servers denote machines M1 and M2 with service rates μ_1 and μ_2; the firing times of the associated transitions may be deterministic or exponentially distributed. As soon as a token arrives to the system, as modeled by the transition arrival with exponentially distributed firing times and arrival rate λ, it will be dispatched to one of the two queueing systems, where it will be processed. The dispatching is accomplished by a policy such as random choice or JSQ (Join the Shortest Queue). Once the token is dispatched to one queueing system, no further jockeying is allowed.

Random Choice

A stochastic Petri net which models the system under control as it implements random choice is shown in figure 1.

The arriving tokens are dispatched to either server with equal probabilities due to equal weights associated with the transitions choice1 and choice2.

JSQ

A stochastic Petri net which models the system under control as it implements JSQ is shown in figure 2.

The places count1 and count2 denote the difference of the queue lengths regarding Q1 and Q2; they influence the transitions choice1 and choice2 by inhibitor

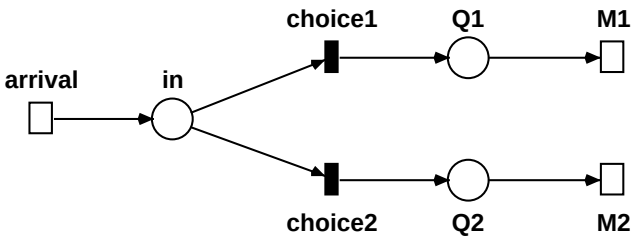

Figure 1: The System under Control Implementing Random Choice

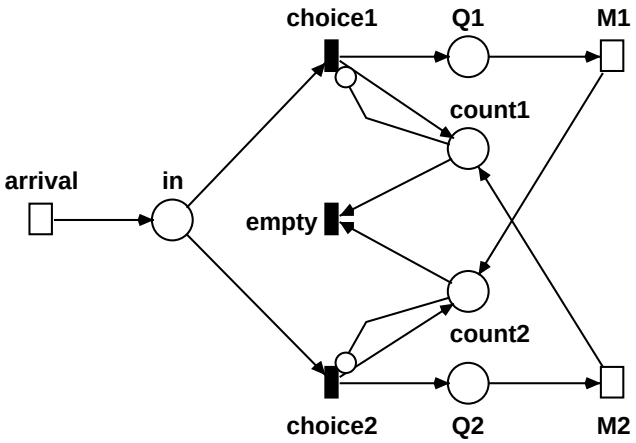

Figure 2: The System under Control Implementing JSQ

arcs according to JSQ. Since we are not interested in the queue lengths but only in their difference, we consider a job leaving server M1 by adding one token to place count2 instead of removing one token from place count1; note that the latter approach fails in the presence of the transition empty, while the former approach leads to a concise model of JSQ.

On-Line Control

A stochastic Petri net which models the system under control as it is controlled by means of on-line simulation is shown in figure 3.

The stochastic Petri net works as follows:

Each newly arriving token triggers the on-line control; i.e., the token is not dispatched before the appropriate choice has been made by on-line control. To this end, the current system state as denoted by the queue lengths (#Q1,#Q2) is copied to the two alternative systems as initial states (#S1Q1,#S1Q2) and (#S2Q1,#S2Q2) for the on-line simulation, respectively; the copying is accomplished using marking dependent arc weights according to the transitions copyQ1 and copyQ2. Note that the alternative systems are reset to empty queues before copying, since former on-line simulation may have left the systems dirty.

Following this, the on-line simulation is started by firing of transition start, which adds a new token to the places S1Q1 and S2Q2, respectively, according to the two alternatives of dispatching to be evaluated. In terms of the introduction, the on-line simulation evaluates two policies, where the first one always chooses the machine M1 for dispatching arriving tokens, while the second one always chooses the machine M2. Note that further arrivals to the on-line simulated systems do not occur. The alternative of which simulation has completed first (in that the respective queues are empty) is considered to be the best choice. This consideration is due to the limited tractability of the stochastic Petri net formalism with respect to performance evaluation as it is integrated as a stochastic Petri net itself. Regarding the transitions choice1 and choice2, the stop condition of the on-line simulation is modeled with inhibitor arcs, while the concurrency of both transitions is considered by the place P4. The transitions transfer the token of which arrival triggered the on-line simulation to either queue Q1 or queue Q2.

The service rates of the transitions SxMx are greater

than that of the transitions M1 and M2 by a factor of $s > 1$, which is the speedup of the on-line simulation. Actually, instead of simulating on-line simulation (it would be a difficult task to do so since we would have to specify for each control decision its required "thinking time" in advance), we simply operate the system under control at a higher speed. Hence, the definition of speedup that we use in this work is quite different from the usual one, which states that speedup is the factor by which the on-line simulation is faster than real-time. But since both definitions are related by some kind of mapping, we expect our approach to be valid.

While the on-line simulation is being performed, arriving tokens according to the system under control have to wait in the place in to proceed until the choice for dispatching is made for the pending token that triggered the on-line control (as controlled by the place ready). Thus, the mean number of tokens in the place in can be used as a measure for the delay of the system's operation due to the execution time of the on-line simulation needed for each token.

Note that the stochastic Petri net model presented here is simplified for sake of comprehension in that just one single simulation run is performed with respect to the on-line simulation. In fact, for the experiments we conducted, we used an extended stochastic Petri net model that employs N simulation runs in order to make the choice for dispatching according to one token. For this purpose, the extended model comprises two additional places (one for each possible choice) that count the results of each simulation run concerning what choice is considered to be the best (just as described above); the place that contains $\frac{N}{2} + 1$ tokens first causes the associated choice to actually be made.

EXPERIMENTAL RESULTS

We use the mean work in progress (WIP) as the performance measure regarding the system under control; in all experiments, the WIP is computed by performing steady-state simulation with a confidence level of 95 percent. The simple system presented here reaches steady-state even under on-line control, so our experimental environment is justified. Since the mean WIP observed in the queues Q1 and Q2 is directly related to the mean waiting time through Little's Law [3], smaller values of the WIP imply better performance of the control policy.

In the following, we consider four cases regarding the processing time distributions of the two machines M1 and M2; these cases differ in the variance (i.e., deterministic vs. exponential processing times) and in the ratio of the mean processing times μ_1^{-1} and μ_2^{-1} (equality vs. inequality). Note that in case of deterministic processing times, only one simulation run is needed in order to perform the on-line simulation, thus $N = 1$ in these cases.

Balanced Deterministic Case

The results are shown in table 1, where $\lambda = 1.75$ and $\mu_1 = \mu_2 = 1.0$ (with the machines having constant processing times).

Note that in the case of balanced deterministic processing times, JSQ is the optimal policy with respect to the mean waiting time of the system under control, since for each machine, the waiting time can be computed by multiplying the number of waiting jobs in the queue with the constant processing time for each job. However, in the experiment, the WIP resulting from applying on-line control is somewhat smaller than that resulting from JSQ; this is due to the simulation error. Basically, the on-line control leads to optimal performance in this case, just as JSQ does. In fact, studying the simulation traces of the experiment, it turns out that the on-line control just behaves like JSQ. Note, however, that while JSQ considers the system's state explicitly in that it compares the two queue lengths, the on-line control just selects one of the two simplest policies possible, which do not consider the system's state explicitly. In the latter case, the information about the system's state is implicitly considered when initializing the on-line simulation to the current state of the system under control.

In addition, the experiment yields WIP(in) = 0.003, given that $s = 100$; thus, in this case, the execution time of the on-line simulation has virtually no impact on the system's operation, compared to the WIP of Q1 and Q2.

Balanced Exponential Case

Here, the machines have exponentially distributed processing times, where, again, $\lambda = 1.75$ and $\mu_1 = \mu_2 = 1.0$. The results are shown in table 2 regarding the random choice and JSQ policies and in table 3 regarding the on-line control under different settings with respect to N and s.

It can be observed that generally, the results obtained by means of on-line control get better, the more simulation runs are performed regarding the on-line simulation. Quite clearly, this is because of the stochastic nature of the processing times associated with the machines. However, with N increasing, the time needed for performing the on-line simulation becomes crucial to the quality of the on-line control. At this point, the speedup of the on-line simulation becomes important, where larger speedup allows for performing more simulation runs according to a fixed

Table 1: Results for the Balanced Deterministic Case

	Random Choice	JSQ	On-Line Control
WIP(Q1)	3.88	2.42	2.35
WIP(Q2)	3.95	2.42	2.35
Σ	7.83	4.84	4.70

Table 2: Results for the Balanced Exponential Case (1)

	Random Choice	JSQ
WIP(Q1)	7.02	3.95
WIP(Q2)	6.99	3.98
WIP(in)	—	—
Σ	14.01	7.93

time period.

The results show that also in this case, on-line control is able to perform as optimal as JSQ does [9], although a considerable number of simulation runs concerning the on-line simulation is needed, requiring an appropriate speedup. However, the WIP according to the place in is not negligible anymore, implying that the on-line simulation has a serious impact on the system's operation in that it causes a considerable number of tokens to wait before processing.

Note that in case of $N = 5$ and $s = 100$, the WIP regarding the place in is about two third of the WIP regarding Q1 and Q2 (with this ratio being considerably greater than in other cases); however, the total WIP is similar to the WIP resulting from applying JSQ. Thus, the on-line control performs quite well despite of its specific dynamics as measured by observing WIP(in). In addition, in case of $N = 15$ and $s = 1000$, the total WIP is similar to the WIP resulting from JSQ, too, while the speedup is considerably larger.

However, large speedup may lead to several problems concerning statistical analysis of the simulation results [1]: While the execution time needed for on-line simulation is reduced as the speedup increases, concurrent validation of the employed simulation models may still be necessary; however, certain events that will possibly occur while the system under control continues to operate, causing the real-time data to be updated, may become rare with respect to the on-line simulation, while the data updates have to be considered by the validation process.

Thus, small speedup leads to the problem of incorrect decision making due to the system's evolution while the on-line simulation is being performed, while large speedup may lead to incorrect decision making due to incorrect simulation results; quite clearly, this stresses the need for research on a trade-off regarding the speedup and the accuracy of on-line simulation.

Unbalanced Deterministic Case

The results are shown in table 4, where $\lambda = 1.75$ and $\mu_1^{-1} = 1.5$, $\mu_2^{-1} = 0.5$. In this case of unequal mean processing times, we additionally give results for the throughputs τ of the machines M1 and M2, since they imply the ratio of the jobs' splitting between the machines.

Note that in the case of random choice, we adjusted the weights associated with the transitions choice1 and choice2 (see figure 1) in that arriving jobs are now dispatched to the slower machine M1 with a smaller probability and vice versa. With this experiment's settings, the probability of choosing the first machine is 0.25, whereas that of choosing the second one is 0.75, corresponding to the reciprocal of the ratio of the mean processing times. (Note that while this is an obvious adjustment, it is not optimal with respect to the sum of the mean waiting times.) As result, the throughputs' ratio is that of the dispatching probabilities, while the WIP of Q1 and Q2 is evenly split.

When comparing the results regarding JSQ to those regarding the on-line control, the following observations can be made: First, the ratio of the throughputs in case of on-line control is greater than that in case of JSQ by a factor of four; second, the main portion of the WIP in case of JSQ is related to Q1, while it is to Q2 in case of on-line control. This is due to the fact that JSQ only relies on the queue lengths; however, longer queues in front of faster servers may be processed in less time than shorter queues in front of slower servers, depending on the ratio of the mean serving times as compared to the ratio of the queue

Table 3: Results for the Balanced Exponential Case (2)

On-Line Control	$N = 1$ $s = 100$	$N = 5$ $s = 100$	$N = 15$ $s = 100$	$N = 15$ $s = 1000$
WIP(Q1)	4.99	3.20	1.90	3.96
WIP(Q2)	4.98	3.00	1.90	3.73
WIP(in)	0.12	2.15	18.00	0.26
Σ	10.09	8.35	21.80	7.95

Table 4: Results for the Unbalanced Deterministic Case

	Random Choice	JSQ	On-Line Control
WIP(Q1)	1.25	1.31	0.29
WIP(Q2)	1.26	0.84	1.62
Σ	2.51	2.15	1.91
τ(M1)	0.43	0.54	0.18
τ(M2)	1.30	1.18	1.56

lengths. While JSQ cannot consider this fact, on-line control can thanks to the on-line simulation. This leads to the on-line control performing better than JSQ, thus being the optimal policy in this case.

Unbalanced Exponential Case

Here, the machines have exponentially distributed processing times, where, again, $\lambda = 1.75$ and $\mu_1^{-1} = 1.5$, $\mu_2^{-1} = 0.5$. The results are shown in table 5 regarding the random choice and JSQ policies and in table 6 regarding the on-line control under different settings with respect to N and s. (Note, however, that these settings are slightly different from that of the balanced case.) Again, the dispatching probability regarding the machine M1 is three times smaller than that regarding the machine M2 in case of random choice.

The results show that in case of stochastic processing times, performing one single simulation run with respect to on-line simulation is senseless; in this experiment, the on-line control even causes the system under control to become unstable, which is due to the fact that the on-line control randomly overloads one of the two machines because of insufficient future projection accuracy. (In this case, the steady-state simulation that we performed is senseless.) With several simulation runs, however, the on-line control performs nearly as good as JSQ. Note that also in this case, the WIP of the place in becomes significant as the number of simulation runs increases, given a fixed speedup; in fact, the WIP regarding in is about one fifth of the total WIP, although the speedup is $s = 1000$.

CONCLUSION

In this work, we built a stochastic Petri net that models on-line control as it performs by means of on-line simulation; for sake of conciseness, both the system under control and its on-line control were integrated into one single stochastic Petri net model. As the system under control, we employed a simple open queueing network consisting of one arrival process and two service processes, where the dispatching of arriving jobs to one of the two servers is subject to one of the policies random choice, JSQ and on-line control. Since our aim was to study basic aspects of on-line control, we confined ourselves to a simple system under control with known properties. We conducted several experiments with this model, varying the processing time distributions associated with the two machines. With these experiments, we studied the impact of the speedup and the number of simulation runs employed in the on-line simulation on the performance of the on-line control.

It turned out that JSQ always outperforms random choice (as expected), while the on-line control performs quite as good as JSQ. In case of unbalanced deterministic processing times, the on-line control even turned out to be optimal, despite of the simple evaluation method of emptying the queues. This is considered as a strong hint towards the profitable applicability of on-line simulation to manufacturing systems, although the system under control that we used here is indeed very simple.

Despite of this simplicity, our experiments lead to an important observation that may turn out to reveal a major issue in the application of on-line simulation:

Table 5: Results for the Unbalanced Exponential Case (1)

	Random Choice	JSQ
WIP(Q1)	1.91	1.79
WIP(Q2)	1.92	1.16
WIP(in)	—	—
Σ	3.83	2.95

Table 6: Results for the Unbalanced Exponential Case (2)

On-Line Control	$N = 1$ $s = 100$	$N = 5$ $s = 100$	$N = 5$ $s = 1000$	$N = 15$ $s = 1000$
WIP(Q1)	(unstable)	1.01	1.19	0.56
WIP(Q2)	(unstable)	1.65	1.79	2.04
WIP(in)	(unstable)	0.32	0.03	0.64
Σ	—	2.98	3.01	3.24

While on-line simulation generally leads to higher performance of the on-line control as it is performed with larger speedup (since the system's ongoing evolution becomes negligible), it may still perform quite well with smaller speedup (or similarily, with more simulation runs). Facing the risk of statistical instability in case of large speedup due to rare events of the system under control, one may benefit from small speedup compared to fast on-line simulation. However, small speedup (or similarily, a large number of simulation runs) may affect the system's operation; in our experiments, the WIP related to the incoming place considerably increased. Thus, the experimental results give hints towards the need for a trade-off for on-line simulation regarding its execution time and its accuracy, where great care has to be taken in case of large speedup with respect to statistical stability.

Furthermore, the results give hints that it is not suitable to trigger on-line control for every single job, since the impact on the system's operation may become undesirably great; instead, the on-line simulation should cover the short-term future of the system's evolution in that it includes arrival processes as well. In this approach, the policies subject to decision making would be assessed regarding their impact on the system's performance every time a certain decision point is reached, as discussed in the introduction.

REFERENCES

[1] T. Bessey. Needs and proposals for theoretical research on on-line simulation. In *Proc. Summer Computer Simulation Conference (SCSC)*, 2003.

[2] T. Bessey. On-line simulation: Towards new statistical approaches. In *Proc. Summer Computer Simulation Conference (SCSC)*, 2003.

[3] G. Bolch, S. Greiner, H. d. Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. Wiley, 1998.

[4] W. J. Davis. On-line simulation: Need and evolving research requirements. In J. Banks, editor, *Handbook of simulation*, chapter 13. Wiley, New York, 1998.

[5] A. A. Desrochers and R. Y. Al-Jaar. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis*. IEEE Press, 1995.

[6] G. R. Drake and J. S. Smith. Simulation system for real-time planning, scheduling, and control. In *Proc. Winter Simulation Conference (WSC)*, 1996.

[7] J. Esser, L. Neubert, J. Wahle, and M. Schreckenberg. Microscopic online simulation of urban traffic. In *Proc. 14th International Symposium on Transportation and Traffic Theory*, 1999.

[8] R. German, C. Kelling, A. Zimmermann, and G. Hommel. TimeNET — A toolkit for evaluating non-Markovian stochastic Petri nets. *Performance Evaluation*, 24:69–87, 1995.

[9] H.-C. Lin and C. S. Raghavendra. An approximate analysis of the Join the Shortest Queue (JSQ) policy. *IEEE Transactions on Parallel and Distributed Systems*, 7(3), 1996.

[10] S. Manivannan and J. Banks. Real-time control of a manufacturing cell using knowledge-based simulation. In *Proc. Winter Simulation Conference (WSC)*, 1991.

[11] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, 1981.

[12] A. I. Sivakumar. Optimization of cycle time & utilization in semiconductor test manufacturing using simulation based, on-line near-real-time scheduling system. In *Proc. Winter Simulation Conference (WSC)*, 1999.

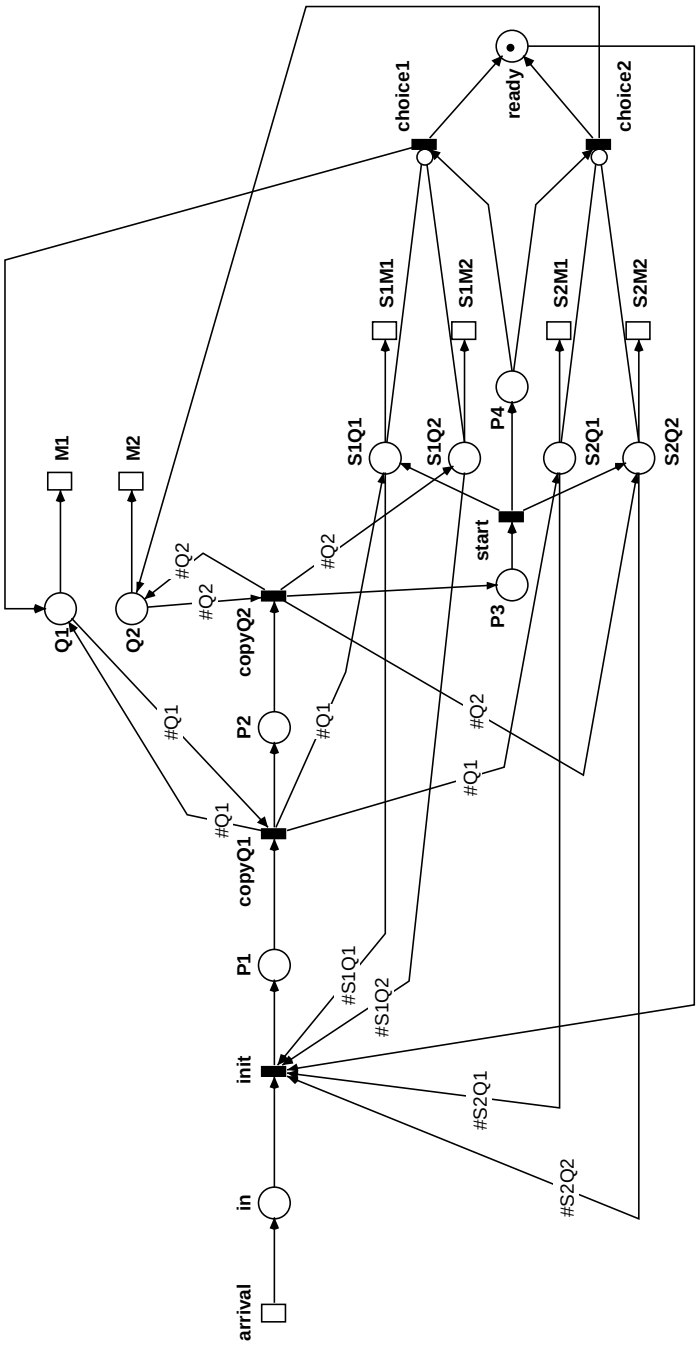

Figure 3: Control via On-Line Simulation