Department of Computing Science,
University of Newcastle upon Tyne

Relative data freshness of asynchronous communication mechanisms

A. Madalinski, F. Xia, A. Yakovlev

Technical Report Series

No. CS-TR-709
August, 2000

Contact:

A.A.Madalinski@ncl.ac.uk
Fei.Xia@ncl.ac.uk
Alex.Yakovlev@ncl.ac.uk

This work was supported by EPSRC GR/L93775, GR/M94366

Published by the University of Newcastle upon Tyne,
Department of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK

Abstract

Data freshness is an important property of asynchronous communication mechanisms (ACMs). It defines how up to date any item of data that the reader obtains from the ACM is. The concept of relative data freshness is introduced, which explores data freshness quantitatively. A method of studying relative data freshness for multi-slot ACMs using stochastic Petri nets (SPNs) techniques is presented. The relative data freshness properties of a three-slot and a four-slot ACM are investigated.

Keywords: asynchronous communication mechanism, stochastic Petri nets, data freshness.

1 Introduction

An asynchronous communication mechanism (ACM) is a scheme which manages the transfer of data between two processes. The data being passed consists of a stream of individual items of a given type. The processes are single thread cycles, one providing an item of data during each cycle, the other making use of an item of data during each cycle. The provider of data is known as the *writer* of the ACM and the user is known as the *reader* of the ACM.

Such techniques as the multiple *slot* mechanisms were proposed in [1] and are meant to communicate without mutual timing constraints. The data is passed between the writer and reader processes, through shared memory locations (slots) where the access to these slots is coordinated by small shared control variables. Schematically, data communication between processes through an ACM is shown in Figure 1.

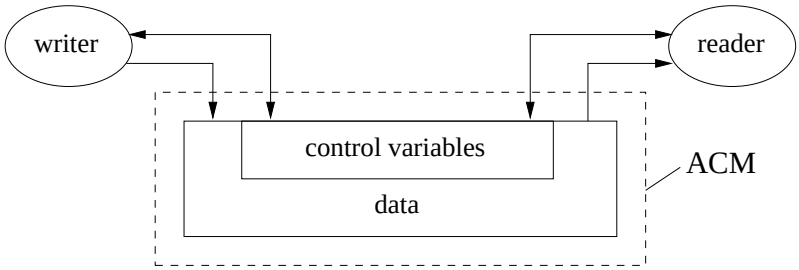


Figure 1: Asynchronous data communication mechanism

The number of data slots is significant to the most important ACM properties, viz. asynchrony, data coherence and data freshness. In order to provide full asynchronism, more than two data slots are needed so that the reader and writer do not access the same slot at the same time. When simultaneous reading and writing occurs on the same slot, it is likely that corrupt data will be passed on. This is known as the data coherence problem.

The other important property of such ACMs is data freshness, which describes how up to date any item of data obtained by the reader is, relative to the latest item of data provided by the writer. Data freshness is defined qualitatively in [2]. In this paper data freshness is studied quantitatively. A concept of relative data freshness is introduced, which classifies freshness grades from the most fresh data to the least fresh data within an ACM.

Using SPN techniques the relative data freshness of a three-slot and a four-slot ACM is studied here. This study is a continuation of previous studies of ACMs. In [3] other properties of an ACM have been studied using SPN techniques.

SPN techniques offer a well-developed approach for the performance analysis of discrete event systems. This approach also seems most appropriate because the principal formal techniques used in former studies [4] for modelling such asynchronous systems are those based on Petri nets.

It has been shown in [1] that a four-slot ACM is capable of maintaining data coherence and data freshness in fully asynchronous operation. The three-slot design, however, fails data coherence requirements in a subtle way. Specifically, if certain control variable statements are not regarded as atomic, simultaneous reading and writing of the same slot may occur. A hardware ACM design using self-timed circuits has been published [5] where mutex elements are used to deal with the issue of metastable control variables. A side effect of this use of mutex elements is that the relevant control variable statements are protected, and may now be regarded as atomic.

2 Background

In this section the four-slot and the three-slot ACMs and their PN models are described. Furthermore, the stochastic Petri nets (SPNs) and the GreatSPN tool are briefly summarized.

2.1 The four-slot mechanism and its PN model

Simpson's four-slot mechanism [1] is asynchronous in that the operation of one process has no timing effects on the operation of the other. The algorithm is shown in Table 1. The writer and reader processes are single loops with three statements each. A four

writer	reader
$wr : d[n, \overline{s[n]}] := input$	$r0 : r := l$
$w0 : s[n] := \overline{s[n]}$	$r1 : v := s$
$w1 : l := n \Vert n = \overline{r}$	$rd : output := d[r, v[r]]$

Table 1: Algorithm of the four-slot mechanism

element array $d[0,0]$ to $d[1,1]$ represents the four data storage slots. The control variables n, l, r, $s[0..1]$, $v[0..1]$, which are either single bits or vectors of two single bits, steer the access to the slots. The variables n, l and r, respectively, indicate which slot is being used to assemble new data, which slot holds the latest data and which has been selected for reading. The control variables s and v indicate, respectively, the slot which holds the latest data and which slot has been selected for reading. The four-slot ACM is shown schematically in Figure 2.

PN model of the four-slot ACM

The Petri net model of the four-slot ACM as presented in [4] is described here. According to the algorithm in Table 1 the ACM is represented as a writer and reader model with three sub-models each.

Writer model The writer process is divided into three statements wr, $w0$ and $w1$. The first writer statement wr is the writer slot access and is depicted in Figure 3. In this

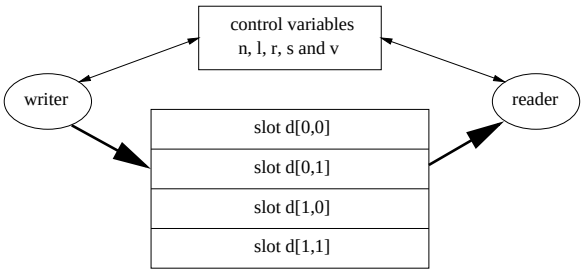


Figure 2: Scheme of the four-slot mechanism

sub-model each slot has its own start writing, writing, end writing and not writing cycle. The value of the control variable $s[0..1]$ determines which slot the writer would go to.

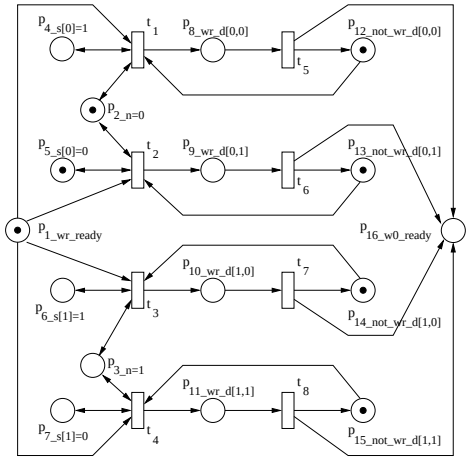


Figure 3: PN of the writer statement wr of the four-slot mechanism

The second writer statement $w0$ is the first part of the write post sequence. In this sequence the control variables are updated by the writer. The value of the variable s changes to its complement. The PN model of this statement is shown in Figure 4(a).

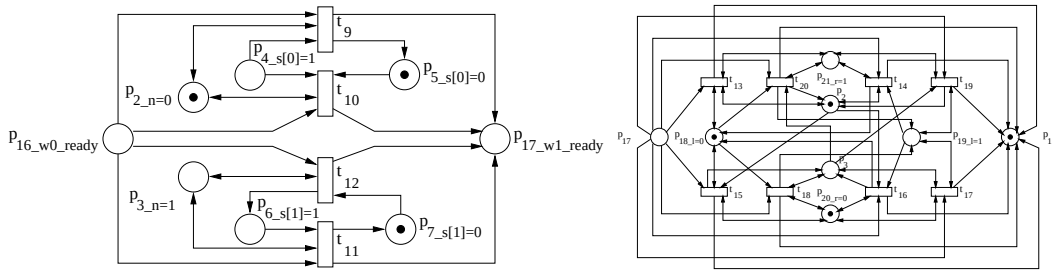
Figure 4(b) shows the third writer sub-model for the $w1$ statement, which is the second part of the post sequence. This action is a parallel assignment of the control variables n and $\overline{r}$. The assignment is performed by firing one transition, depending on the value of the control variables.

Reader model The reader model is established in a similar way to the writer model. The first and second statements $r0$ and $r1$, respectively, denote the read pre-sequence, where the control variables are updated. These sub-models are depicted in Figure 5.

The third reader statement rd, shown in Figure 6, represents the read slot access, where each slot has its own start reading, reading, end reading and not reading cycle.

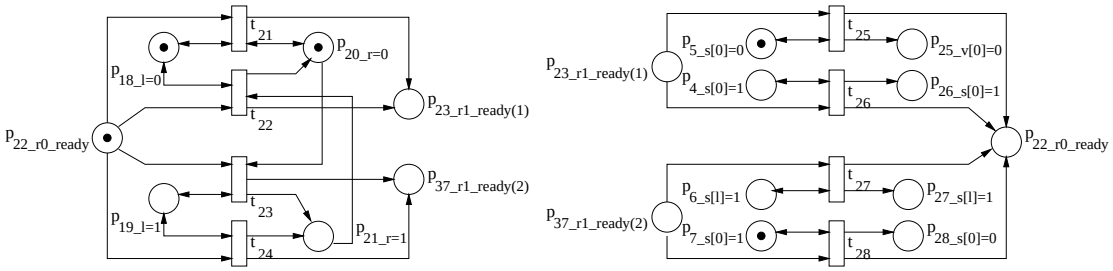
2.2 The three-slot mechanism and its PN model

The algorithm of the three-slot mechanism, as defined in [2], is shown in Table 2. Here the writer and reader processes are single thread loops with three and two statements each, respectively. The data is passed through a three slot array $d[n]$. The control variables l, r



(a) w0 statement (b) w1 statement

Figure 4: PN of the writer post sequence of the four-slot mechanism



(a) r0 statement (b) r1 statement

Figure 5: PN of the read pre-sequence of the four-slot mechanism

and n indicate, respectively, which of the three slots holds the latest data, which has been selected for reading and where new data is to be assembled.

writer	reader
$wr : d[n] := input$	$r0 : r := l$
$w0 : l := n$	$rd : output := d[r]$
$w1 : n := \neg(l \vee r)$	

Table 2: Algorithm of the three-slot mechanism

The PN model represented here is slightly modified from the original one in [4] because a simplified algorithm was used.

Writer model The writer process describes three statements wr, $w0$ and $w1$. The first writer statement wr is the writer slot access and is depicted in Figure 7. In this sub-model each slot has its own start writing, writing, end writing and not writing cycle.

The second and third writer statements are the write post sequence. In this sequence the control variables are updated by the writer, setting the variables l and n, respectively. The models are shown in Figure 8.

Reader model The reader process is divided into two statements $r0$ and rd. The statements $r0$ and rd are modelled in a way shown in Figure 9, according to the algorithm.

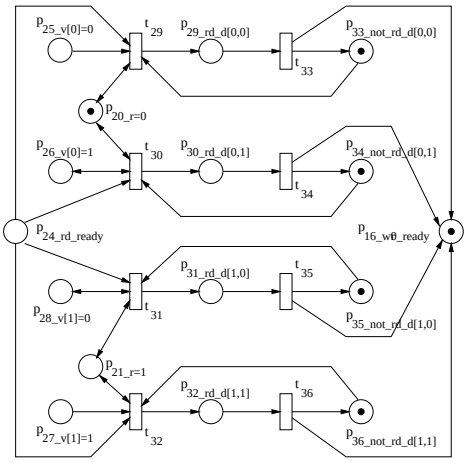


Figure 6: PN of the reader statement $r0$ of the four-slot mechanism

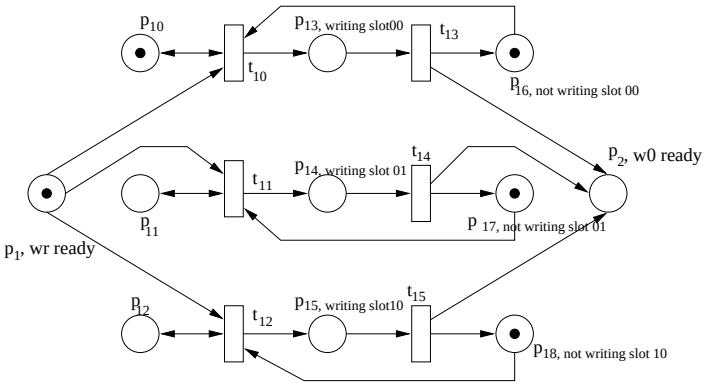


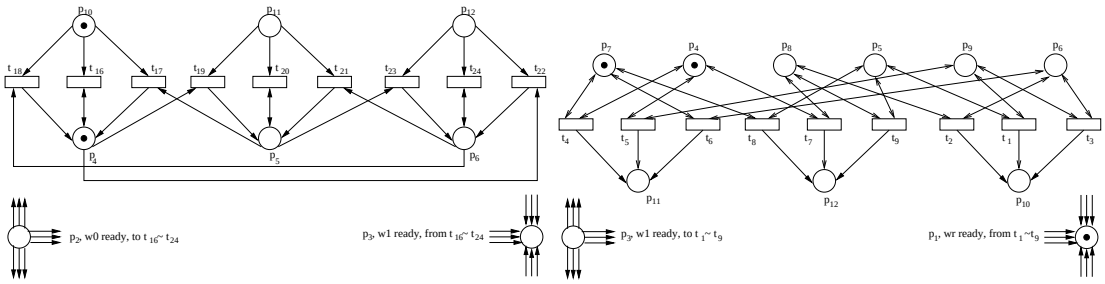
Figure 7: PN of the writer statement *wr* of the three-slot mechanism

2.3 Stochastic Petri Nets and the GreatSPN tool

A Stochastic Petri Net (SPN) [6] is a Petri Net where each transition is associated with a random variable (exponentially distributed) that expresses the delay from the enabling to the firing of the transition. In the case where several transitions are simultaneously enabled, the transition that has the shortest delay will fire first. The reachability graph of an SPN is identical to the one of the underlying PN. Due to the *memoryless* property of the exponential distribution of firing delays, the marking process generated by an SPN is a continuous-time Markov chain with a state space which is isomorphic to the reachability set [7].

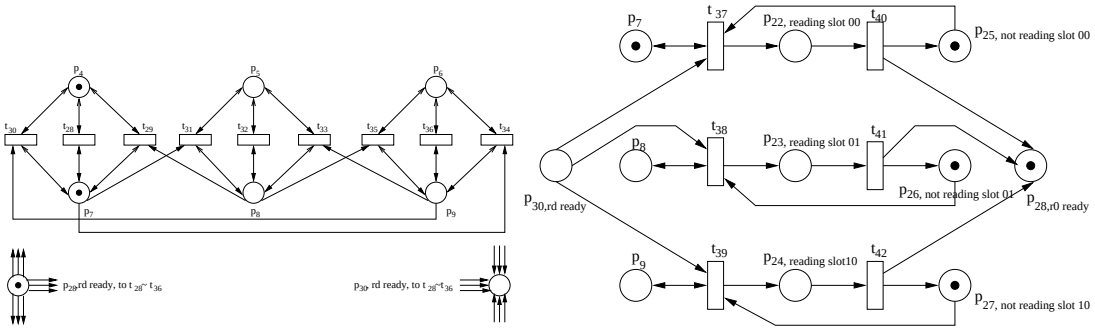
A generalised stochastic Petri net (GSPN) [9][8] is an extension of SPN that copes with the state space explosion problem. A GSPN has two types of transitions, timed and immediate. A timed transition has an exponentially distributed firing delay, as in SPN, and an immediate transition has no firing delay.

The stochastic interpretation of a GSPN model differs from that of an SPN model in taking into account immediate transitions. When a marking is entered, it is first necessary to ascertain whether it enables timed transitions only, or at least one immediate transition.



(a) w0 statement (b) w1 statement

Figure 8: PN of the writer post sequence of the three-slot mechanism



(a) r0 statement (b) rd statement

Figure 9: PN of the reader process of the three-slot mechanism

Markings of the former type are called tangible, whereas markings of the latter type are called vanishing. In the case of tangible marking the transition rule corresponds to the one of SPN. In the case of vanishing markings, the selection of which transition to fire cannot be based on the temporal description (since all immediate transitions fire exactly in zero time). Thus, the choice is based on priorities and weights.

The GreatSPN2.0.2 tool [10][11] is a software package for modeling, validation, and performance evaluation of distributed systems using Generalized Stochastic Petri Nets. The main features which the tool provides are a graphical interface, structural properties analysis and Markovian solvers for steady state.

The GSPN model is created using the graphical interface and validated using structural analysis methods. The performance-related parameter such the rates or weights associated with transitions are set to default values and should be defined to specify the behaviour of the model. The Markovian performance analysis computes the steady state probability which is defined as the percentage of time the system spends in the state i being in equilibrium condition. The performance parameters, including the throughput, the frequency of firing of a transition and the distribution of tokens into places (the probability of an event defined through place marking), are obtained from the steady state distribution. This tool was successfully employed in [3] where its findings were checked thoroughly with the help of simulations.

3 Modelling and analysis of relative data freshness

A monitoring environment must be introduced in order to study relative data freshness of ACMs using SPN techniques.

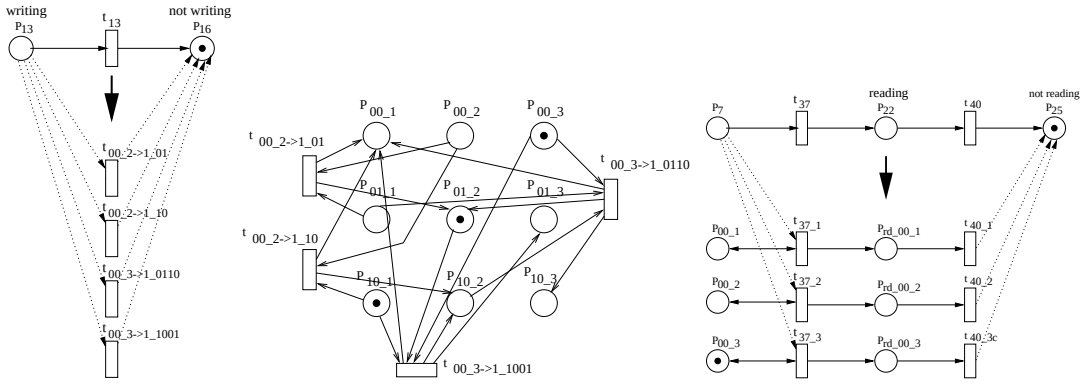
Definition: *Relative data Freshness*

> At any moment in steady state, the slots of the ACM contain data items which were written at different times. At the beginning of an *rd* statement, the n slots of the ACM are labelled 1 to n according to when their contents were written, with the slot containing the most recently written item labelled 1, the slot containing the next most recently written item labelled 2, etc. The reader is said to obtain an item of relative data freshness i during a cycle when the slot accessed by the *rd* statement of the cycle is so labelled i.

This definition is fundamentally different from Simpson's original qualitative definition of data freshness [2], which was concerned with data items being admissible new. Informally, the oldest item of data that satisfies that definition is the second newest in the ACM at the beginning of *r0*, if *r0* and the sequence of *w0* and *w1* overlap in time. This means that in some cases, data items whose relative freshness is greater than 1 may be regarded as fresh in the sense of Simpson.

According to the definition of relative data freshness above, monitoring networks for the three-slot and four-slot ACM were constructed. First the monitoring network for the three-slot ACM is described in detail followed by the one for the four-slot ACM.

The monitoring environment of the three-slot ACM containing, amongst other things, 3 relative freshness labels for each slot. These relative freshness labels are updated after a writer access (wr) and checked before a reader access (rd). Depending which relative freshness label was set at the beginning of the reader access the particular relative data freshness grade is set. In Figure 10 a part of the monitoring network of the three-slot ACM is shown.



(a) extention of wr statement (b) setting relative freshness labels (c) extention of rd statement

Figure 10: Part of the monitoring network of the three-slot ACM

There are four possibilities of changing the relative freshness labels for each slot, depending which slot is accessed by the writer and which relative freshness label is set. An

additional sub-model was introduced for each slot to update the relative freshness labels after a writer access. Figure 10(b) shows this model for the first slot. The places are arranged in a 3 by 3 array. The places in the first row (p_{00_i}, $i = 1, 2, 3$) define the relative freshness labels of the first slot, in the second row (p_{01_i}) of the second slot and so on. The transitions in this sub-model denote the finished writing transitions of the first slot. In the original PN model this action was represented by the transition t_{13} in the *wr* sub-model.

This sub-model was modified by dividing the transition t_{13} (finished writing slot 00) into four transitions. This is shown in Figure 10(a). The method of subscriptions for these transitions is denoted as follows. The first two binary digits define the slot after writer access. The next part denotes the setting of the freshness label of the accessed slot from 2 or 3 to 1. The last part of the index describes the slot(s) involved in updating its (their) freshness label(s). If there are more than two digits the slot with the lowest label value is placed first followed by the one with a higher value.

If one of the transitions $t_{00\_2\rightarrow1\_01}$ or $t_{00\_2\rightarrow1\_10}$ in Figure 10(b) fires, the relative freshness label of the first slot is set from 2 to 1 and the relative freshness labels of the second and third slots, respectively, from 1 to 2 depending on the state of these slots. The transition $t_{00\_3\rightarrow1\_0110}$ or $t_{00\_3\rightarrow1\_1001}$ is enabled if the label of the first slot is set to 3. One of these transitions fires, setting the label of the first slot to 1 and adding 1 to the values of the label of the other slots (setting either slot 01 to 2 and slot 10 to 3 or vice versa). According to the initial markings (p_{00_3}, p_{01_2} and p_{10_1}) the transition $t_{00\_3\rightarrow1\_1001}$ fires if the writer accesses the first slot setting labels denoted by places p_{00_1}, p_{01_3} and p_{10_3}.

The relative freshness labels are checked at the beginning of the reader access. Figure 10(c) shows the extension of the *rd* sub-model. Here the part from starting reading to finishing reading was divided into three. Depending which relative freshness label is set, one of the starting reading transitions t_{37_i} ($i = 1, 2, 3$) fires adding a token to the place $p_{rd\_00\_i}$. The places $p_{rd\_00\_i}$ denote the relative freshness of data obtained by the reader as its token distribution.

The monitoring network for the second and third slot is modelled similar to Figure 10. The *wr* sub-model is modified by dividing the transition t_{14} and t_{15} (finished writing) into four transitions each, and the *rd* sub-model is modified by dividing the part from starting reading transition to finished reading transition into three sub-parts for every slot. The updating of the relative freshness labels is modelled analogously to the one in Figure 10(c).

The monitoring network for the four-slot ACM is modelled in a similar way as the one for the three-slot. However, the model is more complex because of the number of slots, which increases the possibilities of changing the relative freshness labels. In Table 3 the principle of the updating process of the relative freshness labels for the first slot of the four-slot ACM is presented as a textual listing.

The extended PN models of the three-slot and four-slot ACM were constructed and analysed using the GreatSPN2.0.2 tool. All transitions were assumed to have exponential distributions with the mean time given in Table 4. These were obtained from analog simulations [5]. The results of the analysis of relative freshness of the three and four-slot ACM are shown in Table 5.

extension of t_1	input	output
$t_{00\_2\to1\_01}$	p_{00_2}, p_{01_1}	p_{00_1}, p_{01_2}
$t_{00\_2\to1\_10}$	p_{00_2}, p_{10_1}	p_{00_1}, p_{10_2}
$t_{00\_2\to1\_11}$	p_{00_2}, p_{11_1}	p_{00_1}, p_{11_2}
$t_{00\_3\to1\_0110}$	p_{00_3}, p_{01_1}, p_{10_2}	p_{00_1}, p_{01_2}, p_{10_3}
$t_{00\_3\to1\_0111}$	p_{00_3}, p_{01_1}, p_{11_2}	p_{00_1}, p_{01_2}, p_{11_3}
$t_{00\_3\to1\_1001}$	p_{00_3}, p_{10_1}, p_{01_2}	p_{00_1}, p_{10_2}, p_{01_3}
$t_{00\_3\to1\_1011}$	p_{00_3}, p_{10_1}, p_{11_2}	p_{00_1}, p_{10_2}, p_{11_3}
$t_{00\_3\to1\_1101}$	p_{00_3}, p_{11_1}, p_{01_2}	p_{00_1}, p_{11_2}, p_{01_3}
$t_{00\_3\to1\_1110}$	p_{00_3}, p_{11_1}, p_{10_2}	p_{00_1}, p_{11_2}, p_{10_3}
$t_{00\_4\to1\_011011}$	p_{00_4}, p_{01_1}, p_{10_2}, p_{11_3}	p_{00_1}, p_{01_2}, p_{10_3}, p_{11_4}
$t_{00\_4\to1\_011110}$	p_{00_4}, p_{01_1}, p_{11_2}, p_{10_3}	p_{00_1}, p_{01_2}, p_{11_3}, p_{10_4}
$t_{00\_4\to1\_100111}$	p_{00_4}, p_{10_1}, p_{01_2}, p_{11_3}	p_{00_1}, p_{10_2}, p_{01_3}, p_{11_4}
$t_{00\_4\to1\_101101}$	p_{00_4}, p_{10_1}, p_{11_2}, p_{01_3}	p_{00_1}, p_{10_2}, p_{11_3}, p_{01_4}
$t_{00\_4\to1\_110110}$	p_{00_4}, p_{11_1}, p_{01_2}, p_{10_3}	p_{00_1}, p_{11_2}, p_{01_3}, p_{10_4}
$t_{00\_4\to1\_111001}$	p_{00_4}, p_{11_1}, p_{10_2}, p_{01_3}	p_{00_1}, p_{11_2}, p_{10_3}, p_{01_4}

Table 3: A part of the monitoring network of the four-slot ACM

statement	three-slot [ns]	four-slot [ns]
wr	4.42	4.42
w0	2.08	3.67
w1	2.08	5.97
r0	1.38	2.18
r1	/	2.59
rd	3.56	3.56

Table 4: Timing Data

4 Conclusions

This paper presented a method of quantitative analysis of data freshness for multi-slot ACMs using SPN techniques. The concept of relative data freshness was introduced. Relative data freshness described how up to date any item of data that the reader obtains from the ACM is, and classified it in grades from freshest data to least fresh data within the ACM.

The existing PN models of the three-slot and four-slot ACM have been extended by a model for monitoring the relative data freshness in order to perform the analysis using SPN techniques. The extended PN models have been implemented and analysed using the

relative data freshness	three-slot		four-slot	
1	0.0684	59.9%	0.0497	60.6%
2	0.0435	38.2%	0.0302	36.8%
3	0.0022	1.91%	0.0021	2.51%
4	/	/	0.2e-4	0.02%

Table 5: Results of the analysis

GreatSPN2.0.2 tool.

The analysis results show that the relative data freshness behaviours of the three-slot and four-slot ACM are essentially the same.

In the next stage, other ACMs could be studied by using this method. Furthermore, other properties of ACMs could be analysed using a similar method of modelling.

References

[1] H.R. Simpson. Four-slot fully asynchronous communication mechanism, IEE Proceedings, Vol. 137, Pt. E, No.1, pp.17-30, January 1990.

[2] H.R. Simpson, Correctness analysis of class of asynchronous communication mechanisms, IEE Proceedings, Vol. 139, Pt. E. No. 1, pp.35-49. January 1992.

[3] A. Madalinski, F. Xia, A. Yakovlev, Study the data loss and data re-reading behaviour of a four-slot asynchronous communication mechanism using stochastic Petri net techniques, Proceedings of the Seventh UK Asynchronous Forum, Department of Computing Science, University of Newcastle, December 20-21, 1999.

[4] F. Xia, Supporting the Mascot Method with Petri Net Techniques for Real-Time Systems Development, PhD Thesis, London University, London, 2000.

[5] Xia, F., Yakovlev, A., Shang, D., Bystrov, A., Koelmans, A., Kinniment, D.J., Asynchronous Communication Mechanisms Using Self-timed Circuits, Proc. Sixth Int. Symp. on Advanced Research in Asynchronous Circuits and Systems (Async'2000), April 2000, Eilat, Israel, IEEE Computer Society Press, pp. 150-159.

[6] T. Murata. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, Vol. 77, pp.541-580, April 1989.

[7] M. A. Marsan. Stochastic Petri Nets: An elementary introduction. In G. Rozenberg, editor, Advances in Petri Nets 1989, Lecture Notes in Computer Science. Springer Verlag, 1990.

[8] M. A. Marsan, A. Bobbio, S. Donnatelli, Petri Nets in Performance Analysis: An Introduction, In W. Reisig, G. Rozenberg, editors, Lectures on Petri Nets I, Lectures Notes in Computer Science. Springer Verlag, 1998.

[9] M. Ajmone Marsan, A. Bobbio, G. Conte, S. Donatelli, G. Franceschinis. Modelling with Generalized Stochastic Petri Nets, Wiley, 1995.

[10] University of Turin, GreatSPN2.0.2, http://www.di.unito.it/~greatspn.

[11] G. Chiola. GreatSPN Users' Manual, Version 1.3, University of Turin, September 1987.