

# Network Calculus for Real Time Analysis of Embedded Systems with Cyclic Task Dependencies

Henrik Schioler                      Jan Jessen                      Jens Dalsgaard Nielsen                      Kim G. Larsen  
CISS/AAU                      CISS/AAU                      Dept. of Control Eng./AAU                      CISS/AAU  
henrik@control.aau.dk                      jjj@control.aau.dk                      jdn@control.aau.dk                      kgl@cs.aau.dk

CISS: Center for Embedded SW Systems    AAU: Aalborg University, Denmark

## Abstract

The paper addresses performance analysis associated with analysis and design of embedded real time systems with processor sharing among a number of processes. The main contribution concerns the application of Network Calculus (DNC) principles to the performance analysis of such systems, where complex data flow graphs prohibit the use of standard scheduling analysis techniques. Cyclic flow graphs take the problem into new grounds to which the theoretical basis for DNC is extended in this paper. An example is provided to support the practical relevance of the developed approach denoted CyNC and comparative results for CyNC and alternative methods are presented and commented.

## 1 Introduction

Embedded systems environments induce performance demands in the specifications for such systems. Therefore performance analysis is an indispensable part of the design process. Analysis of the time-related performance during design of embedded systems has been approached from various angles, such as scheduling analysis [1], completion time analysis [2] and model checking in timed automata [8, 9].

Completion time analysis assumes no or limited dependence between tasks. Task dependence is allowed through blocking in critical regions protecting shared resources. Results for fixed or dynamic priority scheduling, fixed cyclic schedules and round robin exist for that case. Automata-based model-checking in real time systems is based on timed automata modelling. Various externally observable time related properties can be expressed as timed logical propositions, which are then transformed into timed automata augmenting the system automata model.

Frequently task dependency models allowing only mutual exclusion by critical regions are not feasible. F.ex. in communication devices *producer-consumer* communication patterns between tasks are often met. Such dependencies bring completion time analysis off grounds and tends to enlarge to reachability sets in timed automata significantly.

Within the field of communication systems a theoretical framework for non-stochastic performance analysis has emerged with the network calculus (DNC) of [4], [3]. DNC is based on specifications in the shape of lower and upper bounds on traffic as well as service. From specifications, upper bounds on backlogs, waiting time in queues and overall response time for specified transactions may be deduced.

This work explores how to bring DNC to use in the analysis of embedded real time systems. Earlier work including [5] introduced DNC for embedded systems with heterogenous architectures and acyclic task dependencies. A related approach is presented in [6], where a *periodic with jitter* model is adopted. Results for cyclic flow graphs are provided under the *periodic with jitter* assumption in [7] including phase relations between events in the analysis. [3] provides results for cyclic dependencies for affine upper flow constraints and lower service constraints. Results for that approach (ADNC) are provided in this paper for comparison on a common example. This work provides theoretical results for cyclic flow graphs under the more general assumptions of arbitrary minimum and maximum flow constraints and service guarantees.

Initially a short introduction to performance analysis for embedded systems is given through an example illustrating a typical situation. DNC is introduced and a suggestion for migrating this technique to the analysis of embedded systems is presented. A general theoretical basis for the presented approach (CyNC) is given. CyNC is illustrated for a relevant example

and results are compared to ADNC and model checking. Algorithmic complexity issues are discussed before conclusive remarks are given along with suggestions for directions of future research.

## 2 Analysis and Design of Embedded Real Time Systems

The context of embedded real time systems is constituted by the set of external interfaces connected to the system [11]. Timing analysis of external interfaces is typically initiated by categorizing devices according to the time process describing how stimuli from such devices are generated. Sensor and actuator devices for control purposes are typically periodically generated, whereas network interfaces are often aperiodic and finally alarm sensor devices are sporadic. Application domain analysis specifies functional as well as performance demands, so that all user and system transactions are specified in both value and time domain. Control transactions f.ex. include sensor reading, computing response and writing the response to actuator devices. Timely behaviour is specified by an upper bound  $\Delta$  less than the period  $T$  for the time between control request and initiating the sensor reading as well the maximum response time  $\delta \ll T$  between sensor-read and -write. Control subsystems defined during design typically map to a number of independent control tasks, which are eventually determined during a task structuring phase, merging tasks according to various cohesion criteria [11]. Under fixed priority scheduling, completion time analysis provides response bounds for independent tasks and may even account for bounded blocking from lower priority tasks. However completion time analysis fails to handle more complex transactions including various patterns of inter-task communication. Consider the Task Architecture Diagram (TAD) shown in figure 1. The task architecture constitutes the result of a preliminary design for an embedded controller unit accesible through LAN connection. Asynchronous inter-task communication has been decided to provide decoupling of *device driver* tasks from internal tasks. The design is divided into: a number of periodic *control* tasks, a *network transport layer interface*, a *communication agent* distributing messages to and from the network interface, an *information agent* compiling data into formatted replies upon requests from local user interface or from the network and finally a *data abstraction server* receiving and storing measurement data from control tasks.

The design supports various use cases including the *in-*

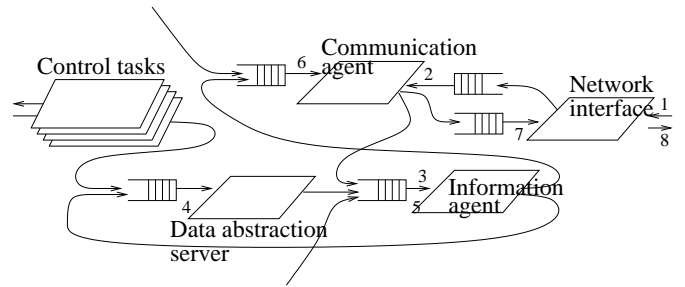


Figure 1: Task Architecture Diagram (TAD) for LAN accessible controller unit.

*formation retrieval* transaction exhibiting a data flow passing through all tasks except control tasks as indicated by the sequence numbering in figure 1. For bounded external traffic, performance analysis should provide response time bounds for entire transactions like the information retrieval indicated in figure 1 and others.

## 3 Deterministic Network Calculus (DNC)

DNC assumes bounded traffic/job flows. Defining  $R(t)$  as the accumulated server/cpu time-requests within  $[0, t]$ , we say that such a flow ( $R$ ) is constrained by  $\alpha_L$  and  $\alpha_U$  iff

$$\alpha_L(t-s) \leq R(t) - R(s) \leq \alpha_U(t-s) \quad \forall 0 \leq s \leq t \quad (1)$$

i.e.

the traffic within  $(s, t]$  is confined to  $[\alpha_L(t-s), \alpha_U(t-s)]$ .

Similarly the service/cpu time available within  $(s, t]$  to some flow may be bounded by functions  $\beta_L$  and  $\beta_U$ . Provided bounded external input flows, flow and service bounds may be propagated through a network of service elements/tasks. Computational models for various service disciplines exist in [4],[3] and [5].

### 3.1 DNC for Modelling and Analysis of Embedded Systems

DNC defines for network communications a number of generic network element types. In the following a map between DNC elements and their counterparts in embedded systems is constructed. (DNC elements are written in *italic.*)

- Interfaces to external devices are mapped to external

*generators* generating external traffic flows, specified by upper and lower bounds.

- Message buffers are mapped to network buffers receiving bounded traffic and served under some discipline (FIFO, FP, WFQ) by a *service* element.
- Bounded *internal flows* connecting buffers or escaping system boundaries model message streams between processes.
- Execution processes map to *service elements* serving buffers and specified by lower and upper service bounds.
- Flow *convergence* elements, where traffic flows are aggregated, model message flows entering a common buffer.
- Flow *divergence* elements where traffic flows are split into separate subflows model the internal routing of messages in the system.

Migrating the DNC framework from network communication to the task communication in embedded systems involves re-interpretation of some of the element types above and modification of some as well. Network service elements naturally bound output flows  $R'$  below corresponding input flows  $R$ , i.e.

$$R'(t) \leq R(t) \quad (2)$$

This is not generally the case for tasks in an embedded system, since a single input message may stimulate the transmission of a variable number of output messages as well as require variable service time at various tasks. We suggest scaling factors associated to pairs of corresponding input and output flows. Consider a number of flows  $\{f_i\}$  entering an ingress buffer of some task  $\tau$  producing a number of output flows  $\{g_j\}$  in return. Each message from  $f_i$  is then assumed to produce  $n$  messages to  $g_j$  each requiring  $s$  time units of service at the receiving task. In that case arrival constraints for the output flow should be multiplied to  $S_{ij}^\tau = n \cdot s$ . Scaling factors are allowed to be non-integer in which case the effect of packetizing can be accounted for by adding/subtracting the size of one packet to the upper and lower constraints.

## 4 Performance Analysis

Performance analysis in network calculus is achieved by propagating constraints through and between service elements of the modelled system along data and service flows.

### 4.1 Propagating through Service Elements

Considering a service element generating an output flow  $R'$  in return for an input flow  $R$  bounded as in 1. When service is bounded by  $\beta_L$  and  $\beta_U$  the output flow  $R'$  is bounded by  $\alpha'_L$  and  $\alpha'_U$ , where

$$\begin{aligned} \alpha'_L(t) &= \inf_{0 \leq u \leq t} \{\alpha_L(u) + \beta_L(t - u)\} \\ \alpha'_U(t) &= \inf_{0 \leq u \leq t} \{\sup_{v \geq 0} \{\alpha_U(u + v) - \beta_L(v)\} + \beta_U(t - u), \beta_U(t)\} \end{aligned} \quad (3)$$

Likewise remaining processing resources are bounded by  $\beta'_L$  and  $\beta'_U$  where

$$\begin{aligned} \beta'_L(t) &= \sup_{0 \leq u \leq t} \{\beta_L(u) - \alpha_U(u)\} \\ \beta'_U(t) &= \sup_{0 \leq u \leq t} \{\beta_U(u) - \alpha_L(u)\} \end{aligned} \quad (4)$$

Expressions 3 and 4 are derived [5], and used for propagating constraints along job flows for systems with acyclic task dependencies.

### 4.2 Propagating between Service Elements

Flow between tasks may in our model be scaled (by factors  $S_{ij}^\tau$ ) and aggregated so that flow constraints are computationally propagated through affine operators, i.e.  $\alpha_U = \mathbf{A}\alpha'_U + Ua$  and  $\alpha_L = \mathbf{A}\alpha'_L + La$ .  $\mathbf{A}$  is a stable matrix of scaling factors defined by system data flow, vectors  $\{\alpha_U, \alpha_L\}$  and  $\{\alpha'_U, \alpha'_L\}$  comprise arrival bounds on flows entering and leaving service elements respectively and  $\{Ua, La\}$  account for external flow constraints and additive effects from packetizing. Bounds on remaining processing resources are propagated from higher to lower priority tasks, i.e.  $\beta_U = \mathbf{B}\beta'_U + Ub$  and  $\beta_L = \mathbf{B}\beta'_L + Lb$ .  $\mathbf{B}$  is a stable matrix with boolean entries defined by priority order and vectors  $\{\beta_U, \beta_L\}$  and  $\{\beta'_U, \beta'_L\}$  comprise service bounds passed on and received by service elements respectively.  $\{Ub, Lb\}$  account for externally given bounds on processing resources.

Composing mappings 3 and 4 with the affine transformations above gives an overall system mapping  $\Gamma$  defined by 5

$$(\alpha'_U, \alpha'_L, \beta'_U, \beta'_L) = \Gamma(\alpha'_U, \alpha'_L, \beta'_U, \beta'_L, Ua, La, Ub, Lb) \quad (5)$$

For acyclic dependency a finite number of applications of the overall system mapping  $\Gamma$  produces flow and service constraints in an order defined by the dependency

graph. When dataflows are cyclic or are directed from lower to higher priority tasks under processor sharing, cyclic dependencies between constraints appear. In that case constraints are given implicitly by 5 as a fix point to  $\Gamma$ .

## 5 Theoretical Basis

From a theoretical perspective we should ask for existence and uniqueness of solutions to 5 as well as an algorithm solving 5 in some reasonable sense. We shall subsequently leave out the question of existence and establish results for uniqueness and algorithm.

### 5.1 Relevant set of solutions

We shall define upper and lower bounds to be *linearly separated*, when there are positive rates  $A_U \geq A_L$  so that

$$\alpha_L(t) \leq A_L t \leq A_U t \leq \alpha_U(t) \quad \forall 0 \leq t \quad (6)$$

$$\lim_{t \rightarrow \infty} \frac{\alpha_L(t)}{t} = A_L, \quad \lim_{t \rightarrow \infty} \frac{\alpha_U(t)}{t} = A_U$$

and likewise for service bounds. We shall assume externally given constraints to be linearly separated. (Linear separation is a fairly general assumption and fulfilled under e.g. the *periodic with jitter* assumption of [6] and [7].) Under this assumption the long term rates of external flows may be as high as  $A_U$  and as low as  $A_L$ . In a stable (sufficient service rates) lossless system, long term bounds  $A_U$  and  $A_L$  on internal flow rates may be found from standard flow equations, i.e.

$$\begin{aligned} A_U &= \mathbf{A} A_U + r_U^A \Rightarrow A_U = (\mathbf{I} - \mathbf{A})^{-1} r_U^A \\ A_L &= \mathbf{A} A_L + r_L^A \Rightarrow A_L = (\mathbf{I} - \mathbf{A})^{-1} r_L^A \end{aligned} \quad (7)$$

where vectors  $r_U^A$  and  $r_L^A$  contain external upper and lower flow rates.

Service rate bounds may be found similarly, i.e.

$$\begin{aligned} B_U &= \mathbf{B} B_U + r_U^B - \mathbf{F} A_L \\ \Rightarrow B_U &= (\mathbf{I} - \mathbf{B})^{-1} (r_U^B - \mathbf{F} A_L) \\ B_L &= \mathbf{B} B_L + r_L^B - \mathbf{F} A_U \\ \Rightarrow B_L &= (\mathbf{I} - \mathbf{B})^{-1} (r_L^B - \mathbf{F} A_U) \end{aligned}$$

where the boolean matrix  $\mathbf{F} = \{F_{ij} \text{ iff flow } j \text{ enters task } i\}$  and  $r_U^B, r_L^B$  denote external long-term service-rate bounds.

Thus

**Lemma 1** *Internal flows  $R$  may exhibit long term rates within  $[A_L, A_U]$  and internal services may have rates in the interval  $[B_L, B_U]$ .*

Now assume for some internal flow  $R$  and positive time  $T$  that  $\alpha_U(T) = A T \leq A_U T$ , so that  $\frac{R(t)}{t} \leq \frac{A \lceil \frac{t}{T} \rceil T}{t} \leq A \frac{t+1}{t}$  for  $t \geq T$ , which is less than  $A_U$  for large  $t$ . So by contradiction to lemma 1  $\alpha_U(t) \geq A_U t$  for all  $t \geq 0$ . An equivalent argument holds for lower flow bounds and service bounds so that internal flow and service bounds are linearly separated by  $A_L, A_U$  and  $B_L, B_U$  respectively, which in turn should hold for any solution  $x^*$  to 5. Such solutions are termed *relevant* in the sequel.

### 5.2 Uniqueness and Algorithm

Initially we rewrite 5 into an iteration, i.e.

$$\begin{aligned} &(\alpha_U^{n+1}, \alpha_L^{n+1}, \beta_U^{n+1}, \beta_L^{n+1}) \\ &= \Gamma(\alpha_U^n, \alpha_L^n, \beta_U^n, \beta_L^n, Ua, La, Ub, Lb) \end{aligned} \quad (8)$$

Inspecting 3 and 4 reveals that  $\Gamma$  is monotone in each of its components when relations ( $\leq, \geq, =$ ) are interpreted elementwise for vectors and pointwise for real functions. If  $+, -, 0$  denote non-decrease, non-increase and invariance, we may describe the monotonicity of  $\Gamma$  by the matrix  $M$

$$M = \left[ \begin{array}{cccc|cccc} + & 0 & + & - & + & 0 & + & - \\ 0 & + & - & + & 0 & + & - & + \\ + & - & + & 0 & + & - & + & 0 \\ - & + & 0 & + & - & + & 0 & + \end{array} \right] \quad (9)$$

indicating e.g. by  $M(1,1) = +$  that  $\alpha_U^{n+1}$  (in 8) does not decrease with  $\alpha_U^n$  and by  $M(2,5) = 0$  that  $\alpha_L^{n+1}$  is invariant to changes in  $Ua$  etc.

Assume the existence of one or more finite relevant solutions to 5 and let

$$\begin{aligned} x^* &= (\alpha_U^*, \alpha_L^*, \beta_U^*, \beta_L^*) \\ &= (\min(\alpha_U), \max(\alpha_L), \min(\beta_U), \max(\beta_L)) \end{aligned} \quad (10)$$

where min and max are taken over all relevant solutions of 5.

Also assume the existence of an initial point  $x^0 = (\alpha_U^0, \alpha_L^0, \beta_U^0, \beta_L^0)$  so that

$$(\alpha_U^0 \leq \alpha_U^*, \alpha_L^0 \geq \alpha_L^*, \beta_U^0 \leq \beta_U^*, \beta_L^0 \geq \beta_L^*) \quad (11)$$

and

$$(\alpha_U^1 \geq \alpha_U^0, \alpha_L^1 \leq \alpha_L^0, \beta_U^1 \geq \beta_U^0, \beta_L^1 \leq \beta_L^0) \quad (12)$$

then the sequence  $\{\alpha_U^n, \alpha_L^n, \beta_U^n, \beta_L^n\}$  (by monotonicity properties of 9 of  $\Gamma$ ) fulfills

$$(\alpha_U^n \leq \alpha_U^*, \alpha_L^n \geq \alpha_L^*, \beta_U^n \leq \beta_U^*, \beta_L^n \geq \beta_L^*) \quad (13)$$

as well as

$$(\alpha_U^{n+1} \geq \alpha_U^n, \alpha_L^{n+1} \leq \alpha_L^n, \beta_U^{n+1} \geq \beta_U^n, \beta_L^{n+1} \leq \beta_L^n) \quad (14)$$

which means that the iterands of 8 converge to  $x^*$  which is therefore itself a uniquely determined *best* fixpoint comprising lowest upper bounds and highest lower bounds.

### 5.3 Initial point

Our result so far relies on the existence of some initial point  $x^0$  fulfilling 11 and 12 as well as at least one relevant finite solution to 5.

Consider as an initial point  $x^0 = (A_U t, A_L t, B_U t, B_L t)$ , where  $(A_U, A_L, B_U, B_L)$  is the solution of 7 and 8. Since  $x^*$  is relevant 11 is directly fulfilled.

It is readily established that  $x^0$  is a fixpoint of  $\Gamma$  for  $U_A(t) = r_U^A t, L_A(t) = r_L^A t, U_B(t) = r_U^B t, L_B(t) = r_L^B t$ . Since external constraints are assumed to be linearly separated, monotonicity properties 9 of  $\Gamma$  imply that  $x^0$  also fulfills 12.

We have in this way established the uniqueness of a best fixpoint  $x^*$  and the existence of an iterative algorithm 8 for finding it.

## 6 EXAMPLE

Performance analysis is illustrated with a simplified abstraction of the information retrieval transaction described above. As shown in figure 2 two tasks

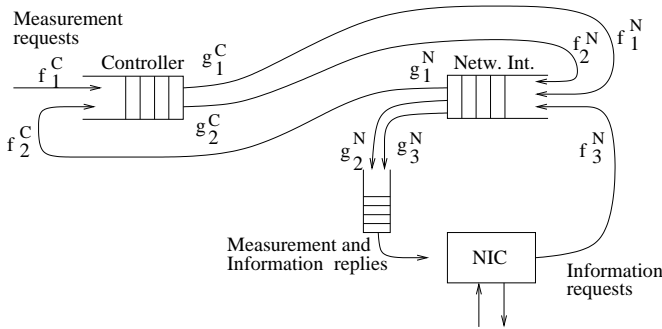


Figure 2: Simplified architecture for LAN accessible controller unit.

are defined; *controller* ( $\tau^C$ ) and *network interface* ( $\tau^N$ ) along with corresponding message buffers. Two external flows are defined; *measurement requests* and *information requests* from the LAN. An independent net-

work interface controller (NIC) with FIFO acts as message sink.

### 6.1 Network modelling

$\tau^C$  serves two input flows; the measurement flow  $f_1^C$  and a flow of information retrieval messages  $f_2^C = g_1^N$  from  $\tau^N$ . (Notation  $f$  and  $g$  is used to distinguish input and output flows respectively. Subscripts denote flow indexing, whereas superscripts indicate task association.) Information retrieval messages flow through  $f_3^N$  to  $\tau^N$  and are forwarded to  $\tau^C$  through  $f_2^C = g_1^N$ . Reply messages from  $\tau^C$  flow across  $g_2^C = f_2^N$  to  $\tau^N$  and produce a direct flow  $g_2^N$  to the NIC FIFO. Every 30 th. measurement is sent unrequested to the network interface, through the flow  $g_1^C = f_1^N$  to  $\tau^N$  and finally by  $g_3^N$  to the NIC.

All together non-zero scaling factors include:  $S_{1,1}^C = 1/30, S_{2,2}^C = 1, S_{1,3}^N = 1, S_{2,2}^N = 1$  and  $S_{3,1}^N = 1$ .

We assume in this example a non preemptive fixed priority scheduling discipline, where tasks are defined to be ready, whenever ingress buffers are nonempty. In the presented example it is assumed that  $\tau^C$  has priority over  $\tau^N$ . Locally tasks serve inflows from a common buffer in a FIFO discipline.

### 6.2 Numerical results

Numerical results are given for the example in figure 2. Message processing times are set to 5 generally. External flows are assumed to have affine constraints, i.e.  $\alpha_U^{sensor} = 1/10t + 1, \alpha_L^{sensor} = [1/10t - 1]^+$  and  $\alpha_U^{network} = 1/40t + 5, \alpha_L^{network} = [1/40t - 5]^+$  given in units of messages of length 5. Note that upper and lower rates are assumed equal, which however is not mandatory. Summing over all rates gives a system utilization  $5(1/10 + 1/40 + 1/10/30 + 2/40) = 0.9$ , which may be characterized as heavy load.

A coarse solution through ADNC is provided by assuming affine internal traffic and service bounds and using only upper traffic and lower service bounds. More information on ADNC is found in [3] pp. 104. Table 1 show selected numerical results. All results are given in units of CPU time. The delay  $d_{InfRet}$  the information retrieval transaction is found by adding delays along the data flow. Simulation and model checking is performed on the same timed automata model, which is abstracted only to account for total backlogs due to limited computational resources. Delay estimates are not provided from the timed automata model since this would require model complexity beyond our present computational abilities.

CyNC estimates generally improve ADNC estimates

	$q_1^C$	$q_2^C$	$q_1^N$	$q_2^N$	$q_3^N$	$q^C$	$q^N$	$d^C$	$d_N$	$d_{InfRet}$
ADNC	105	280	31	396	191	385	618	200	1330	2860
CyNC	5.5	5.5	12	65	63	11	140	6	337	680
Simulation	—	—	—	—	—	10	20	—	—	—
Model checking	—	—	—	—	—	10	35	—	—	—

Table 1: Selected numerical results

significantly, however model checking reveals that not all estimates are tight. Simulation results indicate that extreme values may be hard to capture under simulation.

Iterates of the fix point iteration 8 are shown in figure 3 illustrating how upper and lower bounds diverge monotonously from the initial linear estimate as predicted by formula 14 in section 5. A timed automata

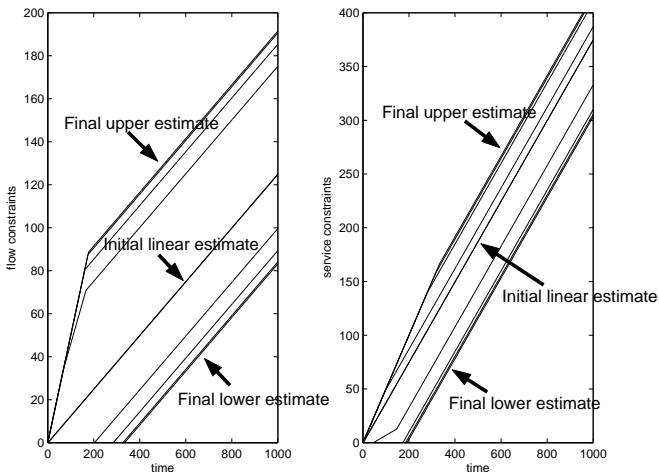


Figure 3: Iterates of fix point iteration for flow and service constraints.

model of the presented example is constructed and UPPAAL [10] is used for simulation and model checking. The UPPAAL diagram of a token bucket filter modelling affine flow constraints is shown in figure 4.

## 7 Complexity Issues

ADNC requires solving a system of an order matching the number of flows  $n$  defined in the system. Linear systems are solved in  $n^3 + n^2$  floating points operations (flops). In the presented example this amounts to 150 flops.

CyNC requires numeric iteration in a space of constraint functions. It is believed that the required num-

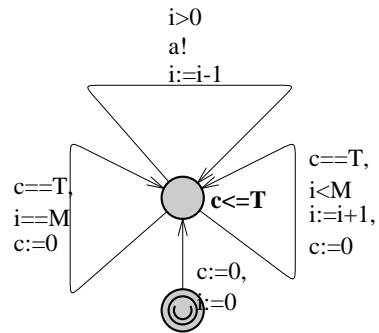


Figure 4: Timed Automata Model of Token Bucket Filter

ber of iterations for a sufficiently precise solution does not depend on the number of constraint functions but rather on the overall load of the analysed system. If so the computational complexity of the presented solution is linear in the number of constraint functions as well as the size of their numerical representation. For the presented example 10 iterations amounting to 1E6 flops are required. The implemented algorithms apply a naive table based representation of constraint functions and it is expected that possibilities for efficiency improvement are large.

Exact backlog bounds 2 and 7 have been found through checking existence of paths in the timed automata model leading to backlogs 2 and 3 for the controller queue and 7 and 8 for the network queue this requires traversing the entire reachability set. Thus exact bounds may potentially require exponential complexity. In the presented case the reachability set includes 1E6 states, which however is for the abstracted model accounting only for total backlogs.

## 8 Conclusion

The application of network calculus (DNC) to the analysis and design of embedded systems has been suggested as well as several modifications of DNC needed

for this application area.

A theoretical basis for the application of DNC for cyclic task dependencies and arbitrary constraint assumptions is provided. The presented theory extends previous work of [3] and [5].

An example is provided for comparison between 3 different approaches; a coarse DNC based approach (ADNC) inspired by [3], the developed CyNC approach and model checking in a timed automata model. Model checking provides exact backlog bounds, whereas the ADNC yields rather conservative bounds for backlog and message delay. CyNC yields significantly less conservative bounds than ADNC. CyNC bounds are close to exact bounds except when aggregating flows with phase relations.

ADNC is by far the lightest from a complexity viewpoint. Complexities for CyNC and model checking are comparable for the presented example. This balance is however expected to point in favour of CyNC for larger system models, since CyNC is believed to be polynomial whereas model checking is exponential.

CyNC is illustrated by an example assuming fixed priority and FIFO scheduling disciplines, which despite their popularity in industrial designs are only two among a number of applied principles like *Round Robin*, *Earliest Deadline First* and *Weighted Fair Queuing*. Generalization of the presented approach to other scheduling disciplines is postponed to future work.

## References

- [1] C. L. Liu and J. W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment., *JACM*, pp.46-61,1973.
- [2] J. P. Lehoczky, L. Sha and Y. Ding, The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behaviour, *Proceedings of the 10th IEEE Real-Time Systems Symposium*, pp. 166-171, 1989.
- [3] J. L. Boudec and P. Thiran *Network Calculus - A Theory of Deterministic Queuing Systems for the Internet.*, Springer-Verlag, 2000.
- [4] R. L. Cruz, *A Calculus for Network Delay Part I: Network Elements in Isolation*, IEEE trans. on Information, 1991.
- [5] Samarjit Chakraborty, Simon Knzli, Lothar Thiele, Andreas Herkersdorf, Patricia Sagmeister, *Performance Evaluation of Network Processor Architectures: Combining Simulation with*

*Analytical Estimation*, Computer Networks, Vol. 41, No. 5, pages 641–665, 2003.

- [6] Marek Jersak and Rolf Ernst, *Enabling Scheduling Analysis of Heterogeneous Systems with Multi-Rate Data Dependencies and Rate Intervals.*,40th Design Automation Conference (DAC), Anaheim, USA, June, 2003
- [7] Marek Jersak, Kai Richter and Rolf Ernst, *Performance Analysis for Complex Embedded Applications*,(to appear in) the International Journal of Embedded Systems, Interscience Publishers, 2004
- [8] Larsen, Kim G. and Steffen, Bernhard and Weise, Carsten, *Continuous Modelling of Real Time and Hybrid Systems: From Concepts to Tools*, International Journal on Software Tools for Technology Transfer, pp. 64–85, Vol. 1, No. 1, 1997.
- [9] R. Alur and D.L. Dill, *A Theory of Timed Automata*, Theoretical Computer Science, pp. 183–235, 1994.
- [10] K. G. Larsen, W. P. Pettersson and W. Li, *UPPAAL in a Nutshell*, Int. Journal on Software Tools for Technology Transfer, Vol. 1, 1997.
- [11] H. Gomaa, *Software Design Methods for Concurrent and Real-Time Systems.*, Addison Wesley, 1993.